VISOKA TEHNIČKA ŠKOLA STRUKOVNIH STUDIJA - SUBOTICA
SZABADKAI MŰSZAKI SZAKFŐISKOLA
SUBOTICA TECH – COLLEGE OF APPLIED SCIENCES

AUTONÓM ROBOT MEGVALÓSÍTÁSA ÉS GÉPI LÁTÁS ALAPÚ ALGORITMUSSAL TÖRTÉNŐ IRÁNYÍTÁSA

Hallgató

Bessenyei Szilárd

Mentor

Dr. Pletl Szilveszter

Szabadka, 2014

Tartalom

Feladatkiírás:

Készítsen egy intelligens, kerekeken guruló autonóm mobil robotot. Tekintse át a kerekeken guruló mobil robotok gyakori megvalósítási módjait. Tervezzen meg egy, az alábbiakban megadott céloknak megfelelő platformot. Válassza ki vagy készítse el a szükséges érzékelőket, beavatkozókat, meghajtókat és vezérlő egységet. Készítse el a robotvezérlő programját. Tervezze meg és valósítsa meg a robot és egy PC közötti kommunikációt. Készítsen PC alkalmazást a robot vezérlésére. Telepítsen és parametrizáljon egy IP kamerát megvalósító Android alkalmazást. Oldja meg a kamera képének valós idejű továbbítását a PC felé. Készítsen képfeldolgozó alkalmazást a PC-n, amely alkalmas képszegmentálásra (színes labda követése) és karakter felismerésre a 3D-s térben (parancsok észlelése). Tesztelje a megvalósított rendszert. Adjon továbbfejlesztési javaslatokat.

1. Bevezető

Napjainkban a mobil robotok egyre inkább jelen vannak az élet számos területén. Megtaláljuk őket az iparban az egészségügyi ellátásban, a modern hadviselésben, az űrkutatásban és a szórakoztató iparban egyaránt. A szakterület fejlődése töretlen. A terület kutatása is jelentős történelemre tekint immár vissza.

A szakterület fejlődését a jelenkorban is zajló intenzív kutatások is biztosítják. A kutatások kiterjednek a mobil robotok megfelelő kinematikai struktúrájának kutatására, az alkalmazható érzékelők és beavatkozók kutatására és nem utolsó sorban az mobil robotoknál hatékonyan alkalmazható irányítástechnikai megoldások keresésére. A kinematikai megoldások közül legismertebbek a lépegető robotok a kerekeken guruló robotok az úszó autonóm járművek valamint a pilóta nélküli repülő járművek.

A dolgozat eredményeként a gyakorlatban megvalósult egy kereken guruló intelligens mobil robot. Kinematikáját tekintve a robot háromkerekű, amiből kettő differenciál hajtással meghajtott egy pedig szabadon guruló. A megoldás, számos jó tulajdonságának köszönhetően igen elterjedtnek számít napjainkban. A robotot vezérlőt egy ATmega2560 mikrovezérlővel rendelkező Arduino lap alkotja. A struktúra további elemei: szenzor-elektronika, két egyenáramú motor, két akkumulátor, egy szervo motor, egy aktív megvilágítást alkalmazó látórendszer és egy Bluetooth modul. Az érzékelők és a beavatkozók kapcsolatát a vezérlővel, egy saját fejlesztésű illesztőkártya biztosítja. A környezet érzékelését végző látórendszer egy aktív fényforrást és egy mobiltelefonba épített kamerát tartalmaz. A dolgozat bemutatja a robot működési elvét, megépítésének folyamatát és tesztelésének eredményeit.

A rendszer hierarchikus irányítási modellel rendelkezik. A stratégiai szintű irányítás programja egy PC-alapú architektúrán fut. A PC és a robot Bluetooth-on keresztül kommunikál egymással. A PC-n futó alkalmazás és a kommunikáció megvalósítása is saját fejlesztés. A robot két üzemmódban dolgozhat: az első szerint kézi vezérléssel, a második szerint pedig autonóm módon. A két üzemmód megvalósítását független programok végzik. Az első program, a kézi vezérlésű, ahol a felhasználó a PC

billentyűzet segítségével irányítja a robotot. Ebben az üzemmódban a robot érzékelői által mért adatokat a PC képes vizuálisan, áttekinthetően megjeleníteni.

A második egy automata irányítással rendelkező program. Ebben az üzemmódban a robotot egy kameraállványra felszerelt mobiltelefon által szolgáltatott kép feldolgozásának eredménye alapján irányítja a PC. A program tartalmaz egy képfeldolgozó modult, mely segítségével valósul meg a gépi látás. A modul rendelkezik a gépi látás esetében használatos leggyakoribb képfeldolgozó algoritmusok implementációjával, melyeknek célja a keresett objektum helyzetének megbízható meghatározása. A rendszer a kamera által küldött képek sorát, az OpenCV könyvtár függvényeinek megfelelő alkalmazásával valós időben dolgozza fel. A képfeldolgozás eredményétől függően a PC egy saját fejlesztésű protokollon keresztül küldi a stratégiai szint utasításait a robotnak. A taktikai szintű irányításhoz a L. Jones, Anita M. Flynn, Bruce A. Seiger csapata által 1999-ben bemutatott algoritmus módosított változata került alkalmazásra. [13]

Köszönetnyilvánítás.

A szerző háláját fejezi ki Istennek, hogy adott neki erőt, egészséget és bölcsességet a munkájához és családjának a támogatásért. Külön köszönetet mond a mentornak, Dr. Pletl Szilveszternek aki támogatta, és kedvező feltételeket biztosított a munka végzéséhez, Szegedi Mihálynak, aki segített a nyomtatott áramkör kimaratásában és tanácsaival, ICB Tech cég munkásainak, és Burkus Ervinnek, akik támogatták ötleteikkel és tapasztalataikkal, Barna Imre gumitechnikusnak, aki jó minőségű gumikat biztosított a robothoz, és Mrđanov Dánielnek, aki segített az alkatrészek beszerzésében, Tóth Dénes tanár úrnak, aki támogatta tapasztalataival, Miskolci Rolandnak, aki segített a robot működését bemutató film felvevésében és Dr. Odry Péter tanár úrnak aki bemutatta a robotika alapjait a főiskolán.

2. Jelmagyarázat

Jelzés/Rövidítés	Jelmagyarázat
OpenCV	Open Source Computer Vision - Nyílt forráskódú számitógépes látás
Threshold	határérték
PWM	Pulse Width Modulation (impulzus szélesség moduláció)
UART	(Universal Asynchronous Receiver/Transmitter)

3. A szakdolgozat témája

Egy gyakorlatban működő mobil robot megépítése, amely differenciál hajtással rendelkezik, ez mellett a robot és a PC számítógép között megvalósítani a kommunikációt, szabályozni a robot mozgását, több szabályozó algoritmus összehasonlítása, és a robot irányítása billentyűzet és képfeldolgozó program segítségével.
A robotot egy ATmega2560 mikrovezérlővel rendelkező Arduino lappal, egy saját fejlesztésű illesztő kártyával, szenzor-elektronikával, két egyenáramú motorral, két akkumulátorral, egy szervó motorral és egy kamerával rendelkezik.
Ez a fejlesztőplatform ingyenes és jól dokumentált. Az Arduino lapot AVR programozó nélkül is fel lehet programozni (mivel van rajta bootloader), és vannak hozzá különböző modulok (Bluetooth, ultrahangos távolságmérő).
A roboton a Bluetooth kommunikációt egy Bluetooth modullal működik, melynek előnye, hogy nem kellett külön összetettebb áramkört kialakítani a modul mellé. Ez mellett könnyen összekapcsolható a mikrovezérlővel.
A vezetéknélküli kamera egy Android operációs rendszerrel rendelkező telefont. IP webcam nevű program segítségével a mobiltelefont IP kameraként működik. A telefon IP címére való csatlakozásnál lehetőség van arra, hogy fogadjuk az általa vételezett videó jelet valós időben.

A gépi látás alapú algoritmus megírásához Emgu CV lett használva, ami egy platformfüggetlen .Net csomag az OpenCV képfeldolgozó könyvtárakhoz. OpenCV-nek nagy előnye, hogy gyors, és képes valós időben feldolgozni a képeket.
A programot C#-nyelvben lett írva, színek szűréséhez, a kép homályosításához, morfológiai műveletekhez beépített függvényeket használtam, a tárgy felismeréséhez, tulajdonságai meghatározásához, pozíciójának meghatározásához, hiba méréséhez saját fejlesztésű függvényeket lettek kifejlesztve, melyek következő fejezetekben lesznek taglalva.

4. Elméleti háttér

4.1. A tématerület elhelyezése

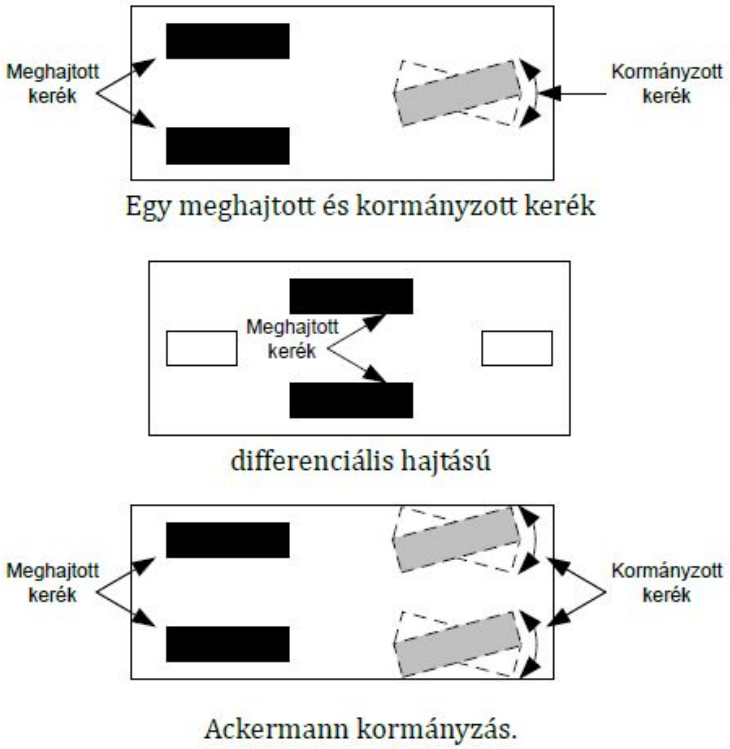
A mobil robotok témaköre egy igazi multidiszciplináris terület. Szükséges hozzá a gépészet, az elektronika, az informatika, irányítás technika, a képfeldolgozás és a kommunikációs technika valamilyen szintű ismerete. Ebben a munkában sem volt ez másként. Gépészeti szempontból főleg statikai és dinamikai ismeretekre, a gépelemek terén való jártasságra volt szükség. Az elektronika terén a nyomtatott áramkör tervezése, alkatrészek felforrasztása, akkumulátortöltő elkészítése, enkóderes lap tervezése jelentett kihívást. A képfeldolgozás témaköréből szükség mutatkozott a színterek ismeretére a képek szűrésének elméletére, a képtranszformációk alkalmazására, morfológiai műveletek ismeretére és szegmentálásra. Az informatika területéhez tartozik az alkalmazás szintű program elkészítése .NET–ben, a Bluetooth-os kommunikáció megvalósítása, a képkockák fogadása, a képfeldolgozási funkciók implementálása, képfeldolgozás eredményeinek továbbítása és a mikrovezérlő mint beágyazott rendszer programozása. Az irányítástechnika területéhez tartoznak a jelfeldolgozási műveletek, a motorok szabályozása, a differenciál hajtást alkalmazó szabályozási körök működtetése.

A témakör szakirodalmi lefedettsége jelentős. A differenciál hajtású robotokkal foglalkozik:

Embedded Robotics - Thomas Bräunl [1] könyve, a robot szabályozásának módszerei jól leírtak a: JONES, J., FLYNN, A., SEIGER, B. Mobile Robots - From Inspiration to Implementation, 2nd Ed., AK Peters, Wellesley MA, 1999 [1] és a http://www.robotc.net/wiki/Tutorials/Arduino_Projects/Mobile_Robotics/VEX/Using_encoders_to_drive_straight linken, a képfeldolgozás elmélete a : http://docs.opencv.org/ címen jól követhető, a munkában felhasznált mikrovezérlővel kapcsolatos információk a: http://arduino.cc/en/Main/arduinoBoardMega2560 címen találhatók.

4.2. A kerekeken guruló mobil robotok kinematikája

A legegyszerűbb mobil robotok a kerekes robotok. Ezek a robotok egy vagy több meghajtott kerékkel, és ez mellett esetleg még passzív kerekekkel rendelkeznek.



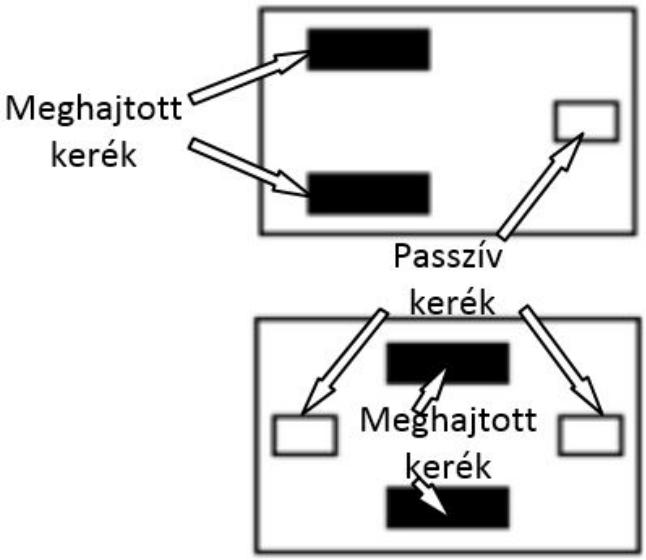
1. ábra: A kerekes robotok gyakori megvalósitásai [8]

Az 1. ábrán legfölül egy olyan háromkerekű robot látható, amelynek két meghajtott kereke és egy kormányzó kereke van. Ehhez a megoldáshoz egy meghajtó és egy kormányzó motor szükséges. E megoldás előnye, hogy itt a fordulás és a meghajtás független egymástól (két külön motort használunk mindkét feladatra), ebben a megoldásban a vezérlő algoritmusa nem bonyolult. A hátránya, hogy nem tud helyben megfordulni, mivel a vezérlő kerék nincs középen.

Az 1. ábrán középen látható a differenciál hajtású robot, mely igen elterjedtek számít napjainkban. A két meghajtott kerék lehetővé teszi, hogy a robot egyenesen vagy íven haladjon, vagy helyben megforduljon. A meghajtási parancsokat például, hogy egy bizonyos sugárral rendelkező íven haladjon, a kerekek fordulatszámának szabályozásával érjük el. Egy másik előnye ennek a megoldásnak, hogy a meghajtott kerekek fixen állnak,

és nincs szükség arra, hogy elforgassuk őket, mint az előző megoldásban. Ez jelentősen leegyszerűsíti a robot mechanikáját. A megoldásról bővebben a következő fejezet szól.
Az ábrán legalul pedig az Ackermann kormányzást láthatjuk, ami a hátsó kerék meghajtással rendelkező autóknál is használatos. Ennél a megoldásnál a kocsi négy keréken gurul. Két kerék a meghajtott, kettő pedig a kormányzott. A meghajtás egy motorral történik, míg a kerekek elforgatása egy másikkal. Evvel a megoldással nem nehéz egyenesen haladni, mivel a hátsó kerekek egy tengellyel vannak meghajtva. Hátránya az, hogy nem tud a helyben megfordulni, szükséges egy minimális sugarú kör. Ez mellett nem stabil fordulás közben, mert a hátsó kerekek képesek kicsúszni ilyenkor. [1]

4.3. Differenciál hajtás



2. ábra: A differenciál hajtással rendelkező robotok két fajtája [1]

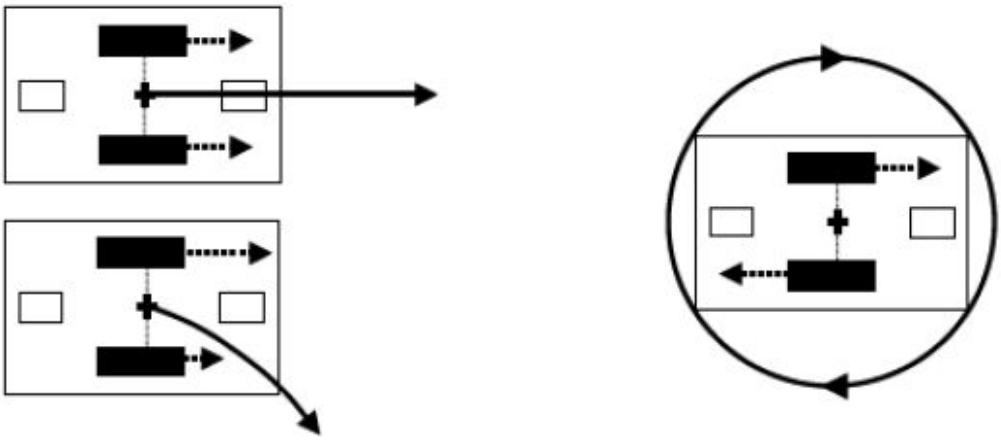
A differenciál hajtásos megoldások esetében a mozgásért és a fordulásért két azonos geometriájú meghajtott kerék a felelős. Mind a két kereket egy-egy külön vezérlésű motor hajtja. A robot mozgása attól függ, hogy melyik kerék milyen irányú és sebességű forgást végez. Bonyolultabb ez az egy kerekes meghajtástól, mert itt a két kerék forgását összehangolni és koordinálni kell ahhoz, hogy a robot a megfelelő mozgást el tudja végezni. A differenciál hajtásnak két fajtáját különböztetjük meg: az egy passzív kerekes

és a két passzív kerekes változatot. A két felépítés között a legnagyobb különbség a robot forgáspontjának elhelyezkedése, mivel az minden esetben a két meghajtott kerék „tengelyén" helyezkedik el és az a négy kerekes változat esetében lehet a kocsi alapjának geometriai közepe, míg a háromkerekes megoldás esetében, a robot súlypontjának helyétől függően az általában el van tolva valamely front irányában.

Mindegyik felépítésre igaz, hogy ha a kerekek egyforma irányba, azonos sebességgel vannak meghajtva, akkor a robot mozgása egyenes irányú. Abban az esetben, ha valamelyik kerék forgását lelassítjuk, akkor a mozgó robot a lassított kerék irányába fog kanyarodni. A kanyarodás élessége függ a két kerék forgása közötti sebesség különbségétől. Ha a kerekeket ellenkező irányba, de azonos sebességgel forgatjuk, akkor a robot egy helyben a saját forgástengelye körül fog forogni. [1]

A kerekek kerületi sebessége a következőképpen írható fel:

v(t) sebesség a t időpillanatban



3. ábra: A differenciál hajtással rendelkező robotok mozgásának fajtái [1]

Forgó mozgás esetében érvényes, hogy a kerületi sebesség megadható a következő összefüggéssel:

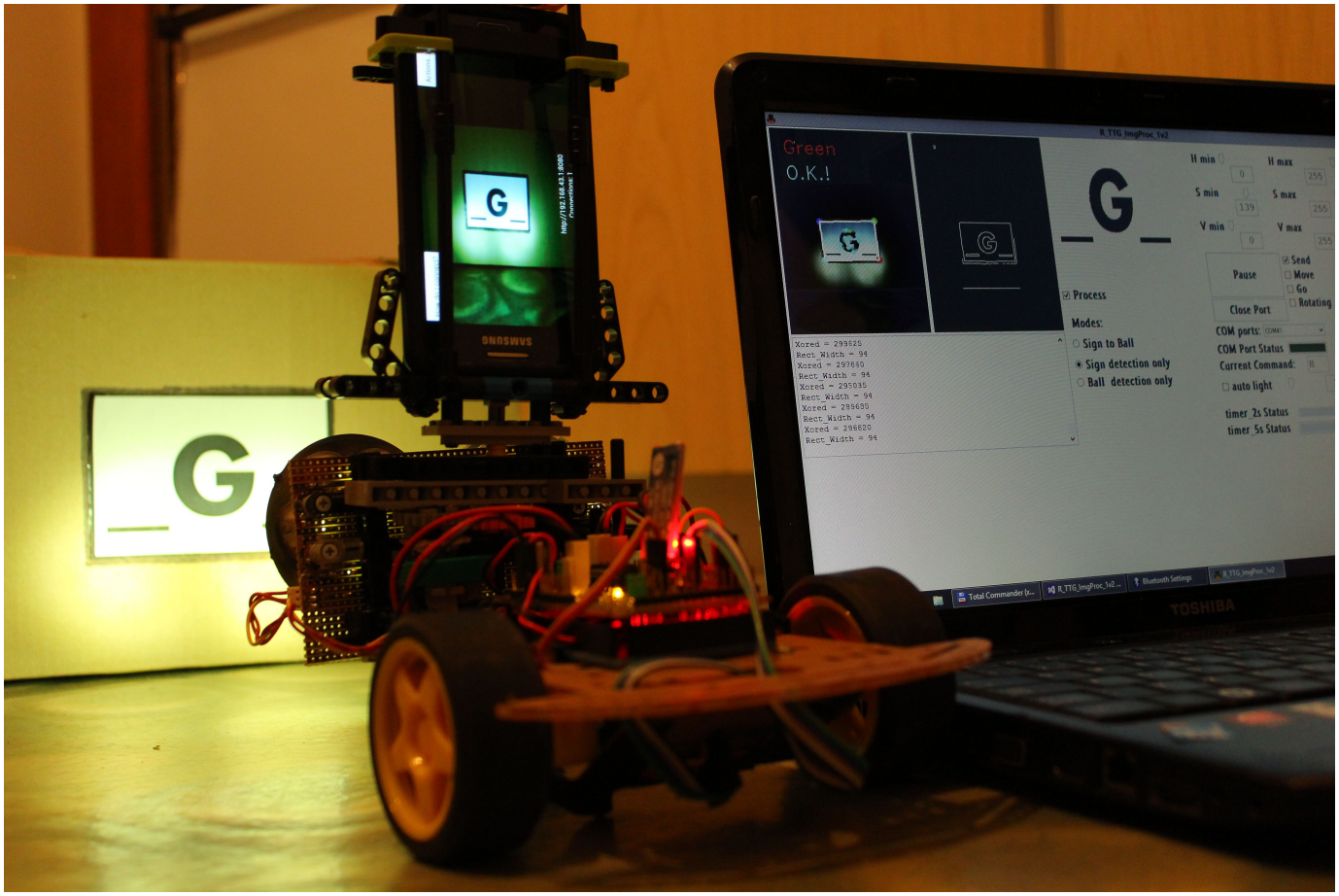
$$v(t) = r \cdot \omega(t)$$

ahol az ω (t) a kerék szögsebessége a t időpillanatban, r pedig a kerék sugara.

Jelölje v_L a bal kerék kerületi sebességét, v_R pedig a jobb kerék kerületi sebességét, akkor a következő mozgási fajták lehetnek:

- Egyenes haladás: $v_L = v_R$, $v_L>0$
- Ívben haladás: $v_L > v_R$ vagy $v_L > v_R$
- Helyben megfordulás (óramutató járásával ellentétesen) $v_L = -v_R$, $v_L>0$

A munkámban olyan robotot elkészítésére és szabályozására került sor, ami egy passzív kerékkel rendelkezik. A robot építésének fontos viszonyítási alapja a váza, az egyes elemek megfelelő pontosságú gyártásával és szerelésével elérhető, hogy a váz vízszintes legyen, így biztosított, hogy az irányításnak nem kell a geometriai pontatlanságból adódó hibák kompenzálásával foglalkozni.



4. ábra: a megvalósított differenciál hajtással rendelkező robot és gépi látás alapú algoritmussal rendelkező képfeldolgozó program működését illusztráló ábra.

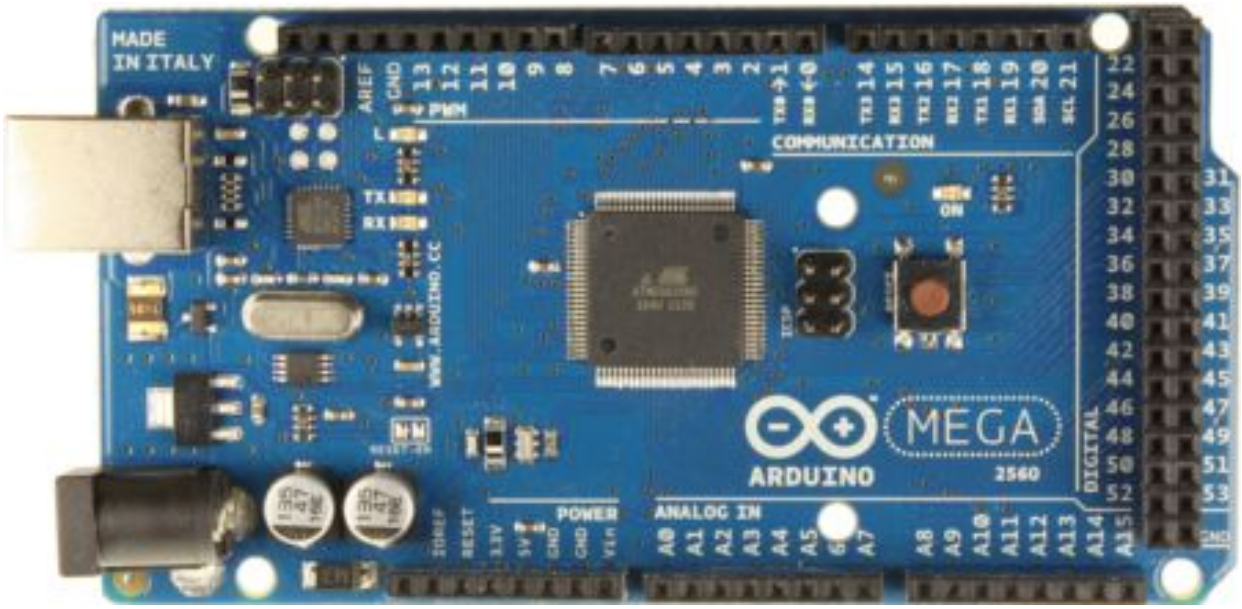
4.4. A felhasznált beágyazott rendszer

A munka megvalósításához alkalmazott, választott beágyazott rendszer az Arduino. Az Arduino egy az Atmel AVR mikrovezérlő családra épülő, szabad szoftveres elektronikai fejlesztőplatform, arra tervezve, hogy a különböző projektekben az elektronikus eszközök könnyebben hozzáférhetőek, kezelhetőek legyenek [2]. Széles körben használják, mivel olcsó, könnyen beszerezhető, nem kell hozzá külső programozó és több eszköz is csatolható hozzá.

A fejlesztői platform az integrált fejlesztői környezetből (IDE) és az Arduino lapból áll.

A fejlesztői környezet lehetőséget ad az alkalmazás fejlesztésére és tesztelésére. Az elkészített programok USB porton keresztül feltölthetők az Arduino lapra.

A fejlesztő környezet ingyenes, felhasználó barát, tartalmaz egy függvénykönyvtárt, a beépített függvényekkel összetettebb feladatokat egyszerűbben meg lehet oldani.



5. ábra: Arduino Mega2560 felülnézetből [3]

A munkámhoz Arduino Mega 2560-os a mikrovezérlőt használtam, amely az egyik legtöbb erőforrással rendelkező Arduino lap. Többek között nagy előnye, hogy egyszerre több soros portot is tud kezelni, ez mellett 5 timer van benne, 6 megszakítással rendelkező bemenet, és a többi Arduino laphoz képest sok ki/bemenettel rendelkezik.

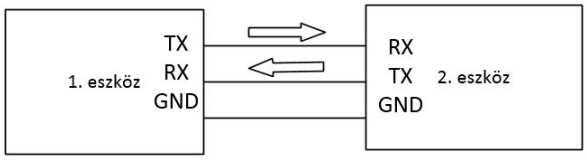
Technikai paraméterek [3]:

Mikorvezérlő	ATmega2560
Üzemi feszültség	5V
Bemeneti feszültség (ajánlott)	7-12V
Bemeneti feszültség (határok)	6-20V
Digitális ki/bemenetek	54 (ebből 15 képes generálni PWM-t)
Analóg bemenetek	16
Max leadható egyenáram I/O Pin-enként	40 mA
Programmemória / Adatmemória	256 KB (ebből 8 KB bootloader)
SRAM	8 KB
EEPROM	4 KB
Órajel sebessége	16 MHz

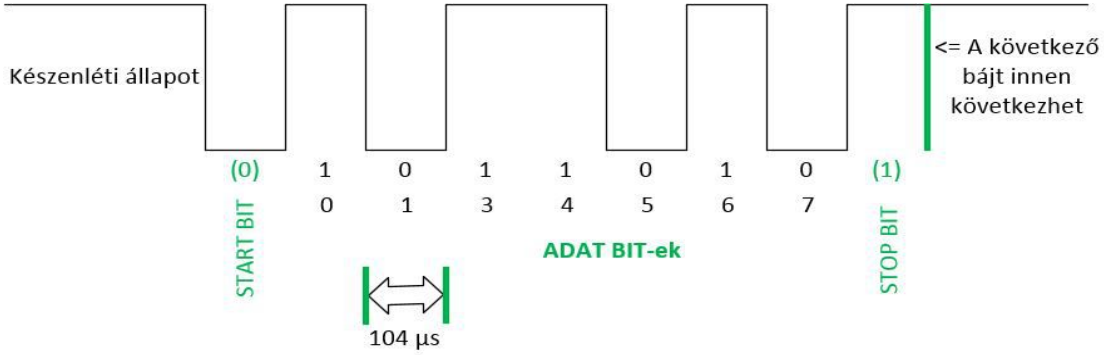
1. táblázat: Arduino Mega2560 Technikai paraméterei

4.5. Kommunikáció

UART kommunikációt használtam a számítógép és a robot között, ami a soros kommunikáció egyik fajtája. Előnye az, hogy full duplex (egy időben az eszközök küldhetnek és fogadhatnak adatot), és ez mellett vezeték nélküli közegben is alkalmazható. Az UART [4] (Universal Asynchronous Receiver/Transmitter) kommunikációban két egyenrangú eszköz vesz részt. Az információt bitenként továbbítjuk. Az információ csere két vezetéken keresztül történik. Mindkét eszköznek van egy RX (receiver-fogadó) és egy TX (transceiver - küldő) lába, amiket a 6. ábra szerint kell bekötni. Mindkét eszköz előre meghatározott sebességen kommunikál. Ezt a sebességet nevezzük baud rate-nek, ami azt fejezi ki, hogy másodpercenként hány szimbólumot, esetünkben bitet tudunk elküldeni. A kommunikációnál 9600 bps-t használtam, amelynek 104 μs kell egy bit elküldéséhez. Ez a kommunikáció full-duplex, miszerint egy időben az eszköz fogadhat, és küldhet is adatot.



6. ábra: Az UART kommunikáció blokksémája



7. ábra: Az UART kommunikáció folyamata

A küldő félen egy bájtnyi adatot bitenként egyesével küldi át sorban, a vételi oldalon pedig sorba összerakja. Az aszinkron kommunikáció miatt a 8 bit adat mellé, még Start bitet és Stop bitet is küldeni kell. Ez így 10 bit-nek felel meg, amit 9600 bps sebességnél 960 bájtot jelent másodpercenként.

Ez a kommunikáció fajta az RS-232 szabványra épül, ami 1969-ben került elfogadásra.

Bluetooth kommunikáció

A Bluetooth RS232-es szabvány rövidtávú vezeték nélküli alternatívája, amely az átvitelhez mikrohullámú rádióhullámokat használ. Sávszélessége 2.4-től 2.485 GHz-ig terjed. Ez is full-duplex kommunikáció. [5]

A munkámhoz JY-MCU Bluetooth Wireless Serial Port (HC-06) modult használtam, amit ha csatlakoztatok a PC-hez virtuális soros portként kezeli. Ezért az UART kommunikáció elve alapján kommunikál a robot a PC-vel.

A kommunikációt saját programmal oldottam meg, amit a következő fejezetben taglalok. Az Arduino Mega2560-ot egy soros porton programozom, de mivel több soros kommunikációra képes egy időben, ezért az mellett, hogy csatlakozva vagyok Bluetooth-on a robotra, egy időben képes vagyok programozni is a lapot.

5. A megvalósított robot bemutatása

A robot két üzemmódban dolgozhat: az első szerint kézi vezérléssel, a második szerint pedig autonóm módon. A két üzemmód megvalósítását független programok végzik. Az az első üzemmód, a kézi vezérlésű, ahol a felhasználó a PC billentyűzet segítségével irányítja a robotot. Ebben az üzemmódban a robot érzékelői által mért adatokat a PC képes vizuálisan megjeleníteni, így áttekinthető módon láthatjuk a mérési eredményeket. A második egy automata irányítást megvalósító üzemmód. Ebben az üzemmódban a robotot egy robot vázára felszerelt kamera képének feldolgozása segítségével irányítja a PC.

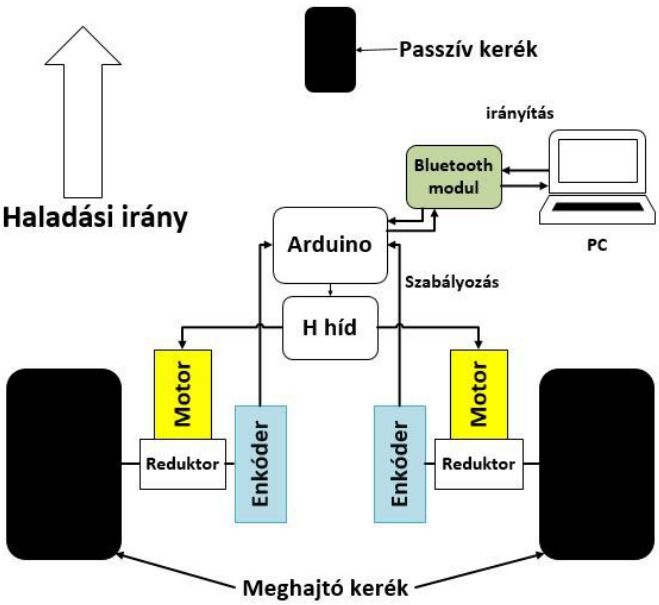
5.1.1. A kézi vezérlésű rendszer terve

8. ábra: Az irányítás hierarchiája

A munkámban egy differenciál hajtással rendelkező robotot valósítottam meg, amely egy passzív kerekékkel rendelkezik. Előnye, az, hogy egy kerékkel is hasonló eredményt el tudunk érni, mint a két passzív kerékkel rendelkező hajtásnál.

A robothoz beágyazott rendszerként egy Arduino alapú ATmega2560 mikrovezérlőre épülő, szabad szoftveres elektronikai fejlesztőplatformot használtam. és ez mellett egy saját fejlesztésű illesztő kártyát, szenzor-elektronikát, és két egyenáramú motort használtam.

A felhasználó irányítja a PC-t, a PC pedig az utasításokat továbbítja a mikrovezérlőnek. A mikrovezérlő pedig végrehajtja az utasításokat a beavatkozó szervekkel. A beavatkozás eredményességét szenzorok által tudjuk követni.



9. ábra: A mobil robot szerkezetét szemléltető ábra (kézi üzemmód)

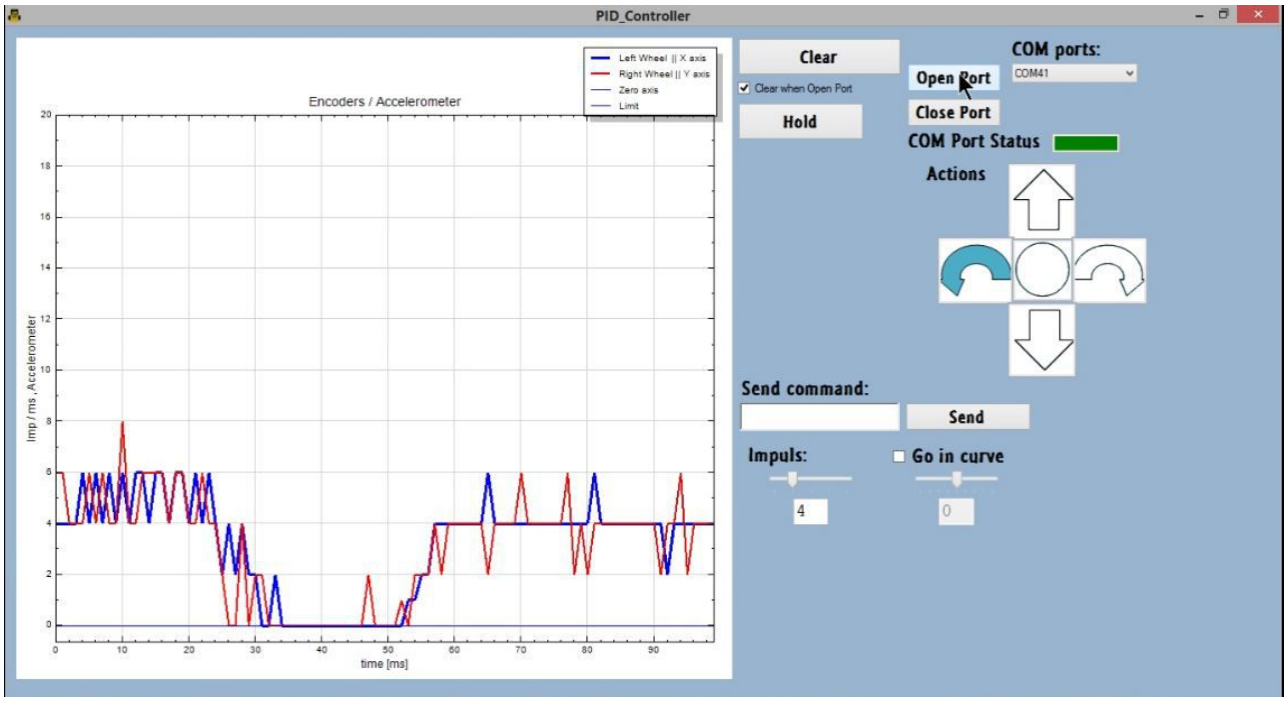
Kézi vezérlés esetében a robotot egy PC-n futó felhasználóbarát alkalmazáson keresztül lehet irányítani. Az alkalmazás adatgyűjtést és megjelenítést is végez, alkalmas a mikrovezérlő által szolgáltatott adatok vizuális megjelenítésére. Az alkalmazás teljes mértékben saját munka. Működése szerint a PC Bluetooth-on keresztül kommunikál a beágyazott rendszerrel. Az irányítás teljes hiearchiáját a 8. ábrán láthatjuk.

Ahhoz, hogy a leadott parancsot a robot pontosan végre tudja hajtani, az Arduino egy szabályozó algoritmust használ. A szabályozó körben az elfordulás érzékelését egy saját készítésű enkóder végzi. Az enkódereket újrahasznosítottam két régi számítógépes egérből. Az egerek eredeti áramkörét tanulmányozva, kis változtatással alakítottam ki az enkóderek elektronikáját. A robot hajtását két egyenáramú motor végzi, melyek integrálva tartalmazzák a hajtóművet. Motorok vezérlésére L293B integrált áramkört használtam, amely dupla H hidat tartalmaz, így egy motormeghajtó IC-vel meghajtható mindkét motor. A mobil robot szerkezetét az 9. ábra szemlélteti.

Munkámban L. Jones, Anita M. Flynn, Bruce A. Seiger csapata által 1999-ben bemutatott algoritmus [13] módosított változatát valósítottam meg. A kézi vezérlés működése szerint a a robot egy saját fejlesztésű PC-n platformú programtól, Bluetooth-on keresztül kapja az utasításokat. A robotot a felhasználó irányítja billentyűzet segítségével.

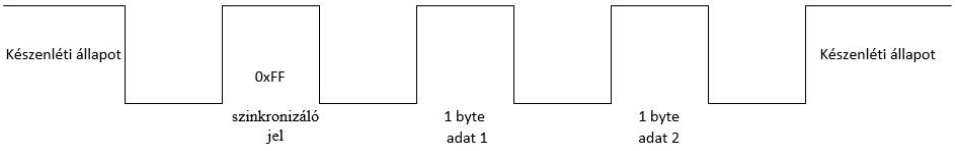
A program a következőket funkciókkal rendelkezik:
- ki lehet választani melyik COM portot akarjuk megnyitni
- kijelezni a port státuszát (zöld-aktív, szürke-inaktív)
- billentyűzet lenyomásával parancsokat küldeni a COM porton keresztül automatikusan
- egy érték megadásával meg lehet adni a haladási sebesség nagyságát, ami a mintavételezésenkénti impulzus számnak felel meg
- meg lehet határozni mekkora körívben menjen a robot (kerekek között hány impulzus különbség legyen)
- robottól fogadott adatokat fogadni, kiírni, és vizuálisan megjeleníteni (vizualizálásra használtam a NPlot nevű szabad felhasználású .NET könyvtár segítségével)

Öt fajta állapot van mozgás szempontjából: előre, hátra haladás, óramutató járásával megegyező és ellentétes irányba fordulás, és egy helyben állás. Azt, hogy a robot éppen melyik mozgást végzi, vizuálisan is láthatjuk a programban.

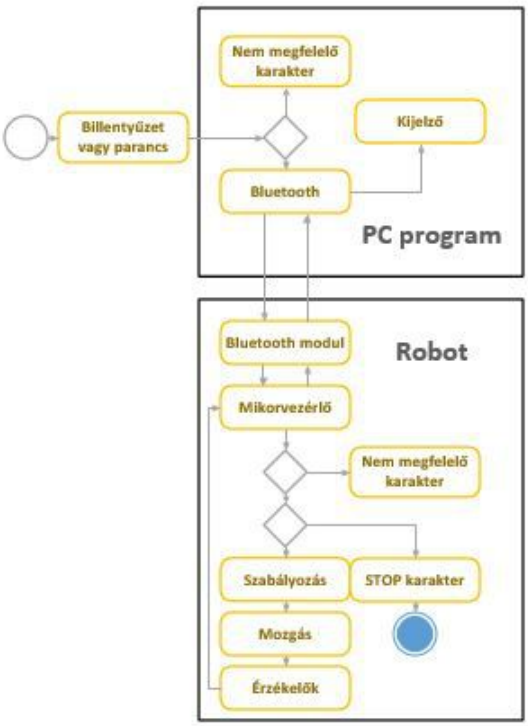


10. ábra: A PC program futás közben

A PC és a robot közötti kommunikáció megvalósítása során egymás után különböző adatot küldök, ezért szükség volt egy szinkronizáló jelre, hogy tudjam, hol kezdődik az üzenet. A szinkronizáló jelnek 0xFF-et választottam (ami 255-nek felel meg), mert a mérések értéke minden alkalommal kisebbek voltak mint 255.



11. ábra: A kommunikáció folyamata



12. ábra: A Mester-szolga (kézi vezérlés) UML diagramja

A rendszer UML diagramja a 12. ábrán látható. Miután csatlakoztunk a megfelelő COM portra, a program a billentyűzet lenyomására várakozik.

A billentyű lenyomása után, a program ellenőrzi, hogy milyen karaktert nyomtunk le. Ha valós parancsot hívtunk meg a billentyűvel, akkor elküldi Bluetooth-on keresztül. A másik oldalon a Bluetooth modul fogadja, és továbbítja a mikrovezérlőnek.

Az mikrovezérlő is leellenőrzi a karaktert, és ha valós utasítást kap, addig végzi, még nem kapja a Stop utasítást, amit akkor küld a PC program ha elengedtük a gombot.

A folyamat közben a mikrovezérlő a mért értékeket továbbítja a Bluetooth Modulnak, ami elküldi a PC-nek. Itt a képernyőn vizuálisan megjelenik az érték.

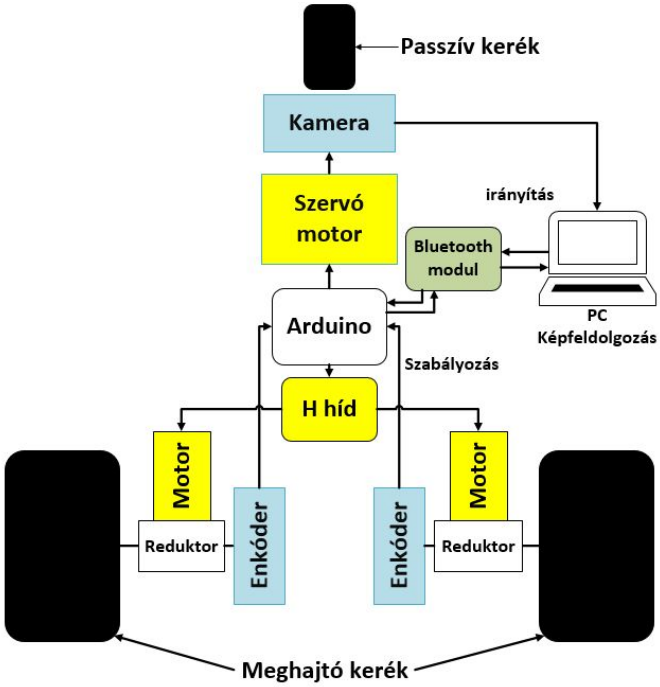
A programot C# nyelvben írtam és elkészítéséhez a Microsoft Visual Studio fejlesztői környezet használtam.

5.2. Az autonóm üzemmód

A második PC alkalmazás automatikusan irányítja a robotot egy robot vázára felszerelt kamera képének feldolgozása segítségével. A munka tartalmaz egy képfeldolgozó algoritmus megvalósítását, amely magában foglalja az élfelismerést, szín szűrést, morfológiai műveleteket, és ezeken a műveleteken alapuló transzformációkat, a képről az eszköz tulajdonságainak meghatározását és más műveleteket.

A kamera által küldött képek valós időben kerülnek feldolgozásra. A képfeldolgozáshoz felhasználtam az Emgu CV könyvtárait. A képekből nyert információ szerint parancsokat küldünk a robotnak.Ebben az esetben a hierarchia tetején nem a felhasználó van, hanem a kamera, amelynek feldolgozzuk a képet.

13. ábra: Az automata irányítás hierarchiája



14. ábra: A mobil robot szerkezetét szemléltető ábra (automata üzemmód)

A megoldáshoz felhasznált kamera egy Android alapú mobil telefon kamerája, mely az IP webcam nevű program segítségével úgy viselkedik, mint egy IP kamera. Az Emgu CV lehetővé teszi, hogy IP alapú kamera által szolgáltatott képet is feldolgozzunk. A csatlakozás úgy történik, hogy a beépített kamera sorszáma helyett az IP kamera IP címét írjuk. Miután feldolgozta a képet, a PC elküldi a parancsot mikrovezérlőnek, és a mikrovezérlő ez szerint vezérli az aktuátorokat (motorokat). A képen láthatjuk, hogy itt a szerkezet ki van bővítve egy kamerával. Ennek a megoldásnak hátránya, hogy nem a roboton történik a képfeldolgozás, így a robot mellett szükséges egy külső eszköz, amely feldolgozza a képet. Ez mellett meg kell oldani a kommunikációt az eszköz és a robot között, amelynél figyelembe kell venni a sebességet, hogy valós időben tudjunk beavatkozni. Mivel a valós idejű képfeldolgozásnak nagy a számítás igénye, ezért nem lett volna könnyű olyan eszközt találni, amely kielégíti ezt az igényt. A mobilra szerelt kamera előnye, hogy könnyen mozdítható, működés közben vizuálisan is láthatjuk a kamera képét. A megoldás előnye az is, hogy nincs szükség olyan akkumulátor beszerzésére, amely elég nagy kapacitással rendelkezik ahhoz, hogy hosszabb ideig tudja táplálni a motorokat, és a képfeldolgozó eszközt is egyszerre. Előnye ennek a megoldásnak, hogy külön végezzük a képfeldolgozást (PC végzi), és a mozgást (mikrovezérlő), így nincs szükség nagy különleges kapacitású akkumulátorra, a PC elég gyors a képfeldolgozásra, és a PC-n vizuálisan láthatjuk az eredményt.

6. Az alkalmazott képfeldolgozás elméleti háttere

A képfeldolgozás a jelfeldolgozás olyan fajtája, ahol a bemenő adat egy kép, ami lehet egy videó frame-je, vagy egy digitalizált kép. A kimenő adat, a bemenő kép módosított változata, vagy információ a kép tulajdonságáról. A legtöbb képfeldolgozó művelet a jelfeldolgozásban megszokott műveleteket, avval a különbséggel, hogy a képet úgy kezeli, mint egy olyan jelet, amely két dimenzióval rendelkezik. [17]

Ilyen műveletek például:

- szűrés
- Transzformáció (Fourier vagy Laplace)
- Morfológia műveletek (dilatáció, erózió)
- Élfelismerés (Canny, Sobel élfelismerő módszerek)

15. ábra: A képfeldolgozás folyamata

6.1. Valósidejű képfeldolgozás

A "valósidejű képfeldolgozás" fogalma nem egyszerűen a valósidejűség és a képfeldolgozás együttesét takarja. Természetesen a képfeldolgozási alkalmazások során is nagyon sok esetben elengedhetetlen a valósidejű reakció egy-egy eseményre, a legtöbb feladat valamilyen szinten megoldható az elvárt időkorlátokon belül, de a megoldás minősége, megbízhatósága sok esetben igen alacsony. Cél tehát az egyes algoritmusok feldolgozási sebességének, és eredményének minőségének növelése. [18]

6.2. Gépi látás

A gépi látás az emberi látás azon funkcióit valósítja meg, amelyek a retinai kép elemzését végzik. Ezek elsősorban a képi tartalom értelmezésére irányulnak: a látott képből következtet az objektumok 3D alakjára (felület rekonstrukció), az objektumok térbeli elhelyezkedésére, egymáshoz való viszonyára (mélységi információ kinyerése), illetve több időben egymást követő képből a mozgás érzékelése és a mozgó objektumok követése. [19] Ahhoz, hogy a robot a térben tudjon tájékozódni, számítógépes elkerülhetetlen.

6.3. Képfeldolgozó könyvtárak

A rendszer a kamera által küldött képek sorát valós időben dolgozza fel. A feldolgozást az Emgu CV programcsomag segítségével oldottam meg, ami az OpenCV képfeldolgozó könyvtárakhoz tartozó, platform független .Net csomag. A képfeldolgozás eredményétől függően a PC egy saját fejlesztésű protokollon keresztül küldi a stratégiai szint utasításait a robotnak.

OpenCV

16. ábra: OpenCV logója [20]

Az OpenCV (Open Source Computer Vision Library - Nyilt Forráskódú Számítógépes Látás Könyvtár) nyilt forráskódú C, C++ könytárak, amely főleg a valósidejű képfeldolgozásra lett kifejlesztve. A gyakorlatban az iparban és tudományos fejlesztéseknél egyaránt használják.
Intel cég kezdte fejleszteni a 90-es évek közepén, és 2000-ben lett nyílt forráskódú a béta verziója. 2008-ban átvette a fejlesztését Willow Garage.

Több nyelvet támogat: C, C++, Python, Java. Különböző funkciókat lehet vele megvalósítani, mint a: Ember-gép kapcsolat (Human–Computer Interaction), tárgy felismerés, arc felismerés, mozgás követés, képfeldolgozás (homályosítás, morfológia stb.). A könyvtár több száz számítógépes látás algoritmust tartalmaz. [20]
Előnyei főbb az Matlab-hoz képest: gyorsabb, alacsonyabb a gép igény ingyenes.
Ez mellett dokumentációja részletes, amely tartalmazza a képfeldolgozás különböző műveleteit és több példa programot.
Főbb hátrányai: nehezebb használni, nincs beépített fejlesztőkörnyezete - pár napot igénybe vett, mire megfelelően sikerült a Visual Studio-ban futtatni a példa programokat. [21]

Emgu CV

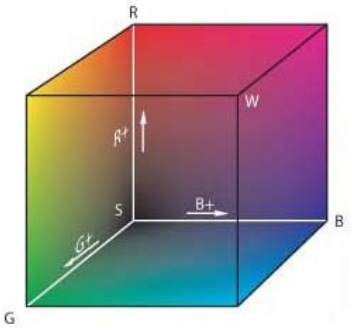
17. ábra: Emgu CV logója [23]

Emgu CV egy platformfüggetlen .Net csomag az OpenCV képfeldolgozó könyvtárakhoz (mint a C#, VC, VC++). A csomag segítségével Windows Form alkalmazásokban is lehet használni az OpenCV könyvtárakat. Így az OpenCV előnyeit (gyorsabb, alacsonyabb gép igény) össze lehet kapcsolni Form alkalmazásokkal, ami lehetővé teszi, hogy GUI (Grafikus felhasználói felület) használatát. [22]
A program megírásához Emgu CV-t használtam, C# nyelvel.

6.4. Szín terek

A szín model egy elvont matematika model, ahol a színeket számokkal ábrázoljuk. Általában három vagy négy értékkel írjuk le a színeket. Gyakorlatban több színteret használnak (RGB, CMY, CMYK, HSV, HSL), de a munkámhoz csak két színteret használtam, ami az RGB és a HSV.

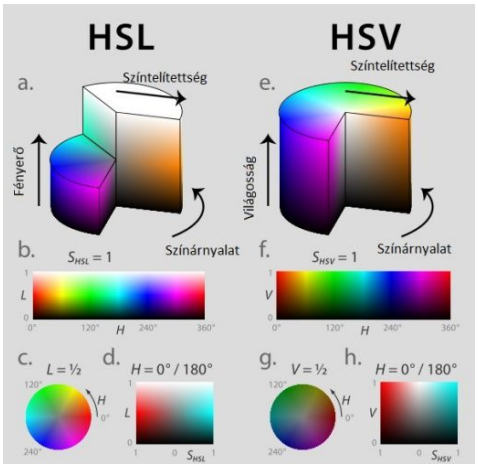
RGB színtér



18. ábra: RGB színtér [24]

Az RGB színtér egy olyan additív színmodell, ami piros (Red), zöld (Green) és kék (Blue) színek különböző mértékű keverésével határozza meg a különböző színeket. Elsődlegesen elektronikai eszközök és számítástechnika terén alkalmazzák (képernyők, kijelzők, érzékelők). [24] Az mindegyik szín összetevő értéke 0-tól 255-ig vehet fel értéket, ami 256 árnyalatnak felel meg. A fehér színnek a (255 ,255, 255) szín felel meg, a feketének pedig (0,0,0). Ezt a színteret egy 3D kockával lehet ábrázolni. A kamerától érkezett kép RGB színteret használ, amelyet beépített függvénnyel HSV színtérbe átalakítok.

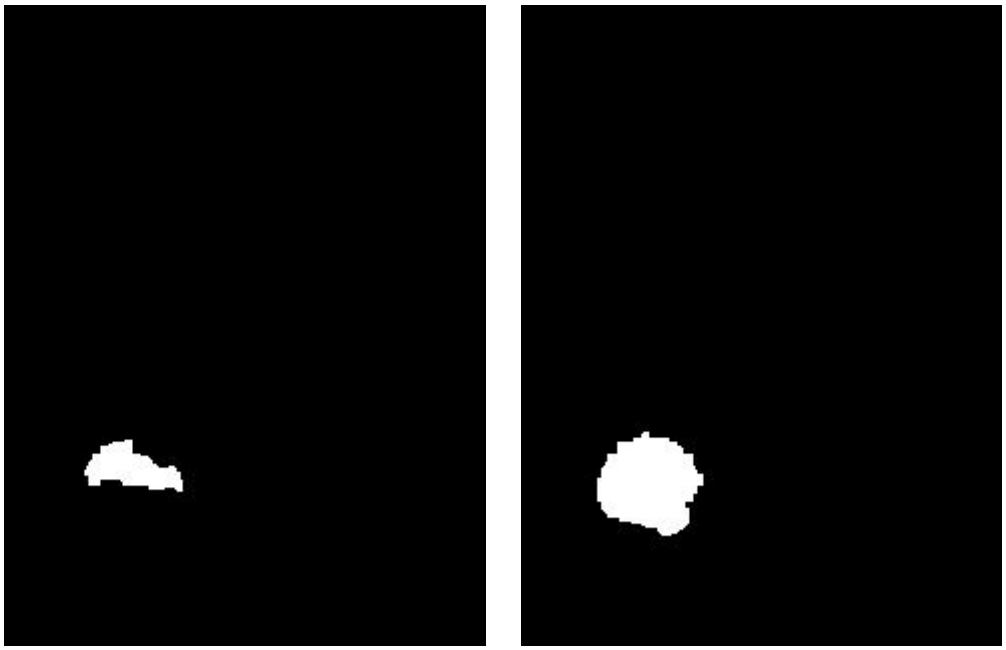
HSV és HSL



19. ábra: HSV színtér [25]

A számítógép grafika kutatók fejlesztették ki 1970-es évek második felében az RBG szín model alternatíváját a HSV-t (Hue-színárnyalat, Saturation-színtelítettség, Value-világosság) és a HSL-t ((Hue-színárnyalat, Saturation-színtelítettség, Lightness-fényerő). Képszerkesztő programokban, képelemzésnél és számítógépes látásnál alkalmazzák.

A színeket henger koordináta rendszerben ábrázoljuk. [25]
A HSV színtérben könnyebben lehet a tárgyakat kiszűrni. Előnye, hogy a színárnyalat külön van választva a színtelítettségtől, és a világosságtól. Ez a valós képeknél jobb eredményt ad, mint az RGB színmodell.



20. ábra: eredmény RGB szűrés után (bal oldalt) és HSV szűrés után (jobb oldalt)

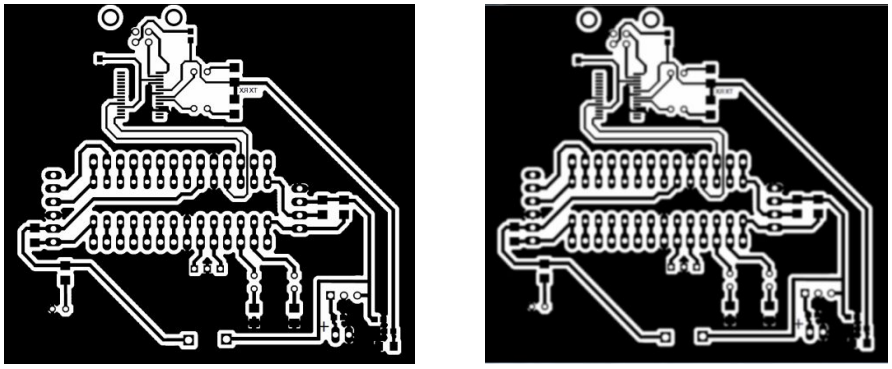
6.5. Szűrés - Threshold

A bemenő képből egy bináris képet készítünk szűrés segítségével. Az alsó és a felső határ megadásával ki tudunk egy bizonyos színtartományt emelni. Mivel HSV színtérbe van átalakítva a kép, ezért a következő hat értéket választjuk ki:
- Minimális H-t (minimális színárnyalat)
- Maximális H-t (maximális színárnyalat)
- Minimális S-t (minimális színtelítettség)
- Maximális S-t (maximális színtelítettség)
- Minimális V-t (minimális világosság)
- Maximális V-t (maximális világosság)

Azok a pixelek amelyek értékei a két határ közé vannak fehérek lesznek, a többi pedig fekete. A szűrés után egy bináris képet kapunk, ahol nincs színárnyalat, hanem csak fehér vagy fekete. (20. ábra)

6.6. Gauss homályosítás (Gaussian blur)

Gauss homályosítást sűrűn használják képszerkesztésben. Fő alkalmazási területe a kép zajának, és részletességét csökkentése. Előfeldolgozásra is alkalmazzák a számítógépes látású algoritmusoknál. Megvalósítását tekintve matematikailag a kép és egy Gauss görbe között egy konvolúció. [26]



21. ábra: eredeti kép (bal oldalt), Gauss homályosítás után (jobb oldalt)

6.7. Morfológia

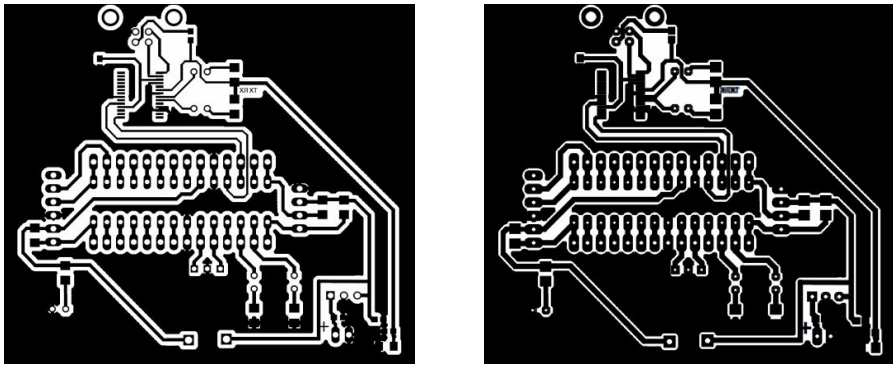
Legtöbbször a tárgy azonosítása a képből elég nehéz feladat. A technikát amelyet az ilyen bináris képeknél használnak morfológiai képfeldolgozásnak hívnak (a görög morphe szóból aminek jelentése forma vagy alak). Ennél az eljárásnál a szürke skálát át kell alakítani az első lépésben kétszínű képpé, amelyben minden pixel értéke korlátozva van vagy 0 vagy 1-re. A morfológiai algoritmusok nagy része egyszerű logikai művelet egyszeri használatra. Másszóval, minden alkalmazás külön megoldást igényel a kísérletekre és hibákra alapozva. [30]

Erózió

Az erózió az objektum körbenyírását jelenti, a strukturáló elem sugarával csökken minden irányban az objektum mérete. Nevezik még hámozásnak és fogyasztásnak is.

Az erózió végrehajtása során az objektumnak azok a képpontjai maradnak meg továbbra is az objektum részeként, amelyekre ráhelyezve a strukturáló elemet, annak minden pontja az objektumhoz tartozó képpontot fed. Más szavakkal: Az objektum azon pontjait, amelyekre ráhelyezve a strukturálóelemet, annak valamely pontja a háttérhez tartozó képpontot takar, a háttérhez kell sorolni. [27]

Az erózió alkalmas a zaj eltávolítására, és különböző elemek egymástól elkülönítésére.

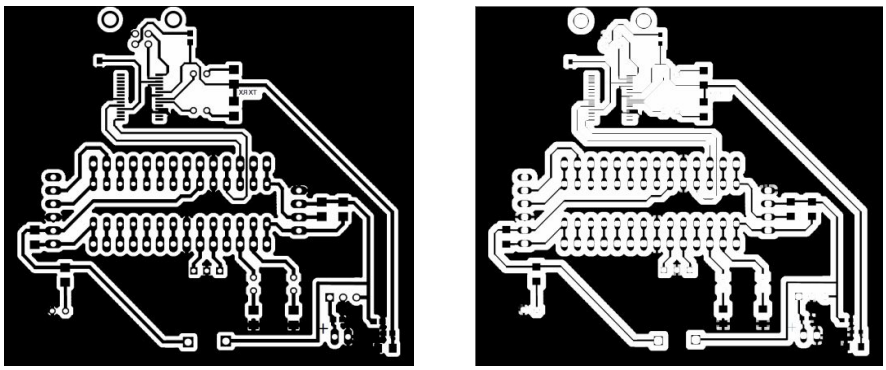


22. ábra: az eredeti kép (bal oldalt), a kép erózió után (jobb oldalt)

Dilatáció

A dilatáció az erózió ellentéte, az objektumot bővítjük a strukturáló elem sugarával. Hizlalásnak is szoktuk nevezni.

A dilatáció eredményeként a háttér azon pontjai, amelyekre a strukturáló elemet ráhelyezve az objektumhoz tartozó képponttal is fedésbe kerül, az objektumhoz fognak tartozni. A dilatáció alkalmas az objektumon támadt lyukak befoltozására. A hizlalás eredménye képen a szétválasztott, közeli objektumok újra összeolvadhatnak. [27]



23. ábra: az eredeti kép (bal oldalt), a kép dilatáció után (jobb oldalt)

6.8. Transzformációk

Ha az eróziót és a dilatációt kombináljuk, akkor különböző transzformációkat tudunk megvalósítani. Ennek előnye, hogy képesek vagyunk kis fehér vagy fekete felületeket eltüntetni, vagy nagyobb fehér vagy fekete felületeket kiemelni.

Nyitás

A nyitást úgy valósítjuk meg, hogy eróziót végzünk a képen és azután dilatációt.

Arra hasznos, hogy kis fehér felületeket eltüntessük.

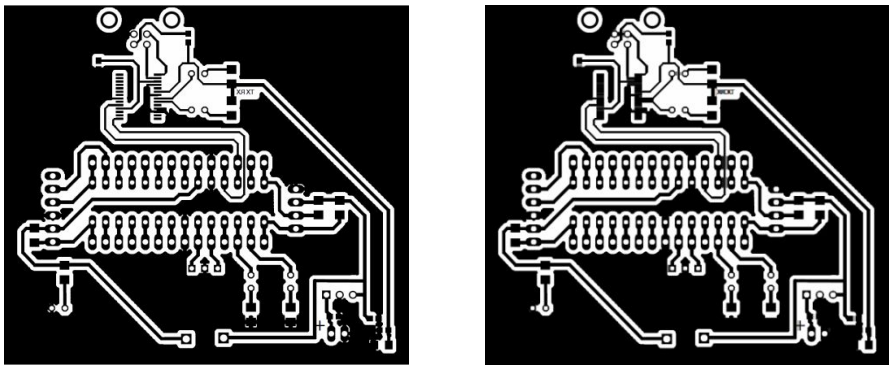
dst - feldolgozott kép

dilate - dilatáció művelet

erode - erózió művelet

src- bemenő kép

$$dst = dilate(erode(src))$$



24. ábra: az eredeti kép (bal oldalt), a kép nyitás után (jobb oldalt)

A munkában ez a transzformációt zavar szűrésére használtam, vagyis azok a felületek eltüntetésére szolgált, amelyek nem tartoztak az eszközhöz.

Zárás

A zárás egy dilatáció és azt követő eróziónak felel meg. Evvel a transzformációval a kis fekete felületeket tünteti el.

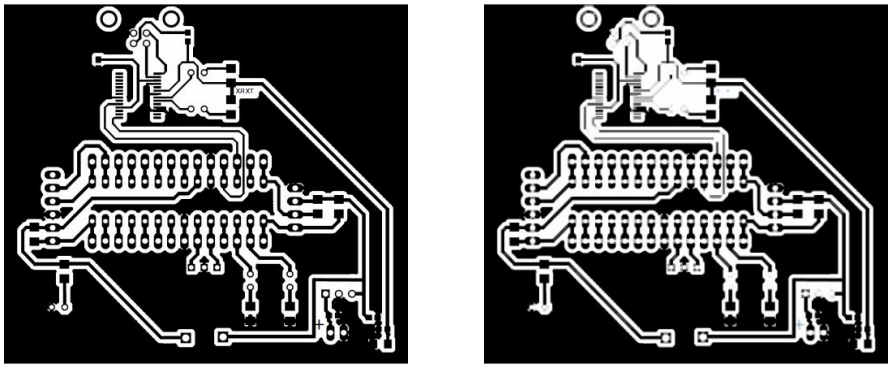
dst - feldolgozott kép

erode - erózió művelet

dilate - dilatáció művelet

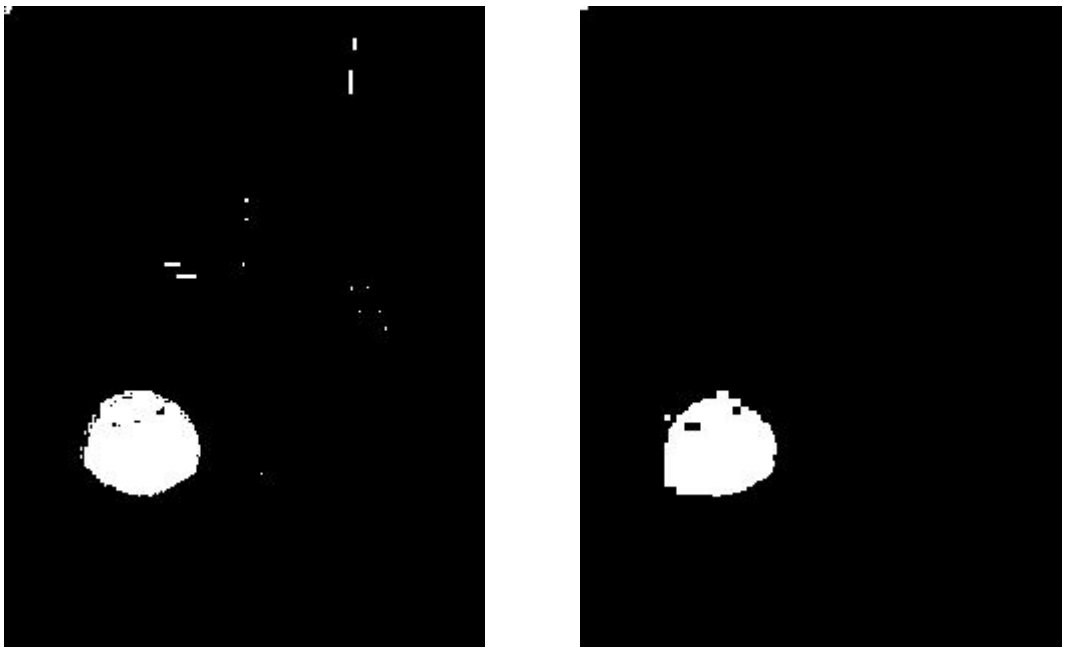
src- bemenő kép

$$dst = erode(dilate(src))$$



25. ábra: az eredeti kép (bal oldalt), a kép zárás után (jobb oldalt)

A munkában a zárást arra használtam, hogy az megnöveljem a valószinűséget, hogy a tárgy összefüggű fehér felület legyen.



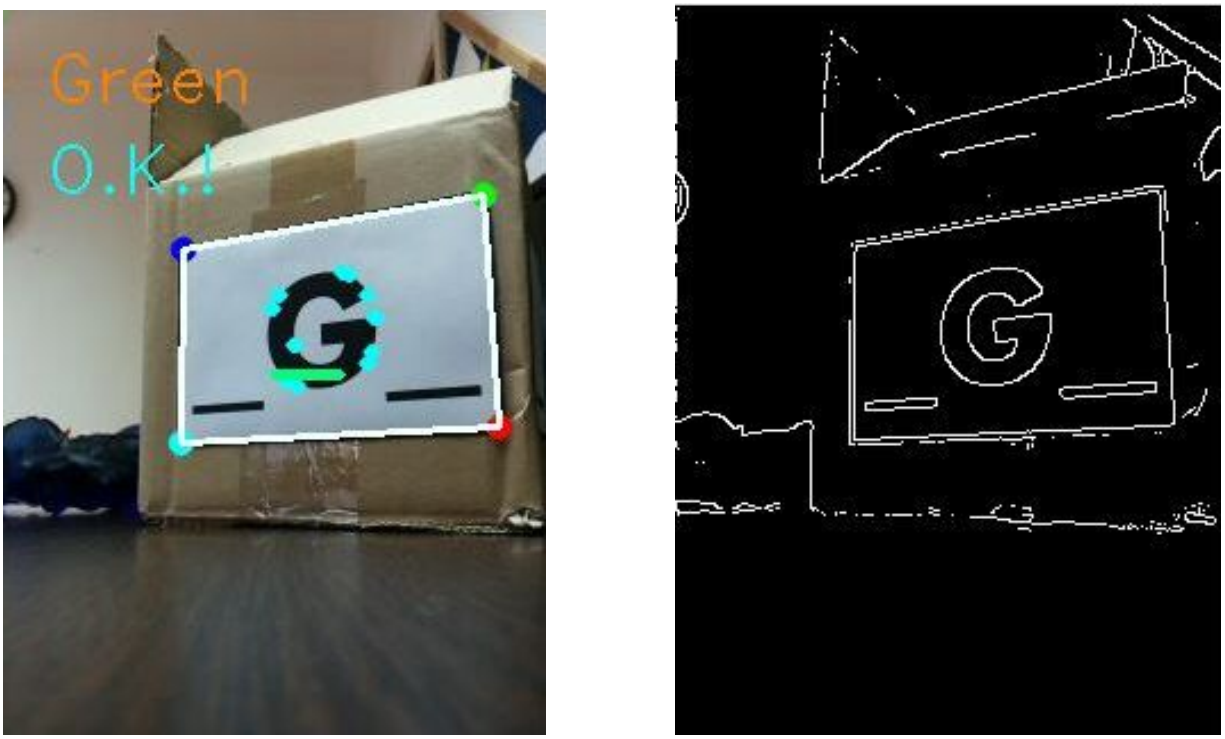
26. ábra: a szűrt kép (bal oldalt), a szűrt kép transzformációk után (jobb oldalt)

Először egy Gauss homályosítást végeztem, amelynél a maszk nagysága 9 pixel volt, utána két nyitást és három zárás alkalmaztam. Az eredeti szűrt képet és az eredményt a 25. ábrán látható,

6.9. Canny élfelismerő módszer

Éldetektálás fogalma alatt - a képfeldolgozás és a számítógépes látás területén - általában olyan algoritmusokat értünk, melyek egy digitális képen képesek azokat a pontokat azonosítani, melyeknél élesen változik a kép élessége illetve ahol megszűnik annak folytonossága. Éldetektálással és további feldolgozással képesek vagyunk tárgyakat felismerni, illetve azok távolságát meghatározni. [30] A Canny élfelismerő módszer John F. Canny fejlesztette ki 1986-ban. Optimális élfelismerő néven is ismert. Előnyei: alacsony hibaarány (jól felismeri a külön álló éleket), jó lokalizáció (az igazi képnek és a feldolgozott kép élei között a távolság minimális) [20]

A munkámban az OpenCV könyvtárak beépített Canny élfelismeréső algoritmust alkalmaztam.



27. ábra: Az eredeti kép (bal oldalt), és a Canny élfelismerő módszer után (jobb oldal)

6.10. Hough Vonal Transzformáció

A Hough Vonal Transzformáció feladata egyenes vonalak keresése. Alkalmazható például metszéspontok keresésére, légifelvételek kiértékelésére.

Az egyenes úgy nevezett normálalakos alakját alkalmazza, ahol r ami az egyenes origótól mért távolsága és a ϑ ami a valamint a pozitív tengellyel bezárt szög által írjuk fel az egyenes egyenletét: [31]

$$r = x \cdot cos(\vartheta) + y \cdot sin(\vartheta)$$

A transzformáció gyorsabban megtalálja az egyenes vonalakat előfeldolgozásként a Canny élfelismerő algoritmussal megszűrjük a képet.

A transzformáció segítségével keressük meg a téglalap kontúrjait, amit később kifeszítünk a megadott képsíkra.

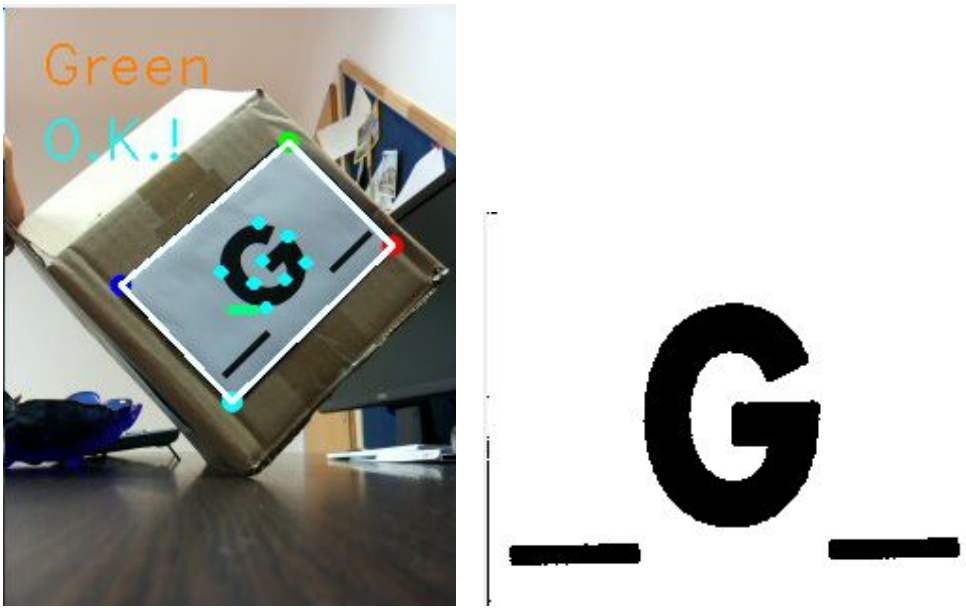
Az OpenCV könyvtárak beépített függvényei tartalmazzák a Hough vonal transzformációt.

7. Perspektív korrekciós algoritmus

A kamerával felvett kép mindig rendelkezik perspektív torzítással. Feladatom során, a robotok táblákra írt alakzatok segítségével utasítottam. Ahhoz, hogy a táblát be tudjuk azonosítani, szükség van perspekciós korrekcióra. A munkám során a tábla keresésére megvalósítottam egy ilyen algoritmust. Evvel a problémával más személyek is foglalkoztak. [29]

28. ábra: példa a perspektív torzításra

A perspektív korrekciós algoritmus három fő részre lehet felosztani:
- eredeti kép módosítása
- téglalap keresése
- megtalált téglalap síkjának kifeszítése a megadott síkon



29. ábra: Eredeti kép (bal oldalt), és a megszűrt kifeszített kép (jobb oldalt)

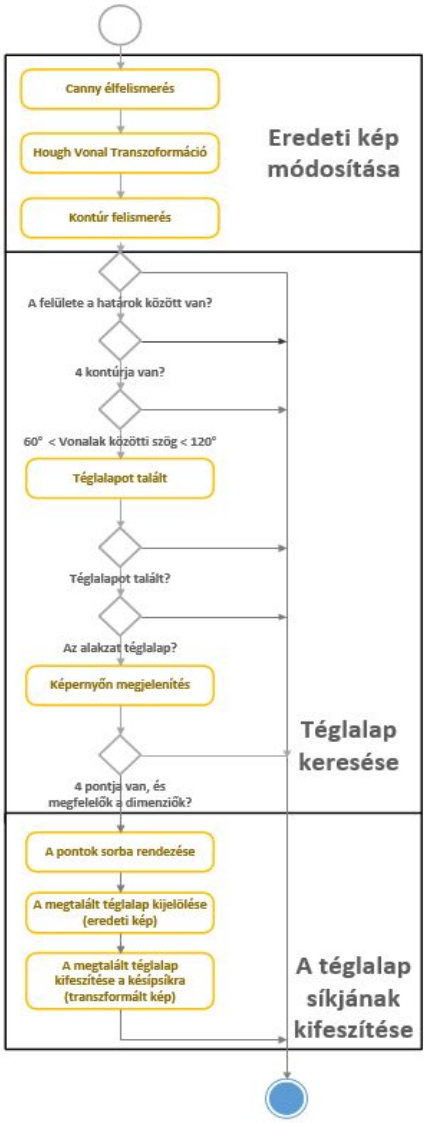
Ahhoz, hogy a tábla kontúrjait képesek legyünk meghatározni, a képet Canny élfelismerő műveletet el kell végezni. Az így kapott képet a Hough Vonal Transzformáció segítségével feldolgozzuk, ami megkeresi az egyenes vonalakat a képen. A kapott

adatoknál meghatározzuk a kontúrokat. A kontúrok egy téglalap kontúrjai legyenek, a következő három feltételnek kell, hogy eleget tegyenek:

- A felülete a határok között kell, hogy legyen
- 4 darab kontúrja van
- A vonalak közötti szög 60 és 120 között kell, hogy legyeknek

Ha a kontúrok eleget tettek a feltételeknek, akkor egy téglalapot talált a program.

Mielőtt kifeszítjük a megtalált téglalapot a megadott képfelületre, meg kell bizonyosodni, hogy a pontok (melyeket kontúrok összekötnek) megfelelő sorrendben vannak. A sorbarendezés után egy beépített függvény segítségével a megtalált téglalapot kifeszítjük a megadott képsíkra. A képsík nagyságát mi határozzuk meg pixelekben.

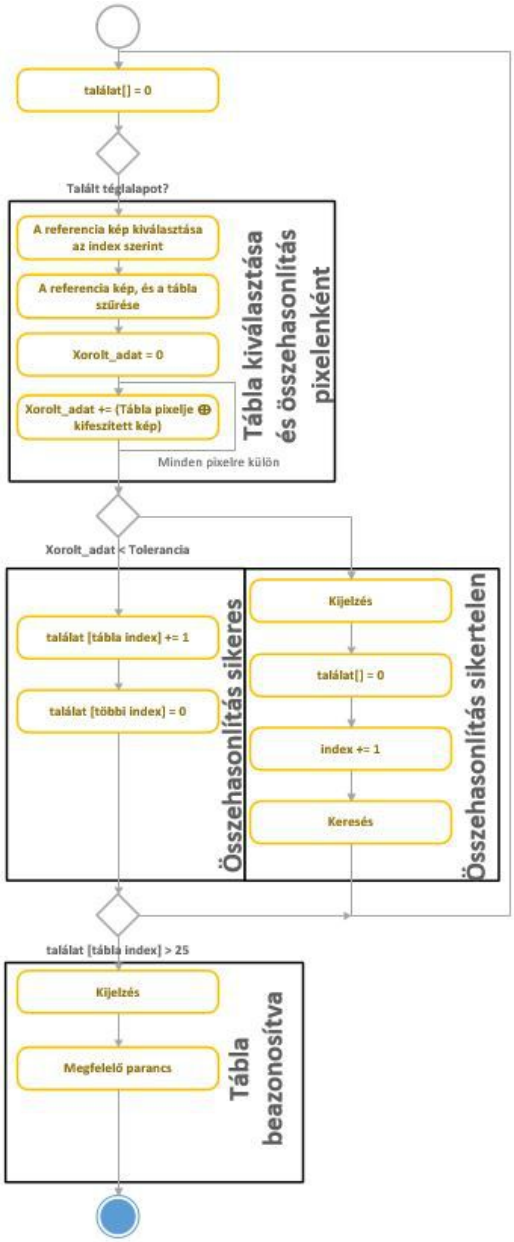


30. ábra: A perspektív korrekciós algoritmus UML diagramja

7.1. A tábla azonosítása

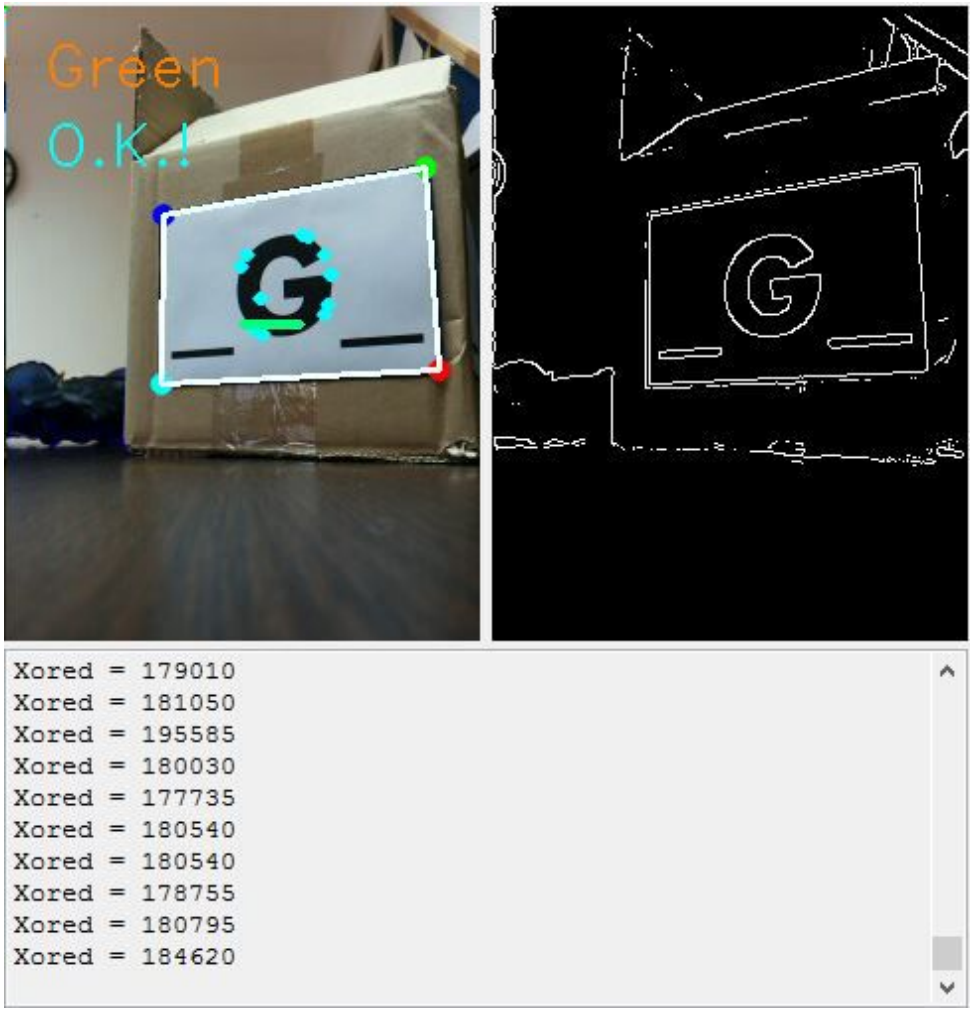
Miután kifeszítettük a táblát a megfelelő képsíkra, előre lementett képekkel (referencia képekkel) összehasonlítjuk. Ezt a folyamatot négy fő részre lehet bontani:

- Referencia kép kiválasztása és összehasonlítása a kifeszített táblára pixelenként
- műveletek, ha az összehasonlítás sikeres
- műveletek, ha az összehasonlítás sikertelen
- ha egymás után ugyan avval a lementett táblával megegyezik (kisebb a toleranciától), akkor beazonosítottuk a táblát.



31. ábra: A referencia kép és a tábla összehasonlítása

Az algoritmus során egy tömbben van eltárolva, hogy hányszor találta ugyan annak a referencia képet, és a kifeszített tábla képét. A folyamat a tömb elemeinek nullázásával indul. Először leellenőrizzük, hogy talált-e téglalapot a program, mivel ha nem, akkor nincs szükség összehasonlításra. Ha talált, akkor a referencia képet és a kifeszített táblát színszűréssel megszűrjük, hogy bináris képet kapjunk.
A két bináris kép minden pixelelei között XOR[1] műveletet végzünk el, és evvel az értékkel növeljük a egy változót. Ha a két pixel értéke megegyezik, akkor 0-át kapunk.



32. ábra: A kapott eredmény

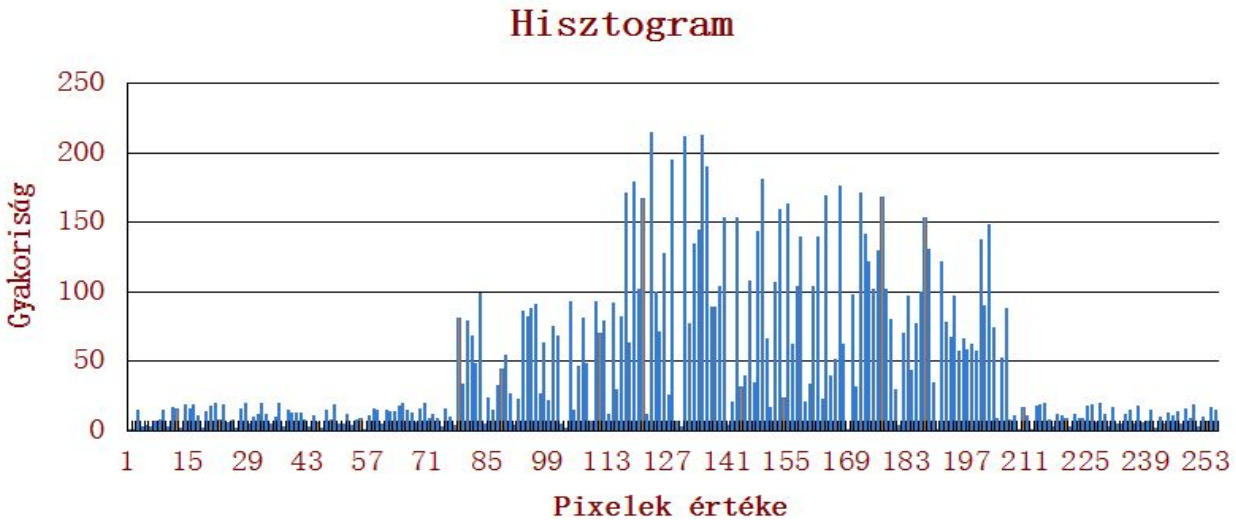
A program három fajta táblát tartalmaz, amelyek meghatározzák, milyen színű eszközt keressen robot.

[1] XOR- kizáró vagy logikai művelet, amelynek akkor 1 a kimenete, ha a két bemenet különbözik

8. Eszköz keresése szín szerint

Az automata PC program a tábla felismerés mellett képes tárgyakat színük alapján megkeresni. Az eszköz keresésének első lépésében az RGB képet átalakítjuk HSV képpé. Ez után színszűrővel megszűrjük a képet, ami egy bináris képet ad. A bináris képen a megmaradt felület eloszlását (hisztogram) vizsgáljuk, mely segítségével megkapjuk az eszköz pozícióját.

8.1. Hisztogram



33. ábra: példa a hisztogramra

A hisztogram a statisztikában az adatok értékeit eloszlásának, grafikon formájában történő ábrázolása. Képfeldolgozásban a hisztogram megadja a világosságkódok eloszlását egy adott képben, tehát a hisztogram egy olyan grafikon, amely a digitális kép különböző tónusú pixeleinek - vagy csoportjainak - számáról ad felvilágosítást. A vízszintes tengelyen a világossági értékek vannak a nullától a maximumig (például a feketétől fehérig). A hisztogram egymás melletti függőleges vonalakból vagy oszlopokból áll. [28]

Ezek magassága arányos azzal, hogy az adott tónusú képpont hányszor fordul elő a kép felületén. A munkámban egy más fajta hisztogramot használtam. Mivel az szűrés után egy bináris képpel dolgozunk, ezért a fehér pixelek (1-esek) eloszlását számoltam. Ezt a képnél függőlegesen és vízszintesen is elvégeztem.

8.2. Eszköz keresése (széleinek, közepének, szélességének, magasságának, pozíciójának keresése)

Eszköz pozícióját úgy kaptam meg, hogy a hisztogramnak megkerestem a maximális értékét. Hogy minél kisebb legyen a tévedés, ezért nem csak a egy sor maximális értéket hanem a környező sorok (vízszintes eloszlásnál) vagy oszlopok (függőleges eloszlásnál) eloszlásának értékét is számításba vettem. Ez az eljárás az átlagoló szűrésnek felel meg.

*2*n + 1 - a szűrő szélessége*

i - a pillanatnyi elem pozíciója

X_{filt} *- a vízszintes hisztogramból a szűrt jel értéke*

Y_{filt} *- a vízszintes hisztogramból a szűrt jel értéke*

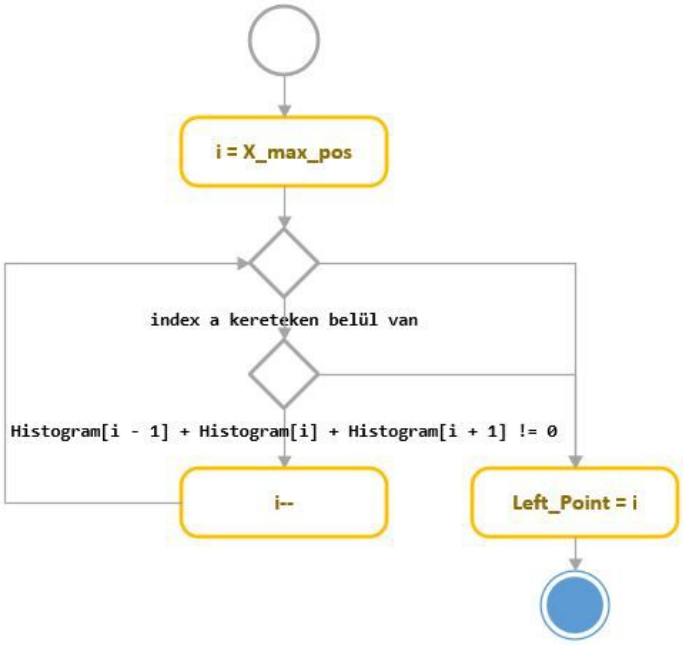
$$X_{\max} = \frac{X_{hist}[i-n] + X_{hist}[i-n+1] + ... + X_{hist}[i] + X_{hist}[i+1] + ... + X_{hist}[i+n-1] + X_{hist}[i+n]}{2 \cdot n + 1}$$

$$Y_{\max} = \frac{Y_{hist}[i-n] + Y_{hist}[i-n+1] + ... + Y_{hist}[i] + Y_{hist}[i+1] + ... + Y_{hist}[i+n-1] + Y_{hist}[i+n]}{2 \cdot n + 1}$$

Mivel a maximális érték nem volt fontos, hanem csak a pozíciója, ezért nem kellett külön vektort létrehozni, ami tartalmazza a szűrt jelek értékét. Ha az így kapott érték nagyobb volt, mint az előző maximális, akkor felülírtam az előző maximumot, és elmentettem a pozícióját. A módszer hátránya, hogy a kép szélétől n értéket nem veszünk figyelembe, ezért egy keret jön létre, ahol nem tudjuk elvégezni a számítást. Mivel a keret nagysága általában elhanyagolható a kép felbontása mellett, ezért ezt nem szükséges, hogy figyelembe vegyük.

34. ábra: az eredeti kép (bal oldal) a hisztogramokkal meghatározott pont (jobb oldal)

8.3. A tárgy széleinek meghatározása



35. ábra: A tárgy széleinek meghatározásának UML diagramja

A hisztogramok maximális értékei megkeresése segítségével, dúrván meglett határozva a tárgy közepe. A tárgy széleit négy ponttal határoztam meg (`Left_Point, Right_Point, Top_Point, Bottom_Point`) aminek az UML diagramja a 35. ábrán látható.
A hisztogramokból középponttól addig haladtam egy irányban, ameddig egymás után következő 3 pixel összege nem 0. Ez bináris képnél annyit jelent, hogy ameddig egymás után következő 3 pixel nem fekete. Előnye az, hogy így ha véletlen a eszközön egy kisebb fekete pont van, nem fogja rögtön szélként érzékelni, mivel a környező pontok fehérek.
Az 35. ábrán a bal pont meghatározását láthatjuk. A folyamat végén, mentjük a pozíciót a megfelelő változóba. A többi három folyamatának pont meghatározása is ezek a feltételek alapján történik. Miután megkaptuk a négy szélét a eszköznek, a következő egyszerű képlet alapján kapjuk meg a eszköz szélességét:
D - a eszköz szélessége
Right_point - a eszköz jobb szélének vízszintes koordinátája
Left_point - a eszköz ball szélének vízszintes koordinátája

$$D = Right\_point - Left\_point$$

A eszköz szélességét azért volt fontos meghatározni mert az eszköz távolságát ez alapján határozza meg a program.
A eszköz pontosabb közepének meghatározásához a következő képletet alkalmaztam:
X_{center} - *a eszköz pontosabb vízszintes közepe*
Y_{center} - *a eszköz pontosabb függőleges közepe*
Right_point - a eszköz jobb szélének pozíciója
Left_point - a eszköz ball szélének pozíciója
Bottom_point - a eszköz alsó szélének pozíciója
Top_point - a eszköz felső szélének pozíciója

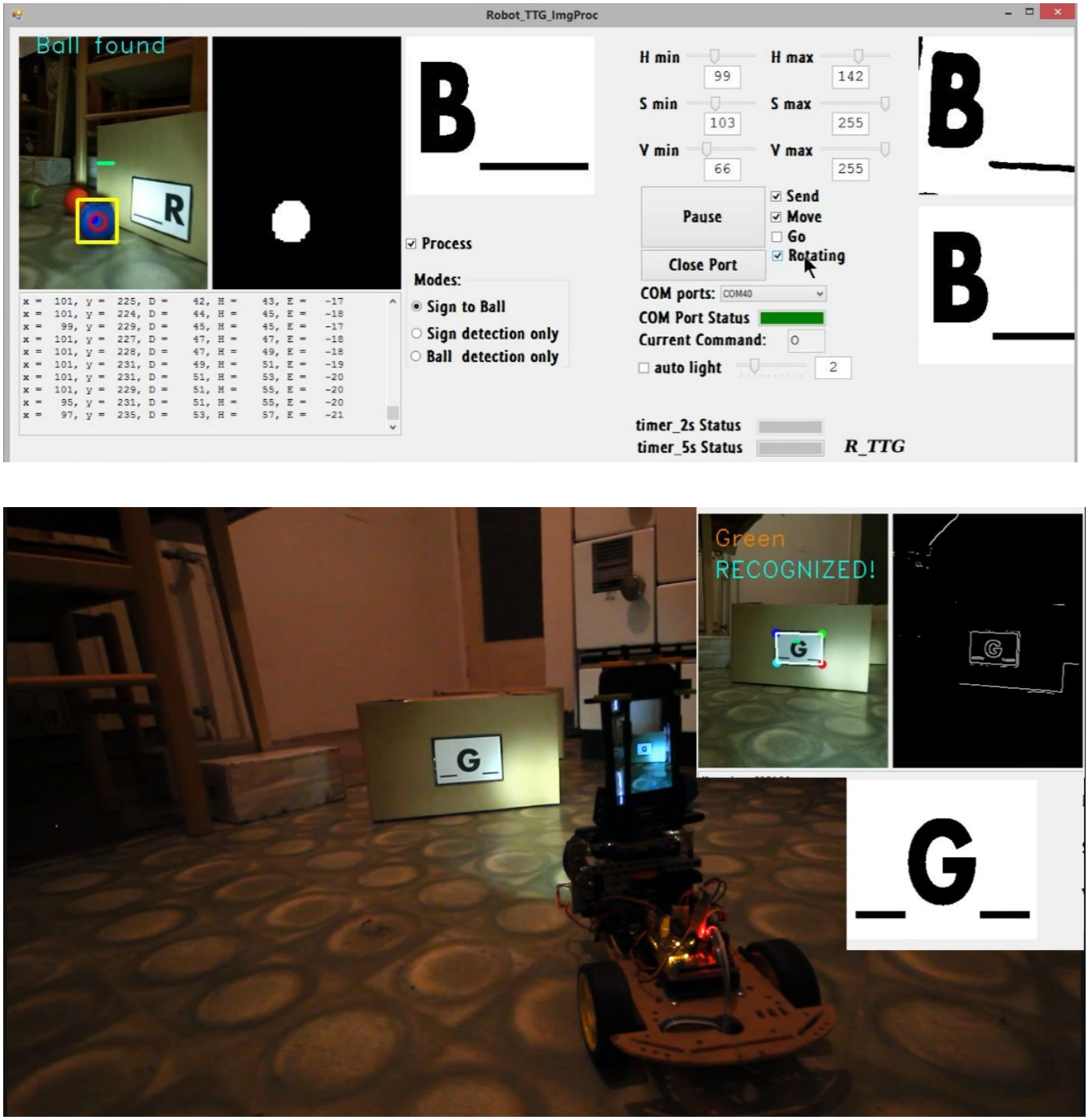
$$X_{center} = Right\_point - \frac{Right\_point - Left\_point}{2}$$

$$Y_{center} = Bottom\_point - \frac{Bottom\_point - Top\_point}{2}$$

A gyakorlat úgy mutatta, hogy ha ez a módszerrel határozom meg a eszköz közepét, akkor stabilabb, és pontosabb eredményt kapok, mint a hisztogramok segítségével egyedül.

36. ábra: Az eredeti kép (bal oldalt), a megtalált eszköz (jobb oldalt)

Azért fontos a eszköz pontosabb közepének meghatározása, mivel az szerint fordul a robot szervo motorja.



37. ábra: A képfeldolgozó program futás közben (felső kép), tábla azonosító üzemmód (alsó kép)

9. Szenzorok

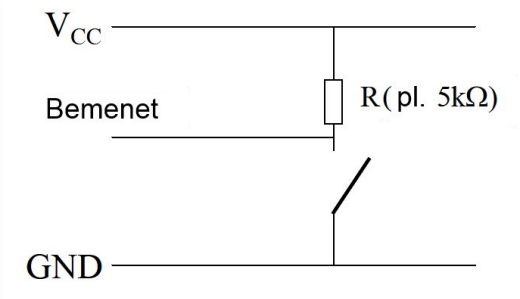
A szenzorok olyan jelátalakítók, amelyek nem villamos jeleket (mint például: kémiai, termikus, mágneses, optikai jeleket), villamos jellé alakítanak át. Jelük tovább vezethető, erősíthető, szűrhető és feldolgozható. [6]

A szenzorok a bemeneti jeleket az információ feldolgozóhoz továbbítják (ez esetben mikrovezérlőhöz), amely meghatározza a szükséges beavatkozásokat az aktuátorokkal.

Szenzoroknak több fajtája ismert: abszolút-inkrementális, bináris, analógok és digitálisak.

A munkámban bináris szenzorokat használtam, amiket részletezni fogok.

Bináris szenzorok

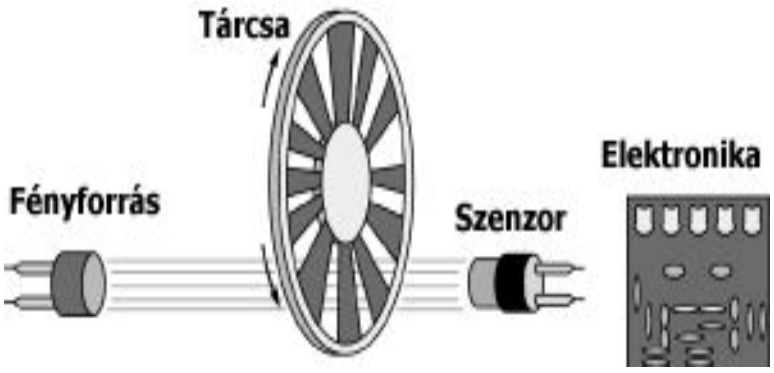


38. ábra: kapcsoló bekötése negatív logika szerint [8]

A bináris szenzorok a legegyszerűbb fajtái a szenzoroknak. Ezek egyetlen bit-et adnak vissza (0-át vagy 1-est). Ilyen például egy kapcsoló (38. ábra). A beágyazott rendszer digitális bemenetére vezetjük az érzékelő kimenetét, és így kapjuk meg jelet.

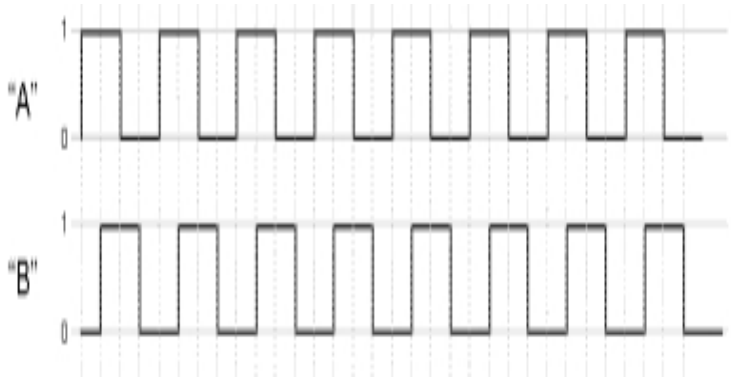
A 39. ábrán láthatjuk, ha be van kapcsolva a kapcsoló, akkor 0-ást kapunk a bementre, ha nincs, akkor 1-est, ami a negatív logikának felel meg. [8]

9.1. Inkrementális enkóder



39. ábra: Inkrementális enkóder [7]

Az inkrementális (növekményes) enkóder három részből áll: fényforrás, fény érzékelő szenzor és egy tárcsa.
Ez a tárcsa olyan, amelyben a rések egyforma méretűek, és egyforma távolságra vannak egymástól.
Ezeket a réseket két optokapu figyeli. A két kapu úgy van elhelyezve, hogy egymáshoz képest 90 fokkal eltolt fázisú jelet adjanak, amikor forog a tárcsa.
Ez a két jel az "A" és a "B" fázis. Az enkóder kell, hogy tartalmazzon egy elektronikát, ami táplálja az optokapu fényforrását és ez mellett tudnia kell a kapu vevőjéről érkező jelet valamilyen szabványos jellé alakítani. Ez leggyakrabban TTL logikai szinteknek felel meg (0-0.8 V-ig alacsony , és 2-5V a magas logikai szint).



40. ábra: Inkrementális enkóder által leadott A és B jel [7]

Az A és a B jelek 50% kitöltési tényezővel rendelkező négyszög jelek, melyek között 90 fokos fáziskülönbség van. A két jel fázisából tudjuk meghatározni az enkóder forgás irányát. Ezt a fajta jeladót azért hívják inkrementálisnak, mert a tengely elfordulásával arányos jel (impulzus sorozat) a tengely helyzetéhez relatív. Ez azt jelenti, hogy az álló enkóder abszolút szöghelyzetéről maga az enkóder nem ad információt.
Ahhoz, hogy tudjuk a pontos helyeztet, nyilván kell tartanunk az előző megszakítások számát. [7]
A robotnál használt enkódereket régi egerekből vettem ki, amelyeknek áramkörét az 11.1-es fejezetben részletezem.

10. Aktuátorok

Ahhoz, hogy a robottal tudjunk a külvilágra hatni szükség van mozgató elemekre, amik segítségével a mozgásállapotukat meg tudják változtatni. Ezeket az elemeket összefoglaló néven aktuátoroknak nevezzük. Az aktuátorok lehetnek:

- Elektromos aktuátorok
- Hidraulikus/pneumatikus aktuátorok [8]

A munkámban csak elektromos aktuátorokat használtam.

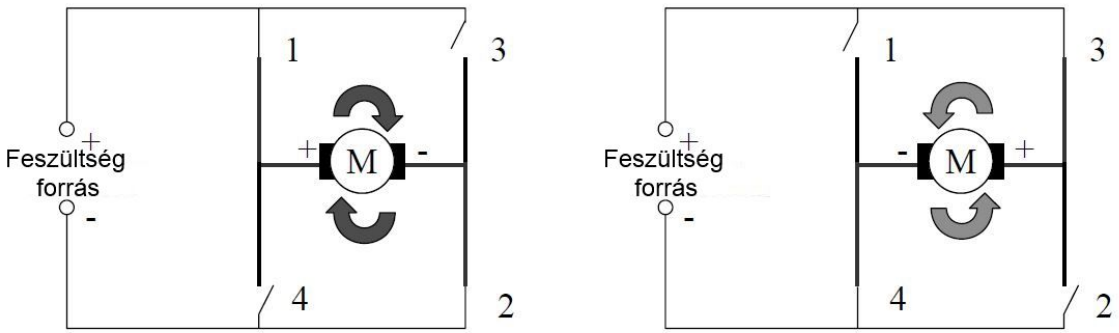
10.1. Egyenáramú (DC) motorok

A egyenáramú motorok a legszélesebb körben használt helyváltoztató rendszerek a mobil robotoknál. Tiszták, csendesek és megfelelő teljesítménnyel rendelkeznek különböző feladatok elvégzéséhez. Jóval egyszerűbben vezérelhetők, mint a pneumatikus szerkezetek. A hagyományos egyenáramú motorok szabadon mozognak a léptetőmotorokkal szemben, ezért a motorvezérléseknek egy visszatápláló mechanizmusra (elfordulás jeladó) van szükség. A motor mellett általában van még reduktor és enkóder. [8]

41. ábra: Motor és enkóder [1] egyesítése

10.2. H-híd

A legtöbb alkalmazásnál két dolgot akarunk megvalósítani a motorokkal:
Előre és határa felé lehessen meghajtani és változtatni lehessen a sebességét.
H-híd kapcsolásra van szükségünk, hogy a motort meg tudjuk hajtani előre és hátrafelé.
A 42. ábra szemlélteti a H-híd felépítését, ami arról kapta a nevét, hogy hasonlít a „H” betűre. Láthatjuk, hogy ha az 1-es és a 2-es kapcsoló van bekapcsolva, óramutató járásával megegyező irányban forog a motor, ha 3-ast és a 4-est, akkor pedig óramutató járásával ellentétes irányban fog forogni, járásával megegyező irányban forog a motor, ha 3-ast és a 4-est, akkor pedig óramutató járásával ellentétes irányban fog forogni.



42. ábra: H-híd felépítése [1]

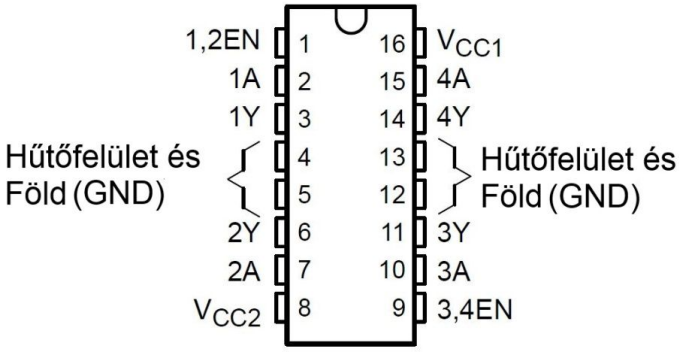
Mivel a mikrovezérlő nem képes annyi áramot leadni, ami direkt meghajtaná a motorokat, ezért műveleti erősítőket használunk a motorok meghajtására. A műveleti erősítők a gyenge áramú vezérlő jeleket, motorokat meghajtó nagy áramú jelekké alakítják át.

A jelek felerősítésére L293B-as motor vezérlő IC-t használtam, amely dupla H-híd kapcsolásnak felel meg. Elektronikus kapcsolókat használ, amelyek az IC megfelelő lábaira adott 0V-os vagy 5V-os vezérlőjelekkel nyithatók és zárhatók. Ezáltal a H-híd IC könnyedén vezérelhető a mikrovezérlővel. Az L293B két H-hidat tartalmaz, ezért egyetlen IC elég a robot két motorjának a vezérléséhez. [10]

Forgásirány:

INPUT1	INPUT2	FORGÁSIRÁNY:
Alacsony (0 V)	Alacsony (0 V)	STOP
Magas (5 V)	Alacsony (0 V)	ELŐRE
Alacsony (0 V)	Magas (5 V)	HÁTRA
Magas (5 V)	Magas (5 V)	GYORS STOP

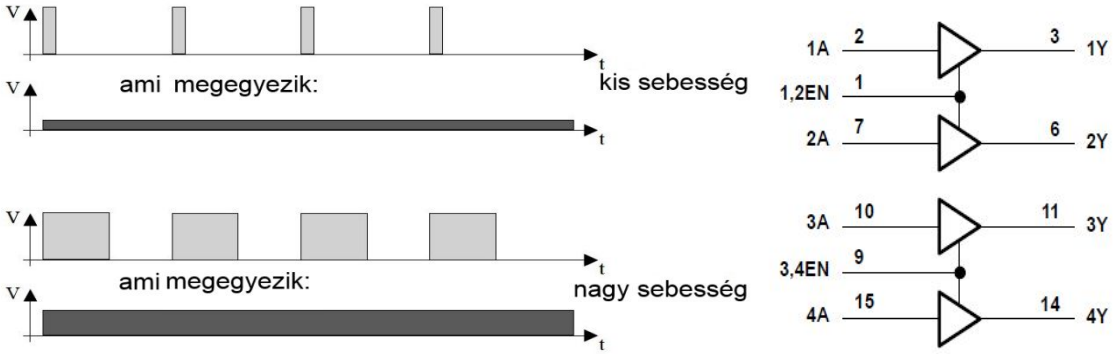
2. táblázat: Az L293B motorvezérlő IC vezérlési táblázata [10]



43. ábra: L293B motorvezérlő IC [16]

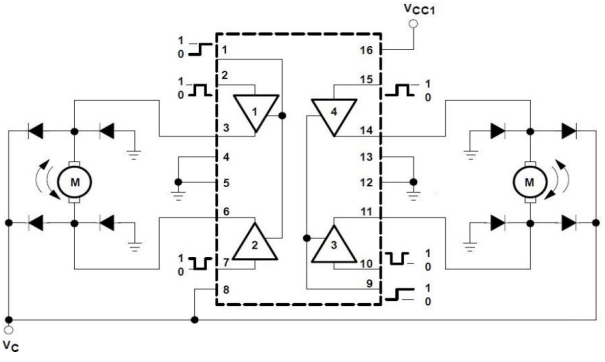
10.3. PWM (impulzus szélesség moduláció)

A PWM az analóg tápkörök elkerülésére szolgáló módszer, mely kihasználja azt a tényt, hogy a mechanikai rendszerek bizonyos késéssel működnek. Egy folyamatos kimeneti jel helyett teljes rendszerfeszültségen (pl. 5V) működő (digitális) pulzusokat generál egy fix frekvencián – pl. 20kHz, így kívül esik az emberi hallástartományon. Az impulzus szélességének szoftveren keresztüli változtatásával tudjuk irányítani a kimenő jelet (jel erősségét). A PWM által létrehozott pulzáló jel által kitöltött jel és nem-jel tér értéke határozza meg a kimeneti jel összesített nagyságát, ami esetünkben a hajtott motor sebessége. [8]



44. ábra: PWM különböző kitöltési tényezőkre (bal oldalt) [1], és Az L293B logikai felépítése (jobb oldalt)[16]

Ezt az elvet használjuk az L293B motorvezérlő IC vezérlésénél is. Ahogy a logikai kapcsolásból látszik, a motorvezérlő IC rendelkezik két Enable bemenettel, amelyeket PWM-mel vezéreljük. Ez által szabályozzuk a motor sebességét. A motor induktív jellegű fogyasztó, és működhet generátoros üzemmódban is, ezért inverz párhuzamos diódákkal el kell vezetni a generált feszültséget. Mivel az L293B motorvezérlő IC nem tartalmaz ilyen diódákat, ezért ezeket kívülről kellett betenni, a következő kapcsolás szerint.

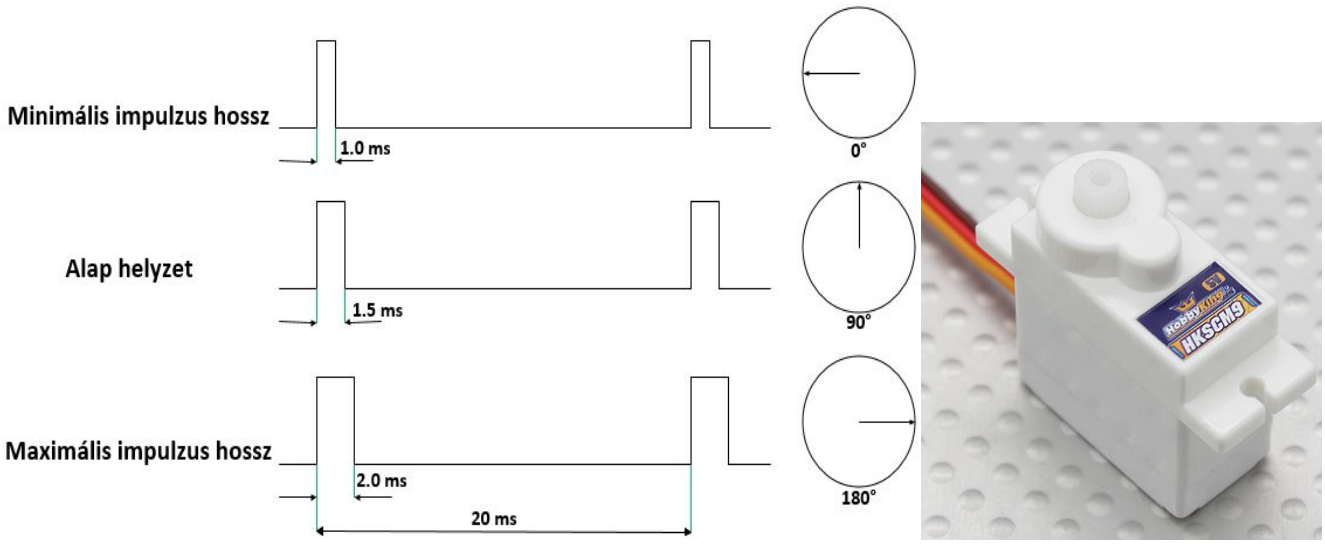


45. ábra: az L293B motorvezérlő IC bekötési rajza

10.4. Szervó motorok

A szervó motorokat régebb óta alkalmazzák több helyen. Kis mérettel rendelkeznek, de méretükhöz képest nagy erőt képesek kifejteni. Három fő részből áll: kis méretű DC motor, potenciométer , és egy szabályozó áramkör. A motor egy reduktor van rakva, amely növeli a motor nyomatékát. A motor forgása változtatja a potenciométer ellenállását, így a szabályozó kör precízen betudja állítani, hogy mennyit mozduljon a motor, és melyik irányban. Amikor a motor elérte a kívánt pozíciót, akkor megszűnik a motor tápellátása, így megáll a motor. A legtöbb szervó motor 180° -os elfordulásra képes. A null pozíciója ahol ugyan annyit tud balra és jobbra fordulni (90-ban). A motor pozícióját úgynevezett jel vezetéken adjuk meg, amelyen elektromos impulzusokat küldünk. Ezeknek az impulzusoknak 20 ms a periódus ideje, és a pozíciót a kitöltési tényező hosszával határozzuk meg. Például ha 1.5 ms akkor 90° -ban lesz a motor pozíciója. A motor sebessége a jelenlegi és a kívánt pozíció közötti távolságtól függ.

A munkámban a szervó motort a kamera forgatására használtam fel. Nagy előnye, hogy pontosan tudhatjuk milyen pozícióban van a tartó a robot testéhez viszonyítva, összesen 180° -ot tud fordulni (a null pozícióhoz ±90°). A pozícionálásának sebessége is megfelelő volt. A szervó motor egy külső 1:1-es áttételen keresztül forgatja a kamerát. Előnye, hogy így a kamera tartó súlya nem terheli a motort.



46. ábra: A pozíció beállítása impulzus szélességgel (bal oldalt), szervó motor (jobb oldalt)

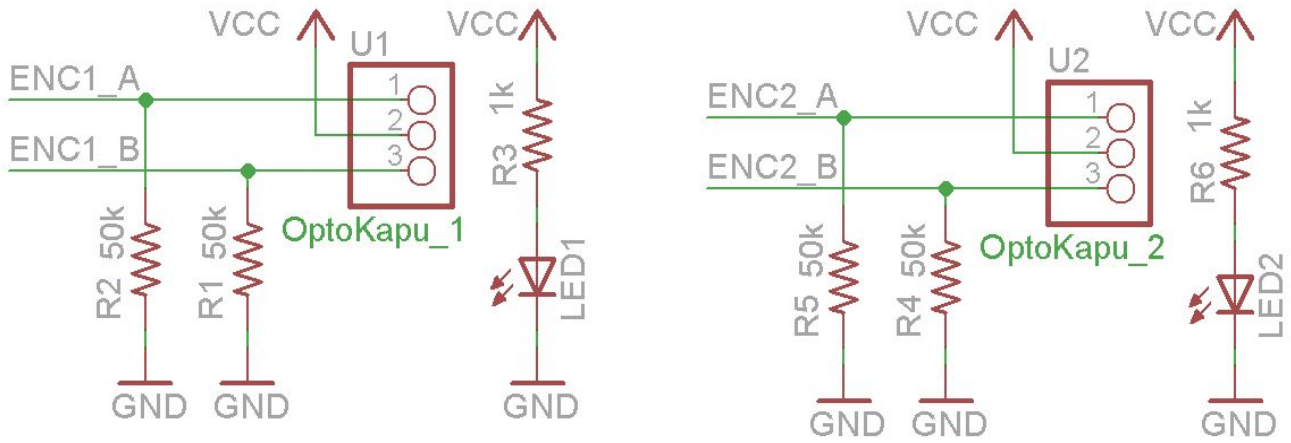
11. A robot megépítése

A robot vázát, motorokat és a kereket (két meghajtó és egy passzív) készen vettem. Mivel voltak szempontok, amiket nem elégítettek ki a készen vett dolgok, és még szükségem volt más kiegészítő eszközökre a robot működéséhez, ezért a következő feladatok elé néztem:

- enkóderekhez külön elektronikát tervezni és építeni
- mikrovezérlő lap mellé egy saját fejlesztésű lap (elektronikus megtervezése, maratása, alkatrészek felforrasztása)
- másik passzív kerék megépítése
- a robot megépítése
- akkumulátor töltő megépítése
- a meghajtó kerekeket modellezni, hogy ki lehessen cserélni a gumikat

11.1. Az enkóderes lap elektronikája

Tanulmányoztam a régi számítógépes egereknek az eredeti áramkörét, és annak alapján raktam össze próba lapon az áramkört. Az eredetihez képest annyival változtattam meg az áramkört, hogy megnöveltem a lehúzó ellenállást 13k óhmról 50k óhmra, mivel így a logikai szintek jobban elkülönülnek, és a TTL logikai szinteknek is megfelel. Az enkóderek által leadott jelet, 6 érrel rendelkező vezetékkel továbbítom az illesztőkártyára. Az enkóderes lap elektronikájában csak furatba szerelt alkatrészeket használtam.



47. ábra: Az enkóderes lap kapcsolási rajza

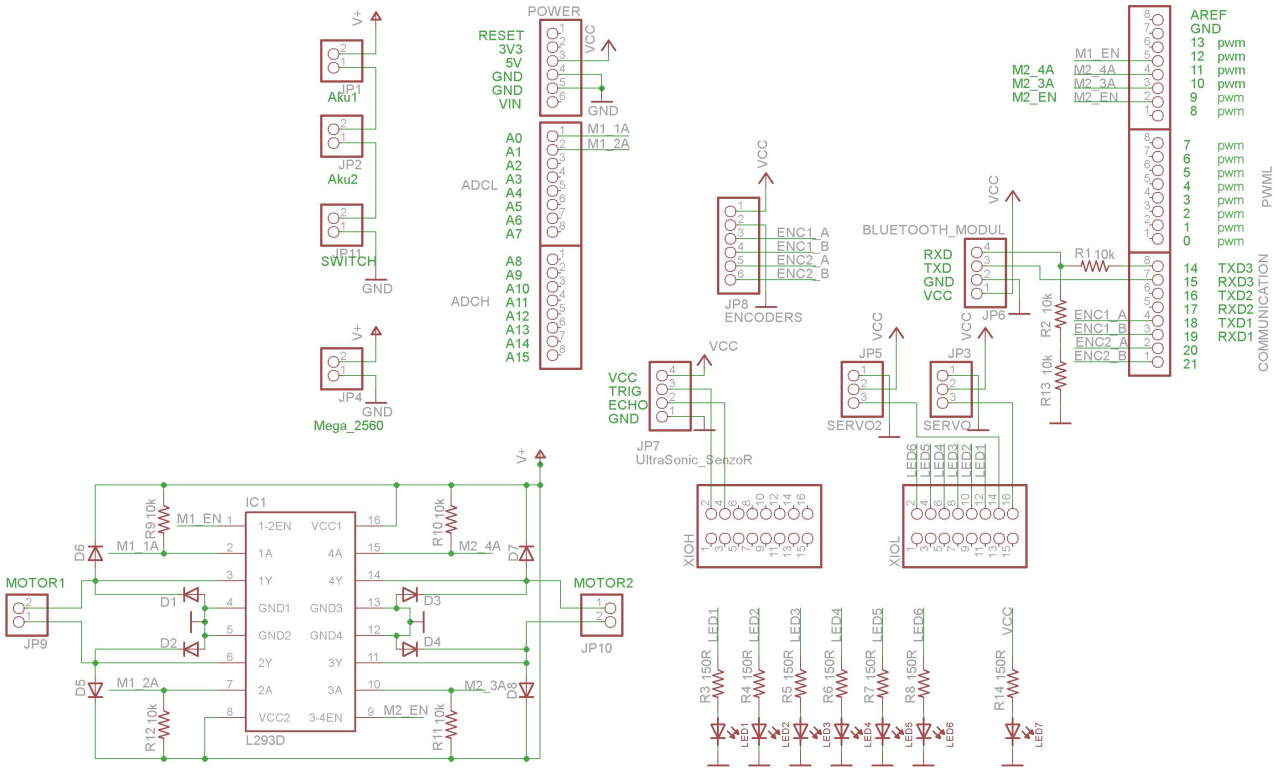
11.2. Az illesztőkártya

A mikrovezérlő áramköre mellé szükséges volt egy másik áramkör, amely egyszerűen csatlakoztatható rá, valamint a következő elemekkel kapcsolja össze a mikrovezérlőt:

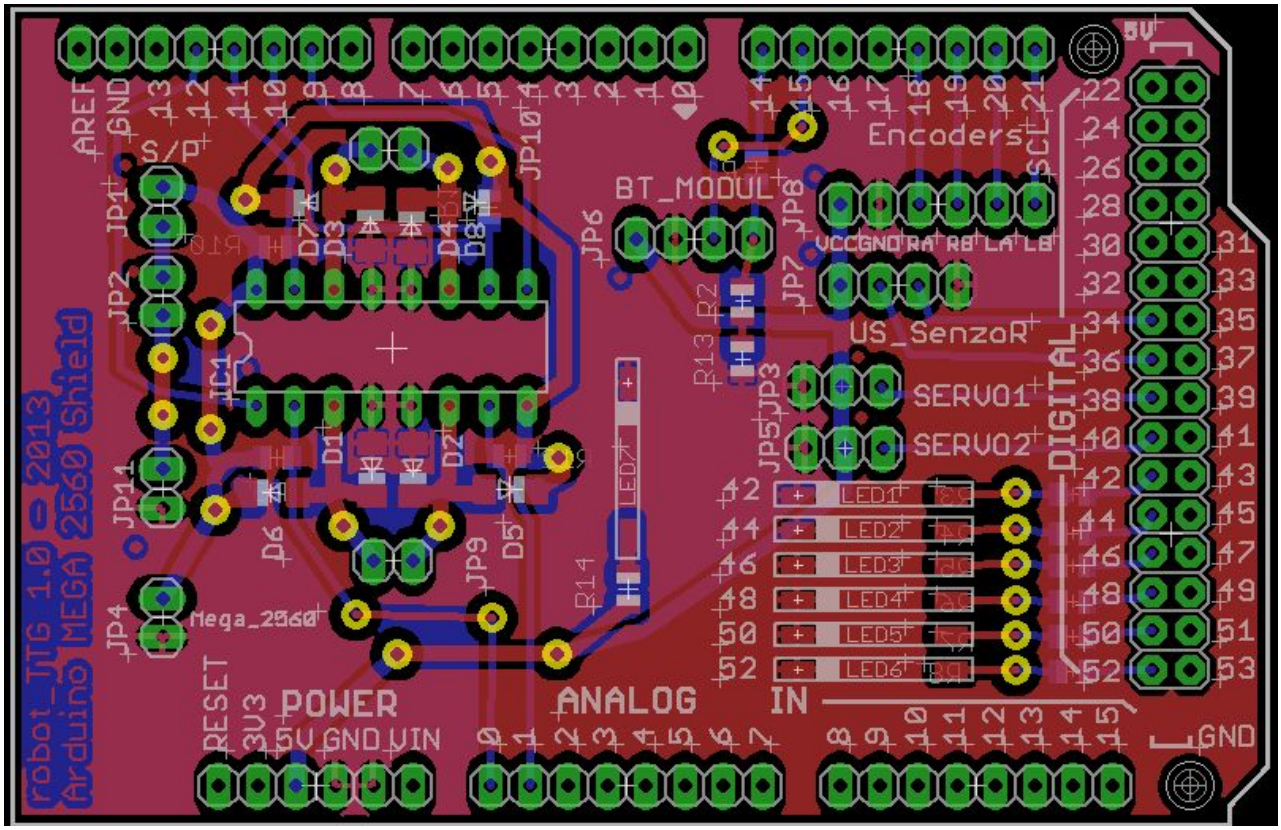
- Motorvezérlő IC (L293B), motorok
- JY-MCU Bluetooth Wireless Serial Port Module (HC-06)
- Enkóderes lap
- Akkumulátorok
- Státusz LED-ek (hibakereséshez)
- 2 szervó motor és egy ultrahangos távolság mérő, amelyek nem lettek bekötve
- Az illesztőkártya megtervezése

Az illesztőkártyát az Arduino Mega2560-as áramkör lap csatlakozóira méreteztem, amely így az áramkör lapjához egyszerűen illeszthető.

Mivel maga az Arduino kapcsolási rajza, és nyomtatott áramköre is az interneten ingyenesen elérhető, ezért annak alapján dolgoztam.



48. ábra: Az illesztőkártya kapcsolási rajza



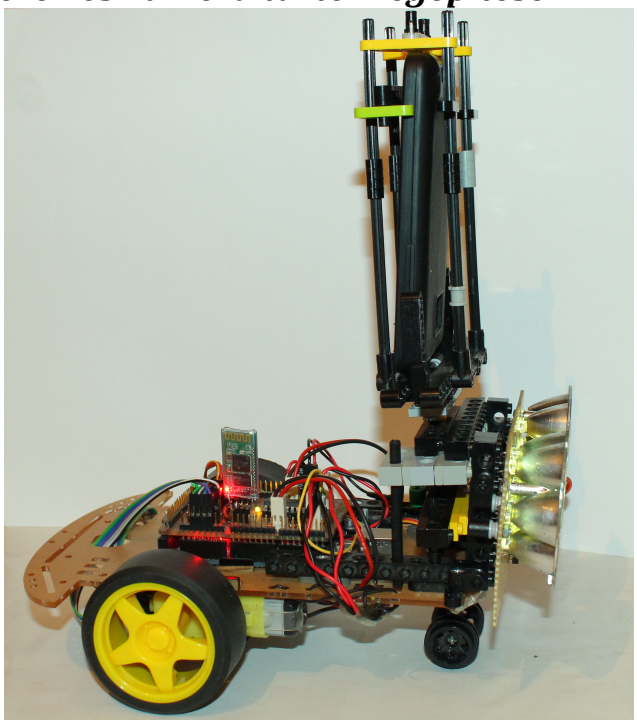
49. ábra: Az illesztőkártya nyomtatott áramköre (PCB)

A lap megtervezéséhez Eagle 6.5.0. laptervező program ingyenes verzióját használtam.

11.3. Az illesztőkártya fizikai kivitelezése

Az PCB megtervezése után, átlátszó fóliára ki lettek nyomtatva a megfelelő rétegek. 15 percig UV fénnyel lett megvilágítva, ami a maszkolatlan lakk réteget meggyengítette. Ezután a maszkolt rétegek előhívása, valamint a felesleges rézfelületek lemaratása következett. Az első lépésként lúgos vízbe lett rakva, ami a legyengítette lak réteget. Második lépésként csapvízbe lett megmosva, ami megtisztította a lapot. A harmadik lépésként pedig vaskloridba lett belerakva, amelynél figyelni kellett, hogy mennyire marta le a maratni való réz réteget. Miután megfelelően lemarta, újból csapvízbe meg lett mosva. Következő lépésként újból UV fény alá lett rakva 15 percig, ami a fennmaradt lak réteget gyengítette meg, majd a maradék lakkréteg is le lett maratva. Ezután a beültetési réteg került a lapra laminálós technikával. Végül ki lettek fúrva a lyukak rajta, és be lettek forrasztva a viák. Utolsó lépésként fel lett forrasztva az összes alkatrész.

11.4. Másik passzív kerék és kamera tartó megépítése

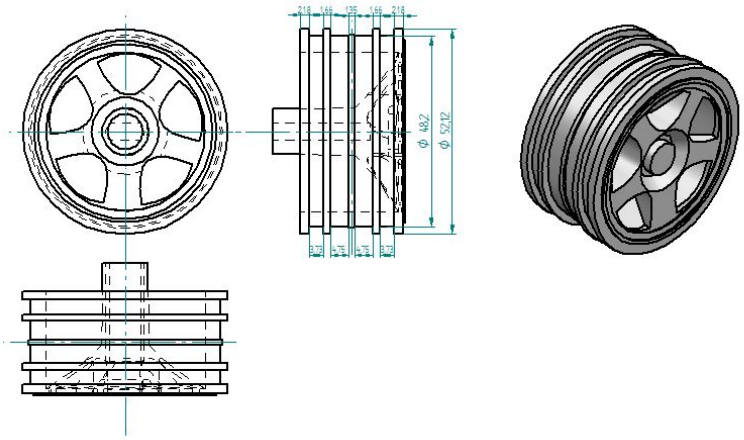


50. ábra: A LEGO kockából megépített passzív kerék

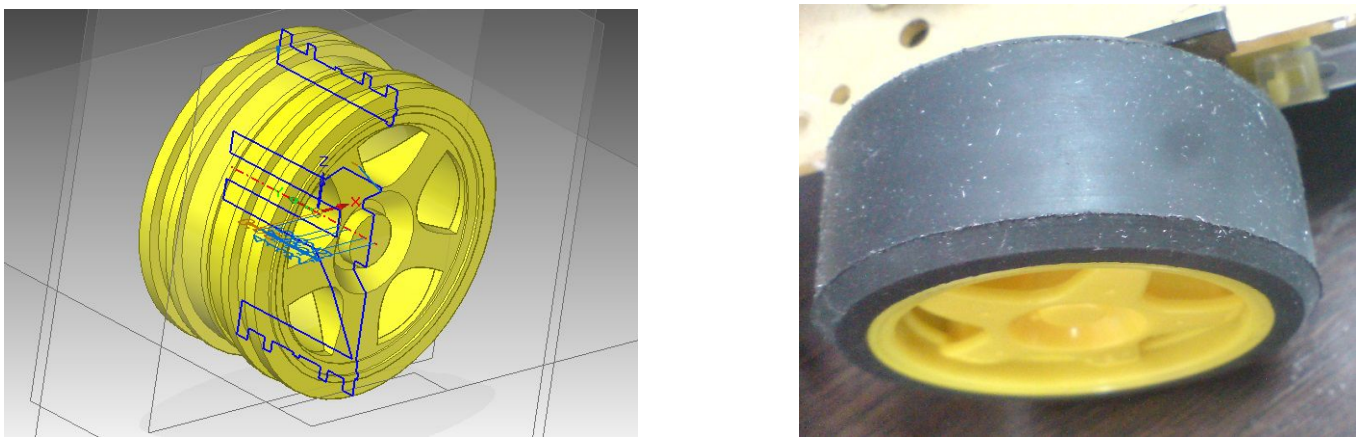
Ahhoz, hogy az irányítás megfelelően működjön, a robot vázának vízszintesen kell állnia. Amikor először összeállítottam a robotot, a váza nem volt vízszintes, ezért szükségem volt egy másik passzív kerékre. Ez mellett még egy olyan tartó szerkezetet kellett összerakni, amely biztonságosan tartja a mobiltelefont, képes forgatni a függőleges tengelye körül. LEGO kockákat használtam fel erre a célra, mivel könnyen lehet változtatni a konstrukción, és csavarokkal is megfelelően fel lehet őket fogatni a vázra.

11.5. Meghajtó kerekek modellezése

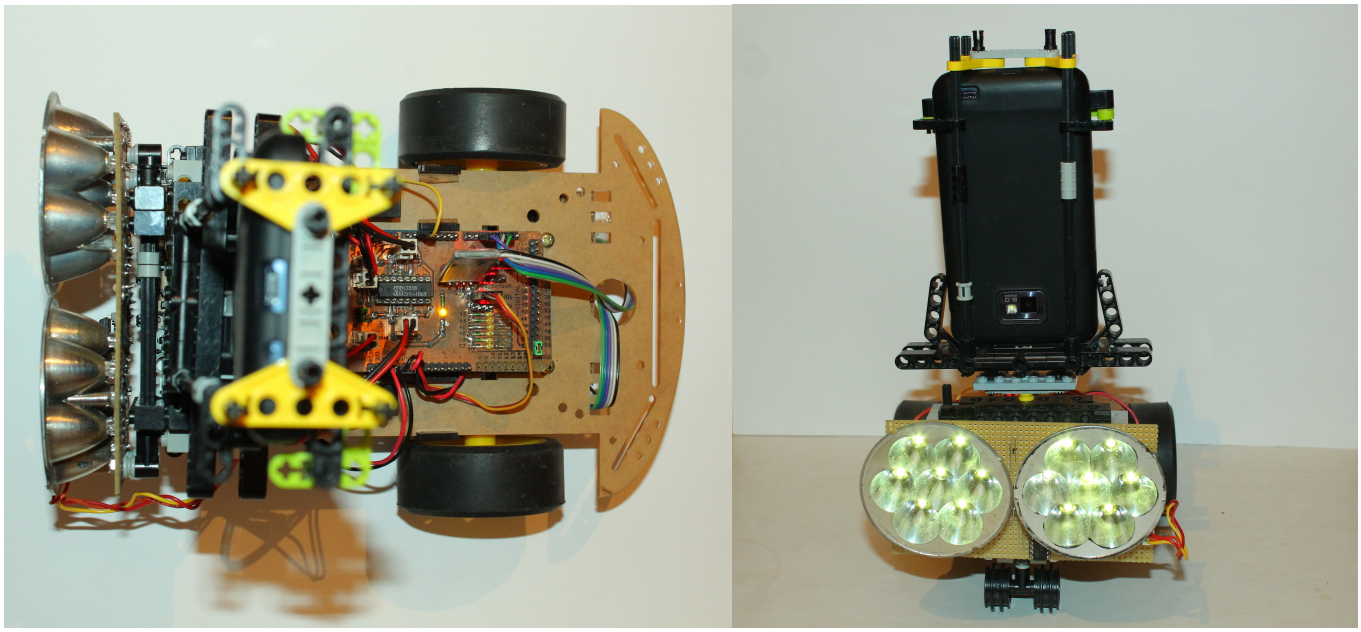
A megrendelt kerekeken a gumik nagy százalékban műanyagot tartalmaztak, ami miatt a kerekek tapadása igen rossz volt. Csúszós talajon kapart amikor megindult, és ez mellett nem állt meg rögtön, amikor elkezdett fékezni a kerék. Ahhoz, hogy az irányítás jól működjön, szükséges a jó tapadás. Egy gumitechnikus személyre szabott szerszámot készített a gumihoz, ami a kerékre lett szabva. Ahhoz, hogy ezt elkészítse, modellezni kellett a kerék dimenzióit. A kiöntött gumi nagy százalékban tiszta kaucsuk, ami miatt rugalmas, és igen jó tapadással rendelkezik. A 3D modellezéshez a Solid Edge ST6 ingyenes diák verziót használtam.



51. ábra: A robot kerekének technikai rajza



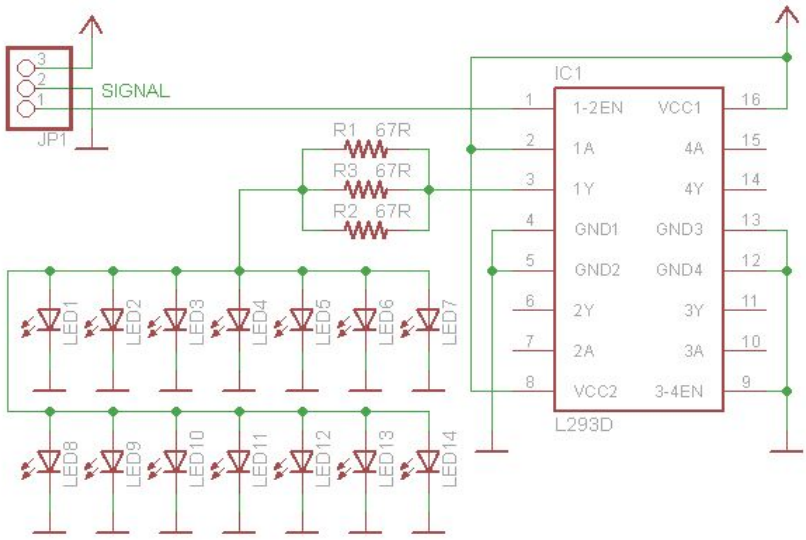
52. ábra: A kerekének 3D modellje (bal oldalt), kiöntött gumi a kerekén (jobb oldalt)



53 ábra: A robot felülnézetből (bal oldalt), a robot elölnézetből (jobb oldalt)

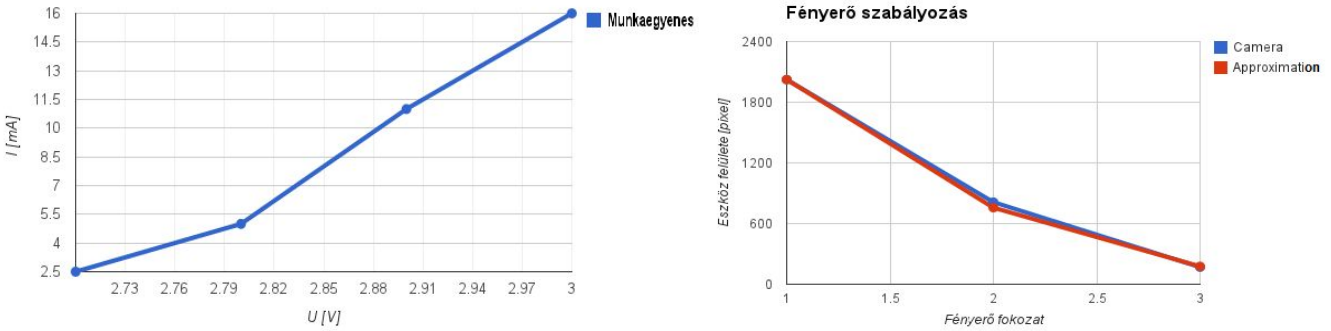
11.6. Aktív megvilágítás

Ahhoz, hogy a képfeldolgozás közben a eszközök ugyan olyan színűek legyenek, fontos, hogy homogén legyen a tér fényerősség szempontjából. Ez okból egy aktív megvilágítást biztosító LED-es lámpa áramkörét terveztem és építettem meg, amelynek a fényerejét mikrovezérlő képes szabályozni. Ez a szabályozás PWM segítségével valósul meg.



54. ábra: A LED-es lámpa sémája

A lámpa áramköre 14 LED-ből, 3 darab 67 óhmos ellenállásból, egy műveleti erősítővel rendelkező IC-ből áll.



55. ábra: A LED-ek mukaegyenese(bal oldalt), és az automatikus fényerő szabályozás megközelítő egyenese (jobb oldalt)

A lámpába SENCART F5 IF-20mA 2160MCD tipusú LED-eket használtam, amelyek üzemi feszültsége 3.0-3.2V között van. A szín hőmérséklete: 6000-6500K közötti, ami sárgás fehérnek felel meg. Több mérést végeztünk, ami alapján meghatároztuk a LED munkaegyenesét. Ezek a mérések azért voltak fontosak, hogy tudjuk, hogy melyik az a feszültség/áram arány, ami megfelel fogyasztás és fényerő szempontjából. Az ellenállás meghatározásával 100 mA-ert fogyaszt a 14 LED maximálisan, ami elfogadható, a motorok fogyasztása mellett.

A LED-ek egy motormeghajtó IC-vel vannak szabályozva, ami lehetővé teszi, hogy szabályozni tudjuk a fényerejüket. Ez egy kifejezetten fontos dolog, mivel ha az eszköz közel van a robothoz és a lámpa fény ereje túl erős, akkor a robot az eszközt színeit fehérnek látja. A munka folyamán ki lett dolgozva egy olyan algoritmus, ami segítségével a robot a lámpa fényerőt automatikusan változtatja annak függvényében, hogy milyen messze van az eszköz a robottól. Itt 3 mérést végeztem, és megnéztem, hogy melyik fényerőnél lesz a képfeldolgozásnak legjobb eredménye. Az eredményekből egy görbét rajzoltam, amelyet approximáltam egy saját görbével. Az görbe képletének inverz kiszámításával kapom meg, hogy mekkora fényerősség fokozat felel meg, az eszköz távolságához.

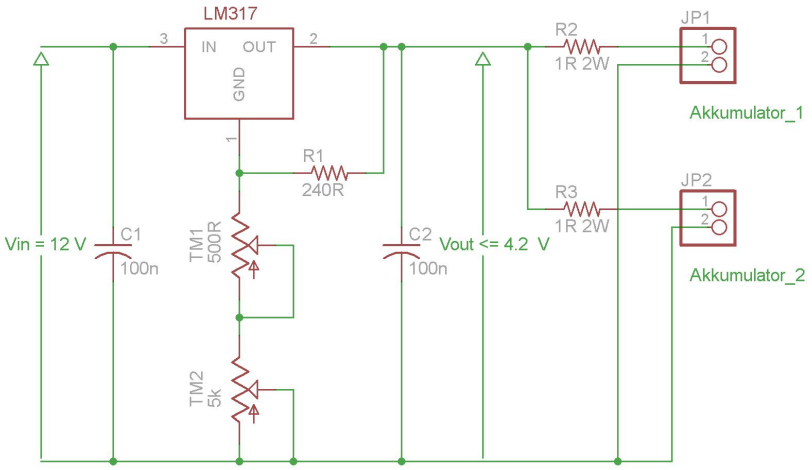
11.7. Az akkumulátor töltő lap

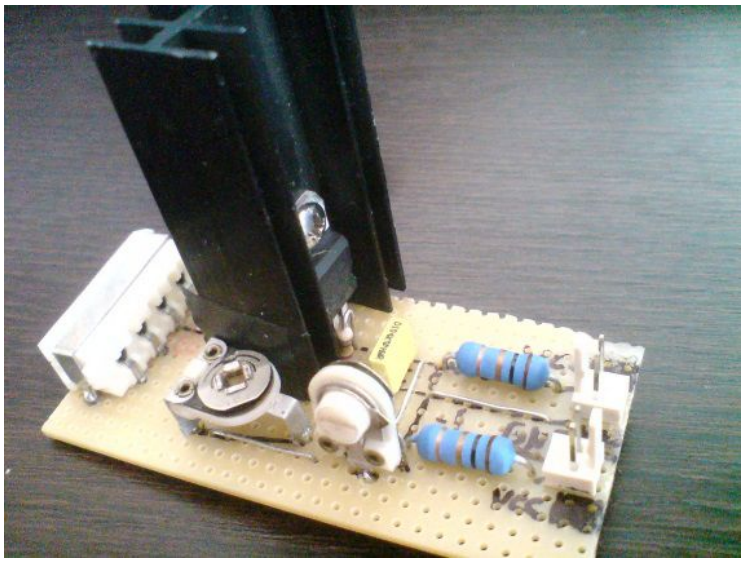
A robot két Lítium-ion akkumulátor soros kötésével van táplálva. Ha mindkét akkumulátor teljesen fel van töltve 8.4 V feszültséget kap, ami a motorokat direktben táplálja. Az Arduino lap is innen kap táplálást, de a lapon a feszültség stabilizátorral le van csökkentve a feszültség 5 V-ra.

A két akkumulátorhoz egy töltő is lett készítve, amely az LM317-es IC-t tartalmazza. A töltő felépítése nem komplikált, és alkatrészei olcsók.

A kimenő feszültséget a trimmerekkel lehet szabályozni. A sorba kötött 1 óhmos ellenállás segítségével tudjuk lemérni, hogy mekkora árammal töltjük az akkumulátorokat. Az áramkör úgy lett kialakítva, hogy a két akkumulátort párhuzamosan lehessen egyszerre tölteni.

Elejében nagyobb árammal töltődik az akkumulátor, ezért az ellenálláson is nagyobb feszültség esés. Ahogy fokozatosan növekszik az akkumulátor feszültsége, úgy csökken a töltő áram. E miatt jól megtudja az áramkör közelíteni az akkumulátor töltéséhez szükséges feszültség és áram változást.
Az áramkör 12 V-ról van táplálva, úgy lett kialakítva, hogy kompatibilis legyen egy PC-táp csatlakozójával.





56. ábra: Az akkumulátor töltő kapcsolási rajza (fent), az akkumulátor töltő fizikailag kivitelezve (lent)

A lapon a kapcsolási rajztól eltérően 10k óhmos trimmert használtam, de mindkét értékkel megfelelően működik az áramkör.

12. Szabályozás

A beágyazott rendszerek elengedhetetlenül fontos része, mely összeegyeztetjük az aktuátorokat és a szenzorokat egy vezérlő algoritmus szoftver segítségével.

A visszacsatolásnak köszönhetően (mely nem történik meg alapvetően az egyenáramú motoroknál) a nem ellenőrzött műveletek paraméterei ellenőrzés (szabályozás) alá kerülnek (fordulatszám, pozíció). [8]

A dolgozatban a szabályozás fő célja a differenciál hajtással rendelkező robot egyenes vonalú mozgásának megvalósítása.

12.1. PID szabályozás

Komolyabb, és az iparban is sok területen használt szabályozás a PID szabályozás. Három tagból tevődik össze: proporcionális (P), integráló (I) és differenciáló (D).

12.2. P (proporcionális) szabályozó

Több szabályozó alkalmazásnál a két állapot közötti hirtelen váltás nem ad megfelelő eredményt. Ezt úgy tudjuk javítani, ha egy lineáris (proporcionális) tagot alkalmazunk. A proporcionális (P) szabályozónak következő a képlete:

R(t) a motor kimenete a t pillanatban

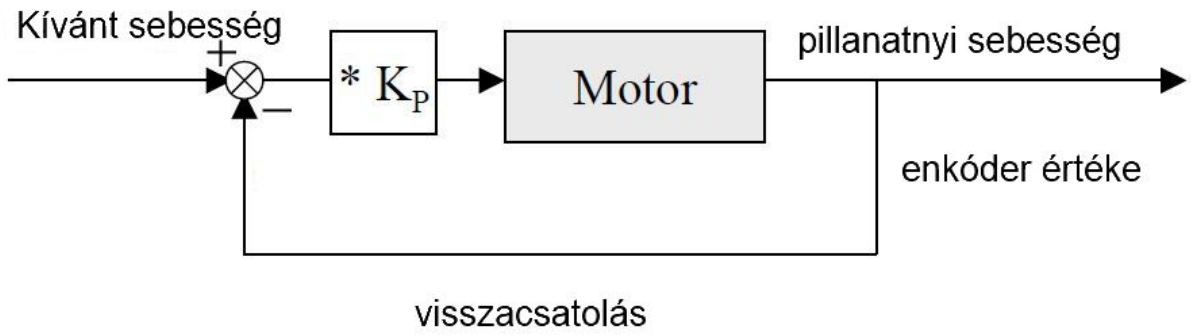
$v_{pill}(t)$ *sebesség a t időpillanatban*

$v_{kiv}(t)$ *kívánt sebesség a t időpillanatban*

Kp konstans szabályzó érték

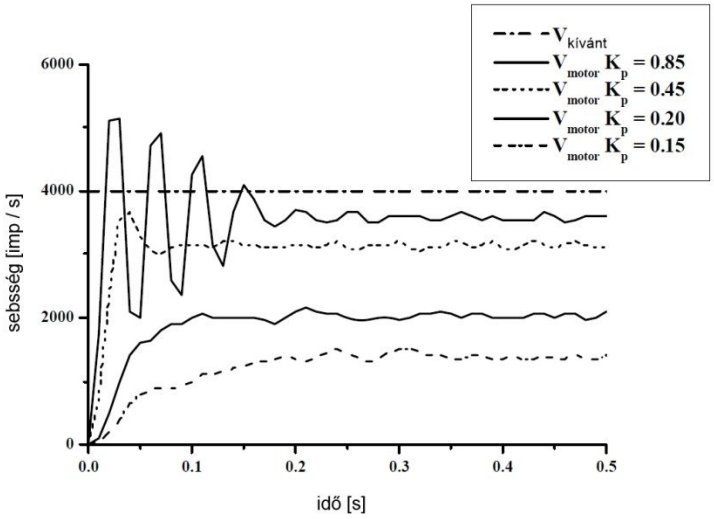
$$R(t) = K_p \cdot (v_{kiv}(t) - v_{pill}(t))$$

A kívánt és a pillanatnyi érték közötti különbséget hívjuk hibának. A 57. ábrán láthatjuk a P szabályzó blokk sémáját.



57. ábra: Az P szabályozó [1]

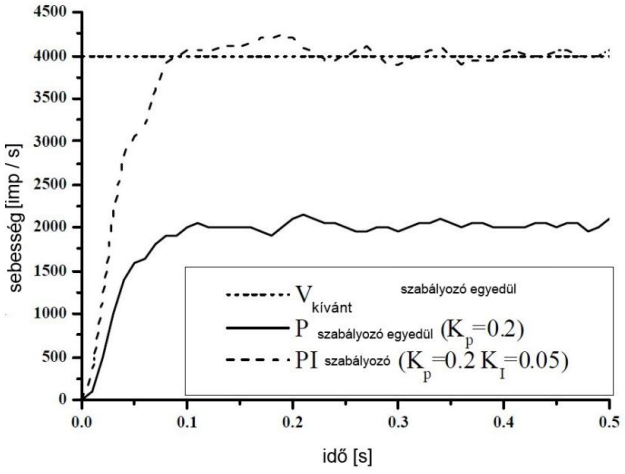
Minél nagyobb a K_p, annál gyorsabban a reagál a szabályzó. Vigyázni kell, mert ha túl nagy a K_p oszcillálni fog a rendszer. Így olyan K_p-t kell választanunk, amely elég gyorsan reagál, de ez mellett ne hozza oszcillációba a rendszert.
Ha P tagot használunk egyedül, akkor nem fogunk soha pontos eredményt kapni, hanem csak megközelítőt. [1]



58. ábra: Az Kp értékű P szabályozó válaszai egységugrás függvényre [1]

I (Integráló) szabályozó

A P szabályzóval ellentétben az I szabályozót keveset használják egyedül, legtöbbször P, vagy PD szabályozókkal kombinálják. Az I szabályzó lényege, hogy a P szabályozónál fellépő egyensúlyi hibákat csökkentse. Egy addicionális integrál kifejezéssel pedig ez akár nullára is csökkenthető. [8]



59. ábra: Az I szabályozó válaszai egységugrás függvényre [1]

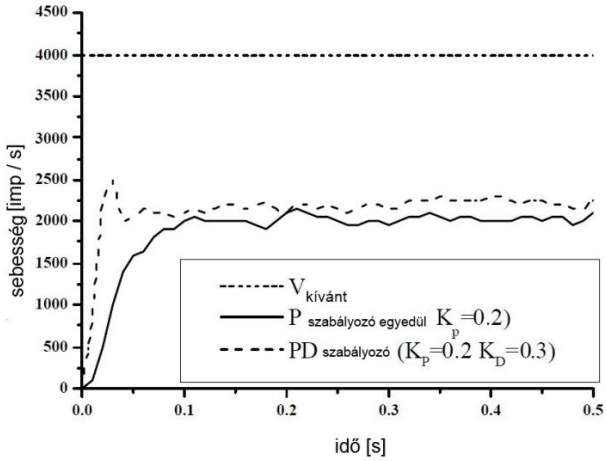
Ha az e(t) a t időpillanatban a hiba, akkor a PI szabályozónak a következő a képlete:

$$R(t) = K_p \cdot [e(t) + \frac{1}{T_I} \cdot \int_0^t e(t)dt]$$

Ha $Q_I = K_p / T_I$ akkor a következő egyenletet kapjuk:

$$R(t) = K_p \cdot e(t) + Q_I \cdot \int_0^t e(t)dt$$

D (differenciáló) szabályozó



60. ábra: A P és PD szabályozók válaszai egységugrás függvényre [1]

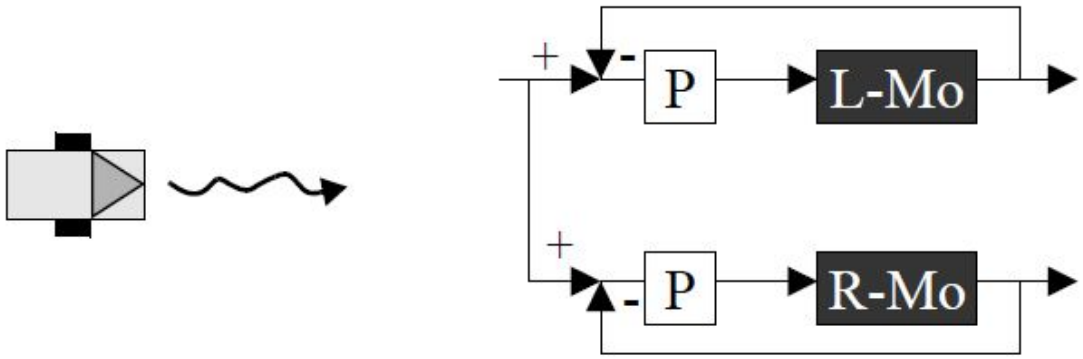
Az I szabályozóhoz hasonlóan a D szabályozót ritkán használják egyedül, általában kombinálják P, vagy PI szabályozókkal. A származtatott (derivált) kifejezés hozzáadása a P szabályozó válaszidejének (a bemeneti jelre való reakcióidejét tekintve) felgyorsítására szolgál. Ahogy az összehasonlító ábrán is látszik (60. ábra), az sokkal előbb eléri az egyensúlyi állapotot, de továbbra is megmarad az egyensúlyi hiba. [11]

12.3. Több motor összehangolása, egyenesen haladás céljából

Azoknál a robotoknál, ahol a két külön motor végzi a meghajtást és a kormányzást, ott nem nehéz egyenesen, vagy bizonyos íven kormányozni. A differenciál hajtásnál ez nem így van. Mivel itt két motort hajtunk meg egyszerre, a motorok aszimmetriája miatt a robot nem fog teljesen egyenes pályán haladni. A motorokat úgy kell szabályozni, hogy megtegyék a megfelelő impulzus számot mintavételezésenként, és ez mellett össze is legyenek hangolva. [1] E feladat megoldását többféleképpen is meg lehet közelíteni.

12.4. Motorok egymástól független szabályozása

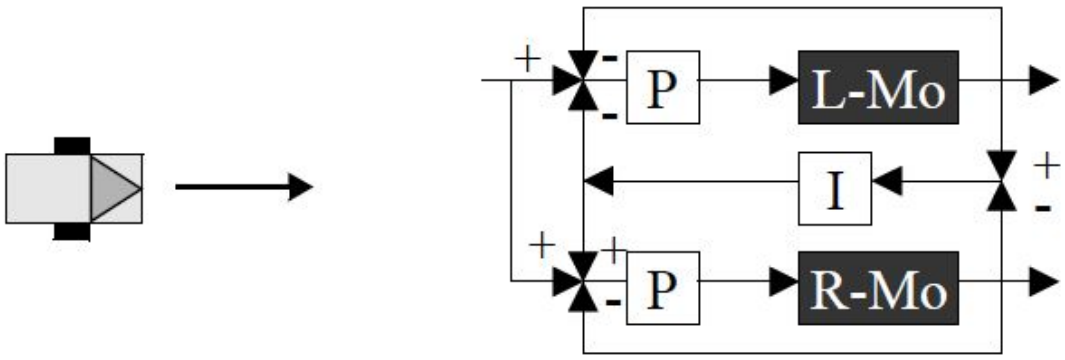
Azoknál a robotoknál, ahol a két külön motor végzi a meghajtást és a kormányzást, ott nem nehéz egyenesen, vagy bizonyos íven kormányozni. A differenciál hajtásnál ez nem így van. Mivel itt két motort hajtunk meg egyszerre, a motorok aszimmetriája miatt a robot nem fog teljesen egyenes pályán haladni. A motorokat úgy kell szabályozni, hogy megtegyék a megfelelő impulzus számot mintavételezésenként, és ez mellett össze is legyenek hangolva. [1] E feladat megoldását többféleképpen is meg lehet közelíteni.



61. ábra: Egyenesen kormányzás első próbája [1]

12.5. Motorok összehangolt szabályozása

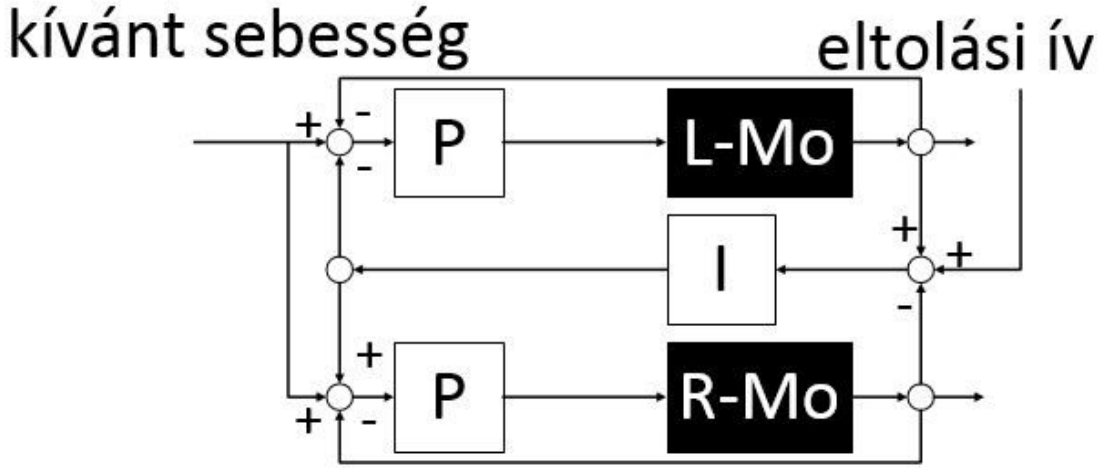
Az előző megoldás kiegészített változatát a 62. ábrán láthatjuk. Itt a motorok közötti elmozdulást is számításba vesszük és visszacsatoljuk ezt a P szabályozóval egy I szabályozón keresztül. Az I szabályozó integrálja, vagy összegezi a hibát, amit a P szabályzó nem vesz figyelembe. Megfigyelhetjük az ellentétes előjeleket az I bemenetén és kimenetén. [1]



62. ábra: Az egyenes kormányzás második próbája [1]

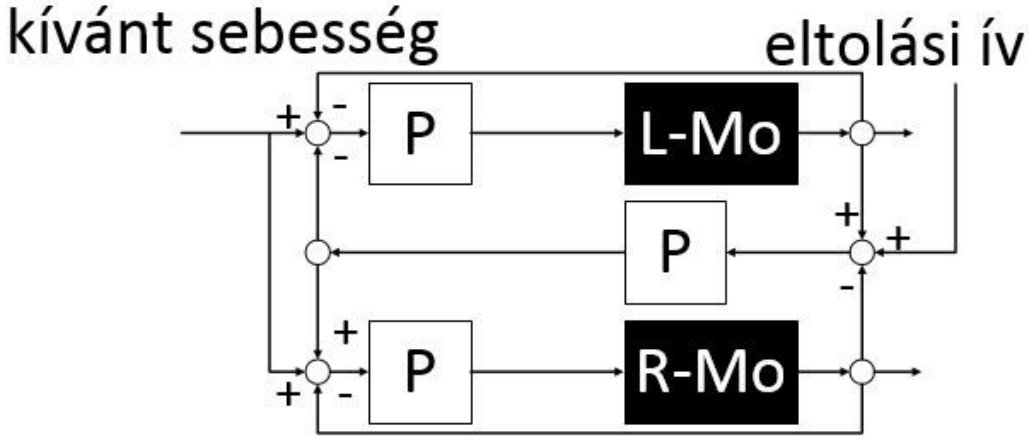
12.6. Motorok összehangolt szabályozása kiegészítve

A végleges verziója a 63. ábrán látható (Jones, Flynn, Seiger, 1999 [13]), ahol még egy bemenettel ki lett bővítve. Ilyen módon képesek vagyunk, nem csak egyenesen haladni, hanem egy bizonyos ívben is. Ha az eltolási ív 0-ával egyenlő, akkor egyenesen fog haladni a robot. Ha pozitív, vagy negatív az eltolási ív, akkor óramutató járásával megegyezően, vagy ellentétesen fog haladni. [1]



63. ábra: Az egyenes kormányzás harmadik próbája [1]

A munkámban, mivel kis mintavételezési frekvenciával dolgoztam (20 Hz) az enkóderek kevés megszakítása miatt, ezért fontos volt, hogy gyorsan reagáljon a szabályzó kör.
Ezt a szabályozó algoritmust annyival módosítottam, hogy I-t, hanem P szabályozót használtam a motorok összehangolásához (64. ábra). Ez azért volt előnyösebb, mert gyorsabban reagált a kerekek között különbségre, és a kamera által könnyebben lehet ívben irányítani.



64. ábra: Az egyenes kormányzás végleges verziója

12.7. Abbé Hiba

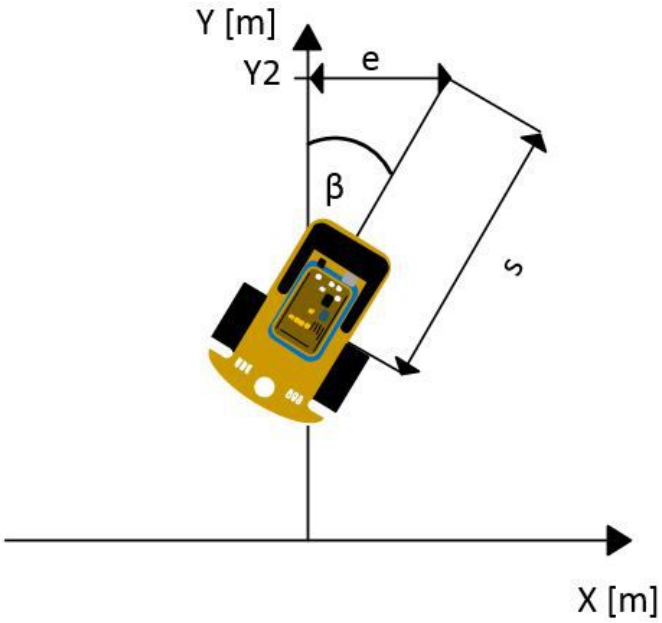
Nem csak a motorok sebességének összehangolását kellet keresni, hanem más hibára is oda kellett figyelni. Attól még, ha összesen ugyanannyi megszakítást is generált a két motor, nem garantálja, hogy a kívánt pozíción leszünk. Példaképpen, ha elején az egyik motor valamennyivel lassabb, akkor a robot kissé elfordul egy szögben, és a végpontnál a hiba a következőképpen számíthatjuk ki:

e - a hiba értéke a végpontban

β - az elfordulás szöge

s - a fennmaradt úthossz

$$e = s \cdot \sin(\beta)$$



65. ábra: Abbé hiba

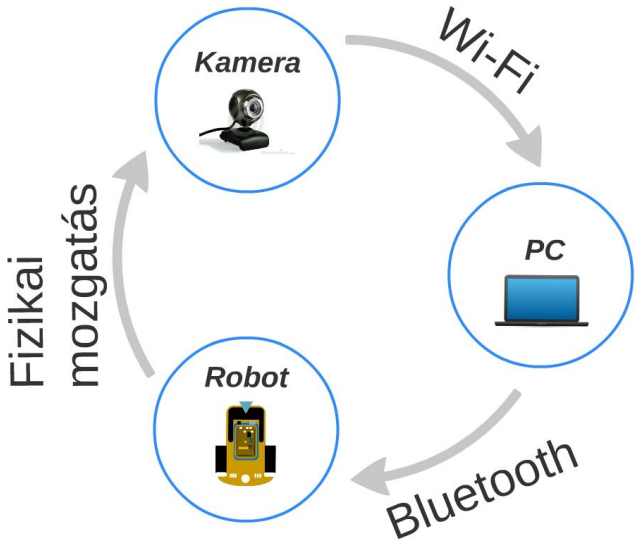
Azt láthatjuk, hogy a hiba nagyságát nem csak az elfordulás szöge befolyásolja, hanem a fennmaradt út hossza is.

Ezek szerint, ha az elején egy kis elfordulási szög megjelenik, akkor ha egyenesen is halad tovább a robot, a hiba egyre csak növekedni fog. Tegyük fel, ha a 10 cm után történik egy 10° -os elfordulás, ami a következő hibákat okozza: 1 méternél 15 cm.

2 méternél pedig 33 cm, ami pontos pozíciónál nem megengedhető. Ezt nevezik Abbé hibának. [14] A hibán az I szabályozó tud csökkenteni, vagy egy szenzor, amellyel megtudjuk határozni a globális koordinátákat.

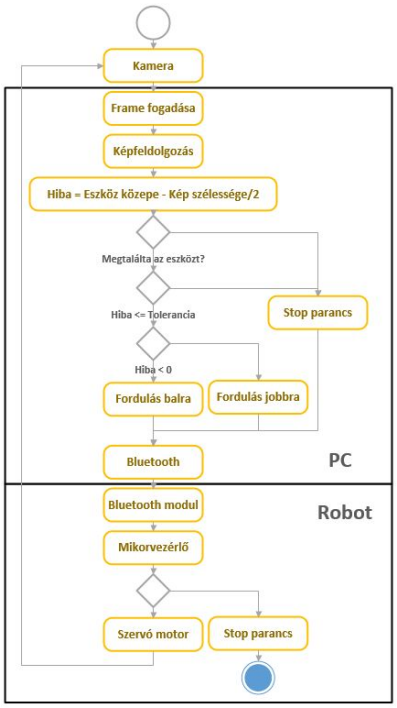
13. Robot és a képfeldolgozó program összehangolása

A folyamatot a 66. ábrán látható. A kamerával felvett képet elküldi a számítógépre Wi-Fi-n keresztül, a PC feldolgozza a képet, és parancsot küld a robotnak. Ez a folyamat ismétlődik másodpercenként többször. Fontos, hogy a folyamat végrehajtásának ideje minél kisebb legyen, hogy a reakció időt csökkentsük.



66. ábra: Robot és a képfeldolgozó program összehangolása

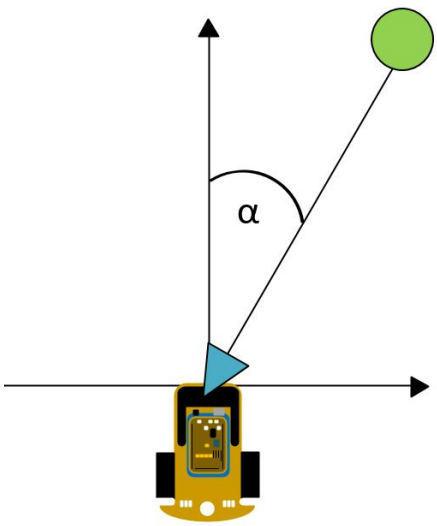
13.1. Kamera irányítása



67. ábra: A kamera irányításának UML diagramja

A kameraként egy mobiltelefon kameráját használtam, ami egy Android-os program segítségével IP kameraként működik.
Az Emgu CV lehetővé teszi, hogy egy IP kamera képét is feldolgozzuk, az által, hogy a megfelelő módon a kamerára csatlakozunk a programon keresztül.
A képfeldolgozás az előzőekben ismertetett folyamat alapján történik (szűrés, elhomályosítás, és transzformációk). Az eszköz közepének meghatározásának segítségével megkaphatjuk mekkora hiba van a eszköz közepe és a kamera kép közepe között. Ez a számítás segítségével megtudjuk határozni, merre kell a szervó motornak fordulnia ahhoz, hogy az eszköz a kép közepén legyen. A távolság segítségével a fordulás sebességét is meghatározzuk (ha távolabb van, gyorsabban fordul).
A program ezután leellenőrzi, hogy megtalálta-e az eszközt vagy a táblát. Ha nem talált, vár egy bizonyos ideig, és elkezd lassan fordulni a robot a saját tengelye körül az óra járásával ellentétesen.
Ha talált, és a képen az eszköz közepének pozíciója meghaladja a toleranciát, akkor elküldi a PC Bluetooth-on keresztül a robotnak az irányt és a sebességet, merre forduljon motor szervó és milyen sebességgel.
Miután fordult a szervó motor, csökken a hiba, és újra kezdődik a folyamat.
Ez a folyamat addig folyatódik, még le nem csökken a hiba a tolerancia alá, vagy még el nem tűnik az eszköz (túl megy rajta).

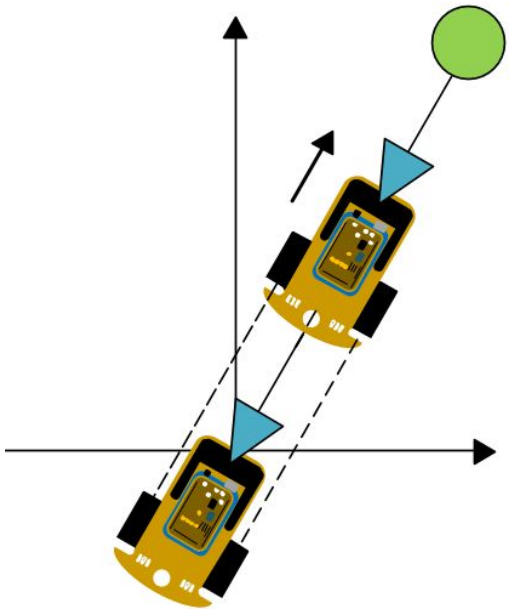
13.2. Robot irányítása



68. ábra: A robot kezdeti pozíciója

Mikor a kamera képének a közepén van az eszköz, a robot „megtalálta” az eszközt. Ez azt jelenti, hogy egy bizonyos toleranciával tudjuk, milyen szög van a robot és az eszköz között. Az 68. ábrán láthatjuk a robot kezdeti pozícióját, és a hibát, ami az α szöggel egyenlő. A PC irányítja a robotot Bluetooth-on keresztül, a visszacsatolást (hogy közelebb ért-e a célhoz) a kamera végzi. Az eszköz megközelítésére két módszert fejlesztettem ki.

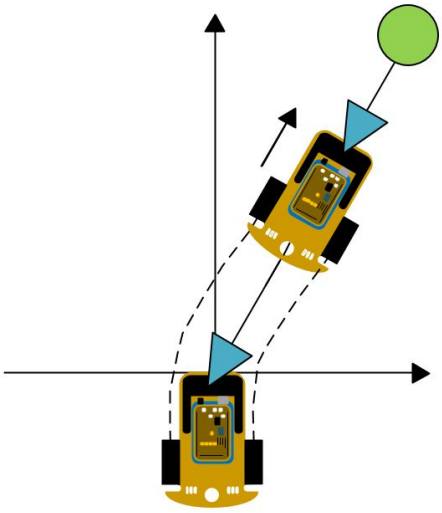
13.3. Az eszköz megközelítése kizárólag egyenes úton



69. ábra: eszköz megközelítése kizárólag egyenes úton

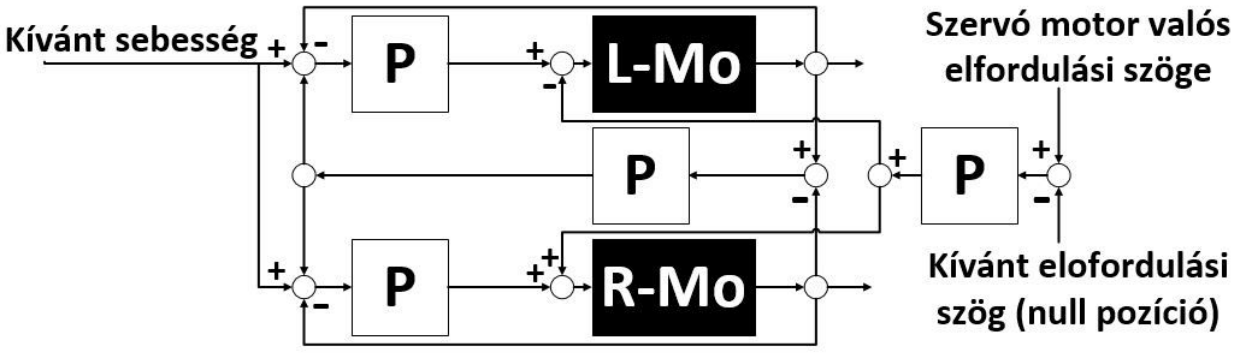
Az első módszer az egyszerűbb és a lassabb. Itt a robot, miután „megtalálta” az eszközt, helyben az α szöget igyekszik közel 0-ra csökkenteni. Ez úgy valósul meg, hogy helyben a két hátsó motor segítségével fordul a robot megfelelő irányban. Folyamatosan fordul, és végül a kamera és a robot között a bizonyos tolerancia alá csökken a hiba.
Miután egyenesen a labda irányába fordult, csak azután indul meg, és halad egyenesen előre. Ha nem megy tökéletesen egyenesen, és az eszköz elmozdul a kép középpontjából, akkor megáll a robot, és újra pozíciónál. Ennél a módszernél nem kell különösebb szabályozó algoritmus.

13.4. Az eszköz megközelítése körív segítségével



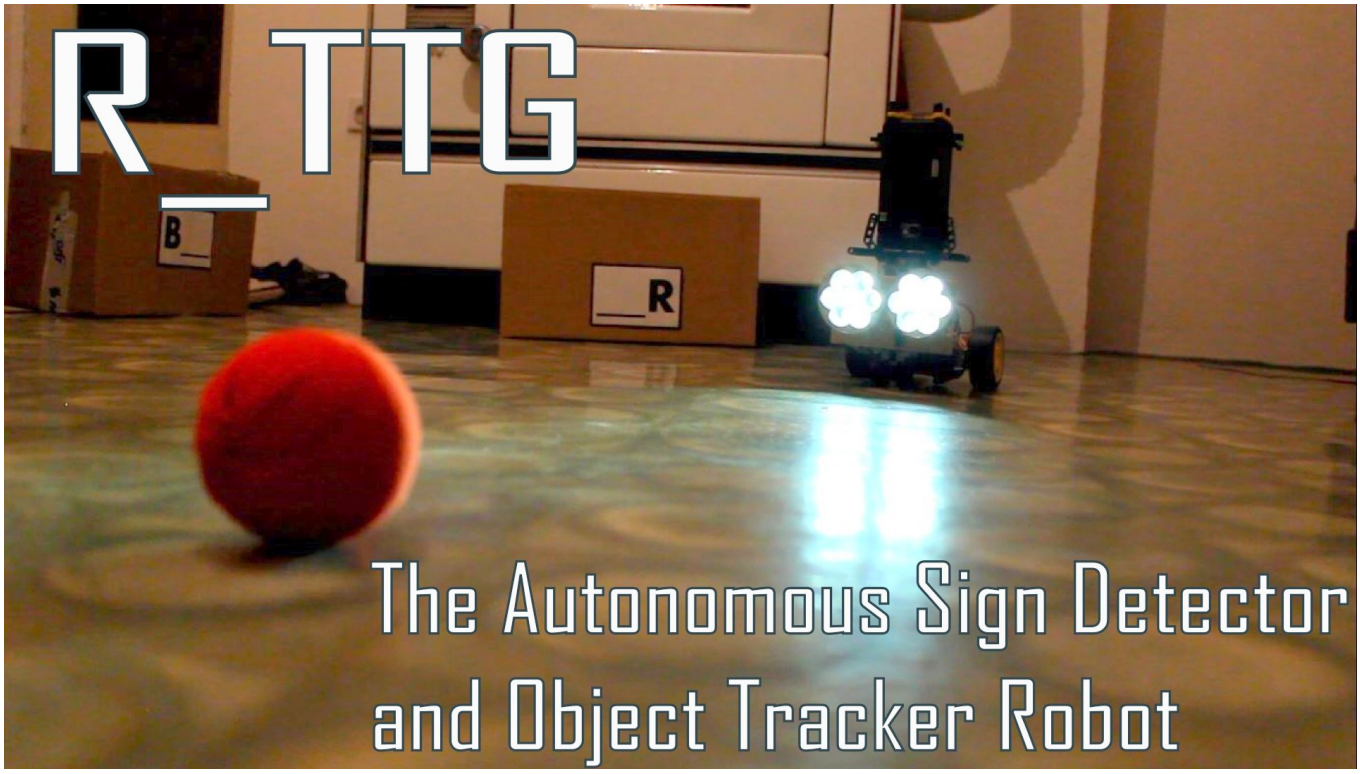
70. ábra: eszköz megközelítése körív segítségével

A második módszernél nem kizárólag egyenesen halad a robot, hanem ha szükség van rá, bizonyos ívben is haladhat. Azt, hogy mekkora ívben halad a robot a hiba határozza meg egy P szabályozón keresztül. Ennél a módszernél kisebb a tolerancia a kameránál, és ez mellett, nem áll le a robot, amikor fordul a kamera, hanem csak csökken a sebessége. A valós idejű képfeldolgozással valós időben tudjuk szabályozni a mozgást. A szabályozási blokksémát az 71. ábrán láthatjuk. Itt megadunk egy kívánt sebességet, és egy kívánt ívet. Fontos dolog, hogy a kalibráljuk a szervó motort, ami úgy történik, hogy megnézzük, mikor van null pozícióban a motor (mikor áll a kamera tartó váz párhuzamosan a kerekekkel), mert ha nem jól határozzuk meg, nem kapunk pontos eredményt. Az eredménybe bizonyos hibát visz be a szervó pontatlansága, és a fogaskerekek holt tere.



71. ábra: A szabályozó kör blokksémája

Ennek a módszernek előnye, hogy folyamatosan (megállás nélkül) és egyszerűen tudjuk szabályozni a robotot. Meg az eszközt megközelítéséhez, nem kell inverz kinematikát alkalmazni, és az eszköz pontos távolsága se fontos a kiindulási ponthoz viszonyítva ahhoz, hogy a robot célba érjen. Itt a visszacsatolásként az enkóderek (valós sebesség mérése) és a kamera (valós elfordulási szög a robothoz pozíciójához képest) szolgál.
Fontos, hogy megfelelően meg legyen választva a P szabályozó értéke a szervó motornál, mert ha túl kicsi, akkor nem fordul megfelelő szögben a cél irányában, ha túl nagy, akkor pedig hullámos lesz a robot útja, és lassabban fog haladni.
Integer változókat használok a nagyobb számítási sebesség miatt. Ha a szervó motort így csatoljuk a szabályozó körhöz, nagyobb skálán tudjuk változtatni az ív nagyságát.



72. ábra: a robotról készített videó fedőlapja

A videó megtekinthető a https://www.youtube.com/watch?v=aYvZUBr8wPQ linken.

14. Összegzés, következtetések

A robot és környezetének megvalósítása széleskörű ismereteket kívánt. **Gépészeti** ismeretek voltak nélkülözhetetlenek a robot modellezéséhez, megtervezéséhez és kivitelezéséhez (váz terve és összeállítása, a kerekek modellezése és kialakítása, mechanikai rendszer összeszerelése), az **elektronika** (az áramkörök megtervezése, nyomtatott áramkör tervezése, alkatrészek felforrasztása, akkumulátor töltő elkészítése, enkóder tervezése és elkészítése), a **képfeldolgozás** (színterek, szűrés, transzformációk, szegmentálás), az **informatika** (a felhasználói alkalmazás elkészítése, Bluetooth kommunikáció megvalósítása, a frame-k fogadása, képfeldolgozás eredményeinek továbbítása, mikrovezérlő programozása, adatok feldolgozása), az **irányítástechnika** (mintavételezés, P szabályozás) és a **robotika** szempontjából (differenciál hajtás tulajdonságai, lehetőségei, és szabályozási körének felépítése).

A munka során törekedtem az egyszerűségre. A váz kiegészítésénél LEGO építőkockát használtam, ami megfelelt a munka követelményeinek. Nagy előnye volt, hogy gyorsan lehetett módosítani, és csavarokkal megfelelően fel lehetett szorítani a vázra. Az új gumik jobb tapadással rendelkeznek, és ez fontos amikor helyben kell megfordulni kis szögben a robotnak. A mikrovezérlő program megírásánál és a hibajavításnál sokat segítettek a LED-ek az illesztőkártyán.

A szűrésnél esetenként helyszínnél újra kell kalibrálni a szűrőt, mivel a fényviszonyok változnak. Azzal, hogy az eszközt hisztogramok legnagyobb értékei keresésével találtam meg, azt jelenti, hogy mindig csak egy eszközt képes a robot megtalálni (aminek szűrés után a legnagyobb a felülete). Előnye, hogy nem beépített függvényt használtam, hogy a eszköz formájától függetlenül megtalálja az eszközt, ha jól beállítjuk a szűrőt. A kameránál 240x320-as felbontást használtam, amit valós időben fel tudott dolgozni a PC.

Ahhoz, hogy a képfeldolgozás sikeresebb legyen szükséges az aktív fényforrás, amelyet a LED-ek biztosítottak a robot elején. Az aktív fényforrás fényerejét egy motormeghajtó IC-vel végeztem, amelyet nem kellett hűteni. Vigyázni kell, hogy ne legyen túl nagy a külső fény, mivel az zavarhatja a képfeldolgozás folyamatát.

A kamera képének pixelenkénti összehasonlítása gyorsabb művelet, mint a PNSR és az MSE meghatározása. Ez a dolgozatban felhasznált módszer bináris képeken lehet alkalmazni.

Az eszközt az nélkül is meg lehet közelíteni, hogy tudnánk a pontos távolságát. Mivel csak egy kamerával dolgoztam, ezért ezt a dolgot nem is bírtam pontosan meghatározni.

A munka során a szabályozást timer megszakításban végeztem, ezért integer változókat használtam a számítás meggyorsítására, ami kerekítési pontatlanságot vitt be a számításba. Ha a mikrovezérlő nagyobb számítási kapacitással rendelkezne, akkor lehetne double, vagy float változó típusokat használni, amik lebegőpontosak, és akkor nem lépne fel a kerekítési hiba.

A kamera használatával az Abbé hibát jelentősen le lehet csökkenteni, mivel így nem csak a lokális koordinátákat tudjuk meghatározni, hanem a globális koordinátákat is.

A kommunikáció sebességét ha megnövelnénk, akkor több adatot megtudnánk jeleníteni a képernyőn.

A következő projektben szeretném a Bluetooth kommunikációt lecserélni. Sajnos a PC részéről (laptop) drivere nem volt Windows 8.1-re ezért nem csatlakozott minden alkalommal. Az árához képest a HC-06-os Bluetooth modul megfelelően működött.

Amikor a robot kereső üzemmódban működik, elegendő, ha 100 ms-os periódus idővel küldjük az utasításokat.

15. Összefoglaló

Autonóm robot megvalósítása és gépi látás alapú algoritmussal történő irányítása

A szakdolgozat bemutatta egy differenciál hajtással rendelkező robot megépítését, szabályozását és gépi látással történő irányítását. A robotot vezérlőt egy ATmega2560 mikrovezérlővel rendelkező Arduino lap alkotja. A struktúra további elemei: szenzor-elektronika, két egyenáramú motor, két akkumulátor, egy szervo motor, egy aktív megvilágítást alkalmazó látórendszer és egy Bluetooth modul. Az érzékelők és a beavatkozók kapcsolatát a vezérlővel, egy saját fejlesztésű illesztőkártya biztosítja.

A stratégiai szintű irányítás programja egy PC-alapú architektúrán fut. A PC és a robot Bluetooth-on keresztül kommunikál egymással. A robot két üzemmódban dolgozhat: az első szerint kézi vezérléssel, a második szerint pedig autonóm módon.

Az első program, a kézi vezérlésű, ahol a felhasználó a PC billentyűzet segítségével irányítja a robotot. A második egy autonóm irányítással rendelkező program. Ebben az üzemmódban a robotot egy kameraállványra felszerelt mobiltelefon által szolgáltatott kép feldolgozásának eredménye alapján irányítja a PC. A képfeldolgozáshoz OpenCV könyvtárakon alapuló Emgu CV segítségével végzi a PC.

Implementation of computer vision-based algorithm for autonomous robot guidance

This thesis describes the construction of a robot with differential drive, its' control mechanism and machine vision based guidance. The robot is controlled by an ATmega2560 microcontroller on an Arduino development board. Its' structure also contains the following elements: sensor-electronics, two DC motors, two accumulators, one servo motor, vision system with active illumination and one Bluetooth module. A proprietary interface card connects the microcontroller with the sensors and the interveners.

The strategic level guidance is accomplished with an application running on PC based architecture. The PC and the robot communicate with each other via Bluetooth. The robot can work in two modes: first one is the manual control mode, the second one is the autonomous mode.

First program is the manually operated mode, in which the user can guide the robot using the PC's keyboard. The second program implements autonomous guidance. In this mode, PC controls the robot based on the result of processing a video feed from a mobile phone mounted onto the robot. Image processing is performed on the PC using Emgu CV which is based on the OpenCV libraries.

Izgrađnja autonomnog robota i upravljanje sa algoritma koji poseduje mašinki vid

Ovaj rad predstavlja izgradnju robota sa diferencijalnim pogonom, kontrolu, i upravljanje sa mašinkim vidom. Deo koji kontrolira robota se sastoji od ATmega2560 mikrokontrolera koji je na Arduino ploči. Dodatni elementi strukture: senzor-elektronika, dva DC motora, dve baterije, servo motor, aktivno osvetljenje koristeći sistem vida i Bluetooth modula. Odnos između senzora, aktuatora i kontrolera isto vremeno obezbeđuje interfejs kartica. Program koji radi na PC arhitekturi zasnovan je na strateškom upravljanju. PC i robot komuniciraju preko Bluetooth-a. Robot može da radi u dva načina: prvi je ručna kontrola, drugi je autonomno upravljanje.
Prvi program, je za ručno upravljanje, gde korisnik sa PC tastature upravlja robota. Drugi program poseduje autonomno upravljanje. U ovom modu, računar kontroliše robota na osnovu rezultata obrade video zapisa sa mobilnog telefona koji je opremljen kamerom. Za obradu slike računar koristi programe sa iz biblioteke Emgu CV na osnovu OpenCV.

16. Irodalom

[1] Embedded Robotics - Thomas Bräunl

[2] http://hu.wikipedia.org/wiki/Arduino letöltve: 2013.11.07.

[3] http://arduino.cc/en/Main/arduinoBoardMega2560 letöltve: 2013.11.07.

[4]http://www.hobbielektronika.hu/cikkek/egyszeru_soros_kommunikacio_avr-rel_uart_oldal2.html letöltve: 2013.11.07.

[5] http://www.bluetooth.com/Pages/Fast-Facts.aspx letöltve: 2013.11.07.

[6] http://www.bosch.uni-miskolc.hu/userfiles/docs/szenzorok.pdf letöltve: 2013.11.07.

[7] http://szirty.uw.hu/Alapfokon/Encoder/encoder.html letöltve: 2013.11.07.

[8] Robottechnika INF-230 - Burkus Ervin

[9] https://sites.google.com/site/robottechnika/robotok/aktuatorok letöltve: 2013.11.07.

[10] http://hobbirobot.hu/content/vonalkoveto-ii-robotika-kezdoknek letöltve: 2013.11.07.

[11] ÅSTRÖM, K., HÄGGLUND, T. PID Controllers: Theory, Design, and Tuning, 2nd Ed., Instrument Society of America, Research Triangle Park NC, 1995

[12]http://www.robotc.net/wiki/Tutorials/Arduino_Projects/Mobile_Robotics/VEX/Using_encoders_to_drive_straight letöltve: 2013.11.07.

[13] JONES, J., FLYNN, A., SEIGER, B. Mobile Robots - From Inspiration to Implementation, 2nd Ed., AK Peters, Wellesley MA, 1999

[14] http://robotics.stackexchange.com/questions/1711/approach-to-using-pid-to-get-a-differential-robot-driving-straight letöltve: 2013.11.07.

[15] Positioning System for 4-Wheel Mobile Robot: Encoder, Gyro and Accelerometer Data Fusion with Error Model Method - Ibrahim Zunaidi , Norihiko Kato, Yoshihiko Nomura and Hirokazu Matsu 2006

[16] L293, L293D QUADRUPLE HALF-H DRIVERS

[17] http://en.wikipedia.org/wiki/Image_processing letöltve: 2013.12.15

[18] http://3dmr.iit.bme.hu/edu/vkf/index.html?lang=hu letöltve: 2013.12.15

[19] http://www.tankonyvtar.hu/hu/tartalom/tamop425/0008_kato_czuni/adatok.html letöltve: 2013.12.15

[20] http://opencv.org/ letöltve: 2013.12.15

[21]http://karanjthakkar.wordpress.com/2012/11/21/what-is-opencv-opencv-vs-matlab/ letöltve: 2013.12.15
[22] http://www.emgu.com/wiki/index.php/Main_Page letöltve: 2013.12.15
[23] www.sourceforge.net
[24] http://hu.wikipedia.org/wiki/RGB_sz%C3%ADnt%C3%A9r letöltve: 2013.12.15
[25] http://en.wikipedia.org/wiki/HSL_and_HSV letöltve: 2013.12.15
[26] en.wikipedia.org/wiki/Gaussian_blur letöltve: 2013.12.15
[27] http://mazsola.iit.uni-miskolc.hu/~czap/HEFOP/Kepfeld1010.pdf letöltve: 2013.12.15
[28] Dr. Szabó Anita és Szakáll Tibor - Kép és hangfeldolgozás gyakorlatok Matlab környezetben
[29]https://opencv-code.com/tutorials/automatic-perspective-correction-for-quadrilateral-objects/ letöltve: 2014.03.12.
[30] Dr. Odry Péter: Hang és kép feldolgozás, jegyzet
[31] http://tfeghough.codeplex.com/documentation letöltve: 2014.03.23.
[32] http://www.dcs.vein.hu/hangos/oktatas/MatlabBevezetes.pdf letötlve: 2014.04.11.

Önéletrajz

Nevem Bessenyei Szilárd. 1991.10.01. születtem Szabadkán.

Az általános iskolát a palicsi Miroslav Antić Általános Iskolában színkitűnő eredménnyel fejeztem be, amiért Vuk-diplomát is kaptam.

A Szabadkai Műszaki középiskolában számítógépes elektrotechnikusi szakon, négy éves képzés keretében végeztem szintén kitűnő tanulmányi eredménnyel.

Középiskola befelyezése után a Szabadkai Műszaki Szakfőiskolán a 2010/2011 évfolyamon az elektrotechnikai szakot írtam be. Az összes tanulmányaimat eddig magyar nyelven végeztem.