ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZET
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

SZAKDOLGOZAT

OE-KVK
2023.

Fehér József Mátyás
T009173/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Fehér József Mátyás
Szakdolgozat száma: SZD2309040923375957PMN10Z
Törzskönyvi száma: T009173/FI12904/K **Neptun kódja:** PMN10Z

Szak: villamosmérnöki (magyar nyelven)
Specializáció: Műszer-automatika specializáció

A dolgozat címe: Delta robot szoftveres problémáinak megoldása
A dolgozat címe angolul: Solving software problems with Delta robot
A feladat részletezése: Készítsen alkalmazást egy delta robot alkalmazására.

Intézményi konzulens neve: Sándor Tamás

A kiadott téma elévülési határideje:2025. 12. 31.
Beadási határidő: 2023. 12. 15.

A szakdolgozat: Nem titkos.

Kiadva: Budapest, 2023. 10. 24.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Elektronikai és Kommunikációs Rendszerek Intézet
P.H

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2023.12.15.

.....
hallgató aláírása

Köszönetnyilvánítás

Elsősorban szeretnék köszönetet mondani az Óbudai Egyetem Kandó Kálmán Villamosmérnöki Karának, hogy elérhetővé tette számunkra az egyetemi delta robotot, amin megvalósíthattuk a projektet.

Köszönöm Sándor Tamás Tanár Úrnak és Borsos Döníz Tanárnőnek, hogy mind a projektben mind az egyetemi tanulmányaimban rengeteget segítettek.

Köszönöm Schriffert Balázsnak a projekten folytatott közös munkát, valamint Kovács Gergő Jánosnak és Schmidt Péternek a mentorálást

Végül pedig hatalmas köszönet a családomnak, hogy egyetemi tanulmányaim alatt mindenben támogattak, hogy el tudjam érni céljaimat.

Tartalom

1. Bevezetés

A dolgozat témája, egy az iparban gyakran használatos robotfajta a delta robot alternatív felhasználása, melyen egy szaktársammal Schriffert Balázzsal dolgoztam együtt. A roboton együtt dolgoztunk, azonban én főleg a szoftveres kérdések megoldásával foglalkoztam, melyről a dolgozatom is szól.

Korábbi szakdolgozatok témáját képezte már ez a robot fajta az egyetemen, így egy a hallgatók által épített delta robotot felhasználva valósult meg ez a projekt is. Az ilyen típusú robotokat, mint majd ahogy arról később is szó esik, az iparban apró, könnyű tárgyak gyors áthelyezésére alkalmazzák. Egy ilyen Pick and Place feladatot valósít meg a dolgozat témáját képező robot is. Az ilyen műveleteket két főbb részre lehet bontani: melyek a mozgatni kívánt tárgy helyének meghatározása és annak pontos mozgatása. Előbbit az iparban olyan képfeldolgozó módszerek segítik melyek képesek alakzatokat és színeket megkülönböztetni, utóbbit pedig precíz matematikai számítások segítik.[1]

A projekt célja egy olyan alternatív felhasználási módszer újra megalkotása volt, amely az előzőekben említett ipari módszereket szoftveresen implementálja, emellett látványos és érdekes a nem hozzáértők számára is. A sakkjáték megfelelő választás erre a célra, hiszen a sakktábla mezőin precízen kell mozgatni a bábukat annak érdekében, hogy a lépések egyértelműek legyenek, mindemellett hat különböző formájú bábú van játékban, két különböző színben, melyeket fel kell ismernie a robotnak, hogy megfelelő lépést tudjon választani. A játék közismertsége végett pedig látványos és szemléletes is a korábban említett módszerek bemutatására.

2. A delta robotok bemutatása

2.1 Történelmi háttér

A „robot” szót mai értelmében a 20. század elején használta először egy Karel Čapek nevű cseh író, aki egyik első sci-fi drámájában ezt a szót mechanikus emberekre használta, akik a gyárakban dolgoztak az emberek helyett. Ez jól párhuzamba állítható a gyárak első ipari forradalom óta tartó törekvésével, miszerint a gyártás minőségének és mennyiségének növelése érdekében egyre több munkafolyamatot és feladatot gépeknek kell ellátnia emberek helyett. Az első ilyen automatizálási megoldások még a gyári dolgozók által üzemeltetett gépek voltak, melyek valamilyen hidraulikus vagy pneumatikus elven működtek. Az elektronika és a digitális technika fejlődésével azonban az 1900-as évek közepétől rohamosan elkezdtek fejlődni az automatizálási technikák, így a robottechnológia is, melyet négy főbb korra lehet osztani: [2]

1950 és 1967 közé tehető a robotok első generációja, mely egyszerű manipulátorokkal indult, melyek numerikus vezérléssel működtek és még nem volt információjuk a környezetükről. Erre a korszakra vezethető vissza a mai robotika eredete, mely az amerikai autóipar magas gyártási igényeinek kiszolgálására kívánt megoldást nyújtani. Az első programozható gépet, (melyet az első ipari robotnak is tartanak) George Devol és Joe Engleberger alkották meg UNIMATE névvel, mely képes volt pick and place feladatokat ellátni és gyorsítani az autógyártás folyamatát. Ennek köszönhetően egyre

1. ábra Az első gyártásban is használt ipari robot [14]

több cég és kutató központ elkezdett érdeklődni és erőforrásokat fektetni a robotika fejlesztésébe. [2]

1968-tól kezdve a robotokat szenzorokkal is elkezdték felszerelni a technika fejlődésének köszönhetően, így képesek lettek a környezetüket érzékelni, illetve saját mozgásukról is tudtak érzékelőkkel adatokat gyűjteni. Így ezzel megindult a robotok második generációja. Ekkoriban a robotok vezérlésére PLC-ket (Programmable Logic Controller) használtak, hiszen ez volt a legelterjedtebb a gyártási folyamat-automatizálásban, megbízhatósága és könnyű programozhatósága miatt. [2]

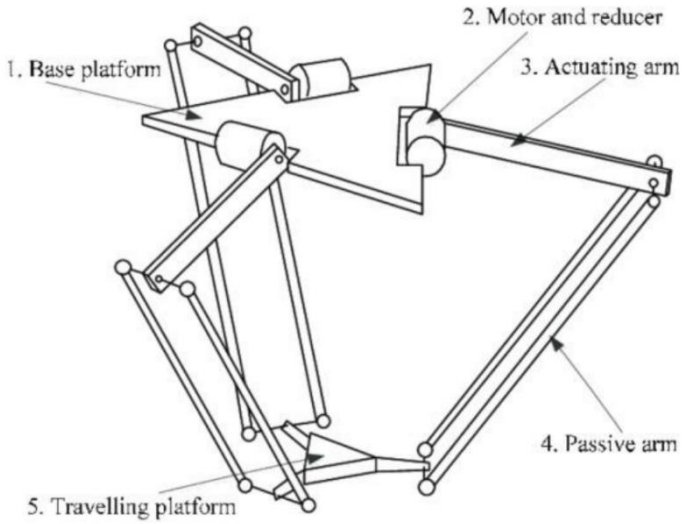
Ugyan a második generáció végéhez közelítve a robotokhoz már elkezdtek mikroszámítógépeket használni, a robotok harmadik generációjával (1978-1999) vált csak általánossá, hogy a robotokat valamilyen számítógépes egységgel vezérelték. Ez és a ’80-as évek technikai fejlődése tette lehetővé, hogy új a robotok programozására dedikált programozási nyelvek alakuljanak ki és újra programozhatóak legyenek. Az egyszerűbb programozhatóság és az ebben az időszakban a robotika fejlesztésére ráfordított hatalmas összegeknek köszönhetően a robotok már nem csak az autóiparban, hanem sok más ipari területen is elterjedtek. Széleskörű felhasználásuk közé tartozott a tárgyak mozgatásán kívül például az összeszerelés, a hegesztés és a festés is. A ’90-es évek végén több cég is elkezdett foglalkozni robotokkal ipari környezeten kívül, ekkor

született meg az első LEGO Mindstorms készlet is (1998) ami segített megtanítani és népszerűsíteni a robotika alapjait a fiatalabb generációnak. [2]

A negyedik generációval (2000-) megkezdődött az intelligens robotok kora. A számítógépek fejlődésével a robotok vezérlésére egyre bonyolultabb algoritmusok is alkalmasak lettek, bonyolultabb logikákat is lehetett alkalmazni. A kor legújabb szenzorjainak és a fejlett számítástechnikának köszönhetően a robotok képesek lettek a környezeti viszonyokhoz jobban alkalmazkodni és ez alapján logikus döntéseket hozni. A mesterséges intelligenciának és a számítógépes látás fejlődésének köszönhetően pedig létrejöttek az egymással, illetve az emberekkel együtt dolgozni képes robotok is. Ugyan a robotok képesek lettek minden eddiginél gyorsabban és pontosabban elvégezni a munkájukat, az innovációk ebben a korszakban jelentősen csökkentek az előző évtizedekhez képest. [2]

2.2 Delta robotok története és felhasználása

A delta robotot az 1980-as években az EPFL svájci egyetem professzora Reymond Clavel alkotta meg. Az ötlet egy tanársegédtől származik, aki egy csokoládégyár látogatása után felvetette a feladatot egy praliné csomagoló gép megalkotására. A cél egy olyan robot létrehozása volt, amely nagy sebességgel képes dolgozni síkfelülettel



2. ábra A delta robot vázlata [15]

párhuzamosan. Ennek érdekében egy könnyű karokkal rendelkező és kis súly gyors mozgatására alkalmas gépet tervezett Clavel, melyre 1985-ben szabadalmat is kapott. [3]

A delta robot fajtáját tekintve egy párhuzamos robot, melyek lényege, hogy egy fix alap összeköttetésben van egy ezzel párhuzamosan mozgatható résszel. Ezt a kapcsolatot több kar és csukló hozza létre egy kinematikai láncként, mely biztosítja a végeffektor és az álló rész állandó párhuzamosságát. A delta robotok alapvető felépítését az határozza meg, hogy a robot karjai a csuklókkal paralelogrammákat alkotnak, melyek az előzőekben említett kinematikai láncot alkotják, így a robot csak az X, Y és Z koordináták mentén tud mozogni, azaz szabadsági foka 3. Ezek a paralelogrammák a robot passzív karjai, azaz nem meghajtottak. A paralelogrammák egyik oldalához kapcsolódik egy-egy csukló segítségével a végeffektor és az ezzel szemben lévő oldalhoz kapcsolódik a hajtott aktív kar. Az aktív karok hajtása általában DC motorokkal történik, minden kart külön motor hajt meg és a motorok elrendezés szerint egy szabályos háromszög oldalainak

közepén helyezkednek el. Ennek a felépítésnek köszönhetően a robot szerkezete sokkal könnyebb, mint a hagyományos soros robotoké, hiszen a nehéz motorok a fix alapon találhatóak nem a mozgásban résztvevő karokon. Mivel könnyebb így gyorsabb mozgásra is képes: ipari körülmények között a végeffektor akár 12 G gyorsulást is el tud érni, így tökéletes választás Pick and Place feladatok ellátására. A szerkezeti felépítésnek azonban van egy hátránya is: más robotokhoz képest a delta robot munkatere sokkal kisebb és korlátoltabb.

3. ábra Csomagoló delta robotok [4]

Sokrétűen felhasználhatóak a delta robotok mind az iparban mind azon kívül: Ipari területeken ezt a típusú robotot legfőképpen Pick and Place feladatok ellátására használják gyorsasága miatt, de ezen felül használják még CNC berendezésekben és csomagoló gépekként is. Egyéb felhasználási területei közé tartozik még a Surgiscope amely egy nagy, műtétekhez használt mikroszkópos orvosi segédeszköz, illetve 3D nyomtatók nyomtató fejének mozgatása is. Az egyetemen lévő delta robot tovább bővíti a felhasználási spektrumot, hiszen az korábban már rajzoló robotként is fel lett használva.[4]

3. Specifikáció

Korábban már képezte szakdolgozatok témáját delta robot az egyetemen, így a projekthez rendelkezésre állt egy hallgatók által épített ilyen robot, melyet többek között sakkozó robotként is felhasználtak. Ennek köszönhetően a robot már rendelkezett a karok mozgatásához, illetve a munkatér vizsgálatához szükséges hardvereszközökkel, valamint képes volt sakkozni emberi ellenfél ellen. A célkitűzés ezen sakkozó robot szoftveres újragondolása volt annak érdekében, hogy működése stabilabb legyen, jól illeszthető legyen jövőbeli fejlesztésekhez és akár más projektekhez is. Így nem csak a robotika népszerűsítésére használható, hanem tanulási eszközként is hallgatók számára.

A projekt megvalósításához a szoftveres kérdéseket 3 fő részre kell osztani, melyből az első a motorvezérlés megvalósítása. Itt elsősorban azt kell figyelembe venni, hogy a vezérlés és a visszacsatolás illeszkedjen a meglévő hardver korlátjaihoz, valamint, hogy a megvalósított program függvényeinek be és kimenetei, illetve az eltárolt változók könnyen felhasználhatóak legyenek a többi modulban, illetve a jövőbeli fejlesztések során.

A második fő kérdés a játéktér és a bábuk detektálása. Itt egy kézenfekvő megoldás valamilyen képfeldolgozási módszer használata, hiszen ez egyszerre nyújt megoldást mind a játéktér mind a bábuk helyzetének megállapítására. Az ehhez rendelkezésre álló hardver és a képfeldolgozási módszerek határozzák meg, hogy melyik módszert érdemes választani ehhez. Továbbá figyelembe kell venni, hogy a modul kimenete jól illeszthető legyen a szoftver harmadik nagy egységéhez.

Ami nem más, mint a sakkozó program és az ezeket a fő program részeket egybefogó modul. Itt fontos figyelembe venni, hogy olyan sakkozó modul valósuljon meg, melynek kimenete és bemenetei illeszkednek a képfeldolgozás és a karok vezérlésének ki és bemeneteihez, valamint, hogy az egybefogó modul kiegészítse az előzőekben említett modulokat és új lehetőséget adjon a fejlesztésre.

4. Logikai rendszerterv

A delta robot a projekt kezdetekor adott volt minden a működtetéséhez szükséges hardvereszközzel. A következőkben szó esik ezen elemek kapcsolatáról és a rendszer logikai felépítéséről.

A delta robot karjai jól definiálják a munkateret, melyben egy az effektor végén található elektromágnes segítségével mozgatja a robot a bábukat.

A robot karjainak mozgatását léptetőmotorok végzik, melyek a karokhoz csatlakozva a robot bázisán, azaz felül helyezkednek el. Ez a motorfajta jól használható precíz mozgások megvalósítására, hiszen a karok állása apró lépésközökkel akár század fokonként is állítható.

Ezen motorok vezérlésére minden motorhoz csatlakoztatva van egy-egy motorvezérlő egység, mely a vezérlő számítógép és a motorok között teremt kapcsolatot. Ezen vezérlőkön keresztül állítható a motorok lépésszögeinek nagysága, a motorok forgásiránya és ezeken keresztül valósul meg a motorok léptetése is.

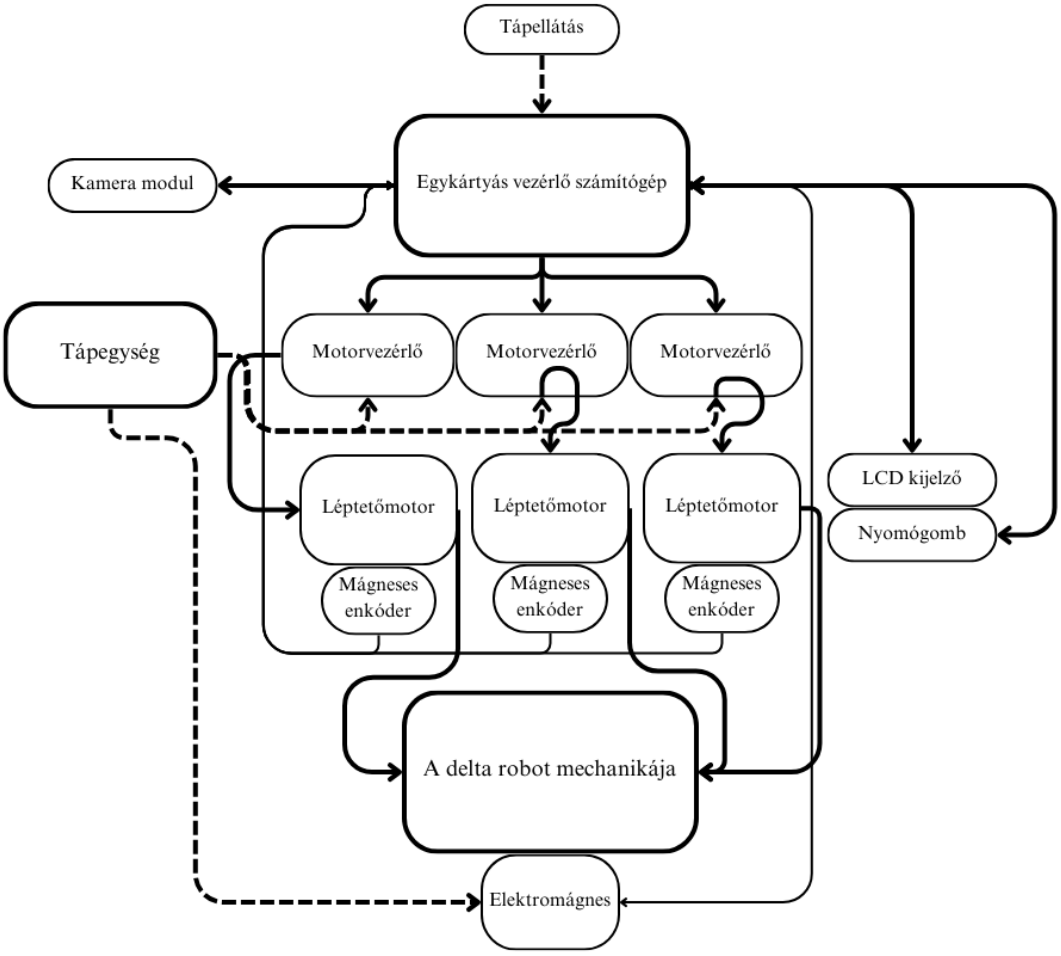
A motorok állásáról abszolút mágneses enkóderek biztosítanak információt, egy a motorok tengelyére rögzített mágnes segítségével. Ezek a szenzorok szintén egyenesen kapcsolódnak a vezérlést végző számítógéphez.

A munkatérben történő változások detektálásához egy kameramodul van használatban, mellyel képek vagy videó készíthető a vezérlőegység számára.

A játékossal való kommunikációra egy LCD kijelző szolgál, hogy tudatni lehessen vele, ő mikor következik és mikor nem szabad a robot munkaterébe nyúlnia. Ezek mellett egy gomb is található, mellyel a felhasználó jelezheti saját köre végét, ezáltal is visszajelzést adva a robotnak.

A delta robot teljes működését egy egykártyás számítógép fogja össze, a motorok vezérlésétől, a képfeldolgozáson keresztül a sakkprogramig, mely nagy számítási kapacitással rendelkezik ezen feladatok kiszolgálásának érdekében.

A 4. ábrán látható a rendszer logikai terve kiemelve a tápellátási viszonyokat is, hiszen ezek eltérnek a perifériák között. A szenzorok, a kamera modul, a gomb és az LCD kijelző a vezérlőegységtől kapják a tápfeszültséget, még a motorvezérlők és az elektromágnes egy külön tápegységről üzemel. A léptetőmotorok pedig a vezérlőktől kapják meg a megfelelő tápfeszültséget és áramot.



4. ábra Logikai rendszerterv

5. Fizikai rendszerterv

A váz fizikai felépítése az első fontos szempont a rendszerterv során, hiszen ez sokat megszab azzal kapcsolatban, hogy a robot mire képes. A robotkarok és a szerkezet váza alumíniumból készültek, ahogyan a munkatér alapja és a karok bázisa is. A felső lapra alulról felszerelve található a három léptető motor, melyek tengelyei geometriailag egy szabályos háromszög oldalainak közepén helyezkednek el a bázis közepén. A motorok tengelyeire szerelve találhatóak a 15cm hosszú aktív (hajtott) karok, melyek végükön csuklótengellyel csatlakoznak a 33 cm hosszú passzív karokhoz. Ezen karok másik végén szintén csuklótengellyel csatlakozik a végeffektor. Ezen karokkal a robot egy 24cm x 24cm x 24cm-es munkateret tud bejárni. Kialakításából adódóan a karok és a végeffektor igen nehezek, így ez azt eredményezi, hogy csak lassan mozgathatók, azonban ez nem probléma a kitűzött feladat megoldásában. A végeffektor aljára egy elektromágnes van elhelyezve, melynek 12 V tápfeszültségre van szüksége, így ezt a vezérlő számítógép egy egycsatornás relén keresztül vezérli.

A karok mozgását három darab 57HS76-2804A típusú bipoláris léptetőmotor biztosítja a roboton. Ez a típus egy 2 fázisú motor, 3V-os fázisfeszültséggel és 2.8A fázisárammal. Nyomatéka 1.9Nm, teljes lépési szöge pedig 1.8°. Az iparban általában erősebb motorokat alkalmaznak a karok mozgatásához, azonban ez a robot oktatási célokra készült, így a bemutatáshoz megfelelő paraméterekkel rendelkezik ez a motortípus.[5]

Egy DM556D típusú motorvezérlő végzi a motorok vezérlését. Ez a vezérlő megfelelő a 4 vezetékes bekötésű motorokhoz, amelyeket használunk. A vezérlőn DIP kapcsolókkal állítható a motorokba kimenő maximális áram és a microstepping felbontása, mely utóbbival állítható a léptető motorok mozgásának finomsága. [6]

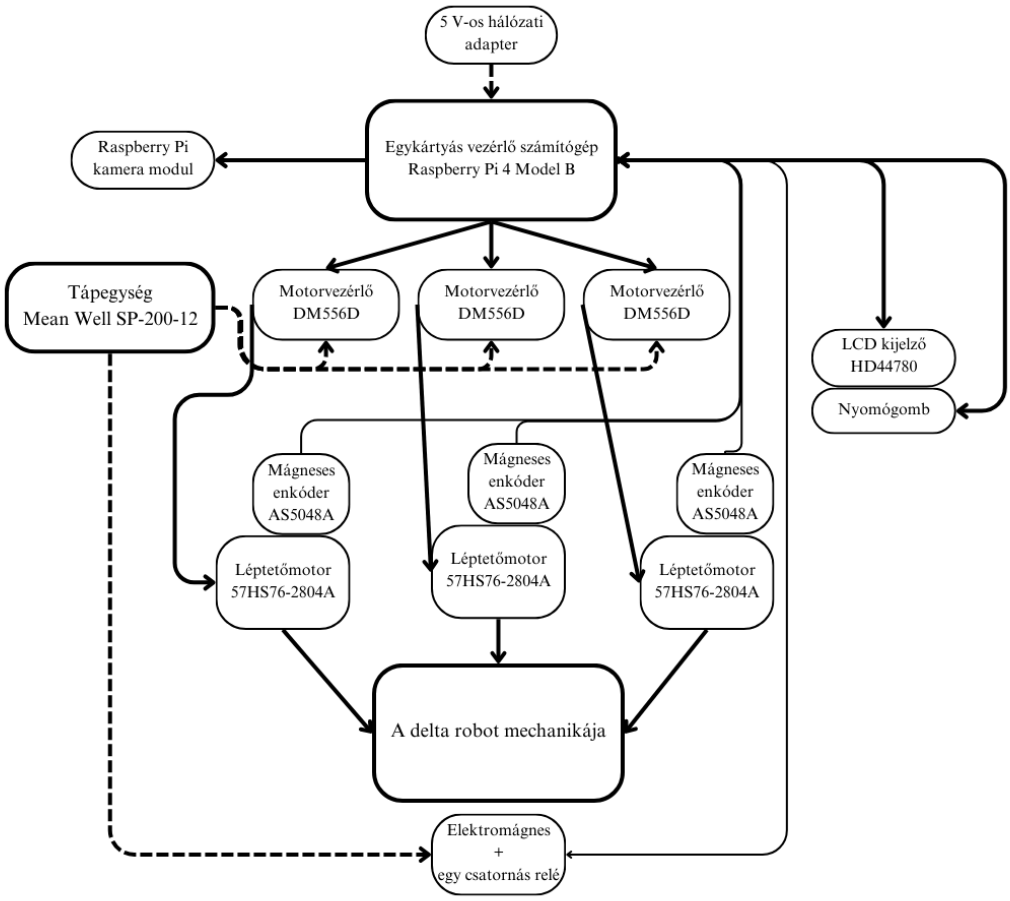
A motorvezérlők és az elektromágnes tápellátását egy Mean Well SP-200-12 kapcsolóüzemű tápegység biztosítja. A tápegység 230V hálózati feszültséget alakít át 12V DC feszültséggé és kimenetén 16.7A áramot biztosít. Mindegyik motor a hozzá tartozó motorvezérlőből kapja a megfelelő tápellátást.[7]

A végeffektor pozicionálásához szükséges a motorok szögének beolvasása, melyet a motorok tengelyére merőlegesen felszerelt mágnes és a vele szemben egy eltartóval rögzített AS5048A típusú mágneses enkóder végzi. Az eszköz teljes körben tudja érzékelni a motorok abszolút elfordulását és a beolvasott szögértékeket 14 bites részletességgel küldi tovább SPI kommunikáción keresztül.[8]

A robot alapján történő változások detektálásáért egy Raspberry Pi kamera modul felel. Ez a kameraegység 3280 x 2464 pixel maximális felbontásra képes. Valamint ez egy a vezérlőhöz kifejezetten gyártott periféria, ami megkönnyíti a szoftveres fejlesztést.[9]

A robot és használója közötti interakciót egy HD44780 LCD kijelző biztosítja, mely szintén a Raspberry Pi 4 Model B- hez van csatlakoztatva. A kettő közötti kommunikáció az egyszerűség kedvéért I^2C interfészen keresztül történik.

Ahogyan az az alábbi ábrán is látható a teljes rendszer működtetéséért egyetlen eszköz, egy Raspberry Pi 4 Model B egykártyás számítógép felel, mely 64 bites Cortex-A72 processzorának és a kártyára szerelt 4GB méretű LPDDR4 SDRAM-nak köszönhetően képes kiszolgálni nagy számítási teljesítményt igénylő feladatokat is. Ennek köszönhetően kifejezetten alkalmas arra, hogy egységesen magába foglalja és elvégezze a motorvezérlés, a képfeldolgozás és a sakkozó modul feladatait is. Ezt és a fejlesztést tovább segíti saját grafikus operációs rendszere és az ehhez szoftveresen tartozó sajátfüggvények is, melyek megkönnyítik az előzőekben felsorolt perifériák illesztését a végső programba. Itt fontos még megjegyezni, hogy a Raspberry Pi 40 GPIO pinnel rendelkezik, melyek 5V-os digitális jelszintet képesek biztosítani, ami kompatibilis a rákapcsolandó eszközökkel: a motorvezérlőkkel, az enkóderekkel, illetve az LCD kijelzővel. Ezen eszközök működtetése csupán 27 GPIO pint foglal le, emellett a grafikus felület és az egy helyen tárolt programok nagyban megkönnyítik a rendszer tovább-fejleszthetőségét is.[10]



5. ábra Fizikai rendszerterv

6. A robotkarok vezérlése

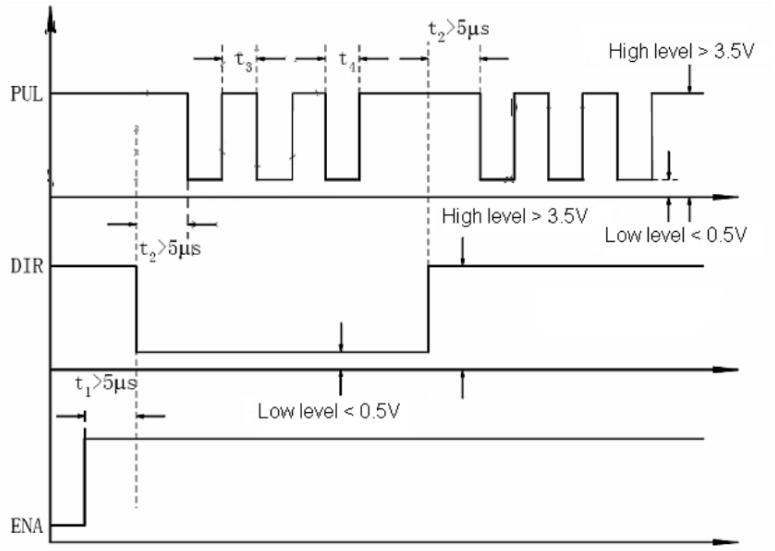
6.1 Motorvezérlési megoldások

Ahogyan arról már korábban is szó esett a robotkarok mozgatását bipoláris léptetőmotorok végzik, melyeket a vezérlő számítógép motorvezérlő egységeken keresztül irányít. A léptetőmotorok működésének szoftveres tekintetben a legfontosabb tulajdonsága az, hogy a motorok elfordulása fix pontok között történik, vagyis a motorok teljes körbefordulása feloszthető adott számú mozdulatra. Az adatlapból kiderül, hogy egy teljes körbeforduláshoz a motornak 200 lépésre van szüksége [5], ezen lépések a motorvezérlőkkel tovább oszthatók mikrolépésekre 2 hatványaival, egy teljes lépés akár 256 további mikrolépésre osztható. [6]

A motorvezérlőknek három jel bemenete van melyekkel a következőket tudjuk állítani: [6]

- PUL (Pulse) Egy impulzus jel, melynek hatására a motor egy a beállítottnak megfelelő lépést vagy mikrolépést tesz. Ennek a bemenetnek magas jelszinten kell lennie alapesetben, mert a lépés 1-0 lefutó élre valósul meg. A stabil működés érdekében mindkét jelszintet legalább 2.9µs-ig kell fenntartani.
- DIR (Direction) Ez a jel a motor forgásának irányát határozza meg. Magas jelszinten a motor az óramutató járásával megegyező irányba teszi a lépéseket, míg alacsony jelszinten ezzel ellentétes irányba.
- ENA (Enable) Ez az engedélyező jel, mellyel a motorvezérlő be, illetve ki kapcsolható. Alapesetben alacsony jelszinten kell lennie, hogy engedélyezze a működést. Fontos tudni, hogy a nem engedélyezett motorvezérlő nem ad áramot

a motornak, így az tartani sem fogja magát, így ennek a működés alatt állandó alacsony szinten kell lennie.



6. ábra A vezérlőjel hullámformája és időzítése [6]

Ezen információk tudatában a motorvezérlő program megírásának első lépése egy setup rész, ahol a megfelelő GPIO kimeneteket be kell állítani a fentieknek megfelelően. A három PUL jelhez tartozó kimenetet tehát alapesetben mindet magas szintre, a DIR-hez tartozó kimenetek tetszőlegesen állíthatóak kezdetben alacsony vagy magas szintre, az ENA- hoz tartozó kimeneteket pedig alacsony szintre kell inicializálni. A motorok forgásirányához tartozó (DIR) kimeneteket alacsony szintre állítottam a program elején, mellyel a karokat a robot annak alapja felé, lefelé mozgatja.

Az inicializálás után a következő lépés a robot karjainak mozgatása, melyhez ugyan fizikailag, de szükséges még beállítani a motorok lépésközét. Ez határozza meg, hogy mennyi impulzust kell kiadnunk a kiinduló és a célpont közötti mozgáshoz, valamint ettől függ az is, hogy a végeffektort milyen pontossággal tudjuk elhelyezni a munkatérben. Itt figyelembe véve a robot és a mezők méreteit egy 8-as osztót választottam, így a léptető motoroknak a teljes körbeforduláshoz 1600 mikrolépésre van szüksége, ami azt jelenti, hogy egy impulzus hatására a motorok 0.225°-ot fordulnak el, ami megfelelően nagy pontosságot jelent.

Az általam készített első motorvezérlés megoldási terv is erre az adatra épült: Hiszen, ha tudjuk a program indulásakor a motorok szögállását és tudjuk mely motorállások szükségesek a végeffektor kívánt pozíciójának eléréséhez, akkor, ha vesszük a két szögérték különbségének abszolút értékét és elosztjuk az egy lépés alatt megtett szögértékkel, akkor megkapjuk, hogy hány impulzus kiadása szükséges ahhoz, hogy a motorok a kívánt állásba kerüljenek. A szögértékek különbségéből továbbá megállapítható az is, hogy az egyes motorokat melyik irányba kell forgatni. Ezzel a módszerrel továbbá nincs szükség visszacsatolásra sem, hiszen a mozgatás végén elért motorállások és a következő mozgatás kiinduló motorállásai megegyeznek.

Ezen vezérlés tesztelése során azonban kiderült, hogy a motorok léptetés közben átugranak lépéseket, mely azt eredményezi, hogy a kívánthoz képest nagyobb elfordulást végeznek a motorok. Ez egy nagy fokú bizonytalanságot okoz a vezérlésben, hiszen az első rossz léptetés végén a motorok már nem a várt állásból indulnak, így nem lesz pontos a vélt és kívánt motorállásokból számított lépésszám sem, így minden hibás lépéssorozat növeli a következő mozgatás hibáját. Ennek a hibának a kiküszöbölésére szükséges bevezetni egy folyamatos visszacsatolást melyet a következő részfejezet tárgyal.

6.2 Visszacsatolás

A motorok helyzetéről a korábban már említett AS5048A abszolút mágneses elfordulásérzékelők tudnak visszacsatolást biztosítani. Ezek az érzékelők SPI segítségével kommunikálnak a vezérlőszámítógéppel, melyet szintén a setup részben inicializálni kell. A Raspberry Pi 4 model B esetében először meg kell adnunk, hogy milyen GPIO kiosztás mellett szeretnénk használni az SPI-t, azaz, hogy az előre definiált pinek közül melyikeken fussanak a MISO, MOSI, SCK és CS jelek. Ezek után meg kell adni az órajel maximális frekvenciáját, melyet a mágneses enkóderek adatlapja alapján 1MHz-re állítottam. Végül pedig az SPI módot kell beállítani, hogy az órajel mely részénél történjen az adatok mintavételezése, és küldése, melyet szintén az adatlap alapján 1-es módra állítottam. [8]

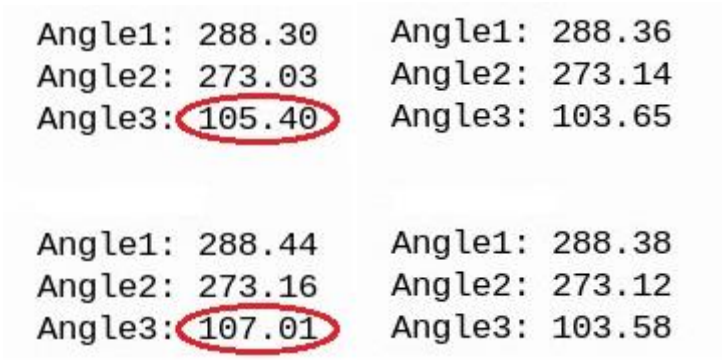
A következő lépés az enkóderek olvasását megvalósító függvény létrehozása volt, mely során a következőket kellett figyelembe venni:

- Az érzékelők Multi slave bekötésben vannak, így a kiválasztó jelek különböző GPIO-kon futnak
- Az adatok bájtonként érkeznek és két bájt méretűek, a bejövő adatok tárolására ennek megfelelő változót kell használni
- A beérkező 16 bitből csak 14 hasznos
- A hasznos 14 bites adatot át kell számítani szögérték formátumba

Ezen pontoknak megfelelően a függvény működését a következőképpen alakítottam ki: A függvény átveszi a vizsgálni kívánt motor enkóderének GPIO számát, majd létrehoz egy két elemű bájt tömb változót az adatok fogadására és egy float típusú változót, melyben a függvény kimenetét fogja eltárolni fok alakban. Ezt követően az átvett adattal engedélyezi a megfelelő érzékelőt és kiküldi az olvasási parancsot, melyre a választ az előbb említett bájttömbben eltárolja, majd visszazárja a kommunikációs csatornát. A tömb két elemét egy újabb változóba egymás után fűzi, majd az így kapott 16 bit felső 2 bitjét kinullázza, hogy csak a szögértékre vonatkozó bitek maradjanak meg. Az így kapott 14 bites számot megszorozza $360/2^{14}$-nel, ami az LSB-re számított

szögérték nagysága (0,0219°), így végeredményként megkapva a kiválasztott motor abszolút szögállását, melyet a függvény visszatérési értékként tovább is ad.

A függvény tesztelése érdekében a karok álló helyzetében többször is kiolvastam az enkóderek által adott szögértékeket, ezzel vizsgálva, hogy mennyire adnak konzisztens



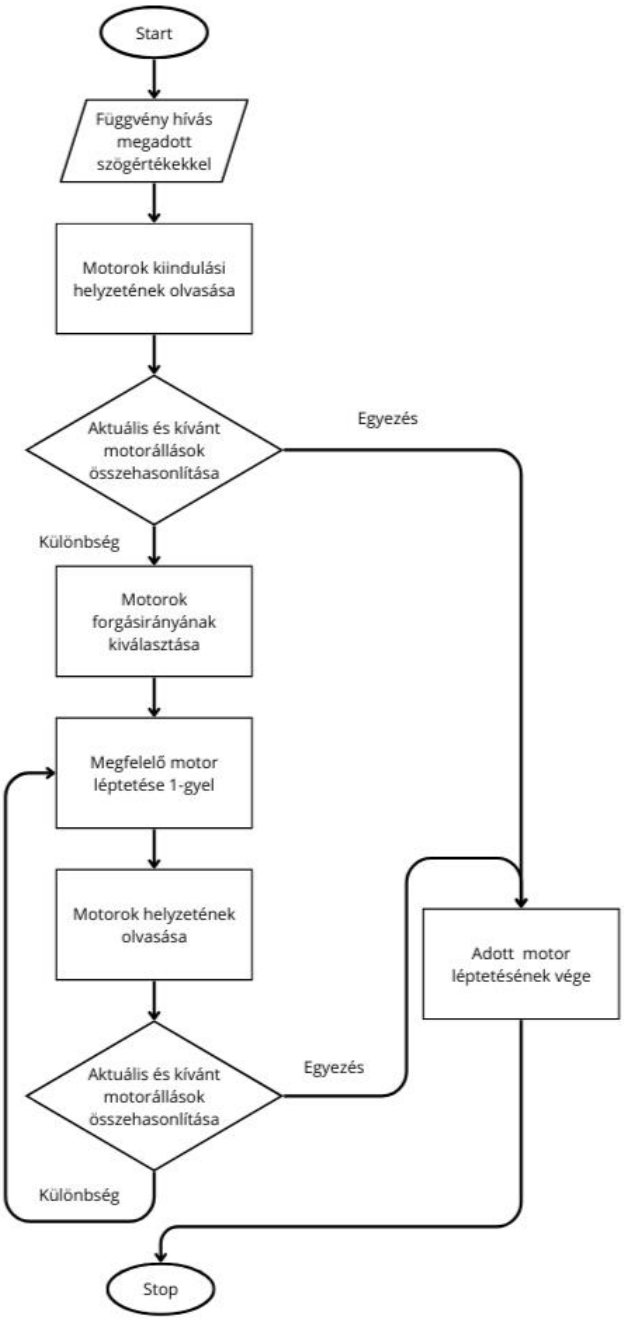
7. ábra A visszacsatolás teszteredményei

adatokat. Azt tapasztaltam, hogy ahogyan az a 7. ábra bal oldalán is látszik akár több egész fokos eltérések is előfordultak két olvasás között. Ennek kiküszöbölésére kiegészítettem a függvényt úgy, hogy egy lefutás alatt 100-szor olvassa le a motorok szögállását, majd ezen 100 érték átlagát vegye és adja vissza. Ezzel a kiegészítéssel a függvény már sokkal konzisztensebb adatokat szolgáltatott. Ahogyan az a 7. ábra jobb oldalán is látható, a hiba kisebb lett, mint egy mikrolépés, mely korábban beállításra került, így biztosított, hogy helyes szögértékek érkezzenek a motorvezérlés visszacsatolásaként.

6.3 A megvalósított szabályozott mozgatás

A visszacsatoló függvény megléte után már könnyebb volt megvalósítanom a motorvezérlést, hiszen nem kellett figyelembe vennem, hogy a motorok mekkora lépésekben haladnak. Tovább segítette a fejlesztést, hogy így akár minden léptetés után kiolvasható lett a motorok szögállása, ezáltal az sem okoz gondot, ha a motorok átugranak egy vagy több lépést a mozgás során.

Az új motorvezérlő függvényt a 8. ábrának megfelelően kezdtem el kialakítani. Ahogyan az első megvalósított motorvezérlésnél, úgy az újnál is, a függvény bemenetének a végeffektor kívánt helyzetéhez tartozó motorállások szögadatainak kell lenniük. Ez azért praktikus, mert a végeffektor bárhol is legyen a munkatérben, már kiolvashatók az érzékelők segítségével a hozzá tartozó motorállások, így egyszerű pont és pont közötti mozgásokat megvalósítani. A sakktáblán való játékhoz pedig ez megfelelő, hiszen a mezőkhöz tartozó szögadatokat eltárolva a robotkar könnyedén képes lesz mozgatni a bábukat a megfelelő helyre. Ezt követően szükséges beolvasni a motorok kiindulási helyzetét, hogy meg tudjuk határozni, a motorok forgási irányát. Ezen beolvasott értékek segítségével továbbá az is megállapítható, hogy szükséges-e mozgatni az egyes motorokat, hiszen előfordulhat, hogy valamely motor már az elérni kívánt állásban van. Ha ezt megtettük a végső feladat, hogy az egyes motorokat addig léptessük amíg el nem érik a kívánt szögállásokat, melyet a visszacsatolással minden lépés után le tudunk ellenőrizni. Ezzel a módszerrel a robot két pont között úgy valósítja meg a



8. ábra Motorvezérlés folyamatábrája

mozgást, hogy mindig a legkevesebb lépést kelljen megtenni a motoroknak, azonban nincs egy meghatározott pályája a mozgásnak a pontok között.

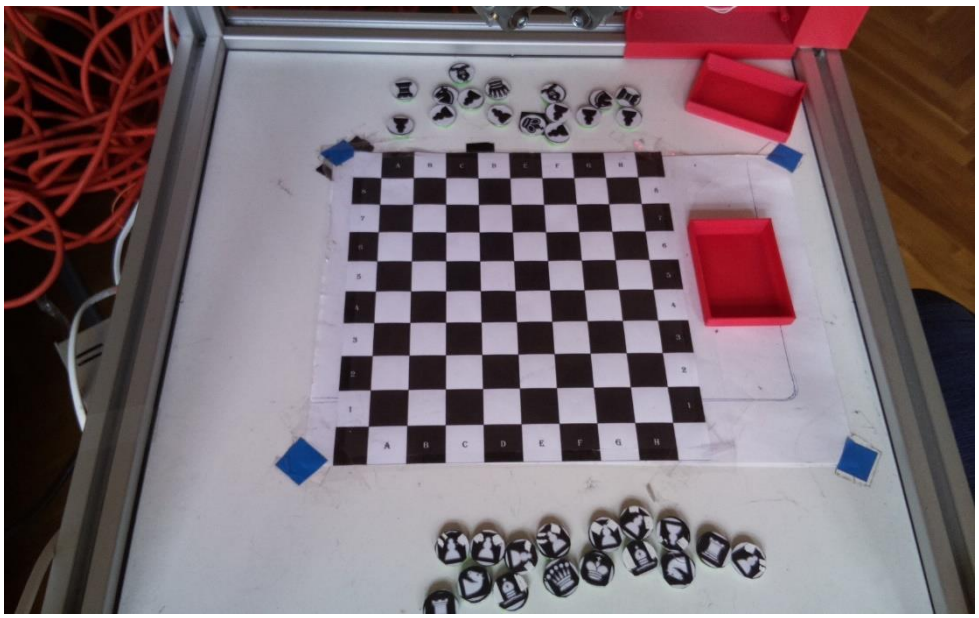
A motorvezérlő függvény megvalósításakor először úgy ellenőriztem azt, hogy az adott motor elérte-e a kívánt állást, hogy az elérendő szögérték egy környezetével hasonlítottam össze az aktuális motorállást. Annak érdekében, hogy a karok pontosan álljanak meg, a célértékhez képes 2 mikrolépéssel nagyobb és kisebb határok között vizsgáltam az aktuális szöget, így bármely irányba is forgott a motor elég volt egy feltételt vizsgálni: a motor éppen az elérni kívánt érték körüli sávban van-e. Ez a megoldás azonban nem bizonyult elég stabilnak a tesztelés során, mert ugyan ritkán, de előfordult, hogy a motor több lépést is átugrott, mint amennyit ez a sáv lefedett, így a ciklus nem állt meg. A lépések kihagyása nem konzisztens, így a sáv növelésével csak a robot pontosságát csökkentettem volna. Ehelyett egy újabb függvényt hoztam létre, mely a motor forgásának iránya, a jelenlegi és az elérni kívánt szögérték alapján meghatározza, hogy a motor túllépett-e a kívánt értéken. Így, ha a motor átugrik lépéseket akkor is biztosan megáll, ha pedig nem hagy ki lépéseket a végeffektor kellő pontossággal megérkezik a kívánt helyre.

Az így megvalósított vezérlőfüggvény a tesztelés alatt már kellő pontossággal tudta mozgatni a végeffektort az elérni kívánt pontba. A mozgatás után a motorok, a függvénynek kiadott cél 1°-os környezetébe érkeztek meg a tesztek során.

7. Képfeldolgozás

7.1 Képfeldolgozás megvalósításának alapkérdései

Ahhoz, hogy a delta robot képes legyen sakkozni, szükséges a sakktábla, illetve az azon elhelyezkedő bábuk detektálása. Erre egy célszerű módszer a számítógépes képfeldolgozás, melyhez adott a Raspberry Pi 4 Model B és a hozzá tartozó kamera modul. Elsőként fontos figyelembe vennünk a robot felépítését, hiszen a kamera elhelyezése nagyban befolyásolja a képfeldolgozás sikerességét. Fontos, hogy a kamera és a sakktábla között ne legyen semmi mikor az képet készít, úgy kell elhelyezni, hogy a robotkar nagy részében a készített képen kívül legyen. Erre a legjobb megoldás, ha a kamera elhelyezése az egyik robotkarral szemben történik a robot bázisán, mert így a karok által alkotott háromszög egyik oldala felől készül a kép, ami miatt a robot kart nem szükséges a munkaterület legszélére kimozgatni. A képfeldolgozási módszereknél továbbá nagy előnyt jelent, hogyha a kép mindig ugyan abból a pozícióból készül, illetve, ha az elkészült kép minél inkább a teljes felülnézethez közelít. Előbbi segítségével egyszerűbb képfeldolgozási módszerek alkalmazhatóak, ami csökkenti a számítási időt, utóbbira pedig azért van szükség, mert így a tábláról készült kép kevésbé torzított és ezáltal pontosabban érzékelhető.



9. ábra A Raspberry Pi kameramodul által alkotott kép

Ennek a kameraelhelyezésnek is megvan a hátránya: a tábla messze van, így az elkészített kép csak egy részét teszi ki maga a tábla. A 9. ábrán a sakktábla közelítőleg a kép 20%-át teszi ki, a tábla 10 x 10-es, hogy a sorok és oszlopok megnevezései is láthatóak legyenek, ezért egy mező a tábla századát teszi ki. Ebből és a kamera felbontásából közelítőleg számítható egy mező mérete, mely 3280 x 2464 / 500 = 127 x 127 pixelnek adódik, ami megfelelően nagy az egyes mezőket és a rajtuk lévő bábukat külön is detektálni.

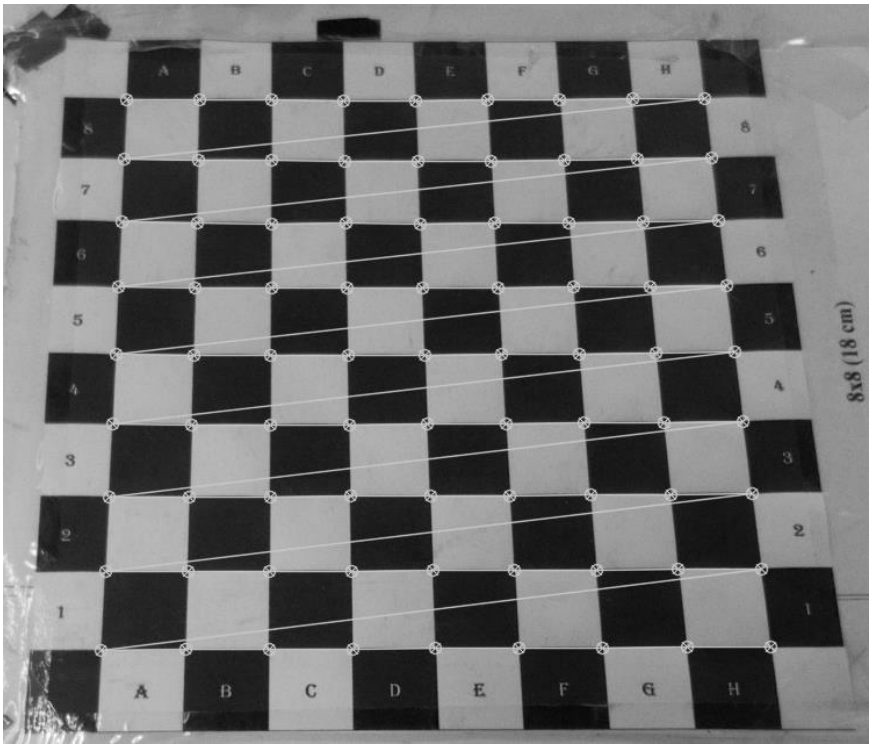
A képfeldolgozásban szoftveresen az OpenCV (Open Source Computer Vision) könyvtárai segítenek, mely nyílt forráskódú és a gépi látást implementálásához tartalmaz függvényeket. Több ezer használatra kész és optimalizált algoritmust tartalmaz képfeldolgozáshoz, mely több programozási nyelven is elérhető. A sakktábla detektálásához például a könyvtárban adott egy findChessboardCorners() nevű függvény, melyet kamerák kalibráláshoz szoktak használni. Emellett a könyvtár típusainak segítségével egyszerű algoritmus készíthető a bábuk vagy sakklépések felismerésére is.[11]

7.2 Tábladetektálás megvalósítása

Ahhoz, hogy lépéseket lehessen detektálni a sakktáblán, szükséges tudni, hogy az egyes mezőket a képen mely pixelek reprezentálják. Ezen terület meghatározásának legegyszerűbb módja a sarokpontok helyeivel lehetséges, hiszen így minden irányban megállapíthatóak a mező határai. A beolvasott kép az OpenCV egy saját típusában tárolható, mely a kép egyes pixeleit egy koordinátarendszerbe rendezi. Ezen koordinátarendszerben az x értékek balról jobbra az y értékek pedig fentről lefelé növekednek, azaz a bal felső pixel koordinátái (0;0). A mezők sarokpontjainak eltárolására ezen koordinátarendszer alkalmas. [12]

Ahogyan arra a függvény neve is utal, az OpenCV könyvtárban található findChessboardCorners() alkalmas egy sakktábla minta belső sarokpontjainak detektálására. A függvény bemeneti paraméterei a tábla sorainak és oszlopainak száma, kimenete pedig egy tömb, amiben a sarokpontok pixelkoordinátái találhatóak. Mivel a robot munkaterében lévő sakktábla minden oldalán ki van egészítve egy sornyi mezővel a sorok és oszlopok neveinek jelölésére, a tábla 10x10 méretű. Ennek köszönhetően a függvény által meghatározni képes belső sarokpontok egybe esnek a játékhoz használt mezők sarokpontjaival. A függvény kimenetéből származó koordináta tömb elemei, a tábla egyes mezőihez rendelhetőek négyesével. Mivel a kamera a játék közben nem

mozdul el, így ezt a detektálást elég egyszer megvalósítani és az így elmentett és összerendelt adatok segítik a további képfeldolgozást.



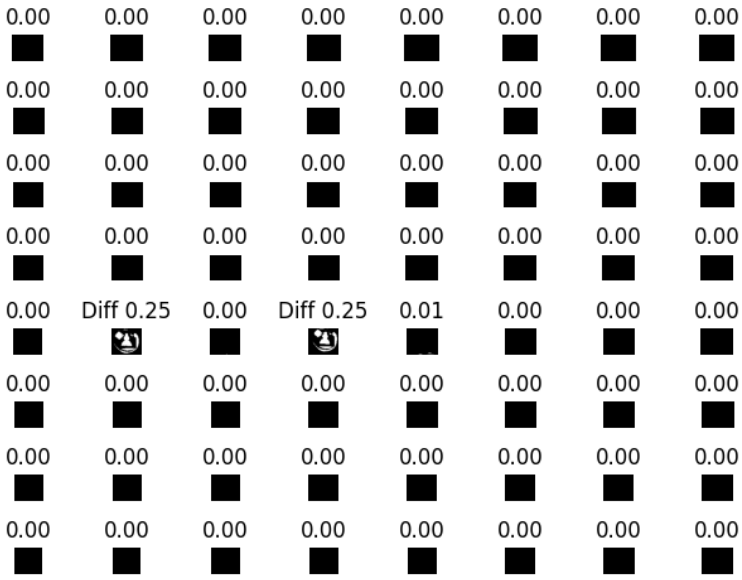
10. ábra A sakktábla belső sarokpontjainak detektálása

7.3 Lépésdetektálás megvalósítása

A tábla detektálása után a következő feladat a sakklépések detektálása, melyhez az előzőekben a mezőkhöz eltárolt adatok is felhasználásra kerülnek. A detektálási módszerek közül egy olyan került kiválasztásra, mely egyszerű, de más módszereknél kevésbé érzékeny a fényviszonyokra és arányaiban kevés számítással ad eredményt a munkatérben történt változások meghatározására. A módszer alapötlete az, hogy minden sakklépés előtt és után szükséges készíteni egy képet, majd a két kép közötti pixelkülönbségek és az előzőekben meghatározott mező helyek segítségével meghatározható, hogy mely mezőkön történt változás. Mivel a sakkban a kiindulási helyzet mindig ugyan az, így abból, hogy melyik mezőkön történt változás és a játék szabályaiból egyértelműen meghatározható, hogy milyen lépés történt és melyik bábuval. Az ezzel a megoldással megkapott mezők száma később feldolgozható sakkprogram segítségével, hogy a lépés helyes-e.

A lépés előtt és a lépés utáni kép közötti különbség a következőképen határozható meg: A két képet először fekete-fehérré kell átalakítani, melyre az OpenCV könyvtár függvényei adnak lehetőséget, így minden képpont fényesség értke lesz eltárolva az adott képet reprezentáló tömbben. A következő lépés, hogy egy harmadik tömbben el kell tárolni az előzőekben említett két tömb azonos indexű elemeinek különbségének abszolút értékét. Így minden pixelhez egy olyan érték lesz rendelve a tömbben mely azt reprezentálja, hogy milyen mértékben tér el a két kép az adott pixelhelyen egymástól fényességben. Ezen értékek négyzetét véve azokon a helyeken, ahol a két kép között nagy fényesség béli eltérés van a meghatározott érték még nagyobb lesz, ahol azonban nem volt nagy különbség az érték nem lesz sokkal nagyobb. Ezeket a négyzetes értékeket egy magas küszöbértékkel szűrve, megkaphatóak azok a tömbelemek azaz pixelek a képen, ahol a detektálni kívánt változás történt. A szűrt tömbben a lényeges változást tartalmazó pixelek 1, a többi pixel pedig 0 értékkel reprezentálható, így egy bináris képet adva arról, hogy hol történt jelentős változás. A bináris képen az adott mezőkhöz tartozó 1-es értékű pixeleket összeszámolva egy százalékos változás számítható, hogy az egyes mezőkön mekkora változás történt. Végül ezt a százalékos értéket egy újabb általunk választott határértékkel összehasonlítva a program el tudja dönteni, hogy mely mezőkön történt releváns változás. Kimenetében pedig a függvény ezen mezők számát adja tovább a sakkozó algoritmusnak.

A 11. ábrán látható a lépések detektálásának vizualizációja, a mezők fölött a hozzájuk tartozó pixelek releváns változásának arányszáma látható, Diff felirat, ahol megfelelő mértékű a változás, valamint a bináris kép a változásról.



11. ábra A mezőkön megjelenő változások

A tesztek során finomhangolásra került a pixelek szűrésére használt határérték, valamint a mezőkhöz használt százalékos érték. Továbbá megvizsgáltam, hogy mely bábuk mely mezőszínen a legkevésbé detektálhatók. Azt találtam, hogy megfelelő fényviszonyok között a világos gyalog és huszár a sötét mezőkön a sötét huszár pedig a saját színén produkálja detektálás szempontjából a legkisebb változást.

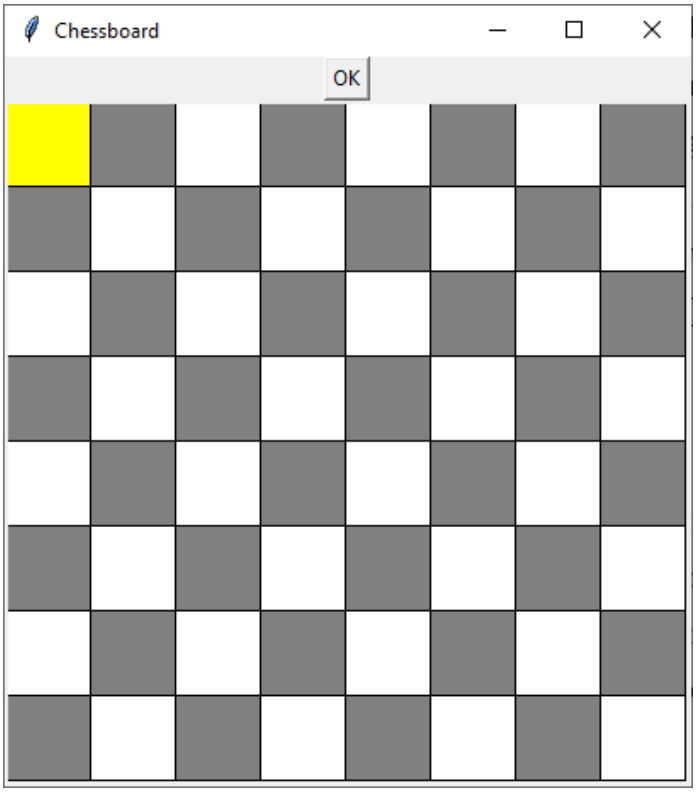
8. Szoftver a sakkozás mögött

8.1 A sakkozó motor

A képfeldolgozás által detektált lépés szabályosságának és helyességének ellenőrzésére egy a sakkozást is megvalósítani képes nyílt forráskódú könyvtár került felhasználásra melynek neve Sunfish. A könyvár kommentek és üres sorok nélkül mindössze 131 sor, azonban sakkmotorja játék közben képes megközelíteni a sakkmesterek szintjét. Ahogyan azt a dokumentációja is írja könnyen átalakítható, így egy olyan kiegészítés lett hozzá implementálva, mellyel a program a játékos lépését nem a billentyűzetről, hanem a képfeldolgozó modultól kapja meg. A program továbbá folyamatosan számontartja a játék állását így a megkapott kiindulási és cél mezők és a sakk szabályainak segítségével meg tudja határozni, hogy mely bábuval történt a lépés, így ellenőrizve annak helyességét.[13]

8.2 Kiegészítő szoftveres elemek

Ahhoz, hogy a modulok együtt tudjanak működni, szükséges volt még további kiegészítő szoftverek létrehozására is: A motorvezérlő modul kiegészítésére létrehoztam egy programot, mely képes beolvasni és eltárolni egy többdimenziós adattömbben a sakktábla egyes mezőihez tartozó motorállásokat, valamint a tábla fölött egy állást a mezők megközelítésére, egy kiinduló állást, hogy egyik robotkar se legyen a kamera és a tábla között a képfeldolgozás alatt és egy állást a tábla szélén, hogy a leütött bábukat is tudja hová tenni a robot. A program indulásakor a 12. ábrán látható grafikus felületet kapjuk, mely reprezentálja a sakktáblát. A végeffektort a megfelelő mezőre helyezése után az OK gombra kattintva a program beolvassa a mezőhöz tartozó szögértékeket, majd kijelöli a következő mezőt, ahol ismét a gombra kattintva beolvashatóak a következő mezőhöz tartozó értékek. A 64 mező beolvasása és eltárolása után pedig a különleges állások értékeit olvashatjuk be az előzőekben említett sorrendben.



12. ábra A betanító program grafikus felülete

A motorvezérlés további kiegészítésére szolgál egy függvény mely illeszti a motorvezérlő modult és a sakkmodult: Az illesztéshez először a sakkmodul által szolgáltatott mező számot kell a megfelelő tömbelemhez rendelni, mely hozzárendelés után a sakklépés megvalósítása a következőképpen működik: A motorvezérlő modul a sakkmodultól megkapja a lépés kiindulási és cél mezőinek számát, valamint azt, hogy a lépés egy bábuleütés-e hiszen ekkor más szekvenciát kell végrehajtani, mint egy átlagos lépésnél. Általános lépésnél a kapott mezőszámokhoz rendelt tömbelemek tartalmával a függvény először meghívja a motorvezérlő függvényt, mely először a kiindulási mező fölé helyezi a végeffektort, majd a mezőn elhelyezkedő bábura. Ekkor a függvény bekapcsolja az elektromágnest, majd a bábut a tábla fölé emeli. Innen a célmező fölé mozgatja a bábut, majd lehelyezi a táblára azt és kikapcsolja az elektromágnest, ezek után pedig visszaáll alap állapotba a tábla szélére, hogy jól látható legyen a tábla. Bábuleütés esetén az előzőekben ismertetett módon először a levenni kívánt bábut fogja meg a robot, majd a tábla széle fölé mozgatja, ahol belehelyezi egy tálkába és ezek után végrehajtja a levett bábu helyére a lépést.

8.3 A szoftveres elemek együttes működése

A modulok együttes működése a következőkben kerül ismertetésre: A játékot a fehér játékos kezdi mindig, azaz a felhasználó. A szoftver az LCD kijelzőn keresztül jelzi a játékos felé, hogy kezdődhet a játék. A kijelző alatt található gomb megnyomásával elkezdődik a játék és a kamera készít egy képet a tábla állásáról. Ezek után a felhasználó lép egyet, majd a gomb ismételt megnyomásával a program még egy képet készít. Az algoritmusban meghívásra kerül a képfeldolgozó szoftvert modul, mely a lépés előtt és után készült képek segítségével meghatározza, hogy mely mezőkön történt a változás és ezt átadja a sakk modulnak. A modul a kapott értékeket átalakítja a sakkban használt mező nevekre. Például: az négyes oszlop hetedik mezője az d7. Ha a képfeldolgozó szoftver ’47’-et ad vissza, akkor azt a sakkmodul ’d7’-re alakítja át. Az átalakított mezőszámot a sakkmotor veszi át, ellenőrzi a lépés helyességét, majd, ha helyesnek találja generál egy ellenlépést. Ha nem találja megfelelőnek a lépést azt az LCD kijelzőn keresztül jelzi a felhasználónak. Az ellenlépés generálása után a program kiadja az utasítást a motorvezérlőknek, hogy az előző fejezetben említett módon helyezze át a bábut az egyik mezőről a másikra. A mozgás után a kar alaphelyzetbe áll vissza, hogy ne takarja ki a kamera képét, majd a program készít ismét egy lépés előtti képet, hogy a felhasználó következő lépését is tudja detektálni, kiírja a kijelzőre, hogy ismét a felhasználó jön, majd a folyamat kezdődik elölről. Ha a játéknak vége, mert valamelyik fél nyert, az eredmény az LCD kijelzőn megjelenítésre kerül.

9. Összegzés

A specifikációban meghatározott feladatokat maradéktalanul sikerült teljesítenem a projekt megvalósítása során, mely az egyes modulok fejlesztés közbeni folyamatos tesztelésének köszönhető. A robot képes egy sakkjátszmát lefolytatni emberi játékos ellen. A megvalósított képfeldolgozás segítségével képes detektálni a felhasználó által megtett sakklépést, melyre a sakkprogram segítségével képes válaszlépést generálni. Végül a lépéshez szükséges bábuáthelyezést képes megtenni a motorvezérlő modul segítségével. Mindeközben az LCD kijelzőn információt szolgáltat a játék menetéről ellenfelének. Ezzel a működéssel játékosan szemléltetve egy ipari robot működését és bemutatva egy alternatív felhasználását.

A szoftveres modulok fejlesztése során kifejezetten figyeltem arra, hogy könnyebben továbbfejleszthető legyen a robot. A motorvezérlés függvényének bemenetei kompatibilisek inverz kinematikai számítások eredményével, így ez könnyen illeszthető a már megvalósított programhoz. Az inverz kinematika segítségével a robot képes lenne a munkatér teljes területén előre mentett pontok nélkül is bármilyen pontba eljutni. Egyszerűbb fejlesztések tekintetében pedig a betanító program tud segíteni, mivel úgy alakítottam ki, hogy a beolvasott állások száma állítható legyen, illetve a beolvasott tömbelemekre indexükkel könnyen és univerzálisan hivatkozhatunk.

A képfeldolgozás fejlesztésére is van lehetőség. A megírt modul felhasználásával megvalósítható más változások esetleg objektumok detektálása a képen, hiszen a program visszaadja a megváltozott pixelek helyét a képen. Ezek mellett a jelenlegi rendszer is javítható és finomítható a határértékek további tesztelésével és finomhangolásával vagy egy fix szórt fényű fényforrás beszerelésével a munkatér fölé, hiszen egy bizonyos mértékben ez a képfeldolgozási módszer is érzékeny a fényviszonyokra és azok változására.

A robot ismeretterjesztési funkcióját is képes ellátni. A projekt bemutatásra kerül az 59. Tudományos diákköri konferencián, valamint több egyetemi és szakmai rendezvényen is, például az Óbudai Egyetem Szakkollégiumok napján és a 27. Tavaszi Szél Konferencián.

10. Summary

In my project, my task was to implement an alternative use for a delta robot, which is commonly used in industrial environments. My main focus was on the software aspects of the device, which this thesis is about. The goal was to create a complex system consisting of different modules that would enable a delta robot to play chess against a human opponent.

At the University of Óbuda, Kandó Kálmán Faculty of Electrical Engineering, a delta robot built by former students was available for me to work on and conduct studies with. These types of robots are commonly used to perform pick and place operations in the industry. The process can be divided into two parts: first, you need to locate the object within the robot's workspace, then you need to grab and relocate the object with high precision. The goal was to create a similar process but with a twist, aiming to make robotics more popular among young people and anyone else interested.

A chess-playing robot meets all the criteria mentioned above because on a chessboard, precise movement is required to place the pieces, and a computer vision solution is needed to detect movements on the board so that the computer can calculate a response move against the player. The well-known game also helps visually demonstrate how an industrial robot actually works.

As detailed above, the project fulfills the expectations set beforehand: it is able to play a game of chess against a human player. The robot accomplishes this with three separate software modules. The first one uses image processing to detect the opponent's move. The second one allows it to generate a response to that move. The third one executes this move by lifting the chess pieces to the correct squares. These software components were designed to facilitate further development, so although the robot meets the specified requirements, there is still room for improvement.

The chess-playing delta robot was showcased at many events on campus, demonstrating its educational value to all generations.

11. Irodalomjegyzék

[1] A KUKA robotics hivatalos magyar nyelvű oldala

https://www.kuka.com/hu-hu/termékek-szolgáltatások/eljárástechnológiák/pick-and-place-robotok [Megtekintés utolsó dátuma: 2023.12.15.]

[2] Irati Zamalloa, Risto Kojcev, Alejandro Hernandez, Inigo Muguruza, Lander Usategui, Asier Bilbao and Víctor Mayoral: Dissecting Robotics - historical overview and future perspectives, (A robotika történelmi áttekintése és jövőbeli kilátásai) 2017. április

https://arxiv.org/pdf/1704.08617.pdf

[Megtekintés utolsó dátuma: 2023.12.15.]

[3] A svájci EPFL hivatalos oldalának archivált cikke a delta robot megalkotójáról

https://web.archive.org/web/20181027103218/http://sti.epfl.ch/page-76362-en.html

[Megtekintés utolsó dátuma: 2023.12.15.]

[4] Párhuzamos robotokról szóló tudományos blog.

https://www.parallemic.org/Reviews/Review002.html

[Megtekintés utolsó dátuma: 2023.12.15.]

[5] Solectro webáruház: a használt léptetőmotorok dokumentációjával

https://solectroshop.com/en/stepper-motors-for-3d-printers/619-motor-pap-nema-23-193kg-57hs762804.html

[Megtekintés utolsó dátuma: 2023.12.15.]

[6] A motorvezérlők adatlapja

https://www.powerbelt.hu/uploads/source/catalogs/DM556D_felhaszn%C3%A1l%C3%B3i_k%C3%A9zik%C3%B6nyv.pdf

[Megtekintés utolsó dátuma: 2023.12.15.]

[7] TME webáruház: SP-200 series tápegység dokumentáció

https://www.tme.eu/Document/4fc31215ee0db3ea3b629e14a72e2457/SP-200-SPEC.PDF

[Megtekintés utolsó dátuma: 2023.12.15.]

[8] Mouser webáruház: AS5048A dokumentáció

https://hu.mouser.com/datasheet/2/588/AS5048_DS000298_4_00-2324531.pdf

[Megtekintés utolsó dátuma: 2023.12.15.]

[9] Raspberry Pi kamera modul adatlapja

https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification

[Megtekintés utolsó dátuma: 2023.12.15.]

[10] Raspberry Pi 4 model B specifikáció, innen elérhető az adatlap is

https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[Megtekintés utolsó dátuma: 2023.12.15.]

[11] A teljes OpenCV dokumentáció

https://docs.opencv.org/4.9.0/

[Megtekintés utolsó dátuma: 2023.12.15.]

[12] Az OpenCV pixelkoordináta rendszere

https://medium.com/red-buffer/mastering-3d-spaces-a-comprehensive-guide-to-coordinate-system-conversions-in-opencv-colmap-ef7a1b32f2df

[Megtekintés utolsó dátuma: 2023.12.15.]

[13] Sunfish forráskód és dokumentáció

https://github.com/thomasahle/sunfish

[Megtekintés utolsó dátuma: 2023.12.15.]

[14] The Unimate robot that revolutionized the technological world (Az Unimate robotról szóló újságcikk)

https://www.koha.net/en/tech/331522/roboti-unimate-qe-beri-revolucion-ne-boten-teknologjike

[Megtekintés utolsó dátuma: 2023.12.15.]

[15] Xingguo Lu, Ming Liu: Optimal Design and Tuning of PID-type Interval Type-2 Fuzzy Logic Controllers for Delta Parallel Robots, Harbin Institute of Technology, Harbin, Heilongjiang, 2016.04.25.

https://www.researchgate.net/publication/303469087_Optimal_Design_and_Tuning_of_PID-Type_Interval_Type-2_Fuzzy_Logic_Controllers_for_Delta_Parallel_Robots

[Megtekintés utolsó dátuma: 2023.12.15.]

12. Ábrajegyzék