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

Szatmári Bendegúz Bence

VEZETÉK NÉLKÜLI KOMMUNIKÁCIÓS EGYSÉGGEL ELLÁTOTT MODELLAUTÓ ELEKTRONIKAI TERVEZÉSE

Konzulens

Dr. Orosz György

BUDAPEST, 2014

Tartalomjegyzék

Kivonat **5**
Abstract **6**
1 Bevezetés **7**
2 Specifikáció és rendszerterv **9**
3 Mechanikai alkatrészek **11**
3.1 Alkatrészek rögzítése 11
4 Elektronikai tervezés **13**
4.1 Mikrovezérlő választása 13
4.1.1 Periféria interfészek 14
4.1.2 Nyolc bites mikrovezérlők 16
4.1.3 Harminckét bites mikrovezérlők 19
4.1.4 STM32F407VGT6 22
4.1.5 STM32F4Discovery 24
4.2 Tápellátás 25
4.3 Feszültségmérés 27
4.4 Gyorsulásmérés 27
4.4.1 Lehetőségek a piacon 28
4.4.2 LIS3DSH 28
4.5 Motorok, motormeghajtás 29
5 Kommunikációs egység és grafikus felhasználói felület **34**
5.1 Vezeték nélküli kommunikációs típusok 34
5.1.1 Bluetooth modulok 36
5.1.2 HC-06 típusú Bluetooth modul 38
5.2 Grafikus felhasználói felület 39
5.2.1 MATLAB 39
6 Megvalósított rendszer **41**
6.1 Fő programrész 41
6.2 AD átalakító 43
6.3 Bluetooth parancsok 44
6.4 Külső megszakítás 44
6.5 LED-ek 44

6.6 Motor vezérlés 45
6.7 Gyorsulásmérés 45
6.8 Időzítő egység 46
6.9 UART interfész 46
6.10 Grafikus felhasználói felület 47
6.11 Megépített rendszer 51
7 Mérési eredmények 52
7.1 Kommunikációs tulajdonságok mérése 52
7.1.1 Maximális kommunikációs távolság 52
7.1.2 Kommunikációs sebesség 52
7.2 Mozgási tulajdonságok mérése 55
7.2.1 Mechanikai eredetű egyenesfutási hiba és javítása 55
7.2.2 Gyorsulás X irányban 56
7.2.3 Gyorsulás Y irányban 58
7.3 Akkumulátor feszültségének mérése 59
8 Összefoglalás, továbbfejlesztés 61
Irodalomjegyzék 62
Függelék 64

HALLGATÓI NYILATKOZAT

Alulírott **Szatmári Bendegúz Bence**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot 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, 2014. 12. 19.

…….
Szatmári Bendegúz Bence

Kivonat

A dolgozat témája egy modellautó elektronikai tervezése, amely vezeték nélküli kapcsolaton keresztül egy PC-vel kommunikál. A cél egy könnyen kezelhető, grafikus felületből irányítható autó megvalósítása volt. Legelőször az autó alvázát és a motorok rögzítését terveztem meg és építettem meg, könnyű anyagokból. Következő lépésként megmértem a motorok áramfelvételét és az autó által nagyjából elérhető maximális gyorsulás értékét. Erre alapozva megterveztem és megépítettem a meghajtó áramkört, majd kiválasztottam a lehetőségnek megfelelően a legjobb felbontású és érzékenységű gyorsulásmérőt. Ezután a vezérlőegységek terén végzett kutatásom alapján választottam egy erre a feladatra alkalmasat, majd nekiláttam az ehhez és egyben a PC-hez illeszthető vezeték nélküli kommunikációs típus és egység kiválasztásának. Munkámat a rendszer tápellátó áramkörének tervezésével és megépítésével folytattam, ügyelve arra, hogy a motorok hirtelen nagy teljesítményfelvétele ne befolyásolja a rendszer működését. Miután elkészültem a hardver tervezésével és építésével, megírtam a vezérlő szoftverkomponensét, valamint létrehoztam PC-n egy grafikus felületet az autó irányítására és a mérési adatainak nyomon követésére, tárolására. Végül méréseket végeztem, ahol az autó kommunikációs és mozgási tulajdonságait mértem föl.

Abstract

The topic of this thesis is the electronic design of a model car, which communicates through a wireless connection with a PC. The purpose was to create a car which is easily manageable and can be controlled from a graphical user interface. First of all I have designed and constructed the chassis and the framework of the engines from light materials. In the next step I measured the current consumption of the engines and the acceleration of the car. Based on this I have designed and built the motor driving circuit, and I have chosen an accelerometer with the best resolution and sensitivity according to the possibilities. After this based on my researches on controller units, I have chosen one which fits the task, then I turned to choose a wireless communication type and module. I have continued my work with the design and construction of the power supply circuit with taking care that the sudden high power consumption of the engines should not influence the function of the system. After I have finished with the design and construction of the hardware, I wrote the controller's software component and also I set up a graphical user interface on a PC for car controlling and measurement data tracking and storing. At last I did measurements about the communication and movement properties of the model car.

1 Bevezetés

Manapság nagyon sokféle vezeték nélküli kommunikáció létezik, amivel autonóm eszközöket lehet vezérelni, mint például IrDA, Bluetooth, Zigbee, Wifi. Ezek különböző teljesítményű, fogyasztású, adatátviteli sebességű típusok, ebből kifolyólag különböző alkalmazásoknál más-más típust használnak kommunikációs célokra. A feladatok nagy részét meg lehet oldani szinte bármelyik vezeték nélküli kommunikációval, egymás helyettesítésére képesek, ha megfelelnek az adott feltételeknek. De az adott szituációban lehet, hogy az egyik használata fölösleges bonyolítást vagy elkerülendő plusz költséget (fogyasztás, kiegészítő hardver) von magával, míg a másik egy egyszerű megoldást biztosítana.

Tipikus példája a vezeték nélküli alkalmazásoknak egy szenzorhálózat, ami általában egy központi feldolgozó egységből és ehhez az csatlakozó mérő végeszközökből áll. Nincs mindig lehetőség vezetékes összeköttetésre vagy ha van, akkor drága megoldás lenne, így kézenfekvő vezeték nélküli kommunikációt alkalmazni. A nagyteljesítményű központi egység lehet például PC, DSP (Digital Sign Processor), FPGA (Field-programmable Gate Array), de akár mikrovezérlő is. Költséghatékonyság szempontjából például a PC a legalkalmasabb ha rendelkezésre áll, de például valós idejű, időkritikus feldolgozás megvalósítására az operációs rendszerek késleltetése miatt nem a legjobb opció. A végeszközök tipikusan egy mikrovezérlőből és az e mellé illesztett szenzor perifériákból állnak. Ezek a vezérlőegységek többnyire adatgyűjtést végeznek, akár valamilyen algoritmus szerint, majd továbbítják a feldolgozó egység felé.

Szakdolgozatom keretein belül, az általam megvalósított rendszer egy Bluetooth-on keresztül kommunikáló modell autó, ami az egyszerű alkatrészekből létrehozható autonóm egység tervezési és megvalósítási lépéseit kívánja bemutatni. Központi feldolgozó egységnek PC-t választottam, végeszköznek pedig egy mikrovezérlőt. A modellautó tekinthető egy mérőrendszernek, amely egy gyorsulásmérővel van felszerelve, ezzel nyomon követhető a mozgása, valamint a mikrovezérlőbe integrált analóg-digitális átalakítóval az akkumulátor feszültsége is.

A 2. fejezetben átfogóan, egy blokkvázlat segítségével bemutatom a rendszer működését, a részletekre a következő fejezetekben térek ki.

A 3. fejezetben mechanikai alkatrészeket, alapanyagokat mutatom be, valamint a rendszer összeszereléséről írok.

A 4. fejezet elektronikai alkatrészek részletes bemutatásáról szól, hogy mit és miért választottam ennek a rendszernek a megvalósításához.

Az 5. fejezet a vezeték nélküli kommunikációs egységek és grafikus felületek világába ad egy kis betekintést, majd leírom mit találtam megfelelőnek.

A 6. fejezetben részletesen leírom miként valósítottam meg a rendszer szoftveres részét, itt kitérek minden perifériára is részletesen.

A 7. fejezetben a rendszer végzett mérésekről, az ebből levonható következtetésekről írok.

A 8. fejezetben összefoglalom az általam végzett munkát, valamint szót ejtek a továbbfejleszthetőségről

2 Specifikáció és rendszerterv

Az autót irányító vezérlő egységnek nem kell nagy számítási teljesítménnyel, adatfeldolgozási képességgel bírnia, mivel ezt a PC fogja végezni. Ebből a megfontolásból kiindulva nem szükséges DSP vagy FPGA alkalmazása erre a célra, hanem elegendő egy mikrovezérlő.

A rendszer tekinthető master-slave viselkedésűnek, mivel a mikrovezérlő kommunikációt és adatátvitelt nem kezdeményez, hanem parancsokat vár a PC-től és ezeknek megfelelően feladatokat hajt végre, tehát slave egységként működik. Alapvető feladata az adatgyűjtés (gyorsulás- és feszültség értékek) és továbbítás, motor vezérlés, valamint a perifériák kezelése.

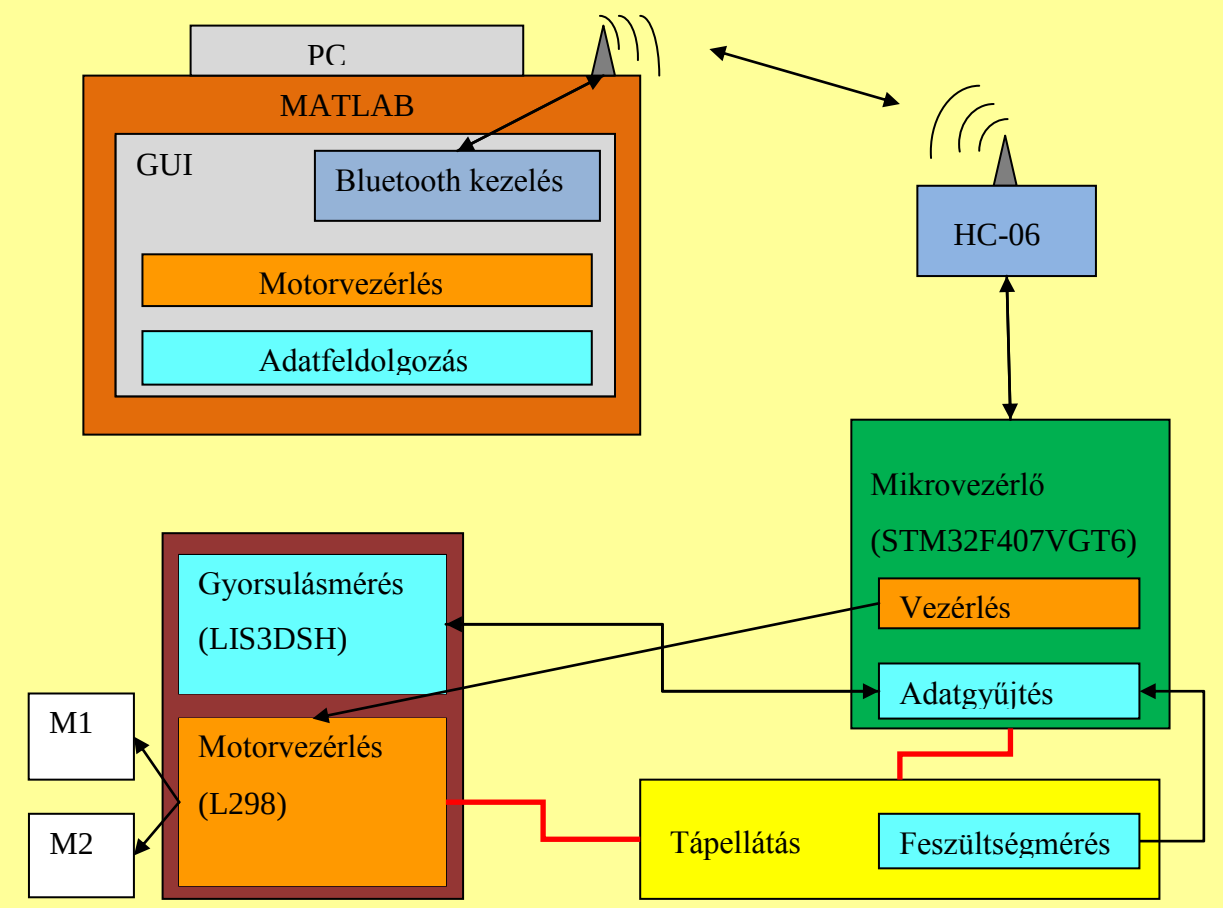
A rendszerben a Master szerepét a PC tölti be. A PC-n MATLAB-ban létrehozott grafikus felületen keresztül lehetőség adódik az autó magas szintű vezérlésére egyszerű parancsokkal (például billentyűlenyomással), valamint a mért adatok tárolására és grafikonokon történő megjelenítésére.

A két egység Bluetooth-on keresztül fog vezeték nélküli kommunikációt végrehajtani, ami a Bluetooth eszköz osztályától függően akár 10 m-es távolságból is történhet. Ezen a csatornán küldi az autó az adatokat a PC felé, valamint fogadja a parancsokat.

Az autó három kerékkel fog rendelkezni, ezek közül egy támasztó szerepet tölt be, míg a másik kettő egy-egy motor tengelyére van felszerelve. A motorvezérlés lehetővé teszi, hogy két motor egymástól függetlenül lehessen működtetni. A motorok meghajtásának mértékét a vezérlő PWM jel kitöltési tényezőjével lehet állítani.

Mivel egy mozgó, vezeték nélküli rendszerről van szó, ezért egy akkumulátor lesz a rendszer tápegysége, amiről nagyjából fél órán keresztül lesz képes folyamatosan működni.

A rendszer blokkvázlat felépítését a 2.1-es ábra mutatja be. A narancssárga a vezérlési, míg a világoskék az adatgyűjtési, feldolgozási feladatokat jelöli. A Bluetooth kommunikációban résztvevő egységek a kék színű egységek az ábrán. A nyilak a kommunikáció (adatátvitel) és a vezérlés irányát mutatják. A két piros vonal a tápellátást szimbolizálja.

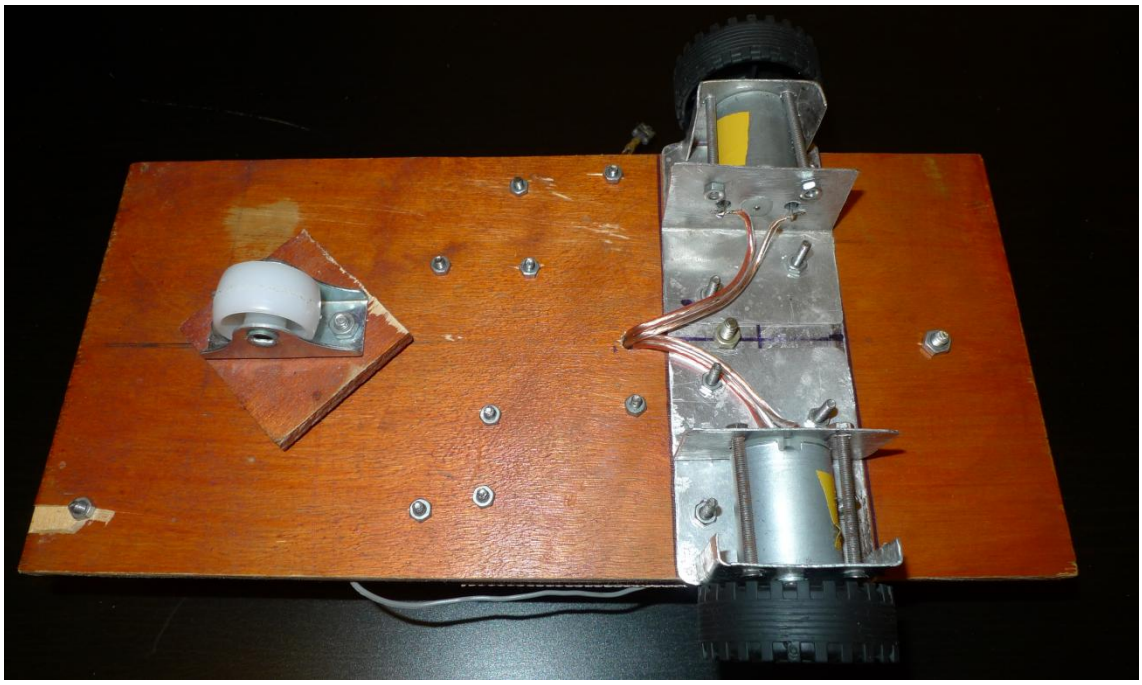


2.1.ábra: Blokkvázlat

3 Mechanikai alkatrészek

A feladat magába foglalta egész autó önállóan történő megépítését. Mivel ilyen téren nem rendelkezek tapasztalattal, ezért egy egyszerű modellt alkottam meg. Törekedtem az egyszerű, könnyű, de erős anyagok felhasználására, mint például a rétegelt lemez és az alumínium. Az így elkészített alváz célja az elsődleges funkciók tesztelése. Amennyiben az elkészített elektromos rendszer működőképesnek bizonyul, lehetőség nyílik jó minőségű, előre gyártott modellkaszni vásárlására.

3.1 Alkatrészek rögzítése



3.1. kép: Alváz és motor rögzítés

A 3.1-es képen látható az autó alvàza egy négy rétegű rétegelt lemez. Ez megfelelően vastag ahhoz, hogy megmunkálható legyen anélkül, hogy eltörne, de továbbra is elég könnyű ahhoz, hogy ne jelentsen többlet súlyt.

Az autó három kerékkel rendelkezik, elöl egy, hátul kettő elrendezésben. Az első kerék támasztó szerepet tölt be, erre a célra megfelelőnek találtam egy bútorkereket, amit csavarokkal lehet rögzíteni. A hátsó két kerék a 44 mm átmérőjű puha PVC kerék, a jobb tapadás érdekében. Ez a két kerék a motorok tengelykivezetéseire illesztőhüvely segítségével csatlakoztatható.

A motorok rögzítéséhez alumínium lapból hajlítottam tartószerkezetet (a 3.1-es képen szintén látható), amit elölről és hátulról a motorhoz kell szorítani, utána pedig már rögzíthető is az alvázhoz. A két motor közös tartószerkezettel szereltem fel, célom ezzel az volt, hogy csökkentsem annak esélyét, hogy a tengelyek nem lesznek párhuzamosak.

Az áramköröket próba NYÁK lapokon valósítottam meg, ezeket csavarozással rögzítettem, de még beraktam egy 0,5 cm vastagságú műanyag alátétet minden csavarozási pont alá, hogy a forrasztási oldal ne legyen az alvázhoz nyomva.

A rendszer akkumulátorról működik, választásom egy UPS Power márkájú, 12 V-os 1.3 Ah-sra esett. Ez egy nagyjából 10x4x5 cm-es "zselés" akkumulátor, tömege 0,61 kg. Perforált vas pánttal rögzítettem, amelyet mindkét végén az alvázhoz csavaroztam. Lévén, hogy ez az autó legnehezebb alkatrésze, a hátsó kerekek fölé, a motorok tengelyének vonalához minél közelebb úgy helyeztem el, hogy az autó súlypontja a motor tengely vonalától előre essen. Ezzel az elhelyezéssel az első kerék terhelései is csökkenthetők, ami növeli a fordulékonyságot, csökkenti az első keréken a gördülési veszteséget és a főként a hátsó kerekekre jutó terheléssel csökkenti azok megcsúszásának esélyét.

4 Elektronikai tervezés

Ebben a fejezetben a az elektronikai alkatrészek választási szempontjait fogom leírni, a 2. fejezet alapján a rendszernek a következő funkciókra, elemekre van szüksége:

- mikrovezérlő, mint központi egység,
- feszültségmérés, AD átalakítás,
- gyorsulásmérő: SPI, I2C kommunikációval,
- Bluetooth kommunikációs modul, UART interfésszel,
- motormeghajtás PWM jelekkel.

Ezek az irányelvek alapján bemutatom az egyes alkatrészeket és azok tulajdonságait. A rendszer teljes kapcsolási rajza a Függelékben megtalálható, a következő alfejezetekben a tervezési megfontolásokat és az egyes áramköri részleteket ismertetem.

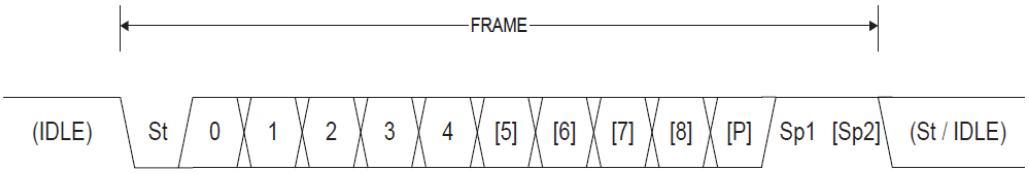
4.1 Mikrovezérlő választása

A mikrovezérlők adatbusz szélesség alapján négy csoportba sorolhatók: 8, 16, 32 és 64 bites. Mindegyiknek megvan a maga előnye és hátránya, én részletesebben a 8 és a 32 bitesekkel fogok foglalkozni.

A mikrovezérlők előtt még kommunikációs protokollokat szeretnék tárgyalni, amelyek szinte minden mikrovezérlőben megtalálhatóak és elengedhetetlenek, ha perifériákat szeretnénk illeszteni a vezérlő egység mellé.

4.1.1 Periféria interfészek

4.1.1.1 UART (Universal Asynchronous Reciever/Transmitter)

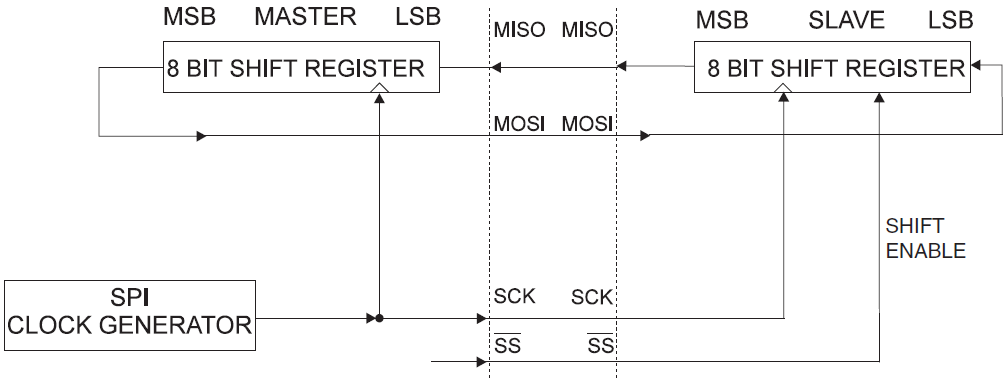


St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

4.1.ábra: UART keret [1]

Az Univerzális aszinkron adóvevő két adatvezetéken (TX - adó, RX - vevő) folytatott full-duplex, soros kommunikációs típus,a keret felépítése a 4.1-es ábrán látható. A keret a következő egységekből áll: Startbit (St), 5-9 Adatbit (n), Paritásbit (P, páros, páratlan vagy elhagyható) 1 vagy 2 Stopbit (SP). Amikor nincs adatátvitel, tétlen (IDLE) állapotba kerül a busz(logikai magas szint), ebből Startbittel kerülhet ki, ami minden esetben logikai alacsony szintű. A Startbit mellett a Stopbit(ek)nek van kötött értéke, ez szigorúan logikai 1. A kommunikáció sebessége néhány száz bps-tól egészen a Mbps-os értékig terjedő skálán változtatható, de minden esetben diszkrét értékeket vesz fel. Számítógépek általános esetben (hardveresen kiegészítés nélkül) 115200 bps-mal képesek kommunikálni. Ha egy számítógéphez valamilyen beágyazott rendszert szeretnénk illeszteni, ez az egyik leggyakrabban alkalmazott protokoll. Részletesebben ezt a témát a Kommunikációs egység fejezetben fogom taglalni.

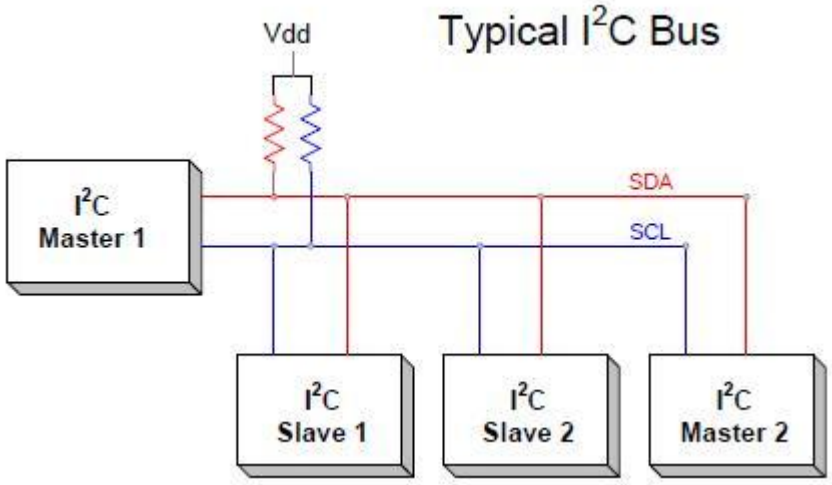
4.1.1.2 SPI (Serial Peripherial Interface)



4.2.ábra: SPI kommunikáció [1]

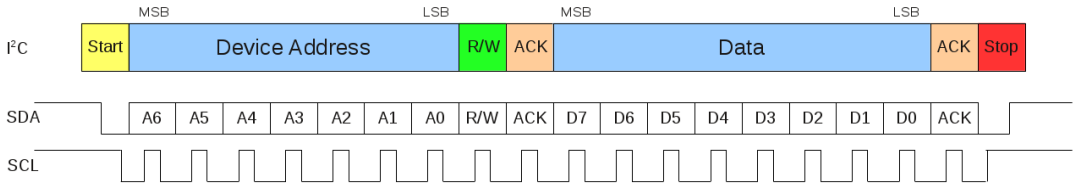
A shiftregiszter-szerűen, Master-Slave elvek alapján működő soros kommunikációs forma maximális kommunikációs sebessége MHz-es nagyságrendű is lehet. Mint ahogy a 4.2. ábrán is látható négy vezetéken valósítható meg a protokoll: MISO (Master Input Slave Output), MOSI (Master Output Slave Input), SCK (Serial Clock), nSS(negated Slave Select). A kommunikációhoz az órajelet (SCK) a Master generálja és biztosítja a Slave számára is. Az adatok átvitele csak nSS logikai 0 értéke mellett lehetséges, kezdődhet LSB-vel (Least Significant Bit) vagy MSB-vel (Most Significant Bit), a mintavételezése pedig órajel fél és lefutó élein történhet.

4.1.1.3 I2C (Inter-Integrated Circuit)



4.3 ábra: I2C busz [2]

Az 4.3-as ábrán is látható, hogy az I2C egy Multi-Master, Multi-Slave busz interfész típus, amelyen a kommunikációhoz két vezeték szükséges (SDA- Serial Data Line, SCL - Serial Clock Line). 0.1, 0.4, 1.0, 3.4 és 5.0 Mbps-os adatsebességgel kommunikálhatnak az eszközök, üzenetváltást csak Master eszköz kezdeményezhet. A Slave csak akkor "szólhat bele" a kommunikációba, ha túl gyors számára. Ekkor ideiglenes lelassíthatja a bitek érkezését azzal, hogy alacsony szinten tartja az órajelet (az adat csak magas órajelszint esetén érvényes). Lévén, hogy Multi-Master, így több csomópontból is indulhat üzenet, az üzenetek összeütközése ellen arbitrációt alkalmaznak.



4.4. ábra: I2C keret felépítése [3]

Az üzenetek címzések alapján haladnak a kezdőponttól a végpontig, felépítésük 4.4. ábrán látható. Az Startbit utáni első bájt a cél címet (7, kibővített esetben 10 bites, ekkor természetesen 2 bájtban fér el a cím) adja meg, valamint, hogy olvasni vagy írni szeretne az üzenetküldő az adott címről. Erre egy ACK (Acknowledge), nyugtázó jellel válaszol a "célpont" majd a bájtban elküldi az adatot a kezdeményező. Ha megkapta a fogadó fél az adatot egy ACK jellel nyugtázza azt, majd a kapcsolat lezárul egy Stopbittel.

A 4.1.1.2 pontban ismertetett SPI és az itt bemutatott I2C gyakori kommunikációs típusai a mikrovezérlők mellé illesztett perifériáknak.

4.1.2 Nyolc bites mikrovezérlők

A 8 bitesek a maximális teljesítmény és párhuzamosság elérése céljából Harvard architektúrát alkalmaznak, azaz a program és adat külön memóriában és külön buszon érhetők el. RISC (csökkentett) utasításkészletet használnak: utasítások lehetőleg egy órajelciklus alatt fussanak le, memóriaelérés csak load/store utasításokkal, nagy-számú általános célú regiszterrel rendelkeznek. Alacsony frekvencián működnek (4-től 40 MHz-ig, de a legáltalánosabb a 8 vagy a 16 MHz), így fogyasztásuk is alacsony. Kevés belső adat- és programmemóriával rendelkeznek (EEPROM, FLASH), de

egyszerűségük miatt nincs is szükségük sokra. Szabályzási algoritmusok ugyan implementálhatók, de a hatékonyságuk 8 bites ALU mellett, ha nem is megkérdőjelezhető, de nem garantált. Mérsékelt számú lábkivezetéssel rendelkeznek, általában egy vagy két dedikált funkcióval, azon kívül, hogy GPIO-ként (General Purpose Input/Output) lehet kezelni őket. Egyszerűségük miatt könnyen, gyorsan fejleszthetők.

Az ATmega128 egy elég ismert, széles körben alkalmazott példája a 8 bites mikrovezérlőknek. RISC architektúra szerint működik: a 133 utasításának nagy része egy órajelciklus alatt végrehajtható, 32 általános célú 8 bites regiszterrel rendelkezik, a kevés speciális célú regiszterek pedig a periféria vezérlésért felelnek. Maximálisan 16 MHz-en működik, ekkora órajel frekvencia mellett az átbocsátási képessége akár 16 MIPS is lehet. 128 kB programmemóriával (FLASH) és 4 kB adatmemóriával rendelkezik (EEPROM). Bővítésre is van lehetőség, 64 kB-ig tud kezelni külső memóriát. SPI-on (ISP - In-System Programming) vagy JTAG-en keresztül programozható, utóbbi debuggolásra is ad lehetőséget.

Programozható I/O vonalainak száma 53, ezekből 8 ADC, 8 PWM csatorna, 2 USART interfész, ezeken felül rendelkezik SPI master/slave és TWI (Two-wire Interface) interfésszel is.

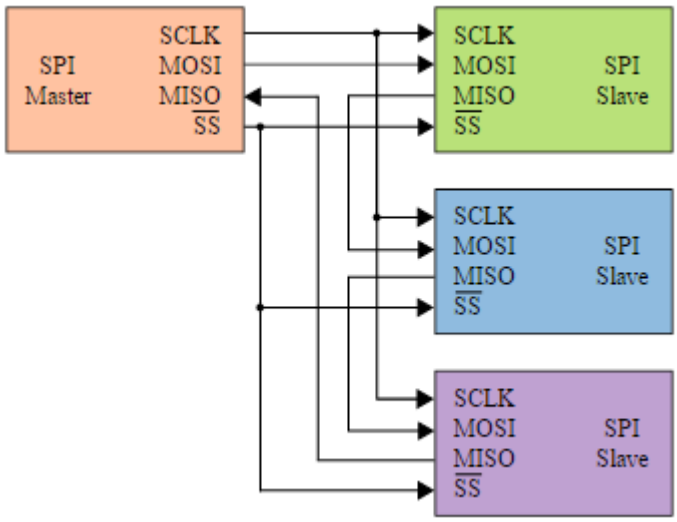
A nyolc csatornás, 10 bites AD átalakító 15 kSPS-os mintavételi sebességet képes elérni maximális felbontáson. Választható 2.56 V-os belső-, VCC értékű belső-, valamint külső referencia feszültség, de ez nem lépheti át a VCC maximális értékét (5.5 V). Ebben a feladatban, ha a mintavételi sebesség nem is annyira számít, a felbontás annál lényegesebb tulajdonság.

A mikrovezérlő két 8 és két 16 bites timer egységgel rendelkezik. A 8 bitesek fixen 8 bit felbontású PWM jelet képesek generálni, egy-egy csatornán. A két kibővített 16 bites időzítő egység 2-től 16-es bites felbontásig képes PWM jelek előállítására, három-három csatornán. Motorok meghajtásához előnyt jelent, ha egyszerre képes generálni a mikrokontroller a PWM jelet és annak negált párját, hardveres előállítással (ez teljesül az ATmega128-ra), mert ez szoftveresen késleltetéseket jelenthet, ami ha nem is ennél a felhasználásnál, de más téren problémát okozhat.

A full-duplex master- és salveként működni képes USART interfész támogatja a számítógépek által is elbírt adatsebességet (maximum 115200 bps). Az általános

UART (0-es pont) tulajdonságokon kívül képes szinkron módon is kommunikálni, ehhez még egy órajelre van szükség (XCK), ami alapján történik beérkező adat mintavételezése.

A TWI interfész kompatibilis a 4.1.1.2-es pontban bemutatott, I2C kommunikációval. A 7 bites címzést támogatja, azaz 128 különböző Slave-et képes kezelni. Maximális adatátviteli sebessége 0.4 Mbps lehet, képes master-ként- és slave-ként is funkcionálni.



4.5. ábra: Daisy-chain [4]

Egy SPI interfésszel rendelkezik, ami azt jelenti, hogy egy nSS jelet tud generálni. Amennyiben több SPI-on kommunikáló eszközt szeretnénk alkalmazni, ekkor Daisy-chain-be kell kötni ezeket, a 4.5. ábrán látható módon. A kommunikáció sebességének 7 programozható értéke van, amit az oszcillátor frekvenciájából leosztással (2, 4, 8, 16, 32, 64, 128) generál. A legnagyobb sebességre, tehát a kétszeres leosztással előállított értékre, csak Master módban van lehetőség.

Jellemzői alapján megfelelő egy autó egyszerű vezérlésére, valamint mérési adatok (gyorsulás és feszültség) továbbítására.

4.1.3 Harminckét bites mikrovezérlők

Alapvetően a 32 bites mikrovezérlőket két fő csoportba lehet osztani:

1. magas órajel frekvenciával rendelkező (több száz MHz), nagyteljesítményű rendszerek, amin beágyazott operációs rendszer fut minden esetben az erőforrások megfelelő kihasználtsága végett,
2. viszonylag magas órajel frekvencián működőek (pár száz MHz maximum), amin ugyan futhat beágyazott operációs rendszer, de anélkül is teljesen hatékonyan működhet.

Az első csoportba tartozik például a RaspberryPi[5], BeagleBone[6] és az ezekhez hasonló mikrovezérlős rendszerek, amik már szinte komplett számítógépnek számítanak. Ezek az eszközök az Internet of Things-re[7] (IoT) fókuszálnak, így nem rendelkeznek sok alacsony szintű kommunikációhoz alkalmazott interfésszel (például: I2C, SPI, UART, LIN, CAN, esetleg még az ADC is ide sorolható), helyette inkább USB-, Ethernet-, audio- és MMC/SD kártya interfészekkel vannak felszerelve.

A második csoportba tartozó mikrovezérlők elmennek a sok alacsony szintű kommunikációs interfész irányába, annak érdekében, hogy minél több periféria illeszthető legyen mellékük. Ebbe a csoportba tartoznak például az Atmel-nek a SAM3-as, SAM4-es mikrovezérlői, az STMicroelectronics STM32F szériája vagy a Texas Instruments által gyártott Stellaris-ok is. A fejlesztések gyorsításának érdekében függvénykönyvtárat, kódgenerátort is biztosíthatnak a fejlesztők az eszközök mellé. Szenzorhálózatokban való alkalmazásokra kiváló, szabályzási, számítási algoritmusok futtatására hatékonyan végezhető ezen a típuson. Egyszerű beágyazott operációs rendszerek futtatására (például FreeRTOS) van lehetőség, amelyek memóriaigénye alacsony, gyors végrehajtás mellett biztosítanak például multitaszkingot, több szálon történő végrehajtást vagy ütemezési algoritmusokat.

Manapság a legszélesebb körben[9] az ARM[10] architektúrát alkalmazzák a 32 bites mikrovezérlők magjaként. A klasszikus architektúra RISC tulajdonságokkal rendelkezik (csak néhány példát említek):

- memóriaelérés csak Load/Store utasításokkal,
- egységes 32 bites utasítások, amik többnyire egy órajel-periódus alatt végrehajthatóak,

- nagy regiszterkészlet.

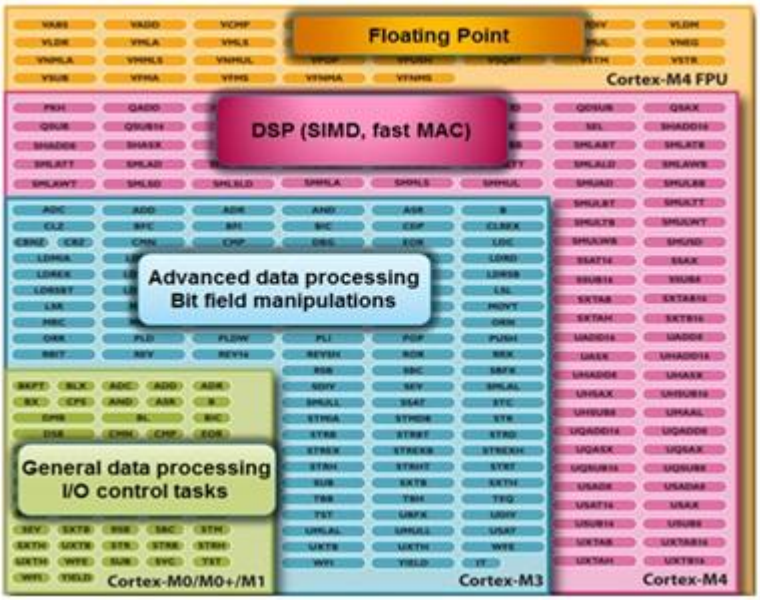
Ezeken kívül néhány figyelemre méltó egyéni tulajdonsággal is rendelkeznek (ezeket is lehetne bővebben felsorolni, de most csak néhányat emelnék ki):

1. a legtöbb utasításkód végrehajtása feltételhez köthető, ennek hatására hatékonyabb a pipeline műveletvégzés,
2. barrel shifter funkció: egy lépésben történő 1-32 bites eltolás, aritmetikai műveletek hatékonyságának növelése,
3. sok, magas szintű nyelveket támogató címzési mód.

Az újabb ARM magos processzorok rendelkeznek egy Thumb nevű, az eredeti ARM készletből tömörített utasításkészlettel is. Az utasítások visszafejtését egy dekompresszor végzi, ami a 16 bites utasításokból 32 biteseket állít elő. A "felbővítés" a hiányzó részek alapértelmezett bitekkel való feltöltésével valósulnak meg. Ez az utasításkészlet használatával lényeges javulást lehet elérni nem 32 bites adatbusz szélesség esetén, köszönhetően a nagyobb kódsűrűségnek. A továbbfejlesztett változata, a Thumb-2, az első verziót egészíti ki új utasításokkal, mint például ugrótáblákkal, feltételes futtatással. Ezzel már majdnem ugyanakkora hatékonyságot el lehet érni 32 bites buszokon, mint az ARM utasításkészlettel, amellett, hogy a kód mérete nagyjából a 75%-ára csökken.

Az ARM Ltd a mikrovezérlős világba a Cortex típusú processzorok gyártásával lépett be. Három fő családba sorolhatók:

1. Cortex-A (Application): high-end alkalmazások, pl.: okostelefonokban,
2. Cortex-R (Real-time): valós idejű alkalmazások,
3. Cortex-M (Microcontroller): beágyazott alkalmazások.



4.6.ábra: Cortex-M sorozat fejlődése [11]

Az M sorozat az általános célú beágyazott rendszereknek az egyik legelterjedtebb típusa. Fejlődésüket a 4.6.ábra mutatja, ahol jól látható, hogy mindig megőrizték a korábbi verzió tulajdonságait és újabbakat illesztettek mellé. Az M3-as verziótól kezdve ARMv7-es architektúrát használ:

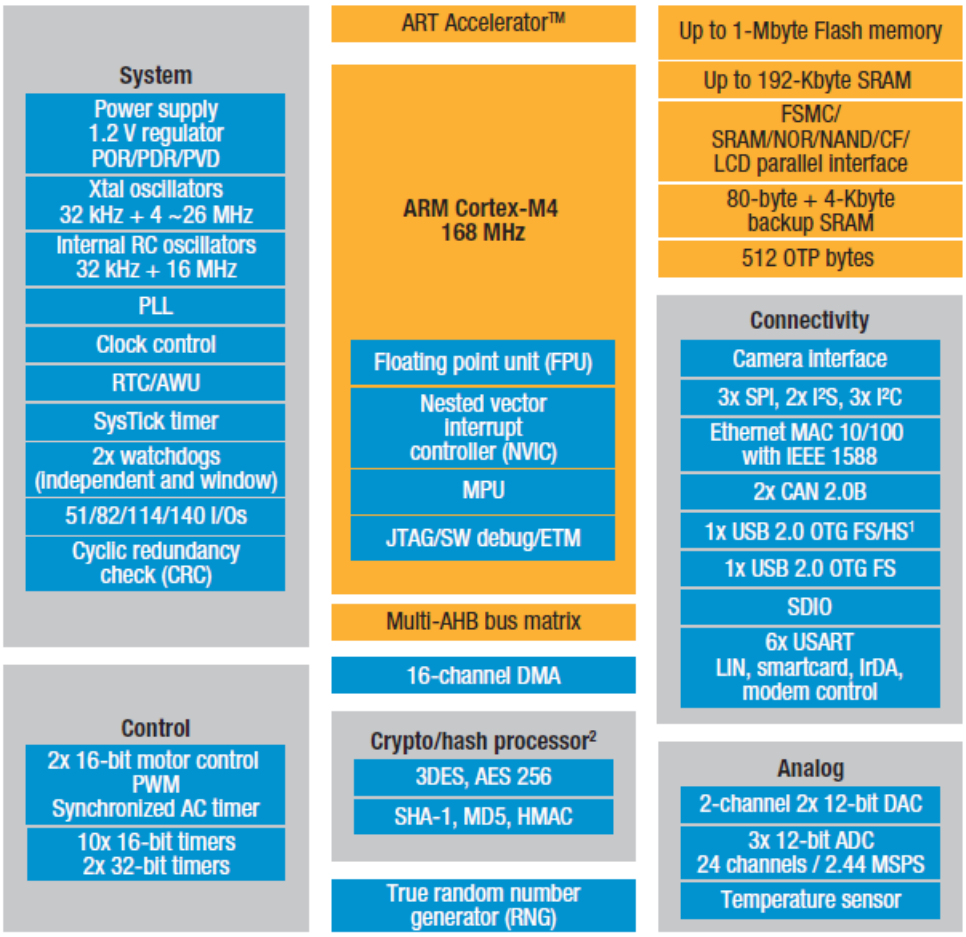
- Harvard architektúra,
- Thumb és Thumb-2 utasításkészlet,
- hardveres szorzás 1 órajel alatt,
- hardveres osztás támogatása.

A Cortex-M4-es verzió az M3-as DSP-vel (Digital Signal Processor - Digitális Jelfeldolgozó Egység) kiegészített változata. Rendelkezik DSP utasításkészlettel, valamint néhány, a DSP-kben jellemzően megtalálható hardveregységekkel, mint például MAC (Multiply and Accumulate), SIMD (Single Instruction Multiple Data), barrel shifter. Az ilyen típusú mikrovezérlőket szokták DSC-nek (Digital Signal Controller), mert nem is igazán sorolható se az általános célú mikrokontrollerek közé, se a DSP-k közé. Jelfeldolgozás, motorvezérlés vagy más szabályzási funkciók a célterületei az ilyen típusú mikrovezérlőknek. Választható JTAG és Serial Wire Debug (SWD) porttal is fel van szerelve, az ilyen maggal rendelkező mikrovezérlőkre fejlesztés gyorsításáért.

Tulajdonságai alapján az ideálisnak találtam egy modellautó vezérlésére a Cortex-M4 magos, STM32F407VGT6 mikrovezérlőt választani. A következő, 4.1.4-os fejezetben ezt a mikrovezérlőt fogom bemutatni.

4.1.4 STM32F407VGT6

Az STM32F407VGT6[12] egy Cortex-M4F magú mikrovezérlő, ami azt jelenti, hogy az M4-es FPU-val (Floating Point Unit - Lebegőpontos Egység) is ellátott változata. Tulajdonságai a 4.7 ábrán láthatók:



4.7.ábra: STM32F407 Blokk diagram [12]

A mikrovezérlő többféle tokozásban és lábkivezetés számmal érhető el, ez a típus LQFP tokozású, 100 kivezetésű verzió. A kivezetései közül nagyjából 80 használható I/O célra.

A processzormag FPU egységével gyorsabb műveletvégzést biztosít az egyszeres pontosságú aritmetikai számokon. A szintén Cortex-M4 tulajdonság, a

NVIC (Nested Vector Interrupt Controller) lehetőséget ad az interruptok szabadon konfigurálására, 16 prioritási szinten, a néhány fix prioritásútól eltekintve.

Flash memóriából 1 MB áll rendelkezésre, itt tárolható a programkód valamint a konstansok értéke (Advanced RISC Machine tulajdonság), RAM-ból pedig 192 kB: ez a részleg a változók tárolására van fenntartva. EEPROM memóriával nem rendelkezik, mert képes írni a saját Flash-ét programfutás közben. A programmemória (Flash), visszafogná a rendszert, ha közvetlenül ebből történne meg az utasítás felhozatal, mivel az olvasása például a mikrovezérlő maximális órajel frekvenciáján már nem megvalósítható. Ennek kompenzálására fejlesztették ki az ART (Adaptive real-time) memory accelerator-t. Ez betölti (bemásolja) a cache-be (nagy sebességű RAM) a programmemória egy részét, majd a vezérlő innen hozza fel az utasításokat, így a kívánt órajel frekvencia elérhető.

Sok periféria interfésszel ellátott mikrovezérlő. A két I2C interfész Master és Slave módban is képes működni, a 0.1 (standard mode) és 0.4 kbps (fast mode) adatsebesség támogatásával. Master-ként 7 és 10 bite, míg Slave-ként kettős 7 bites címzési módot támogatnak. Három SPI interfésszel rendelkezik, ebből egy maximálisan 42, kettő pedig 21-21 Mbps-mal képes kommunikálni. Ezekre lehet illeszteni SD kártyát vagy MMC-t, amennyiben memóriabővítésre van igény. A hat soros interfész (4×USART, 2×UART) közül kettő 10.5 a többi 5.25 Mbps sebességgel képes kommunikálni. LIN, IrDA és ISO 7816 kompatibilis SD kártya eszköz is illeszthető ezekre.

Tizenkét timer egységgel rendelkezik, melyek közül kettő 32 bites. A többi 16 bites időzítő közül kettő kifejezettem motor vezérlési funkciókkal rendelkezik (PWM jel és annak negált párjának generálása).

Három 12 bites AD átalakítóval rendelkezik, amelyek 16 megosztott csatornáról képesek konvertálni adatokat. A konverziók történhetnek folyamatosan (scan mode), valamint egyszeri alkalommal (single-shot mode). DMA vezérlő is kiszolgálhatja, továbbá időzítőkkel lehetséges ütemezni az átalakításokat. Maximálisan 2.4 MSPS-os mintavétel valósítható meg csatornánként, triple interleaved (háromszoros átlapolódás) módban ez az érték pedig 7.2 MSPS.

GPIO kivezetései szoftveresen konfigurálhatóak kimenetként (push-pull, open-drain, fel- és lehúzó ellenállással vagy anélkül), bemenetként (lebegő, fel- és lehúzó ellenállással vagy anélkül) vagy alternatív periféria funkcióként.

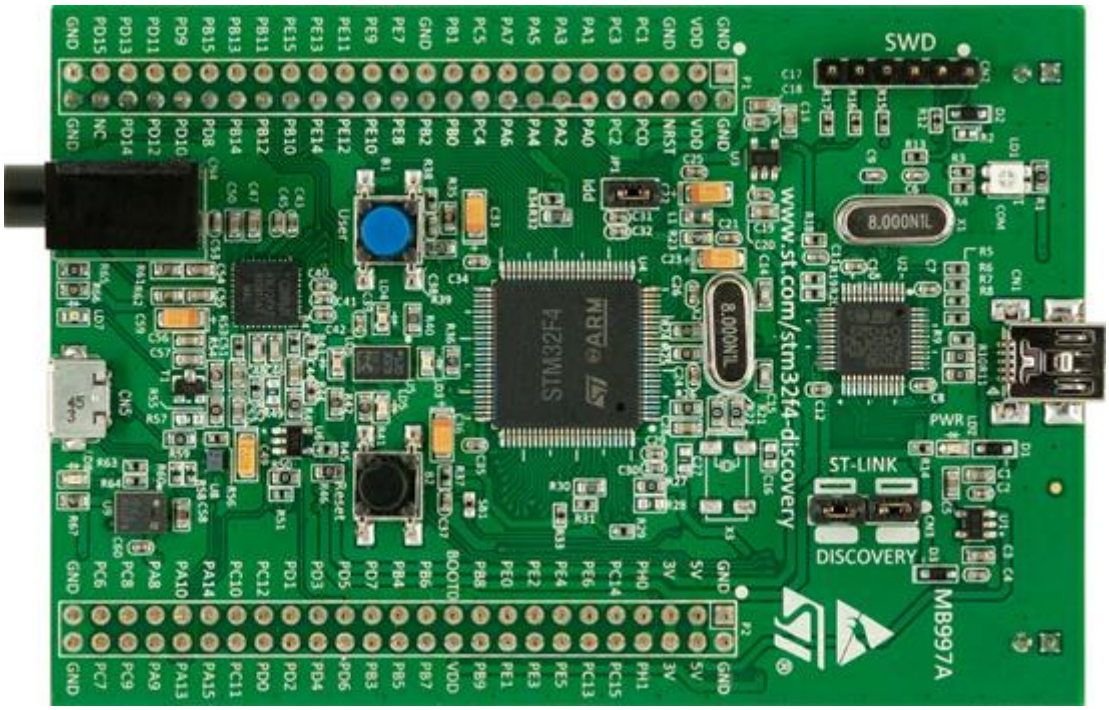
4.1.5 STM32F4Discovery

A tervezés következő lépéseként el kellett döntenem, hogyan szerezzem be a mikrovezérlőt. Egyik lehetőség, hogy csak a chipet szerzem be, másik pedig, hogy egy fejlesztői kártyán jutok hozzá.

Az első opció mellett az szól, hogy helytakarékosan, számomra felesleges perifériáktól mentes NYÁK-ot tudnék előállítani, kizárólag a célhardverrel kiegészítve. A problémát itt az jelentené, hogy a tervezés és az előállítás egy lassú és költségigényes folyamat. Továbbá rendszer tesztelésére, majd csak ezután következhetne, anélkül, hogy biztos lennék benne, hogy működőképes a megvalósítás. Ha valami probléma adódnak, például hibás tervezés, nem várt működés, akkor a javítása is problémás lenne.

Ezeket a problémákat megkerülhetem, ha egy kitesztelt, jól működő fejlesztői kártyát veszek. Ebben az esetben ugyan nagyobb helyigénnyel kell számolni, de az autó alváza elég nagy, így ebből probléma nem fog adódni. A másik hátránya, hogy nem kizárólag az általam igényelt perifériák található rajta. Bár ez nem mindig hátrány, van, hogy az éppen hasznos elemmel van kiegészítve. Az STM32F407VGT6 mikrovezérlő köré is készült egy fejlesztői kártya, az STM32F4Discorvery.

A fejlesztői kártyán, 4.8.ábra, mikrovezérlő mellé integráltak egy programozó és debugger egységet (ST-LINK), de ha ezt nem szeretnénk használni, akkor van lehetőség SWD használatára is, ennek külön csatlakozót építettek ki. A programozón kívül még rendelkezik egy audio kodekkel (CS43L22), egy mikrofonnal (MP454DT02), egy gyorsulásmérővel (LIS3DSH, részletesebben 4.4-es alfejezet lesz róla szó), valamint egy USB OTG interfész micro-AB csatlakozóval. Ezeken kívül rendelkezik egy User és egy Reset gombbal, valamint pár LED-del, amiből 4 van a mikrovezérlő lábkivezetéseire kapcsolva, a többi egyéb jelző funkciókat lát el. A kártya tápellátása történhet USB-ről (az ST-Link felőliről, nem a OTG-ről) vagy külső 3 vagy 5 V-os forrásról.



4.8.ábra: STM32F4Discovery

A mikrovezérlőtől jobbra, a 4.8. ábrán látható egy 8 MHz-es kvarc, ebből generálja a mikrovezérlő PLL-ek segítségével a kívánt órajelét (szoftveresen állítható, maximálisan 168 MHz). A két nyomógomb között található a gyorsulásmérő, a 4 programozható LED-del körbevéve(LD3, 4, 5, 6). Ettől balra helyezkedik el az audio kodek IC, ami egy fejhallgatót képes meghajtani a saját beépített erősítőjével.

A kártyán elhelyezett tüskesör segíti a lábkivezetésekhez való csatlakozást, tesztelést. A fejlesztéséhez egy szükség van egy USB-mini kábelre, Windows operációs rendszerre (XP, Vista, 7), valamint egy fejlesztői környezetre, ami lehet például Keil vagy IAR. Driverek telepítése nem szükséges, automatikusan települnek az első csatlakoztatásnál.

4.2 Tápellátás

Az autó tápellátása egy 12 V-os 1.3 Ah-s ólom-savas akkumulátor. Ha a terheléssel mért áramfelvételét nézzük a motoroknak, akkor nagyjából fél óráig képes működni a rendszer.

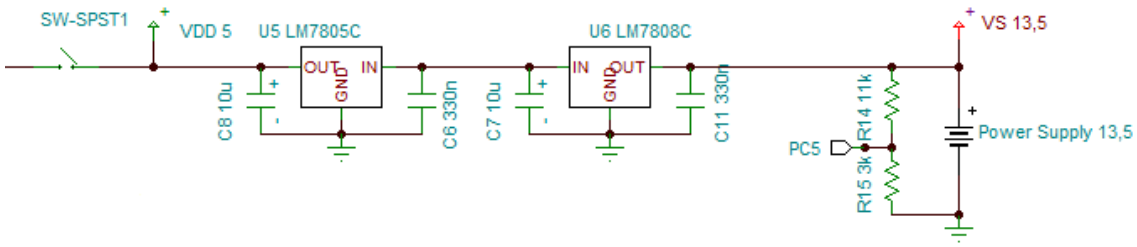
A fejlesztői kártya 3 vagy 5 V-os táplálást igényel, ezt a feszültséget érdemes stabilizáltan előállítani. A mikrovezérlő és perifériái, feltételezve, hogy maximális teljesítményen működnek, közelítőleg 100 mA-t vesznek föl. Ez azt jelenti, hogyha 5 V-

ra szeretnénk stabilizálni 12 V-ról, ami névleges érték(általában 1-1.5V-tal több), akkor 0.7-0.8 W disszipálódik a stabilizátor IC-n.

A L78-as egy feszültség stabilizátor IC széria. Több cég is gyártja, én az STMicroelectronics-félét tanulmányoztam részletesebben.

A TO-92 tokozású L7805-ön 100 mA átengedése lehetséges, de maximum 0.625 W teljesítmény disszipálását tűri el. Ez az érték alacsony, hűtőbordát se lehet szerelni rá, ezért nem jó megoldás.

A TO-220 tokozású típusok ennél már nagyobb áramtűrésűek, tipikus 300, 500, 800 mA változataik vannak. Erre már szerelhető hűtőborda, ami megengedi akár a 15 W leadását. Igazából ekkora teljesítmény leadására nincs szükség, de minél alacsonyabb hőmérsékleten van a stabilizátor, annál kevésbé lesz zajos a mikrovezérlő tápja. Ez nem elhanyagolandó tényező, mivel a tápból állítja elő a mikrokontroller az AD átalakításhoz a referenciafeszültséget. A zaj minimalizálásának érdekében a stabilizálás két lépésben történik meg: először 9 V-ra, az L7809-es IC-vel, majd onnan 5-re, az L7805-ös IC-vel. A stabilizátorokra még hűtőbordákat is szereltem, ezzel tovább csökkentve a zaj mértékét.



4.9.ábra: Feszültség stabilizátor áramkör

A 4.9.ábrán látható a stabilizátorok kapcsolása. Az ábra bal szélén található a mikrovezérlő tápkapcsolója. Azért szükséges az 5 és nem 3.3 V előállítására, mert a H-hídnak egyik lábkivezetésére a logikai 1-es jelszintet kell kötni, ami minimum 4., maximum 7 V-ot jelent. A többi bemeneti lábat lehetséges mikrovezérlő logikai jelszintjéről (3.3 V) vezérelni, mert ott a tűréshatár minimuma 2.3 V a logikai 1 jelszint eléréséhez.

4.3 Feszültségmérés

Ahogy a 4.1.4-os pontban is említettem a mikrovezérlő 3 AD átalakítóval rendelkezik. Mindhárom 12-12 bites, 15-15 csatornás, 3 V-os referencia feszültségű átalakító. A tápfeszültség mérését kell megvalósítani, ami lassan változik, de minél pontosabban szeretnénk tudni az értékét. Az adatok tárolhatók a PC-n, nincs szükség a mikrovezérlőn megőrizni azokat.

A fenti megfontolások alapján egy AD átalakító egy csatornájára van szükség. Nagyobb felbontás mellett pontosabb tudjuk meghatározni a tápfeszültség értékét, ezért a 12 bites felbontást választottam. Folyamatos konverzióra nincs szükség, mert nem fog gyorsan változni az értéke a jelnek, például ha 100 ms-onként indítunk egy konverziót és kiolvassuk az értékét, az megfelelő erre az alkalmazásra. Mivel nem kell folyamatosan sok adatot kezelni, ezért DMA támogatásra sincs szüksége az átalakítónak, nem fogja lassítani a rendszert (a konverzió ideje µs-os nagyságrendben fog mozogni).

A 4.9.ábra jobb szélén látható a feszültség méréshez szükséges áramkör, mely az R14 és R15 ellenállásokból álló feszültségosztó. A feszültségosztóban szereplő ellenállások névleges értékeit figyelembe véve 4.666 részére osztják le a tápfeszültséget. A 13.5 V-os maximális akkumulátor feszültséget 2.89 V-ra osztja le a két ellenállás, így pontosan belefér az AD átalakító működési tartományába. A mikrovezérlő a PC5-ös lábkivezetésén mérhető leosztással előállított analóg jelet mintavételezi és konvertálja digitális értékké.

4.4 Gyorsulásmérés

Végeztem közelítő méréseket az egyenes irányú mozgás során a maximális gyorsulás meghatározására. A mérést úgy végeztem, hogy az akkumulátort közvetlenül rákapcsoltam a motorokra (mindenféle átalakító elektronika nélkül), és mértem, hogy adott távolságot mekkora idő alatt tesz meg az autó. A megtett út és az eltelt idő alapján számoltam az értékeket, amik így nem lettek túl pontosak. Problémák az idő mérésével akadtak, és azt sem állt módomban ellenőrizni, hogy a gyorsulás mennyire tekinthető konstansnak a megtett út alatt, de méretezéshez már megfelelőek voltak. Több mérés elvégzése után megállapítottam, hogy a 60 mg-s maximális gyorsulásra számíthatok.

4.4.1 Lehetőségek a piacon

A legelső és legfontosabb kutatási elvem az volt, hogy minél több bites felbontást érhessek el minél kisebb mérési tartomány mellett, mivel igen kicsit értéket kell majd mérnem.

Először a Freescale által gyártott gyorsulásmérőket [14] vizsgáltam meg. A FXLS8471Q és MMA8451Q 14 bites felbontásra képes +/-2g-s méréshatárok mellett, ami azt jelenti, hogy $\frac{4000\,mg}{2^{14}\,digit} = 0.98\,mg/digit$ pontosságot lehet elérni. A 14-ből 6 bit használható így effektíven $\left(\frac{4000\,mg}{2^{14}\,digit} * 2^6\,digit = 62.5\,mg\right)$, ezt kevésnek találtam.

Az Analog Devices által gyárott ADXL családból [15] a 313-as, 362 és 363 tűnt ígéretesnek. A 313-as verzió 10 bites felbontásra képes, $1024\,LSB/g$ érzékenység mellett. Ezt visszaszámolva $\left(\frac{1000\,mg}{2^{10}\,digit} * 2^6\,digit = 62.5\,mg\right)$ 6 bitet lehetne effektíven használni. A 363-asban található egy hőmérő szenzor, ez különbözteti meg a 362-es típustól, a gyorsulásmérő ugyanaz mindkét eset. Az adatlapjuk szerint $1000\,LSB/g$-s a maximális érzékenységük, ami szinte ugyanaz az eset, mint a 313-as, ebből következik, hogy ezekkel is csak 6 bitet lehetne a méréshez effektíven használni.

Harmadjára az STMicroelectronics által gyártott gyorsulásmérőket[16] tanulmányoztam. A LIS3 sorozat újabb elemei (2011 után gyárottak) mindegyike 16 bites felbontásra képes, változó mérési tartományok mellett. Részletesebben a LIS3DSH-t fogom tárgyalni, a következő pontban (4.4.2).

4.4.2 LIS3DSH

A szenzor legkisebb mérési tartománya +/-2 g, érzékenysége 0.06 mg/digit. Ez azt jelenti, hogy 9 vagy 10 bitet $\left(\frac{4000\,mg}{2^{16}\,digit} * 2^{10}\,digit = 62.5\,mg\right)$ lehet effektíven használni a 16-ból, attól függően, hogy az autó gyorsulása eléri-e a 62.5 mg-t. SPI és I2C kommunikációs interfészekkel rendelkezik, az adat kimeneti formája lehet 8 vagy 16 bit (ez nem változtat a felbontáson, legfeljebb kétszer szükséges olvasni róla). A regiszteres szervezés egyszerű programozásra ad lehetőséget, négy vezérlő regiszterében a következőket lehet beállítani:

- megadható, hogy mekkora frekvenciával állítsa elő a gyorsulási adatokat,
- képes interrupt-ot generálni, ha befejeződött a konverzió,

- kiválasztható, hogy az interrupt vonal logikai alacsony vagy magas szintre húzásával jelezze a kész adatot,
- állítható az Anti-aliasing szűrő sávszélessége(50, 200, 400, 800 Hz),
- állíthatóak a mérési határok(+/-2, 4, 6, 8g),
- valamint, hogy I2C-s vagy SPI-os kommunikációt szeretnénk alkalmazni.

A gyorsulásmérő egyik beszerzési lehetősége egy 20$-os adapter kártya, valamint megtalálható az STM32F4Discovery-n. Mivel ezt a fejlesztői kártyát választottam a vezérlőegység megvalósításához, ezért nem jelent plusz költséget a beszerzése. A szenzor bekötése a mikrovezérlőre lehetőséget ad I2C-n vagy SPI-on történő kommunikációra, valamint a kész adat interrupt-os jelzésére. Ezzel maximum 6 lábkivezetést vesz el a mikrovezérlőről, de mivel működik megszakítások nélkül is, ezért 4 lábkivezetésről is működtethető. SPI-os kommunikációt választottam, mert feltételeztem, hogy a kommunikáció során nem lépnek fel nem kívánatos jelenségek, ezért nem szükséges nyugtázó (Acknowledge-os) kommunikációt alkalmaznom. Ezt azért tehettem meg, mert egy kiforrt, jól tesztelt, jól működő fejlesztői kártyát alkalmazok.

A méréshatárokat a legkisebbre vettem (+/-2 g), mert így már egy jó felbontást érhetek el. A 12.5 Hz-es adatgenerálási időt állítottam be, interrupt generálása nélkül. Nem kritikus tulajdonság a rendszer szempontjából, hogyha a gyorsulásérték hirtelen változik. Amennyiben például 100 ms-onként olvasom ki a szenzor értékeit, akkor ez az ideális frekvencia, mivel az ennél alacsonyabb érték (6.25 Hz) már azt eredményezheti, hogy kétszer olvasom ki ugyanazt az adatot.

A gyorsulásmérő minden egyes rendszerindításkor inicializálandó. Ez a fejlesztői kártya feszültség alá helyezésekor rögtön nem tehető meg, ezért érdemes manuálisan vagy egy automatikus késletetés segítségével megtenni ezt.

4.5 Motorok, motormeghajtás

RP280-CN-12280 12 V-os DC motorokon alapszik a rendszer. Ezek a kommutátoros egyenáramú motorok adatlapjuk alapján a legnagyobb teljesítményt 530 mA-en tudják leadni, 40-45%-os hatékonysággal. Ez természetesen nem tudja felvenni a

versenyt a kifejezetten erre a célra kifejlesztett nagy hatásfokú motorokkal, de kevesebb, mint tizedannyiért beszerezhető.

Próbaméréseket végeztem, hogy tisztában legyek azzal, hogy mekkora áramot képesek felvenni a motorok, mert erre szükségem volt a további alkatrészek méretezéséhez. Az adatlapja szerint maximum 1 A ez az érték, de a biztonság kedvéért megmértem három fontosabb értékét is:

- terhelés nélkül 50 mA-t,
- terheléssel induláskor, rövid ideig 2 A-t,
- terheléssel, a tranziens jelenségek lecsengése után nagyjából 1 A-t vett fel.

A fenti mérések után meg tudtam tervezni a motorok meghajtását. DC motorokat H-híddal lehet egyszerűen, logikai jelekkel irányítani. A H-híd egy kapcsolókból (tranzisztorokból) és logikai kapukból álló áramkör induktív terhelések, mint például relék, szolenoidok, DC- és léptetőmotorok meghajtására. Egy híd egy vagy több, de párhuzamosan kötött motort képes vezérelni. Rendelkezni szoktak áramkörvédelmi funkcióval: semmilyen logikai kapcsolással nem lehet rövidre zárni a tápot.

A számomra elérhető forrásokban több gyártó H-hídja megtalálható: National Semiconductor Technologies-tól az LDM18245, Infineon Technologies-tól TLE5205-2, STMicroelectronics-től pedig az L6203, az L293 széria és az L298. Ezeket fogom most röviden bemutatni.

A Multiwatt15-ös tokozású LDM18245 folyamatosan 3 -rel, maximum 6 A-es csúcsértékekkel terhelhető, TTL és CMOS kompatibilis bemenetek mellett. Rendelkezik áramkör- és túlmelegedés-védelemmel, a tokozása előnyt jelent hűtőborda felszereléséhez. Az autó esetében fölösleges túlméretezés lenne, nem fog ekkora áramterhelés fellépni.

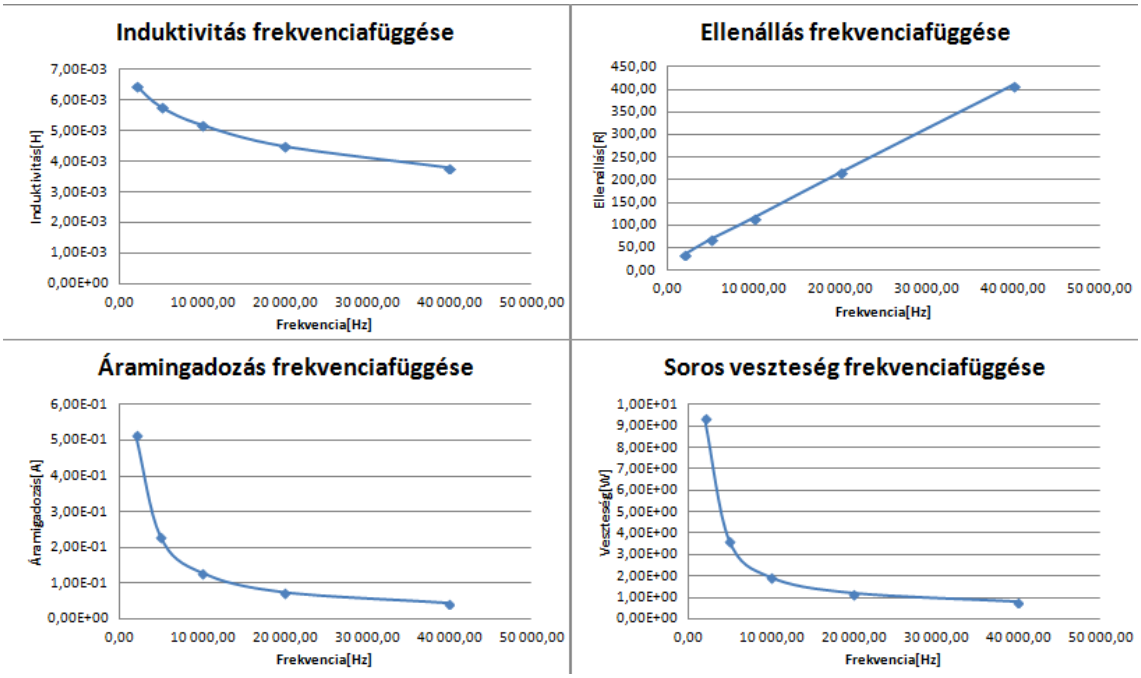
A szintén TO-220-as a TLE5205-2 és az LDM18245 között tudásban annyi a különbség, hogy folyamatosan 5 A, míg csúcsértékben 6 A képes átengedni meghibásodás nélkül, de ez szintén fölösleges túlméretezés.

Az L6203 egy Multiwatt11 tokozású IC, TTL, CMOS, mikrokontroller kompatibilis bemenetekkel. 4 A-rel terhelhető folyamatosan, 5 A-s csúcsérték mellett,

szintén rendelkezik keresztáramok és túlmelegedés elleni védelemmel. Adatlapja szerint magas hatékonyságot ígér, akár 100 kHz-es működési frekvenciával. A motorok szempontjából ez az IC túlméretezést jelentene.

Az L293-as széria 600 mA-t képes folyamatosan átengedni, ami viszont kevés, valamint DIP tokozású, így hűtőborda nem szerelhető rá. Nagy eséllyel hibásodna meg működés közben, ezért ez nem lesz alkalmas motormeghajtásra.

Az L298-as működés közben 2 A átengedésére képes, 10 ms-ig 2,5 A-es, 100 µs-ig 3 A-es tüskéket bír el. A tokozása Multiwatt15, hűtőborda felszerelésére alkalmas. Két H-híd van beleépítve, így egymástól függetlenül két motort képes meghajtani. Választásom erre az IC-re esett, a fenti tulajdonságok alapján ez a legalkalmasabb erre a célra.



4.10 ábra: Soros helyettesítő kép frekvenciafüggő értékei

A H-hidakat PWM (Pulse Width Modulation - Pulzus szélesség moduláció) jellel szokás meghajtani. A PWM jel kitöltési tényezője határozza meg, hogy egy perióduson belül mekkora százalékban kell a tápfeszültséget a motorra rákapcsolni, a frekvenciája pedig ki-be kapcsolgatás sebességét, tehát a jel periódus idejét határozza meg.

A motor tekercsekből (induktivitásokból) áll, ezért az impedanciája induktív jellegű. Az induktív jellegű impedanciát soros vagy párhuzamos R-L helyettesítő képpel

lehet ekvivalenssé tenni, ahol az elemek ideálisnak tekinthetőek. A méréshez a soros helyettesítő képet választottam, aminek a frekvenciafüggő értékei a 4.10-es ábra első sorában láthatóak. A méréseket 2, 5, 10, 20, 40 kHz-en végeztem egy WK6425 típusú precíziós impedancia analizátorral. Az ábrákból kiindulva az állapítható meg, hogy minél nagyobb frekvenciával hajtjuk meg a motort, annál kisebb lesz az induktivitása, de annál nagyobb lesz a veszteségi ellenállása.

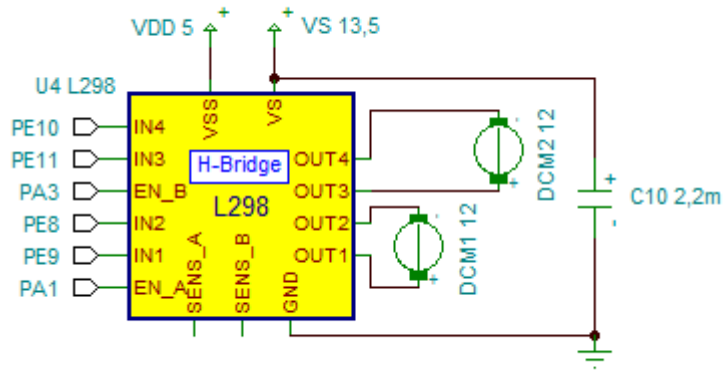
$$P = R * \Delta i^2 \tag{1}$$

$$\Delta i = \frac{U}{L} * \Delta t = \frac{U}{L} * \frac{T}{2} = \frac{U}{L} * \frac{1}{2 * f} \tag{2}$$

Az (1)-es képlet mutatja az összefüggést a soros veszteség és az ellenállás és az áramingadozás között. Közelítő értékének számítását a (2)-es egyenlet alapján tettem meg. Habár az R-L kör időállandója a mért értékek alapján egy nagyságrendbe esik a kapcsolási időkkel pár 10 kHz-en, a $\Delta i = \frac{U}{L} * \Delta t$ közelítés kvalitatív következtetések levonásához még nagyságrendi közelítésre megfelel. Az ellenállás értéke lineárisan növekszik a frekvencia növekedésével, az áramingadozás pedig lineárisan csökken, de mivel a képletben az áramingadozás négyzetesen szerepel, ezért a soros veszteség is csökken. A magas PWM frekvencia mellett szól szintén, hogy nem halljuk, ezért nem zavaró az autó jelenléte.

Hátrányként jelentkezik, hogy a magas PWM frekvenciával együtt növekszik a híd belső tranzisztorain fellépő kapcsolási veszteség, továbbá csökkeni fog a kerekeken fellépő forgatónyomaték. Ezek a szempontokat figyelembe véve kell megválasztani a frekvenciát.

A méréseket azért csak 40 kHz-ig végeztem, mivel a kiválasztott híd nem képes nagyobb frekvencián működni. Ez az abszolút maximum értéke, ami mellett még rendeltetésszerűen működik, de nem biztos, hogy a legnagyobb hatékonysággal. Közelítő értékekkel számolva, az adatlapja alapján a kapcsolási idők összege a 40 kHz-es frekvencián worst case esetben megközelítik a PWM jel periódusának felét, ezért mindenképpen érdemes kisebbre venni a ki-be kapcsolgatás frekvenciáját. Ezek alapján 20 és 30 kHz közötti érték ideálisnak tűnt, én 28 kHz-es értéket választottam, mivel ez egyszerűen és pontosan programozható volt a mikrovezérlő jelgenerálása kiválasztott időzítő egység órajeléből (84 MHz).



4.11.ábra: H-híd és a motorok kapcsolása

A 4.11-es ábrán látható a H-híd és mikrovezérlő, valamint a motorok kapcsolása. Az IN1, IN2, IN3 és IN4 bementét hajtja meg a hídnak mikrovezérlő a PWM jelpárokkal, ezekkel lehet a motorok forgási irányát megadni. Az EN_A és EN_B bemenet a motorok engedélyezéséért felelnek. A C10-es kondenzátor (a táp és a föld közé van kötve) stabilizáló funkciót lát el a rendszerben

5 Kommunikációs egység és grafikus felhasználói felület

A fejezet első részében sorra veszek néhány a beágyazott rendszerek mellé illeszthető vezeték nélküli kommunikációs típust, ami alkalmas lehet egy autó vezérlésére, valamint mérési adatok továbbítására.

A második része pedig a grafikus felület megvalósítási lehetőségekről fog szólni.

5.1 Vezeték nélküli kommunikációs típusok

Az egyik legegyszerűbb vezeték nélküli kommunikációs típus az IrDA (Infrared Data Association[17]), aminek neve egyben utal is a kommunikáció típusára, nem csak az ezt fejlesztő vállalatok csoportjára. Megbízható, optikai kommunikáció, ahol az adónak és a vevőnek látnia kell egymást. Fél-duplex, tehát csak egyirányú üzenetváltás valósítható meg vele, de üzenetváltás előtt még szinkronizálni kell a feleket. A maximális távolság a két fél között 1 m lehet. A tulajdonságai alapján nem megfelelő az autó vezérlésére.

A Zigbee egy alacsony fogyasztású és költségű csillag, fa vagy háló topológiájú hálózaton alapuló, magas szintű kommunikációs protokoll. Három fajta csomópontot különböztetnek meg a szabványban:

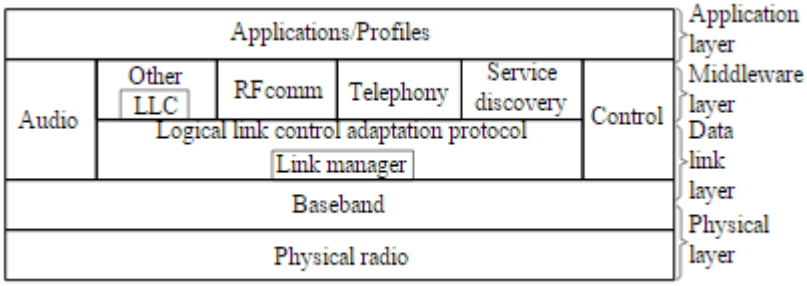
- koordinátor, minden rendszerben csak egy szerepelhet belőle, inicializálja és irányítja a hálózatot,
- router, útválasztó és adattovábbító szerepe van,
- végpont vagy csökkentett működésű eszköz, például szenzor, csak egy kapcsolata lehet, valamint energiatakarékosság céljából alvó módba kapcsolhatnak.

Egy több elemből álló hálózatban akár 100 m-es távolságról is tud kommunikálni két eszköz, de természetesen szükségesek a köztes egységek, mert egy egység nagyjából 10 m-es hatáskörben képes az információcserére. Lehetséges titkosítani a 128 bites szimmetrikus kulcsú rejtjelezést alkalmazva.

A kommunikáció akár 16 csatornán is folyhat a 2.4 GHz-es sávban, maximálisan 250 kbps adatkapcsolati sebességgel. Csomag alapú kommunikációt folytatnak egymással az egyes csomópontok, ezért a tényleges adatsebességet lecsökkentik a késleltetések és a csomag overhead-ek.

Ez a típusú kommunikáció alapvetően jó volna az autóhoz vezérlési célokra, de összetettsége és sokrétűsége megnehezíteni a használatát, arról nem is beszélve, hogy nem lehetne kihasználni és nem is biztos, hogy kellene használni az összes tulajdonságát (például titkosítás), így nem ezt választottam.

A Bluetooth[19] egy rövid hatótávolságú, adatcseréhez használt, nyílt, vezeték nélküli szabvány. Alapvetően az RS232 vezeték nélküli alternatívájának szánták. Hatótávolság avagy teljesítmény alapján három osztályba oszthatóak, 1 m (1 mW), 10 m (2.5mW), 100 m (100 mW). Verziószám szerint a 4.2-est adták ki 2014.december 2-án, de a legelterjedtebb, legtöbb eszközben megtalálható az a 2.0-s és a 2.1-es verzió, amelyek 3.0 Mbps-os adatátviteli sebességre képesek. A Zigbee-hez hasonlóan a 2.4 GHz-es frekvenciasávban működik. Master-Slave kommunikációs típus, egyszerre 7 Slave eszköz csatlakozhat 1 Master-re.



5.1.ábra: Bluetooth protokoll stack [19]

Az 5.1-es ábra mutatja a kommunikáció rétegeit. A kommunikáció alkalmazásánál nem kötelező az applikációs szintet alkalmazni, lehet alacsonyabb szinten is üzenetváltást végrehajtani. A rétegek közül a Middleware-ben (középréteg) található RFComm-ot (Radio Frequency Communications - rádió frekvenciás kommunikáció) emelném ki. Ez a protokoll virtuális soros adatfolyamot generál, ezért vezetékek helyettesítésére használható soros kommunikációban. Soros porton kommunikáló eszközök egyszerűen portolhatóak RFComm-ra. A TCP-hez hasonlóan egyszerű és megbízható adatfolyamot biztosít a felhasználó felé. Bluetooth eszközök

között igen elterjedt a széleskörű támogatása miatt. Ezt a protokollt alkalmazó eszközök tipikusan AT parancsokkal, egyszerűen programozhatók.

Az SPP (Serial Port Profile) egy RFComm protokollt használó profil, ami soros kábelt emulál. Amikor egy SPP-t használó eszközt csatlakoztatunk egy számítógéphez, ami rendelkezik Bluetooth rádióval, az eszközt egy virtuális soros portként ismeri fel az operációs rendszer, kezelése pedig megegyezik a soros porttal.

A fenti megfontolások alapján a Bluetooth kommunikációt választottam az autó vezérlésére, azon is belül törekedtem olyan eszközözök feltérképezésére, használatára, amik SPP-t használnak az egyszerű kezelhetőség céljából.

5.1.1 Bluetooth modulok

Elsőnek egy 32 bites DSP maggal ellátott Liard által gyártott BTM511-es modult tanulmányoztam. 2.1-es Bluetooth verziót támogat, rendelkezik például SPP, HFP, AVRCP profilokkal, AT parancsokkal programozható. Integrált antennájával akár 30 m-es hatótávolságra is képes, maximálisan 3 Mbps-os adatsebességgel. UART interfészen keresztül lehetséges kommunikálni a modullal, 300 kbps-os adatsebesség fölött is akár. Beépített audio interfésszel rendelkezik (16 bites sztereó kodek, beépített erősítők hangszórók meghajtására, sztereó mikrofon bemenet), ezért inkább audio alkalmazásokban használják. Az autó szempontjából fölösleges tulajdonságokkal is rendelkezik, valamint ára se túl kedvező (~20$), ezért nem ezt a modult választottam.

Következő modul, amit megvizsgáltam, a Texas Instruments által gyártott LMX9838 chip volt. A szintén Texas Instruments által gyártott CompactRISC 16 bites processzorra épül a rendszer, teljes Bluetooth 2.0 támogatással. UART-on keresztül lehetséges kommunikálni a modullal, ami az RFComm protokollal akár 704 kbps adatsebességgel képes továbbítani a beérkező adatokat. Pont-pont és csillag elrendezésű kommunikációs rendszerek megvalósítását támogatja, legfeljebb 7 Slave eszköz csatlakozásáig, de az eszköz maga is képes Slave-ként funkcionálni. Szintén rendelkezik beépített audio kodekkel, a BTM511-hez hasonlóan, ami kiegészítve a Digital Smart Radio nevű technológiával ideálissá teszi audio alkalmazásokra. Ára 40$ körül mozog, ezért és az általam nem kihasznált funkciók miatt nem ezt választottam.

A Cambridge Silicon Radio által gyártott BC417-es chip egy jól olcsóbb (~5$) megoldás, mint az előző két példa, de továbbra is megfelel a Bluetooth 2.0-s

verziójának. Kétféle modulációt támogat, amikkel 2 és 3Mbps maximális adatsebesség érhető el. Többféle interfésszel, amin keresztül programozni, kommunikálni lehet vele:

- USART, 4 Mbps-ig,
- UART, 3 Mbps-ig,
- USB 2.0,
- I2C kompatibilis interfész.

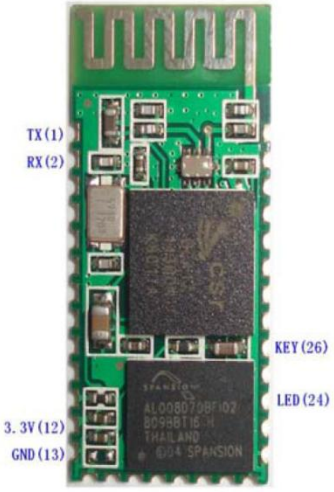


Figure 1 HC-06

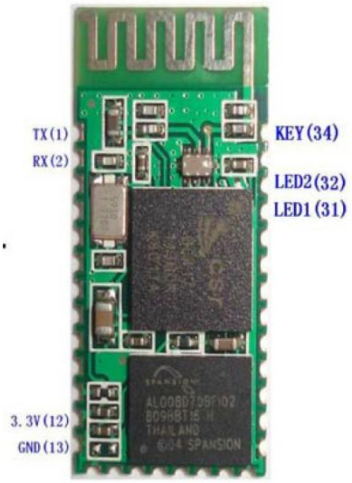


Figure 2 HC-05

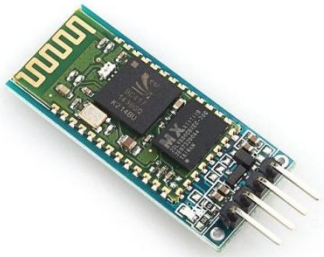
5.2.ábra: HC-05 és HC-06[20]

A modul készre szerelt állapotban beszerezhető HC Bluetooth sorozatban. A szériában ipari (HC-03, HC-04) és civil (HC-05, HC-06) felhasználásra alkalmas modulok is találhatók, ebből kifolyólag részletesen a polgári verziókat vizsgáltam meg. A két modul (HC-05, HC-06) hardveres felépítése ugyanolyan, mindössze a rajtuk található firmware-ben, amiből kifolyólag például lábkiosztásban különböznek (5.2.ábra). Nem rendelkeznek integrált antennával, de ez a NYÁK lapon megvalósítható, mint ahogy a 5.2.ábra is mutatja A széria moduljai kompatibilisek egymással. A LED1-es kivezetés a HC-05-ön és a LED kivezetés a HC-06 jelzi a kapcsolat állapotát. Ezekre a kivezetésekre egy-egy LED-et kötve nyomon követhetjük, hogy sikerült-e felépíteni a kapcsolatot (folyamatosan világít a LED) vagy még nem (nagyjából 2 Hz-es frekvenciával villog a LED). Programozásuk a Bluetooth kapcsolat kiépítése (eszközök párosítása) előtt történhet meg. A KEY lábkivezetés a HC-05 szempontjából érdekes, ennek segítségével léphetünk be az úgy nevezett AT módba,

amiben lehetséges programozni a kommunikációs modult. A HC-06 kapcsolat felépítés nélkül alapból AT módban van, ezért rögtön programozható.

A HC-06 csak Slave üzemmódban képes kommunikálni, míg a HC-05 Master módban is. A modulok csak pont-pont kommunikációra adnak lehetőséget (mintha soros portként működnének), de csillag topológiájú rendszerekben végpont szerepét is betölthetik. Csatlakozás közben az hullámzó az áramfelvétel, 30-40 mA-t is elérheti, majd csatlakozás után folyamatosan 8 mA-t vesz föl akár van kommunikáció, akár nincs. Nagyjából 3.3 V-ról (3.1-4.2 V a tűréshatár) működik, alvó üzemmódra nincs lehetőség. A modul mérete 28 mm*15 mm*2.35 mm, úgyhogy kis helyet foglal, de amúgy se jelentene problémát, mert az autó alváza elég nagy. Ezek a tulajdonságok alapján a HC-06-os típusút alkalmasnak találtam az autó vezérlésre.

5.1.2 HC-06 típusú Bluetooth modul

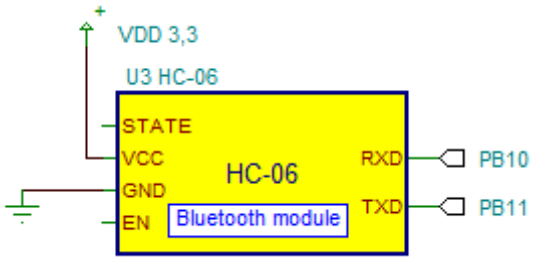


5.3.ábra: HC-06 adapterre szerelt változata

Az 5.3-as ábrán látható a HC-06 adapterre szerelt változata, ami már furatszerelt, tehát próba NYÁK-os megoldásokban egyszerűen használható. Az adapteren található egy feszültség stabilizátor, néhány kondenzátor a zajszűréshez, valamint ellenállások az UART kommunikáció jelszintjeinek biztosításához, valamint egy LED a kapcsolat állapotának jelzéséhez. A modul megtáplálása így 3.6 V-tól 6 V-ig terjedő skálán mozog.

Alap beállításként az UART kommunikáció 9600 bps sebességen 8N1 (8 adatbit, nincs paritásbit, 1 stopbit) beállítással működik. Egyedül az adatsebességen változtattam, fölemeltem 115200 bps-ra, mert ez a maximális sebesség, amit még egy számítógép kezelni tud. A laptopommal (Lenovo, Windows 7) párosítottam, virtuális soros portként jelent meg, majd az SPP támogatás beállítása után, terminál programokból tudtam kommunikálni vele. Az adatsebesség beállítását USB-UART

átalakító segítségével végeztem el. Windows operációs rendszer esetén használat előtt érdemes frissíteni a Bluetooth rádió drivereit, mivel az alap Windows Bluetooth Stack nem biztos, hogy tudja kezelni (az én esetemben nem tudta).



5.4. ábra: Modul kapcsolása

Az 5.4. ábra mutatja a modul kapcsolását, látható, hogy kiegészítő áramkörre nincs szükség, közvetlenül lehet a mikrovezérlő mellé illeszteni.

5.2 Grafikus felhasználói felület

A grafikus felhasználói felület megvalósítását az alapján választottam, hogy az adott programozási nyelvet mennyire ismerem. Fölmerült bennem, hogy Android eszközre, például okostelefonra készítek egy alkalmazást, de a nyelv ismeretének hiányában ezt elvetettem. Második opció lehetett volna a C#, de szintén nyelvismereti hiányosságok miatt más lehetőség után néztem. Harmadik lehetőségnek a MathWorks által kifejlesztett MATLAB[21] alkalmazása adódott, mivel MATLAB-ban programoztam már a tanulmányaim során, grafikus felületet is egyszerűen lehet benne létrehozni, valamint az adatok feldolgozása is könnyen végezhető, ezért ezt választottam.

5.2.1 MATLAB

A MATLAB egyszerre jelent egy magas szintű, objektum orientált programozási nyelvet és egy programot is. MATLAB nyelven íródott szkriptek, programok a MATLAB nevű programban futtathatóak. Alapvetően numerikus számításokra, mátrixműveletek végzésére találták ki. Mivel igen nagy népszerűségnek örvendett további funkciókkal egészítették ki, mint például függvények ábrázolása vagy grafikus felhasználó felületek létrehozása.

Grafikus felhasználói felületeket (GUI-kat) a GUIDE nevű grafikus fejlesztői környezetben lehetséges egyszerűen és gyorsan fejleszteni. Lehet pusztán kódolással is

létrehozni GUI-kat, de ilyet még nem csináltam, ezért választottam a GUIDE-t. A program használata nagyon egyszerű, a *guide* parancsot kell beírni a konzolban, az ezután felugró ablakban pedig lehet építeni a GUI-t az adott elemekből. A következő elemekre volt szükségem a grafikus felület megvalósításához:

- jelölőnégyzet (check box),
- nyomógomb (push button),
- grafikon (axes).

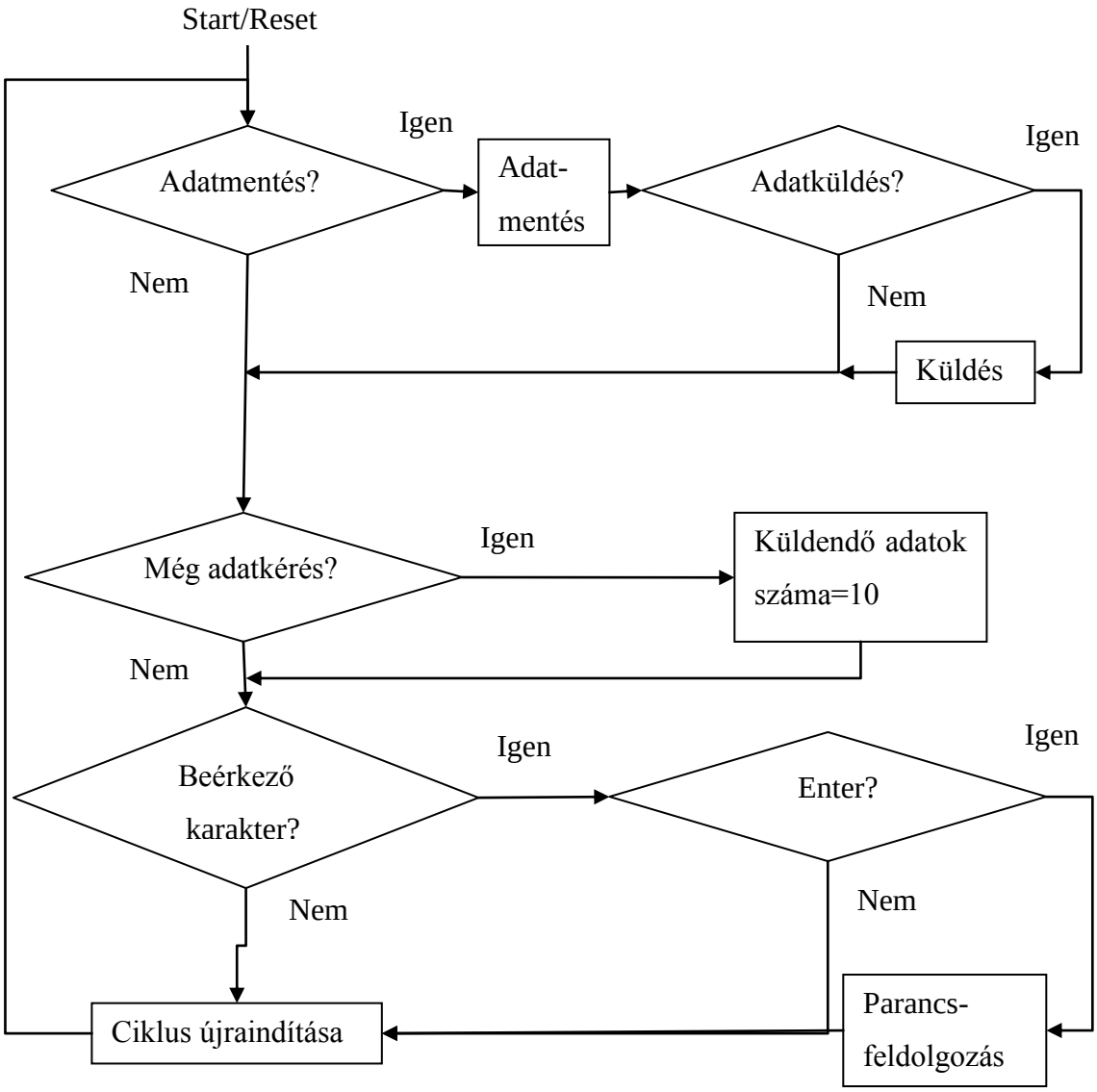
6 Megvalósított rendszer

6.1 Fő programrész

A 6.1-es ábra a fő programrész folyamatábráját mutatja. A fő programrész felel a rendszerkezelő függvények meghívásaiért, azaz a perifériák inicializáló, adat gyűjtő- és küldő valamint a beérkező parancsokat fogadó és feldolgozó függvények végrehajtásáért. Az inicializálásokon kívül a többi feladatot egy végtelen ciklusban végzi. Az adatokat egy 100 elemű cirkuláris tárolóban, egy *Data_Struct* típusú tömbben tárolja. A *Data_Struct* egy általam definiált típus, négy eleme van: a három gyorsulási irány és a tápfeszültség értéke tárolható benne egyszerre. Miután a tömb először feltöltődött, mindig a legrégebbi elemet váltja a legújabb. A tömb feltöltése 10 s-ot vesz igénybe, mert egy 100 ms-onként generált megszakítás következtében írja bele a következő értéket a mikrovezérlő.

Megszakítás beérkezése esetén csak minimális időt szabad a megszakítási szubrutinban eltölteni. A mikrovezérlőn futó program ezt teljesíti, mert csak flag-eket és változókat állít a szubrutinokban, majd utána törli a megszakítás kéréseket, nem itt végzi az adatok feldolgozását. A rendszer három megszakítással dolgozik:

1. külső interrupt (EXTI0), a fejlesztői kártyán található nyomógombbal a MOTOR_STATE_Flag értéke állítható 0 és 1 között,
2. időzítő (Timer2) által 100 ms-onként generált interrupt, DATA_Flag-et állítja 1-be,
3. az UART3-as interfészen beérkező adat generálhat még megszakítást, az UART3_IT_Flag-et állítja egybe, valamint az RX_char nevű változóba betölti a beérkező karaktert.



6.1.ábra: Fő programrész folyamatábrája

A program négy flag-et használ, ebből hármat a fő programrész szolgál ki (1., 2., 3.), kiszolgálás után mindig törlődnek a flag-ek értékei:

1. DATA_Flag: ha 1 az értéke, akkor beolvassa az adatokat a gyorsulásmérőről és az AD átalakítóról és elmenti a cirkuláris tárolóba (folyamatábra: Adatmentés?),
2. UART3_IT_Flag: ha 1 az értéke, lekezeli a főprogramrész a beérkező karaktert (RX_Char) (folyamatábra: Beérkező karakter?),
3. DATA_Request_Flag: ha 1 az értéke, akkor a DATA_To_Send változó értékét 10-re állítja (folyamatábra: Még adatkérés?, Küldendő adatok száma=10),

4. MOTOR_STATE_Flag: ha 1 az értéke lehet irányítani az autót, ha nem akkor a motorokat kikapcsolja a H-híd.

A rendszernek 3 fontos változója van:

1. DATA_To_Send: az értéke (0-10) határozza meg, mennyi adatot (cirkuláris tároló elemet, azaz 3 gyorsulás és 1 feszültségértéket) kér még a PC a mikrovezérlőtől, ha ez nem nulla, akkor elküldi az aktuálisan kiolvasott értékeket (folyamatábra: Adatküldés?),
2. RX_Char: ebbe a változóba tölti be a megszakítási szubrutin az UART-on beérkező adatot, majd ez az érték bekerül egy RX_Str nevű stringbe, ha ez megtörtént a főprogram nullázza az értékékét,
3. RX_Str: ez a string tartalmazza a beérkező karaktereket addig, amíg az utoljára beérkezett karakter egy Enter (folyamatábra: Enter?), ami jelzi, hogy a parancsnak vége, ekkor a főprogram átadja ezt a stringet a parancs feldolgozó függvénynek (folyamatábra: Parancs feldolgozása), ha ez megvolt, akkor törli az értékét.

6.2 AD átalakító

Az AD átalakítót két függvénnyel lehet kezelni:

```
void ADC_Channel_Init(void)
uint16_t ADC_Convert (void)
```

Az első felel az átalakító inicializálásáért. Ebben a függvényben kell beállítani az átalakítás fő tulajdonságait, majd engedélyezni azt:

- felbontás nagysága (12 bit),
- DMA támogatás (nem kell),
- folyamatos átalakítás (nem kell),
- külső trigger forrás (nem kell).

A második függvénnyel lehet végrehajtani a konverziót, ami a függvény visszatérési értéke is lesz egyben. Az adat jobbra zártan kapható meg, a felső biteket nullákkal tölti fel a mikrovezérlő.

6.3 Bluetooth parancsok

Karakteres parancsokat definiáltam az egyszerű kezelhetőség érdekében. A rendszer addig olvassa és fűzi egy stringbe az UART-on beérkező karaktereket, amíg egy Enter karakter-t (0x0D) nem kap. Az Enter karakter jelzi a parancs string végét. Ekkor átveszi a parancs kezelő függvény a parancs sztringjét argumentumként, majd végrehajtja a parancsnak megfelelő műveletet:

```
void BT_Command (char* command_str)
```

A parancsszavak a Függelékben találhatóak.

6.4 Külső megszakítás

A fejlesztői kártyán található USER gombbal lehet külső megszakítást végrehajtani a mikrovezérlőn. Gombnyomással engedélyezhető vagy letiltható a motorok meghajtása. Két függvény szolgál a külső megszakítás kezelésére:

```
void EXTI0_Init(void)
```

Inicializáló függvény, ebben lehet beállítani az megszakítás prioritását, valamint, hogy fel- vagy lefutó élen lépjen életbe.

```
void EXTI0_IRQHandler(void)
```

A megszakítást kezelő függvény, ez hajtja végre a megszakítás kérés törlését, valamint a motor engedélyezését vagy letiltását.

6.5 LED-ek

A fejlesztői kártyán található 4 programozható LED (PortD12-15) segítségével jelzi az autó, hogy milyen irányba halad. Használat előtt inicializálni kell:

```
void LED_Init(void)
```

Három függvénnyel lehet külön-külön kezelni egy LED-et, Set (logikai 1) vagy Reset (logikai 0) állapotba tenni, valamint váltani a két állapot között, mindegyik függvény az változtatni kívánt LED azonosítóját várja argumentumként:

```
void LED_Set(uint8_t LED_ID)
void LED_Reset(uint8_t LED_ID)
void LED_Toggle(uint8_t LED_ID)
```

Két függvény segítségével pedig az összes LED állítható Set (logikai 1) vagy Reset (logikai 0) állapotba:

```
void LED_Full_Set(void)
void LED_Full_Reset(void)
```

6.6 Motor vezérlés

Kilenc függvényt hoztam létre a motor meghajtása kezelésére, azaz a H-híd vezérlésére. Kettő a lábkivezetések és PWM jelek inicializálására való függvény:

```
void MOTOR_Init(void)
void PWM_Init(void)
```

A PWM inicializáló függvényben lehet meghatározni a PWM jel frekvenciáját, ami az autó esetében 25 kHz. A PWM jel kitöltési tényezőjének alapértéke 50%, a továbbiakban a következő függvénnyel állítható, aminek az argumentumai a motor azonosító és a kitöltési tényező kívánt értéke:

```
void PWM_Set_Duty(uint8_t side, uint16_t duty)
```

Motor engedélyezésre és letiltásra két függvény nyújt lehetőséget:

```
void MOTOR_Enable(void)
void MOTOR_Disable(void)
```

Az autó irányítására négy függvény ad lehetőséget, lehet előre-hátra, jobbra előre-hátra, balra előre-hátra mozgatni, valamint megállítani az autót. Mindezek a kitöltési tényező állításával érhetők el, amennyiben egy motoron a kitöltési tényező 50% fölött van, akkor előrefele mozog, ha ez alatt, akkor hátrafele mozog. Ha pont 50%-os a kitöltési tényező, akkor az autó megáll. A függvények, sorrendben előre-hátra, balra, jobbra és állj:

```
void MOVE_FW_BW(uint16_t duty)
void MOVE_L(uint16_t duty1, uint16_t duty0)
void MOVE_R(uint16_t duty1, uint16_t duty0)
void STOP(void)
```

6.7 Gyorsulásmérés

A gyorsulásmérő kezelésére nyolc függvényt hoztam létre. Mielőtt kommunikálni tudnánk a perifériával, inicializálni kell az SPI interfészt:

```
void SPI1_Init (void)
```

Ebben a függvényben van lehetőség a következő paraméterek állítására:

- kommunikáció iránya (Full-Duplex),
- kommunikációs szerep (Master),

- adatméret (8 bit),
- mintavételezés ideje (CPOL_HIGH),
- kommunikációs sebessége (84MHz/256),
- első bit(MSB).

Majd az írási és olvasási függvényeket is meg kell adni, amik argumentumai a cím és az adat, valamint a cím:

```
void SPI1_Send(uint8_t address, uint8_t data)
uint8_t SPI1_Read(uint8_t address)
```

Ha ezek megvannak, akkor lehetséges használni a periféria (gyorsulásmérő) kezelő függvényeket, amelyek funkció sorban: resetelés, inicializálás, X, Y és Z irányú gyorsulás kiolvasása.

```
void MEMS_Reset(void)
void MEMS_Init(void)
uint16_t SPI1_Read_X(void)
uint16_t SPI1_Read_Y(void)
uint16_t SPI1_Read_Z(void)
```

6.8 Időzítő egység

Az időzít egység (Timer2) felel a 100 ms-onként bekövetkező megszakításokért. Két függvénnyel kezelhető az időzítő:

```
void IT_Timer_Init (void)
```

Inicializáló függvény, amiben beállítható a megszakítás prioritása, valamint a sűrűsége.

```
void TIM2_IRQHandler(void)
```

Megszakítás kezelő függvény, ebben szolgálható ki (DATA_Flag 1-be állítása) és törölhető a megszakításkérés.

6.9 UART interfész

Négy függvénnyel kezelhető az UART interfész.

```
void UART_Init(void)
```

Ebben az inicializáló függvényben állítható az adatkapcsolat sebessége (115200 bps) és típusa (8N1), valamint beérkező adat esetén generált interrupt prioritása.

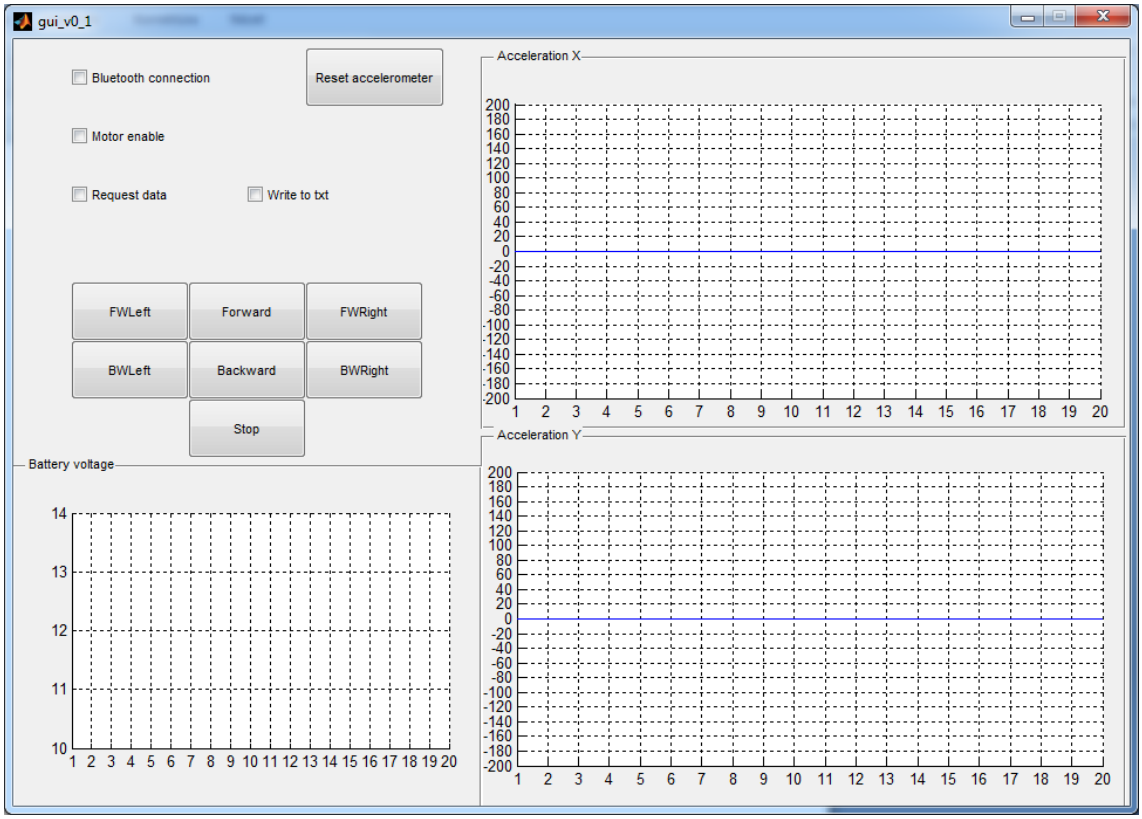
```
void USART3_IRQHandler(void)
```

Megszakítás kezelő függvény, itt kapja meg az RX_Char a bejövő karakter értékét, valamint ez a szubrutin állítja 1-be az UART3_IT_Flag-et, majd törli a megszakításkérést.

Adat küldésre két függvény áll rendelkezésre, egyik egy karakter, míg a másik egy string elküldésére alkalmas. Az argumentumaik az elküldendő értékek:

```
void UART_Send_Char(uint8_t c)
void UART_Send_Str(char* str)
```

6.10 Grafikus felhasználói felület



6.2.ábra: MATLAB-ban készített GUI

Az 6.2-es ábrán látható a megvalósított grafikus felület. A grafikonon lehetséges az autó X és Y irány gyorsulásának, valamint a tápfeszültség értékének ábrázolása. A jelölőnégyzetek a két állapotú funkciók kezelésére szolgálnak:

- "Bluetooth connection": amennyiben a bepipáljuk, a MATLAB kapcsolódik a Bluetooth eszközhöz, kipipálás esetén a kapcsolatot bontja,
- "Motor enable": bepipálásnál engedélyezni a motorokat, azaz mozoghat az autó, kipipálás esetén a mozgás tiltva van,
- "Request data": gyorsulás és feszültségadatokat kér le a mikrovezérlőtől, ha be van pipálva, ezzel párhuzamos pedig megkezdődik az adatok kijelzése a grafikonon,
- "Write to txt": egy .txt kiterjesztésű fájlba írja bele a bepipálás után érkező adatokat, a pipa kivételénél abbahagyja az írást, minden egyes új írásnál törli a korábbi tartalmát a txt fájlnak.

A nyomógombok egyszerű vezérlő funkciókat töltenek be, megnyomásuk után egyszer hajtja végre őket a mikrovezérlő. Az autó mozgásához nem kell folyamatosan irány megadni, elég egyszer. A "Reset accelerometer" feliratú nyomógomb a nevéből is adódóan a gyorsulásmérő alapállapotba állításáért felel. Ezt érdemes az első indítás előtt végrehajtani, mivel van rá esély, hogy nem megfelelő az inicializálása a fejlesztői kártya feszültség alá helyezésével.

Fontos megjegyezni, mielőtt a grafikus felület részleteibe belemennék, hogyan tudja a MATLAB kezelni az SPP-t támogató Bluetooth eszközöket a MATLAB. Ugyan a Windows virtuális soros (COM) portként ismeri fel a HC-06-os modult, a MATLAB-ban történő kezelése mégis máshogy alakul. A MATLAB rendelkezik egy Instrument Control Toolbox[22] kiegészítővel, ami támogatást nyújt például TCP/IP, UDP, I2C, SPI vagy Bluetooth (SPP) eszközök kezelésére. Így a modult nem virtuális soros portként kell kezelni, erre nem is ad lehetőséget a MATLAB, hanem Bluetooth objektumként. Innen már hasonlóan működik, mint a COM portok kezelése, annyi kivétellel, hogy nem az adatsebességet és a UART típusát kell megadni, hanem azt, hogy melyik csatornán folyik a hét lehetségesből a kommunikációs a modullal. Az adatsebesség beállítására nincs szükség, a MATLAB képes a sebesség szinkronizálásra a modullal.

Minden egyes grafikus elem egy objektum, amihez tartozik egy Callback függvény. A GUI indításakor a következő függvényt hívja meg a MATLAB:

```
function gui_v0_1_OpeningFcn(hObject, eventdata, handles, varargin)
```

Ez egy nyitó, inicializáló függvény, amiben végrehajtott utasítások a GUI megnyitásával már érvényre is jutottak. Az objektumokat inicializálni, definiálni itt érdemes. Fontosabb beállítások ebben a függvényben:

- Bluetooth objektum létrehozása,
- Timer objektum inicializálása, 0.5 s-os megszakításokkal,
- grafikonok értékeinek beállítása,
- globális változók definiálása.

Az adatfeldolgozás ütemezését egy *BytesAvailableFcn* függvénnyel valósítottam meg: a MATLAB számolja, hogy hány bájt érkezik be az adott eszköz bemeneti tárolójába, és ha ez elér egy beállított értéket, akkor meghívja az ilyenkor futtatandó függvényt. Ezt azért érdemes használni, mert nagy a MATLAB késleltetése, ha csak szimplán, valamilyen időközönként hívnánk meg az adatbeolvasó függvényt a bemeneten várakozó adatokra. A Bluetooth inicializálásakor kell végrehajtani ezt a beállítást, mert a kapcsolat felépítése után már nem lehetséges a módosítása. A futtatandó függvény:

```
function AXEsUpdate(obj, event, hfigure)
```

Ez a függvény kezeli a beérkező adatokat, átkonvertálja a hexadecimális értékeket decimális értékekké (mg-ben és V-ban számolva, mivel a mikrovezérlő csak hexadecimális számokat küld a PC felé), ha a "Write to txt" négyzet be van pipálva, akkor kiírja egy txt fájlba is. Szintén ez a függvény felel az adatok kijelzéséért a grafikonokon. Az bemeneti tárolóból az *fgetl* függvénnyel lezáró karakterig *(kocsivissza, sorelőre, \r\n)* olvasható az adat, anélkül, hogy a terminátor karakterek belekerülnénk. A mikrovezérlő felől érkező minden adat között lezáró karaktereket helyeztem el *(\r\n)*, így 4 olvasási paranccsal dolgozható fel egy adatsorozat.

Ahogy említettem már, egy időzítő egységet is létrehoztam. Ennek a feladata, hogy amennyiben a "Request data" jelölőnégyzet be van pipálva, kérjen további adatokat a mikrovezérlőtől. Minden 0.5 s után ezt a függvényt hívja meg:

```
function DR_Update(obj, event, handles)
```

A függvény ellenőrzi, hogy be van-e pipálva a négyzet, ha igen, akkor elküldi a *DREQ* parancsot a mikrovezérlő felé.

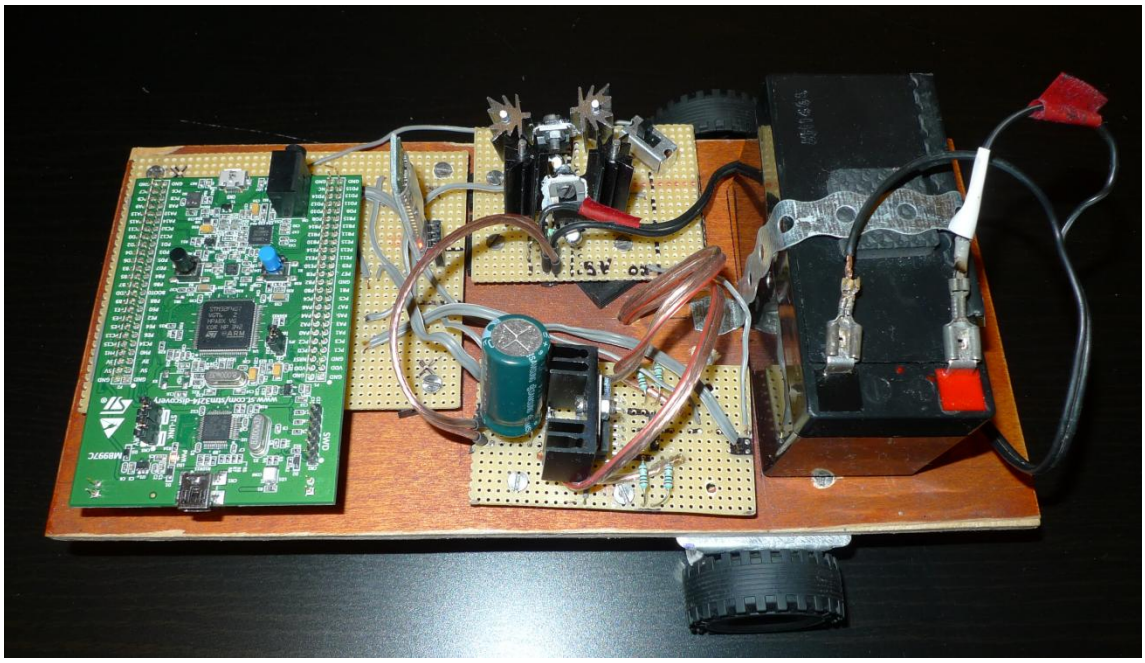
A könnyű vezérelhetőség érdekében billentyűzetről is irányíthatóvá tettem az autót:

```
function figure1_WindowKeyPressFcn(hObject, eventdata, handles)
```

Minden gombnyomásra ezt a függvényt hívja meg a MATLAB, majd, például egy *switch-case* alapján, eldönti melyik billentyű volt lenyomva és az alapján elküldi a parancsot a mikrovezérlőnek. A függvény *eventdata* argumentuma tárolja a billentyű lenyomásról szóló információkat. Az autót a numerikus billentyűzetről lehet irányítani a következő kiosztásban: 0 - stop, 1 - balra hátra, 2 - hátra, 3 - jobbra hátra, 4 - balra előre, 5 - előre, 6 - jobbra előre.

A gyorsulásmérő mozgás közben minden kis sebességváltozást is érzékel, amit például a talaj egyenetlensége okoz, nem a motorok. Ennek következtében a 100 ms-onként mért értékek nem tükrözik a valóságot az autó gyorsulásának szempontjából, hanem csak a gyorsulásmérő mozgását adja vissza hitelesen. Ezért átlagolást hajtok végre a bejövő adatokból, a MATLAB összevár 5 csomagot, ahol egy csomag egy feszültség és három gyorsulás (X, Y és Z irányú) értékeket tartalmaz. A csomagok értékeit kiátlagolja, majd ezt jelzi ki a grafikonokon, valamint megoldottam ezen adatok egy szöveges dokumentumban történő tárolását. Ezzel a módszerrel 500 ms mérési eredményeinek átlaga kerül kijelzésre és mentésre Az így kirajzolt grafikonok már valóságosan adják vissza az autó adatait.

6.11 Megépített rendszer



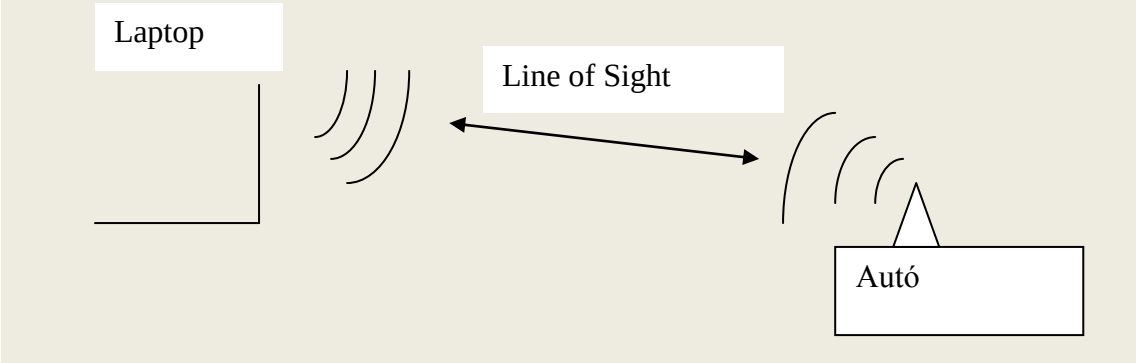
6.3. kép: Megépített rendszer

A 6.3-as képen látható a megépített rendszer. Bal oldalon található a mikrovezérlő, amelynek a két nyomógombja között helyezkedik el a gyorsulásmérő. Ettől jobbra található a függőlegesen beépített Bluetooth modul. Az akkumulátorig fönt a tápáramkör, lent pedig a H-híd látható. A tápáramkör NYÁK-ján a mechanikus főkapcsoló található, ami leválasztja a tápáramkört az akkumulátorról. Ezzel lehet rákapcsolni a mikrovezérlőre a stabilizált 5 V-ot. A H-hidat és a motorokat 0.75 mm^2-es vastagságú hangszóró vezeték köti össze, mind a motorokkal, mind a tápegységgel. A H-híd mellett látható 2.2 mF-os kondenzátor (kékeszöld) a táp és a föld közé van bekötve, ez a tápfeszültség stabilitását biztosítja, arra az esetre, amikor a motor hirtelen nagy teljesítményt vesz föl.

7 Mérési eredmények

7.1 Kommunikációs tulajdonságok mérése

7.1.1 Maximális kommunikációs távolság

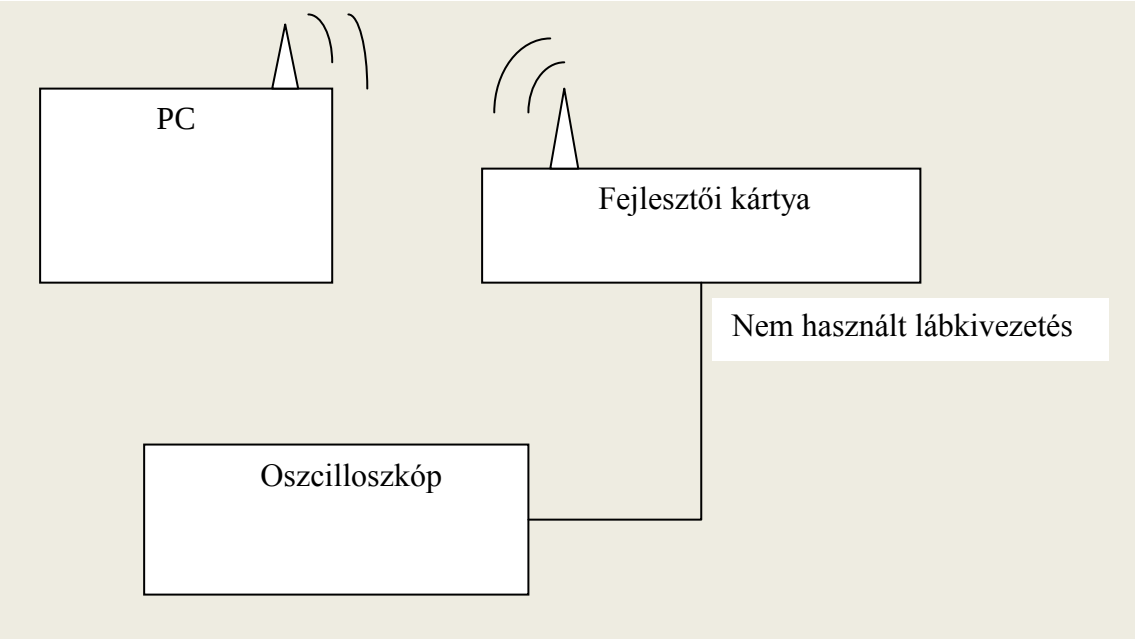


7.1.ábra: Maximális kommunikáció távolság mérési elrendezése

A 7.1.ábra mutatja mérési elrendezést, a Line of Sight címke arra utal, hogy semmilyen akadály nincs a laptop és az autó között. A mérés elvégezhető aulákban, körfolyosókon, lényeg, hogy a két adó-vevő közti távolság meghaladja a 10 m-t, ami a névleges maximális távolsága a kommunikációnak. Méréseim alapján arra az eredményre jutottam, hogy 11 m után elveszíti a kapcsolatot a PC a HC-06-os modullal, innentől kezdve se kommunikáció, se a kommunikáció újraépítése nem lehetséges.

7.1.2 Kommunikációs sebesség

Habár az adatlapokon szereplő értékek adottak, a tényleges kommunikációs illetve adatátviteli sebesség ettől eltérő lehet. Mivel a Bluetooth modullal a mikrovezérlő 115.2kbps kommunikációs sebességre képes, nem fog a modul 3 Mbps-mal kommunikálni. A méréseim célja az volt, hogy hozzávetőleges értéket kapjak néhány bájtnyi adat átvitelének idejére, mert az adatok átviteli és feldolgozási ideje (beleértve a Windows 7 és a MATLAB okozta késleltetéseket) befolyásolja a kommunikációs és feldolgozási módot.

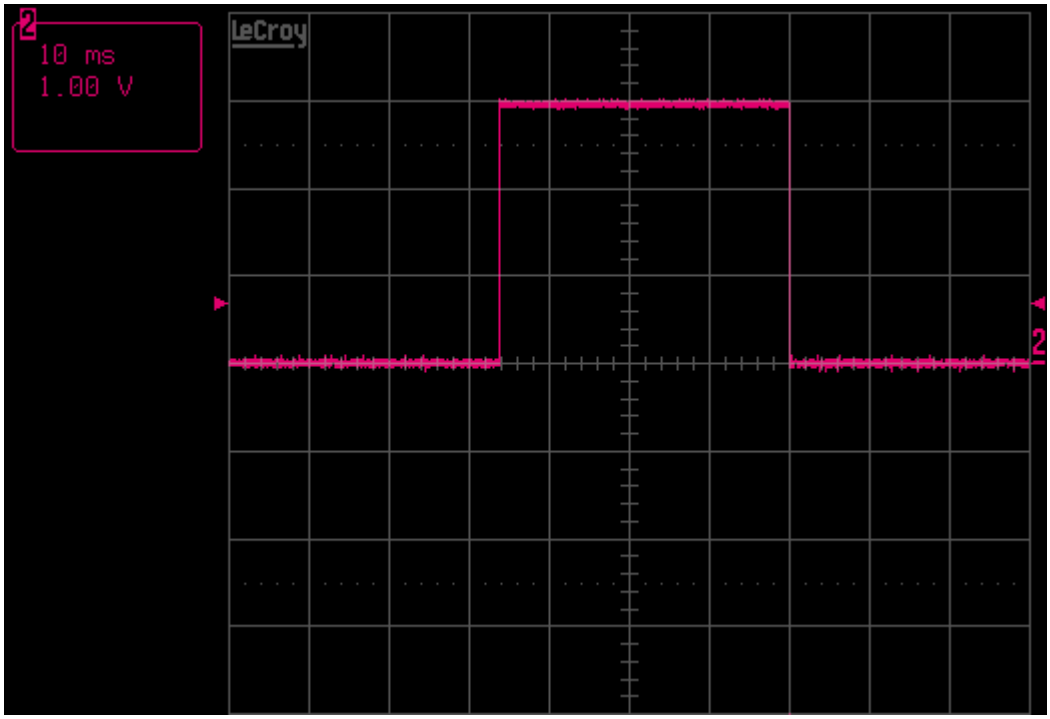


7.2. Kommunikációs sebesség mérés, mérési elrendezés

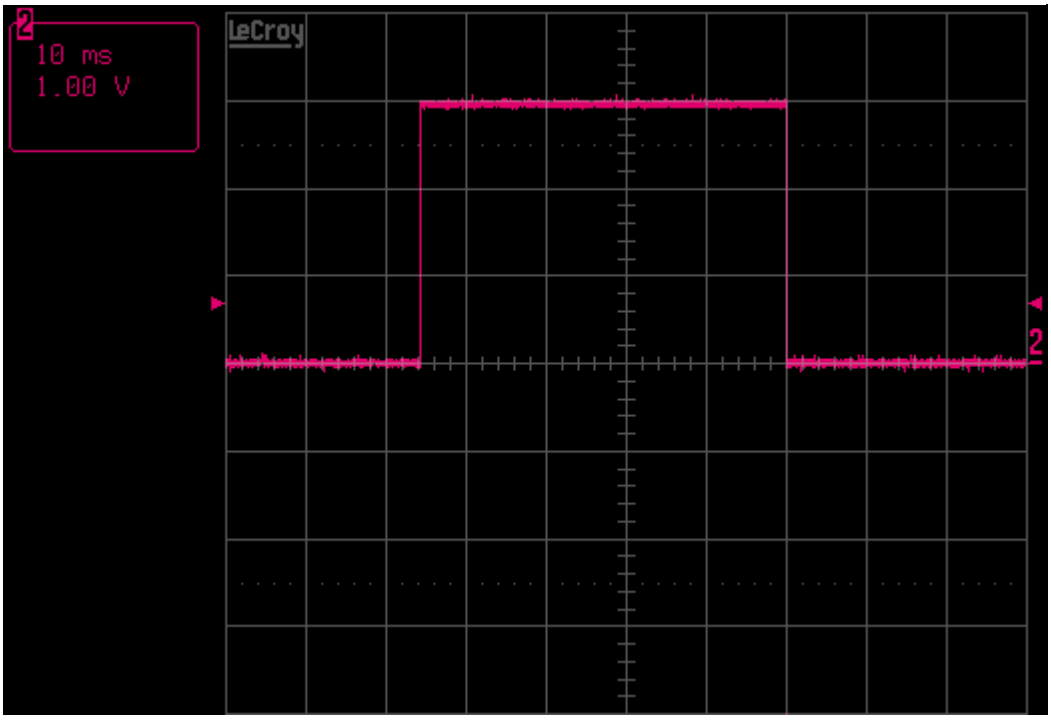
A 7.2-es ábra mutatja a mérési lerendezést. Az oszcilloszkóp a fejlesztői kártya egyik nem használt lábkivezetésére van kötve. A lábkivezetés a következőképpen van programozva: alaphelyzetben logikai 0 szinten van, majd amikor megkezdődik az adatátvitel, logikai 1-be húzza a mikrovezérlő. Amikor pedig a PC felől megérkezett az adat, újra logikai 0 szintre kerül. A 2 bájt adat küldésével és fogadásával teszteltem a rendszerem. A leggyorsabb, illetve leglassabb átvitel sebesség érték nem csak egyszeri eredményként jött ki, 20 mérés alapján a következő értékeket kaptam:

- a leggyorsabb, 7.3.ábra: 36 ms, ~890 bps,
- a leglassabb, 7.4.ábra: 46 ms, ~695 bps,
- a leggyakoribb, 7.5.ábra: 42 ms, ~760 bps.

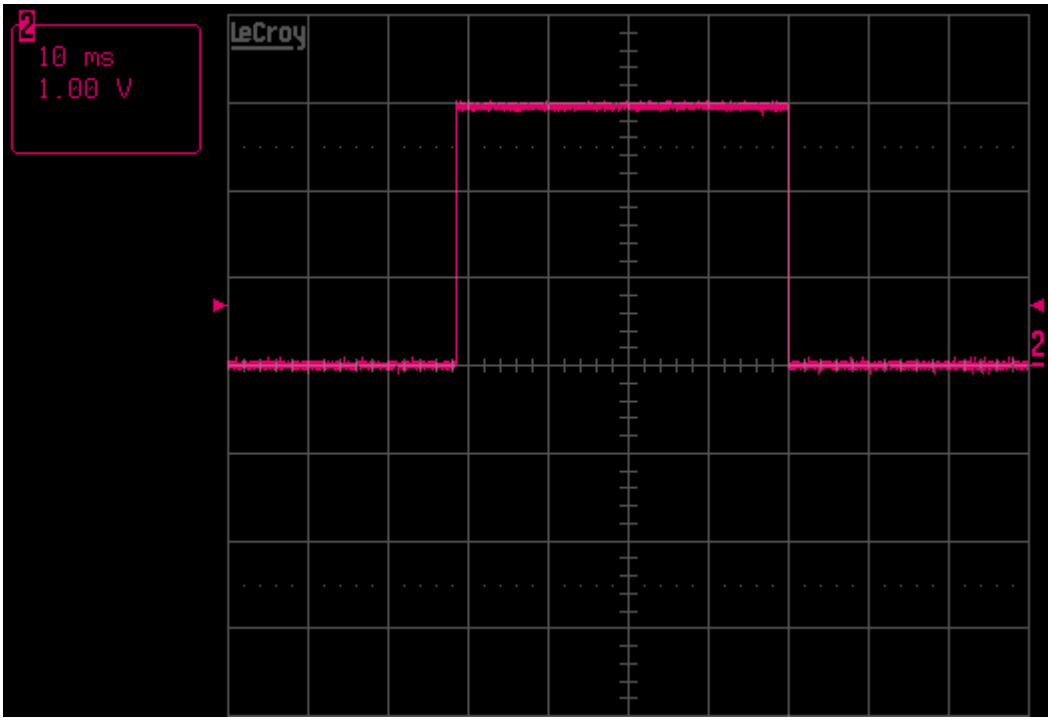
Ezek alapján megállapítható, hogy a rendszer késleltetése elég nagy, feltehetően a PC-s oldal miatt. Ez azonban nem okoz problémát, mivel a PC-nek nem kell a mikrovezérlő által küldött adatcsomagokat visszaküldenie.



7.3.ábra: Leggyorsabb átvitel, 36 ms



7.4.ábra: Leglassabb átvitel, 46 ms



7.5.ábra: Leggyakoribb átvitel, 42 ms

7.2 Mozgási tulajdonságok mérése

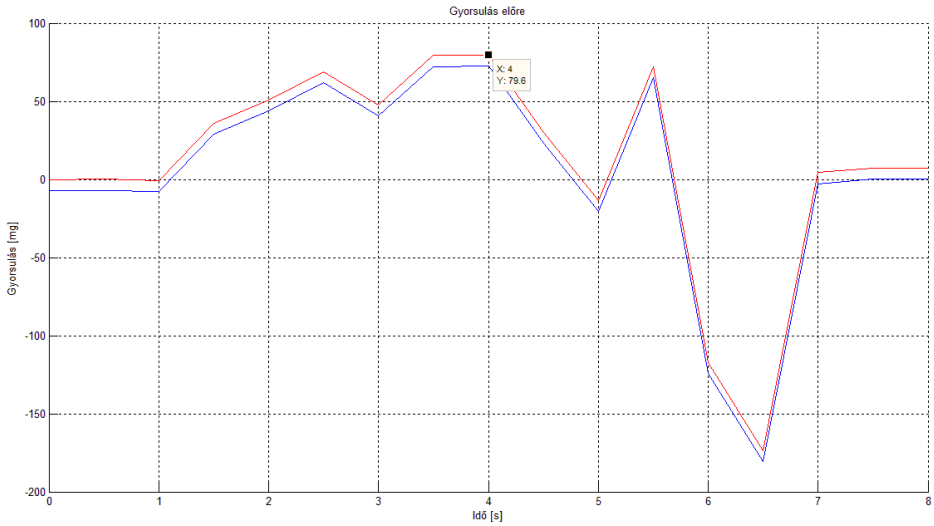
7.2.1 Mechanikai eredetű egyenesfutási hiba és javítása

A tengelyek tökéletes párhuzamosságát a deszkamodell egyszerűsége miatt megoldani nem tudtam, ezért méréseket végeztem, hogy egyenesen történő mozgás során mennyire tér el az autó az egyenestől, amennyiben a motorok vezérlése is egyforma. Szoftveresen ezt a problémát jól lehet kompenzálni: ha valamelyik irányba elkanyarodik az egyenesen előre vagy hátra tartó mozgás helyett, akkor a kanyarodás irányával ellentétes oldali motort meghajtó PWM jel kitöltési tényezőjét csökkentésével javítható a probléma. Amennyiben ezt nem változtatjuk meg, az autó egy kör alakú pályán fog mozogni. A pálya méretét, tehát a kör sugarát nagyban befolyásolják a "talaj" tapadási tulajdonságai. Az autó puha PVC kerekekkel van felszerelve, ami fényes felületű talajokon nem tapad jól, ami elősegíti a csúszást. Például parkettán és laminált padlón máshogy viselkedik, utóbbin kisebb a tapadás, ebből következően jobban csúszik, valamint a kerekek könnyen koszolódnak, ami szintén ront a tapadáson. Ezen problémákból kifolyólag nem biztos, hogy két egymás utáni mérésen ugyanazt vagy ahhoz közeli pályát kapunk, a jelenség pontos mérése nehezen oldható meg.

A párhuzamosság hiánya miatt az autó jobbra tért ki az egyenestől. A pontos mérések helyett inkább teszteléssel oldottam meg a probléma kiküszöbölését. Több teszt elvégzése után arra jutottam, hogy bal oldali motort meghajtó jel kitöltési tényezőjének 3.35%-kal való csökkentése kompenzálta a hibát.

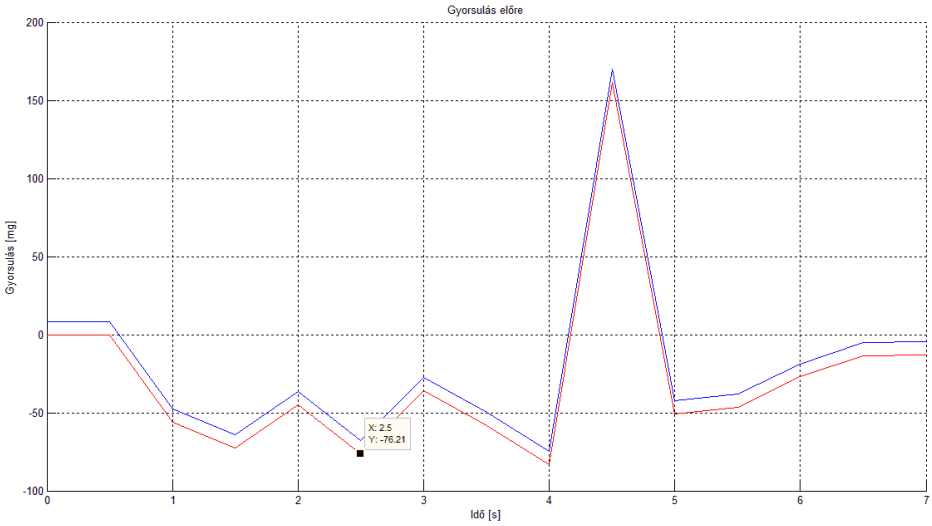
7.2.2 Gyorsulás X irányban

A rendszer koordináta tengelyei jobbkéz-szabály szerint helyezkednek el egymáshoz képest, a X irányú gyorsulás az előre-hátra, míg az Y irányú a jobbra-balra történő sebességváltozást mutatja.



7.3.ábra: Gyorsulás előre (X irány)

Az előzetes mérések alapján X irányban 60 mg nagyságrendű gyorsulás értékre lehet majd számítani ennél a mérésnél. Y irányú gyorsulását a következő alfejezetben (7.2.3) fogom mérni. Mindkét mérés esetén az adatgyűjtést a GUI-ban található "Write to txt" funkcióval végeztem, majd ezek feldolgozást, ábrázolását MATLAB segítségével végeztem el. A 7.3-as ábrán látható a gyorsulás egyenes vonalú közlekedés esetén. Kékkel az eredeti, pirossal ofszet kompenzálás után gyorsulásértékek. A megjelölt pont a maximális gyorsulásérték, ami 79.4 mg. Abban az esetben, ha egy teljes indulás-haladás-leállás ciklust vizsgálunk, a gyorsulásfüggvényének integrálja nulla. Itt egy bonyolultabb mozgás látható. A negyedik másodperc után az autó eléri a pálya végét jelentő ruganyos anyagot, visszapattan, majd ismét nekiütközve megáll, a gyorsulás negatív értékeket vesz fel, majd újra nulla lesz.



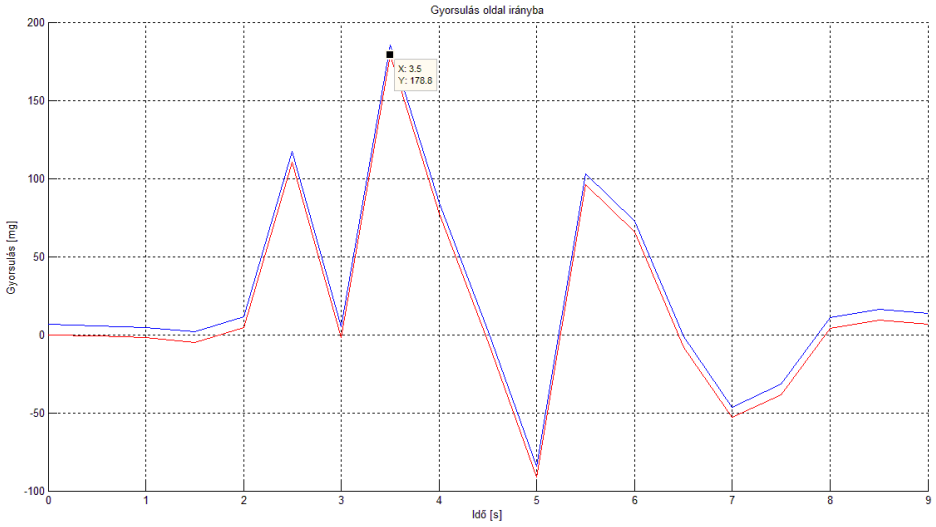
7.4.ábra: Gyorsulás hátrafele(-X irányba)

A 7.4.ábra mutatja a visszautat a hátrafele történő mozgás gyorsulás-idő diagramját. Megint látható, hogy 4 s után eléri a pálya végét, a másik irányba jelentős mértékben lassul (felugrott a szőnyegre), egy picit még megy majd a hátrameneti jel elvétele után megáll. A maximális érték -76.21 mg ami abszolút értékben közel azonos az előrefelé történő mozgás során mért értékkel.

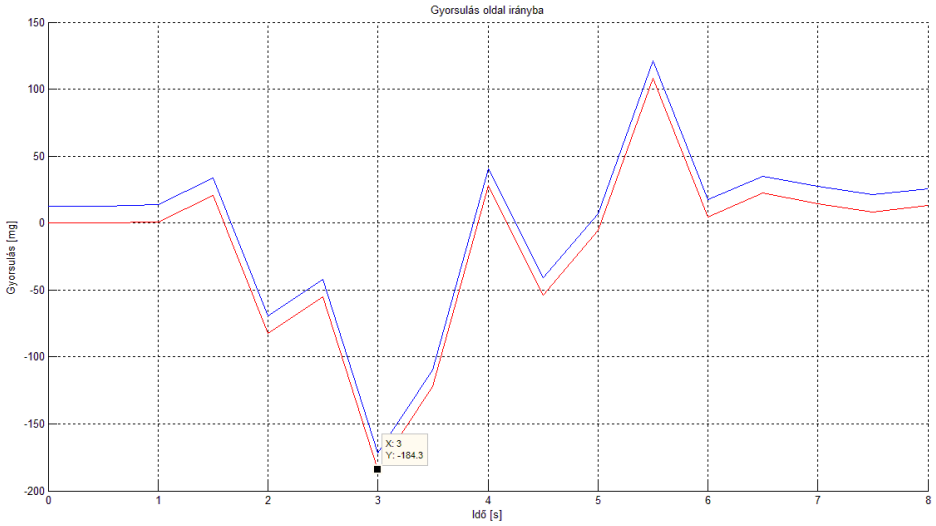
Az előzetesen mért 60 mg-s maximális érték egy hosszabb, 4-5 m-es távra számolt átlagérték volt, ami így a valóságot csak nagyságrendileg pontos közelítéssel tudta tükrözni. Az ábrázolt értékek is átlagértékek. A gyorsulásmérő 80 ms-onként állít elő új adatot. Ha azzal számolunk, hogy stopperes, 5 m-en történt mérés szerint átlagosan 1 m/s-mal halad az autó, akkor ez azt jelenti, hogy átlagosan 8 cm-enként van új információnk. A közvetlen értékek megjelenítése esetén a nagyon szórt eredményű mérések miatt a görbe nem mutatta érzékletesen a gyorsulást, ezért átlagolt értékeket használtam. A beérkező adatokból a MATLAB 5-öt összevár és utána átlagolja ki. Ez azt jelenti, hogy fél másodperces időszakok átlagos gyorsulását ábrázolja a grafikon, amivel kiszűrhetők a talaj felületi és tapadási egyenetlensége és a kommutátoros motor pólusváltása miatt érzékelhető ingadozások. Így ugyan nem lehet pillanatnyi gyorsulásértékeket ábrázolni, de a valóságos mozgást sokkal szemléletesebben bemutató ábrát kapunk.

7.2.3 Gyorsulás Y irányban

A mérés módszere ugyanaz, mint ahogy az X irányú mérésé, csak az oldalirányú mozgásra koncentrál. Főleg az irányváltásokkal lehet előidézni, ami jelentheti akár azt, hogy az autó egy helyben pörög. Ekkor elvileg egy szinuszos jelalakot kellene kapnunk, viszont az átlagolás miatt erre nincs lehetőség, csak hasonló formát fog felvenni a görbe.



7.5.ábra: 180°-os fordulás balra (Y irányba)



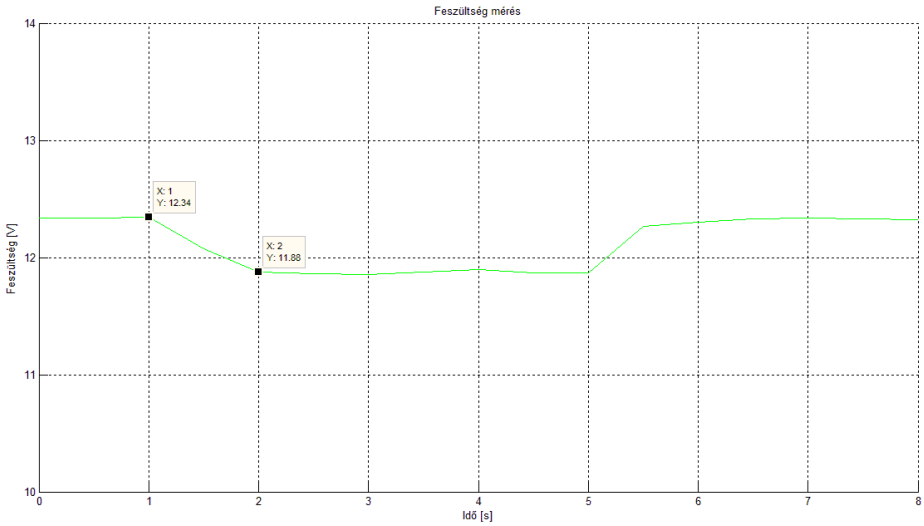
7.6.ábra: 180°-os fordulás jobbra (-Y irányba)

A két ábra (7.5.ábra, 7.6.ábra) mutatja az autó gyorsulás értékeit miközben egy 180°-os fordulatot végez. Itt is látszik, hogy miután megtörtént az elfordulás

visszatérnek az értékek a nullához. Az előző méréshez hasonlóan a kompenzált érték látható pirossal. A két maximális mért érték (178.8 és -184.3 mg) szintén közel azonos abszolút értékű a két irányban. Az autó a nem egyenes vonalú haladás miatt még inkább hajlamos kicsit megcsúszni, de a szinuszos jelleg így is felfedezhető.

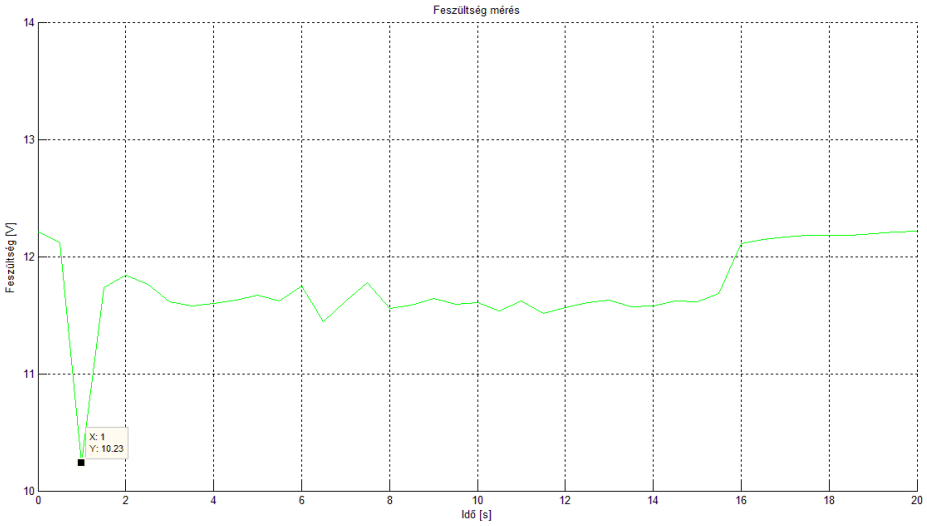
7.3 Akkumulátor feszültségének mérése

A gyorsulásértékekhez hasonlóan átlagolással történik meg a feszültség mérése és kijelzése, így összekapcsolható a mért érték a gyorsulással. Az akkumulátor névleges kapocsfeszültsége 12 V, pedig terheletlenül 13.5 V, kapacitása 1.3 Ah. Működés közben ettől kisebb kapocsfeszültség értékeket is kaphatunk, attól függően, mekkora az akkumulátor belső ellenállása és mekkora áramot vesz fel a terhelés. A kapocsfeszültség értéke a következőképpen számítható ki: $U_{kapocs} = U_{telep} - R_{belső} * I_{terhelés}$. Ebből is látszik, hogy azonos belső ellenállás mellett, minél nagyobb a terhelés, annál kisebb a kapocsfeszültség. Egy mikrovezérlő táplálása például nem okoz nagy feszültségesést, ellentétben például egy motormeghajtással.



7.7.ábra: Kapocsfeszültség előre mozgás közben

A 7.7.ábra mutatja a kapocsfeszültség alakulását, amiből kiszámolható, ha a 2 A-t vesznek fel a motorok akkor nagyjából 200 mΩ lehet az akkumulátor belső ellenállása. Ezek a feszültségértékek a 7.3-as ábrán látható gyorsulásértékek párjai. Nyomon követhető, ahogy az autó elindul, elkezd nagy áramokat felvenni, ettől leesik a kapocsfeszültség, amíg mozog addig közel ezen a szinten marad, majd amikor megtörténik a STOP parancs kiadása ez az érték visszatér az indulás előtti állapotba.



7.8.ábra: Kapocsfeszültség alakulása előre-hátra mozgás közben

A 7.8.ábra egy előre és hátra irány között sűrűn váltogatott mozgás kapocsfeszültség grafikonja. Látható, hogy az első indulásnál nagyobb áramra volt szüksége a motoroknak az autó megmozdításához, mint az előző példában, ezt okozhatta például az előző esethez képest eltérő tapadás. Indulás után hasonló alakot vesz fel a görbe, mint az előző példában. A hullámosságot a sűrűn végrehajtott irányváltások okozzák, mivel először egy ellentétes irányú mozgást kell megállítani, majd elindulni az új irányba.

8 Összefoglalás, továbbfejlesztés

A szakdolgozatomban sikerült a megvalósítani a feladatkiírásban előírtakat. Munkám során megismerkedtem egyszerű elektronikai eszközök (H-híd, DC motor) működésével, viselkedésével, valamint a Cortex magú 32 bites mikrovezérlők felépítésével, programozásával és fejlesztésével. Belekóstoltam a grafikus felületek (MATLAB) létrehozásába és PC-ről irányítható beágyazott rendszerek tervezésébe. Sikerült egy mechanikailag rendkívül egyszerű, de a vezérlés és mérés céljára megfelelő deszkamodellt előállítanom.

A megépített hardver jól működik. De továbbfejlesztésként a motorokat nagyobb hatásfokúra cserélném, ettől függően a H-híd helyett is egy másikat választanék. A mechanikai elemek (alváz, motor rögzítés, kerekek) is fejlesztésre szorulnak, ezért az egyszerű házilagos építés ezek helyett egy mechanikusan kész rendszert kellene beszereznem.

További fejlesztés tárgya lehet a C nyelven íródott mikrovezérlő program fejlesztése. Hatékonyabbá tehető a működés a perifériák megszakításokkal történő kezelésével, valamint egy szabályzási algoritmus implementálásával, ami minimálisra csökkentené az autó mozgását, a STOP parancs kiadása után.

Mivel a mikrovezérlő kapacitása alig van kihasználva ebben a deszkamodellben, ezért tervezem további szenzorokkal felszerelni az autót. Erre a legjobb példa egy ultrahangos távolságmérő, amivel meglehetne oldani, hogy távolságot hagyjon vagy kikerülje a útjába kerülő tárgyakat, vagy minimálisra csökkentse az ütközési sebességet.

Mindent összevetve egy jól működő kísérleti eszközt hoztam létre, ami a továbbfejlesztéshez ideális alapot biztosít.

Irodalomjegyzék

[1] ATmega128 mikrovezérlő adatlapja, URL:
http://www.atmel.com/Images/doc2467.pdf
2011

[2] I2C busz URL: http://www.cypress.com/fckimages/myresources/AN50987.jpg
2013.08.27.

[3] I2C kommunikációs prokoll
URL: http://opencores.org/usercontent,img,1352582784
2014

[4] SPI Daisy-chain, wikipedia:
URL: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
2014

[5] RaspberryPi URL: http://www.raspberrypi.org/
2014

[6] BeagleBone URL: http://beagleboard.org/bone
2014

[7] Internet of Things URL: http://en.wikipedia.org/wiki/Internet_of_Things
2014

[8] Free RTOS hivatalos weboldal URL: http://www.freertos.org/
2014

[9] MCU Market on Migration Path to 32-bit and ARM-based Devices
URL: http://www.icinsights.com/news/bulletins/MCU-Market-On-Migration-Path-To-32bit-And-ARMbased-Devices/
2013.04.25.

[10] ARM architektúra URL: http://arm.com/
2014

[11] ARM Cortex-M család ismertető
URL: http://www.element14.com/community/docs/DOC-68209/l/read-me-first-freedom-beginners-guide
2014

[12] STM32F407VGT6 mikrovezérlő dokumentációja URL: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf
2013

[13] STM32F4Discovery képe,
URL: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419
2014

[14] Freescale gyorsulásmérők
URL: http://www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=01126911821A9&cof=0&am=0
2014

[15] Analog Devices ADXL gyorsulásmérő család
URL: http://www.analog.com/parametricsearch/en/10129#10129/p4510=Accelerometer&p0=ADXL
2014

[16] STMicroelectronics gyorsulásmérők
URL: http://www.st.com/web/en/catalog/sense_power/FM89/SC444
2014

[17] IrDA, wikipedia URL: http://en.wikipedia.org/wiki/Infrared_Data_Association
2014

[18] Zigbee, wikipedia URL: http://en.wikipedia.org/wiki/ZigBee
2014

[19] Bluetooth, wikipedia URL: http://en.wikipedia.org/wiki/Bluetooth
2014

[20] HC-05 és HC-06 adatlap, URL: http://wavesen.com/
2014

[21] MATLAB, MathWorks hivatalos weboldal
URL: http://www.mathworks.com/products/matlab/
2014

[22] MATLAB, Instrument Control Toolbox
URL: http://www.mathworks.com/products/instrument/
2014

Függelék

Mikrovezérlő felé küldhető parancsok

ADC15: kiolvassa az értéket az AD átalakítóról, majd elküldi UART-on keresztül (Bluetooth modul innen továbbítja) a PC felé

DREQ: még 10 adatcsomagot kérése a mikrovezérlőtől

MEMSINIT: inicializálja a gyorsulásmérőt

MEMSRST: reseteli a gyorsulásmérőt

MEMSST: beolvassa a gyorsulásmérő státuszregiszterét, ez a resetelés után történő inicializálás esetén fontos, amikor ezzel ellenőrizhető, hogy sikeres volt-e a művelet

MEMSXYZ: beolvassa a X, Y és Z irányú gyorsulást, majd kiírja az adatokat az UART-ra, ahonnan a Bluetooth modul továbbítja a PC felé

MEN: engedélyezi a motorokat

MDIS: letiltja a motorokat

FW: mozgás egyenesen előre

BW: mozgás egyenesen hátra

FWL: mozgás balra előre, az autó képes megfordulni helyben

FWR: mozgás jobbra előre

BWL: mozgás balra hátra

BWR: mozgás jobbra hátra

STOP: motorok engedélyezve hagyás mellett megállás

Kapcsolási rajz