



INTÉZMÉNYI TUDOMÁNYOS DIÁKKÖRI KONFERENCIA

2022. november 23.

Objektumválogató robot vezérléstervezése

Designing the control of an object sorting robot

Készítette: **Bagi Gergő**
DZSJ1K
Mechatronikai mérnöki, alapképzés
Nappali
Szent István Campus

Témavezető: Tóth János, Egyetemi tanársegéd, Szent István Campus, Műszaki Intézet, Mechatronika tanszék

Lehetséges OTDK szekció: „Műszaki Tudományi Szekció”

**Gödöllő
2022.**

Tartalom

1.	Bevezetés.....	4
1.1.	Téma jelentősége	4
1.2.	Célkitűzés.....	5
2.	Szakirodalom feldolgozása	6
2.1.	Robotok csoportosítása	6
2.2.	Ipari robotok és kobotok alkalmazása.....	9
2.3.	Az ipari robotok típusai	9
2.4.	SCARA ipari robotok	10
2.5.	3D-nyomtatás.....	11
2.6.	Az Arduino.....	12
2.7.	Léptetőmotorok.....	13
2.8.	Gépi látás	15
2.9.	A gépi látás megvalósítása.....	16
2.10.	A gépi látó-rendszer komponensei.....	17
3.	Anyag és módszer	18
3.1.	Robotkar kiválasztás	18
3.2.	Robotkar nyomtatása	19
3.3.	Elektronikai elemek kiválasztása.....	22
3.3.1.	Motorok és vezérlők	22
3.3.2.	Arduino és a Shield.....	24
3.3.3.	Végálláskapcsolók	26
3.3.4.	A mágneset kapcsoló relé.....	27
3.3.5.	Elektromágnes	27
3.3.6.	Munkadarab érzékelő	28
3.3.7.	Tápellátás.....	29

3.4.	Elektronikus elemek kapcsolása	30
3.4.1.	A robotkar áramköre.....	30
3.4.2.	A szállítószalag áramköre.....	32
3.4.3.	A vezérlőeszközök hűtése	33
3.5.	Működtető program	33
3.5.1.	Arduino Mega.....	33
3.5.2.	Arduino Nano	37
3.5.3.	Python.....	38
4.	A megvalósult robot	41
5.	Összefoglalás.....	42
6.	Summary	43
7.	Irodalomjegyzék.....	44
8.	Mellékletek.....	47

1. Bevezetés

„Nem tudom definiálni a robotot, de felismerem, ha látok egyet.” – mondta Joseph Engelberger amerikai mérnök, akire ma a robotika atyjaként tekintenek. Az első robot prototípusát 1959-ben mutatták be, az eszköz a Unimate#001 nevet viselte. Ezt követően a General Motors még abban az évben telepítette az első robotkart, ez az esemény pedig magával hozta a robotipar megjelenését. Ebben az évben tehát a robotokkal kapcsolatos képzetek valósággá váltak és megindul a fejlődésük/fejlesztésük az USA-ban, Norvégiában, Svédországban, Németországban és Japánban is. [1]

1.1. Téma jelentősége

Napjainkban a robotika, mint iparág rohamosan fejlődik. Ennek egyik oka a „jövő gyárai” -nak megjelenése, vagyis a teljesen automatizált gyártás, amely nem igényel a karbantartáson túl emberi beavatkozást és egy gyártási folyamat végeredménye nem csak egy termék lehet, valamint az Internet of Things és az Ipar 4.0 gyakorlati alkalmazásának megjelenése. Ezen gyárak elengedhetetlen eszközei a robotok/robotkarok. Húzó ágazat az autógyártás, de találkozhatunk robotokkal a hadiiparban, űrtechnikában, egészségügyben, anyagmozgatást igénylő feladatoknál a szórakoztatóiparban és gyakorlatilag bármilyen más területen. A robotok és robotkarok elterjedésével és folyamatos terjeszkedésével egyenesen arányosan nő az igény az ilyen típusú gépek üzembehelyezéséhez, üzemeltetéséhez, karbantartásához értő szakemberekre. Ez a változás jelentősen befolyásolja a munkaerőpiacot is, hiszen a kézi erővel végzett munkát a robotok képesek felváltani, üzemeltetésükhöz lényegesen kevesebb emberi munkaerőre van szükség.

Az iparban alkalmazott kisebb méretű robotok egyik legjellemzőbb feladata a Pick-and-Place anyagmozgatási műveletek végrehajtása. A dolgozatom témája egy ilyen feladat ellátására alkalmas robotkar vezérlése. Erre a létező megoldások közül a 3 forgó -rotációs- tengellyel (forgóízület), valamint 1 egyenes vonalú, lineáris mozgást engedő -transzlációs- tengellyel (csúszó-ízület) rendelkező SCARA (Selective Compliance Assembly Robot Arm) mozaikszó pontos jelentése szelektív feladatok elvégzésére kialakított robotkar (RTRR) megfelelő. Ezt a robottípust Hiroshi Makino fejlesztette ki a Yamanashi Egyetemen 1979-ben.

1.2. Célkitűzés

A dolgozatom célkitűzése egy olyan léptetőmotorokkal szerelt négy tengelyes SCARA robotkar 3D nyomtatása és összeszerelése, amely optikai elven egy kamera segítségével képes felismerni az általam készített futószalagon érkező munkadarabok színét, majd a munkadarabokat rendre a saját színükhöz tartozó pozícióra szállítani. További cél, hogy a színfelismerést Python, a robot mozgatását pedig Arduino kód irányítsa.

2. Szakirodalom feldolgozása

Ebben a fejezetben bemutatásra kerül a dolgozatban felhasznált megoldások szakirodalma.

2.1. Robotok csoportosítása

A robotok olyan elektromechanikai szerkezetek, amelyek képesek különböző feladatokat végrehajtani programozásuk alapján. Irányításuk történhet ember vagy számítógép által. A robotokat sok különböző szempont alapján tudjuk csoportosítani. [2] [3]

Mobilitásuk alapján lehetnek:

- Mobil robotok
- Helyhez kötött robotok

Felhasználásuk alapján:

- Anyagmozgató
- Technológiai műveletet elvégző
- Szociális

Helyhez kötött robotok kinematikájuk alapján:

- Párhuzamos kinematikájú
- Láncolt kinematikájú
- Vegyes kinematikájú

Mobil robotok kinematikájuk alapján:

- Differenciális
- Holonomikus
- Egyéb

Koordináta-rendszer szerint:

- Henger
- Gömb
- Derékszögű
- Csuklós gömb

Tápellátás szerint

- Hidraulikus
- Pneumatikus
- Elektromos

Programozási metódus szerint:

- Kézi
- Lead-through
- Walk-through

Vezérlés szerint:

- Non-Servo
- Servo

Napjainkban a robotokat számos feladat elvégzésére alkalmazzák, az új megoldások megjelenésével feladatkörük folyamatosan bővül. Ennek okán feladatkörük szerint érdemes csoportosítani őket:

Ipari robotok (Industrial robots)

Az ipari robotok azonos jellemzői, hogy helyzetük megváltoztatásra nem alkalmasak, rendszerint olyan feladatokat látnak el, amelyek nagy precizitást és gyorsaságot követelnek meg, vagy monotonok, veszélyesek, nehéz fizikai munkát igényelnek. Sok különböző munkafolyamatban alkalmazhatóak, így a gyártástól kezdve, az összeszerelésen át a csomagolásig. Elsősorban az autógyártásban és a mikroelektronikában találkozhatunk velük.

[4]

Takarító vagy háztartási robotok (Domestic or household robots)

Ebbe a csoportba tartoznak a háztartásokban fellelhető robotok. Feladatuk, hogy elősegítsék a ház körüli munkák ellátását. Ilyenek a porszívó robotok, fűnyíró robotok. [2]

Orvosi robotok (Medical robots)

Azon robotok tartoznak ide, amelyekkel a gyógyszeriparban, orvostudományi területeken találkozhatunk, elsősorban a sebészeti robotok. [2]

Service robotok (Service robots)

Ide tartoznak azok a robotok, amelyek feladatkörük alapján nem sorolhatók egyik kategóriába sem. Jellemzően kutatást, szolgáltatást segítő adatgyűjtő, adatfeldolgozó robotok. [2]

Katonai robotok (Military robots)

Minden katonai alkalmazásban részt vevő robot, például a tűzszerész robotok, távirányítású felderítő gépek. [2]

Szórakoztató robotok (Entertainment robots)

Ez a kategória foglalja magába azokat a robotokat, melyek az emberek szórakoztatására készültek. Például ilyenek a táncoló robotok, vagy a kerekeken guruló ébresztőóra. [2]

Űr robotok (Space robots)

Űrkutatásban, űrtechnikában használatos robotok, ide sorolhatók a felderítő robotok, valamint a nemzetközi űrállomáson alkalmazott robotok. [2]

Hobby robotok (Hobby and competition robots)

A hobby robotok a felhasználó által kerülnek összeszerelésre, szintén a felhasználó által szabadon programozhatóak.

Léteznek olyan robotok is, melyeknek feladatköre nem dönthető el egyértelműen, mivel több különböző feladatra is megfelelnek. [2]

Felhasználásuk szerint a robotok kettő nagy csoportra bonthatóak: ipari és szolgáltató robotok. Az ipari robotok régóta jelen vannak az ipari automatizálás területén, a szolgáltató robotok elterjedése ezzel ellentétben sokkal elmaradottabb. A közelmúltban azonban számtalan fejlesztésre és kutatásra került sor a szolgáltató robotok területén, ennek eredményeként feladatkörük bővült és folyamatosan bővül. Az 1. táblázat szemlélteti az ipari és szolgáltató robotok közötti különbségeket. [2]

1. Táblázat Ipari és szolgáltatórobotok összehasonlítása [2]

	Ipari robotok	Szolgáltató robotok
Munkakörnyezet	Ellenőrzött és jól meghatározott környezet	Rendezetlenebb, nehezebben definiálható környezet
Felhasználók	Betanítás speciális feladatokra	A betanítás a tevékenységek széleskörű skáláját öleli fel
Biztonság	Gépfüggő	Robot és felhasználófüggő
Munkafilozófia	Robotok és emberek elkülönítése	Robotoknak és embereknek meg kell osztaniuk a munka-területen, hogy szolgáltatást nyújtsanak/kapjanak
Gép tervezése	Megbízásra rugalmasan reagál	Igényre rugalmasan reagál

2.2. Ipari robotok és kobotok alkalmazása

Ipari robotok a Robotika Nemzetközi Szövetségének (IFR – International Federation of Robotics) adatai szerint az autópárhban, elektronikai iparban és a gumi és –műanyagiparban a legelterjedtebbek, mivel ezek az iparágak rendelkeznek sok, könnyen automatizálható munkafolyamattal. A robotok nagyrésze az emberi munkaerőtől elkülönített robotcellákban végzi szolgálatait. Ennek oka a biztonságtechnikában keresendő. Ezek a robotok a megterhelő, monoton munkát veszik át az emberektől. Az automatizált gyártási folyamatokban a jobb munkavégző képességük mellett pozitívum még, hogy az embereknek több idejük marad a szellemi munkavégzésre. Tanos Áron, a Robotépítők Magyarországi Egyesületének elnöke szerint a robotok nem elveszik, hanem visszaveszik a gépi munkát az emberektől. [5]

Napjainkban kiemelt fejlesztési irány a kollaboratív robot. Amíg az ipari robotok cellákban végzik munkájukat az emberektől elkülönítve, addig a kobotok az emberrel közösen, együttműködve hajtják végre feladatukat. Ezeknél a robotoknál a sokkal nagyobb szerepet kap a biztonságtechnika. [5]

2.3. Az ipari robotok típusai

Az Ipar 4.0 célja, hogy létrehozza a kommunikációt az emberek és a gépek között. Ez a Human-Machine Interface (HMI), amely segíti a gyártási folyamatokban szereplő különböző robottípusok és emberek interakcióját.

A csuklós robotok

A legelterjedtebb robotkar típus a csuklós robotkar. Felépítése nagyon hasonló az emberi karhoz, részeinek megnevezése is azonos. A csuklós kialakítás miatt nagy munkatérrel rendelkeznek, végrehajtók pedig jellemzően a tér minden irányában mozgathatók. Ennek köszönhetően sok különböző gyártási folyamatban alkalmazhatóak.

A gömb koordinátás robotok

Ez a robotkar típus volt az első, amely képes volt részt venni az ipari gyártásban. Ezek a szerkezetek két forgócsuklóval és egy lineáris tengellyel, vagy csak forgócsuklókkal rendelkeznek. Alkalmazhatóak fröccsöntésnél, hegesztésnél.

A hengerkoordinátás robotok

Az ipari alkalmazásokban a hengerkoordinátás robotok jellemző feladata az anyagmozgatás.

A derékszög koordinátarendszerű robot

Ezek a robottípusok három lineáris elmozdulást engedő tengely mentén képesek elmozdulni, ezért nevezik őket x-y-z robotnak, valamint portál robotnak is, munkaterük téglatestként képezhető le. A többi típushoz képest mechanikájuk, működtetésük egyszerűbb és az áruk is kedvezőbb.

A Delta robotok

A Delta robotok vagy más néven párhuzamos robotok egy alpból és hat lineáris tengelyből épülnek fel. Előnyük, hogy nagy gyorsaságra képesek, szerkezetük merev és nagy ismétlési pontossággal rendelkeznek és alkalmazhatóak az élelmiszeriparban, mivel megfelelnek az ottani követelményeknek. Jellemzően pakolási feladatot látnak el szalagok mellett, kis súlyt képesek mozgatni, helyzetük általában függesztett. [6] [7] [8]

2.4. SCARA ipari robotok

A SCARA szó a Selective Compliance Assembly Robot Arm angol kifejezésből származó mozaikszó. Ez a robottípus képes a tér három irányában elmozdulást végezni, x y és z irányban. A SCARA ipari robotok mechanikai felépítése jellemzően két rotációs és egy transzlációs tengelyből áll. Ezek a robotok könnyűek, emellett gyors ciklusidőkre képesek viszont terhelhetőségük kicsi. A három tengellyel rendelkező kivitel után megjelentek a négytengelyes gépek. A negyedik tengely egy rotációs tengely a robotkar végén. A tárgyak megfogásánál, a robotkar végrehajtóját a forgó tengelyek segítségével a mozgatni kívánt tárgy fölé tudják így irányítani, majd a függőleges tengellyel közelítik meg a tárgyat. A rögzített lengőkar kialakítás bizonyos műveletekben akadályokba is ütközhet, korlátozott a hozzáférése bizonyos irányokban és helyzetekben. Ma korszerűnek csak a mikroprocesszorral vagy számítógéppel vezérelt robotok tekinthetők, melyek utasításkészlettel programozhatók. [6] [9] [10] [11]

2.5. 3D-nyomtatás

A 3D-nyomtatás, vagy másnéven additív gyártási folyamat során szilárd testek hozhatók létre digitális fájllokból, amelyeket egy számítógép állít elő. A gyártási folyamat során additívan, vagyis anyag hozzáadásával hozzák létre a kívánt testet. A nyomtatás során az olvadt műanyagból vékony rétegeket hoznak létre melyek egymáshoz tapadnak. A rétegek egymásra épülnek és előáll a kívánt geometria. A hagyományos szubtraktív eljárásokkal szemben, ahol a kívánt alakzat eléréséhez anyagot távolítunk el a munkadarabról, a 3D-nyomtatás nagy előnye, hogy lényegesen kevesebb anyagfelhasználást igényel akár bonyolult alakzat gyártása esetén is. [12] [13]

A 3D-nyomtatással gyártott alkatrészek tehát rétegekből állnak. Ezeket a rétegeket addig rakódnak egymásra, amíg el nem készül a kívánt alakzat. Ez előtt azonban létre kell hozni az alakzatot, amit gyártani szeretnénk. A nyomtatáshoz szükséges fájl létrehozása jellemzően CAD programok segítségével történik. Ezekben a programokban lehetőségünk van a gyártani kívánt alkatrész 3D modelljének létrehozására. Ilyen program a Solid Edge, a Fusion 360 vagy az Inventor. Ezt követően a kész modellt egy szeletelő alkalmazás bontja fel. Ezek az alkalmazások általában automatikusan képesek támasztékot generálni a modell azon részeihez, amelyeknél egyébként a levegőbe kéne rakni az anyagot. Ezek a támasztékok a nyomtatást követően könnyen eltávolíthatók az elkészült alkatrésztől. [12] [13]

Előnyök

- Gyorsabb gyártás
- Minőségi tervek és termékek
- Könnyű tesztelhetőség és integráció
- Könnyű testreszabhatóság
- Különböző méretek és formák
- Különböző anyagok

Hátrányok

- Drága
- Magas fogyasztás
- Károsanyag-kibocsátás

2.6. Az Arduino

Az Arduino egy napjainkban gyakran előforduló nyílt fejlesztőplatform. Az Arduinot eredetileg oktatási célra szánták, de jól alkalmazható személyi felhasználásban, egyszerű automatizálásoknál, valamint alkalmas lehet okosotthon kialakítására is. Ezen felhasználások mellett az iparban is megtalálható, több IoT (Internet of Things) eszköz alapjaként szolgál. Az Arduino fejlesztőplatform része az elektronikai eszközök és az ezekhez tartozó Arduino IDE szoftver. Az Arduino IDE egy C++-ra épülő Arduino programozási nyelvet használ. A nyílt forráskódnak köszönhetően az ehhez tartozó könyvtárak száma folyamatosan nő és bővül. A nyomtatott áramkörü lapok számos kivitelben elérhetők, ezeknek a legfontosabb része a mikrovezérlő. Ez egy integrált áramkör, ebben található a processzor, a memória és a perifériák. A leglényegesebb perifériák ezekben az eszközökben az időzítők, kommunikációs perifériák, ki- és bemenetek (analóg/digitális). Ezek a mikrovezérlők szinte kivétel nélkül minden digitális eszközünkben megtalálhatók a kisgépektől kezdve a nagygépekig egyaránt. Ezekre lehet példa a mobiltelefon, mosógép vagy a távirányító. Alkalmazásuk lehetőséget ad különböző szenzorok jelének feldolgozására, nyomógombok állapotának lekérdezésére, motorok vezérlésére, más eszközökkel való kommunikációra és még számtalan feladat ellátására. [14] [15] [16] [17]

Az Arduino áramkörü lapok véges csatlakozószáma miatt korlátozott a felhasználhatóságuk. Ennek a problémának a kiküszöbölésére hozták létre a hozzájuk csatlakoztatható bővítő áramköröket. Ezek a bővítő áramkörök, vagy más néven Shield-ek többlettudást biztosítanak az Arduino-nak. Sok különböző bővítő, kiegészítő érhető el hozzájuk, ezek lehetnek csak csatlakozószám növelők, de lehetnek egy speciális feladathoz rendelve, például CNC vezérléshez. [18]

2.7. Léptetőmotorok

A léptetőmotorok olyan egyenáramú szénkefe nélküli motorok, melyeknek egy fordulata lépésekre van felosztva. A lépések száma a motor kialakításától függ. Leggyakrabban egy fordulat, vagyis 360° 200 lépésre oszlik fel. Ebből könnyen meghatározható, hogy egy lépés $1,8^\circ$ -ot jelent. Ettől eltérően léteznek olyan motorok is, amelyek egy lépésre 2; 2,5; 5; 15; 30° -os szögelfordulást végeznek.

A léptetőmotor szerkezeti felépítését tekintve két részből áll, rotor és stator. A stator a motor álló része, ezen belül található a rotor, vagyis a forgórész. A rotor a statorral létrehozott forgó mágneses mezővel forog. A fémből készült állórészben találhatók az elektromágnesek, melyek meghatározott helyet foglalnak el. Az állórész elektromágnesének tekercseire feszültséget kapcsolva áram kezd folyni és létrejön a mágneses mező. A mágneses erőhatás és a mágneses fluxus a tekercsekben folyó áramerősségtől és áramiránytól függ. A tekercs energizálását követően az elektromágnes vonzza a forgórészben lévő permanens mágnes. Ez a hatás hozza létre az elfordulást. Ezt követően a forgórész úgy áll az állórészen belül, hogy a mágneses fluxus taszító hatása a legkisebb legyen. Amint egy másik elektromágnes bekapcsol, a forgórész új helyzetbe áll. Az elektromágnesek kapcsolásával változtatható a forgás iránya, valamint a részleges és teljes forgás. A teljes léptetés mellett megvalósítható féllépéses üzemmód is, ebben az esetben a forgórész két aktív tekercs közé áll be. A felbontás tovább növelhető, ezt mikrolépés üzemmódnak nevezik. A léptetőmotorokat olyan területeken alkalmazzák, ahol elengedhetetlen a pontos pozicionálás, ilyen például a CNC (Computer Numerical Control) vezérléssel rendelkező eszterga vagy marógép. A motorok vezérlése bonyolult vezérlőelektronikát igényel. [19] [20] [21]

Léptetőmotor típusok

A léptetőmotoroknál több típust különböztethetünk meg.

Állandó mágneses léptetőmotorok

Az állandó mágneses léptetőmotor két tekercs esetén négy változó pólussal rendelkezik, rotorja sugárirányban mágnesezett. A forgórész helyzetének megváltozását a tekercsben folyó áram irányának változása idézi elő, mivel a változik a mágneses pólusok orientációja. Az áram irányának egyszeri változtatása 90° -os elfordulást eredményez. Ilyen kevés póluspárral rendelkező motor esetén a pozicionálás pontatlan. Erre a problémára megoldást jelent a

póluspárok számának növelése. Több póluspár alkalmazása esetén a motor felbontása és pozicionálási lehetősége nagy mértékben növelhető. A műszaki gyakorlatban az ilyen, több póluspárral rendelkező motorokat alkalmazzák, ezeknél az elérhető léptetési szög $7,5-15^\circ$, amely lépésszámban 24-48 lépést jelent fordulatonként. A permanens mágnessel szerelt forgórész pólusai növelik a mágneses indukciót. Ennek eredménye, hogy ezek a motorok nagy nyomaték kifejtésére képesek. A léptetőmotorok ezen típusai egyszerű konstrukciójuk végett könnyen elérhetőek, viszont az egy fordulatra vonatkoztatott lépésszámuk alacsony. [19]

Változó reluktanciájú léptetőmotorok

A változó mágneses ellenállású (reluktancia) léptetőmotorok jelentek meg először a műszaki alkalmazásokban, azonban napjainkban nem terjedt el használatuk. A léptetőmotorok ezen típusában a forgórész fogait lágy vasból készítik. Az állórész tekercsiben folyó egyenáram mágneses mezője vonzza a forgórész fogait. Mivel a motor kapcsolása nem fokozatmentes, a rotor meghatározott szögelfordulást végez. Ezekre a típusokra jellemző az egyszerű felépítés, irányításuk könnyű, viszont alacsony felbontás és nyomaték érhető el velük. [19]

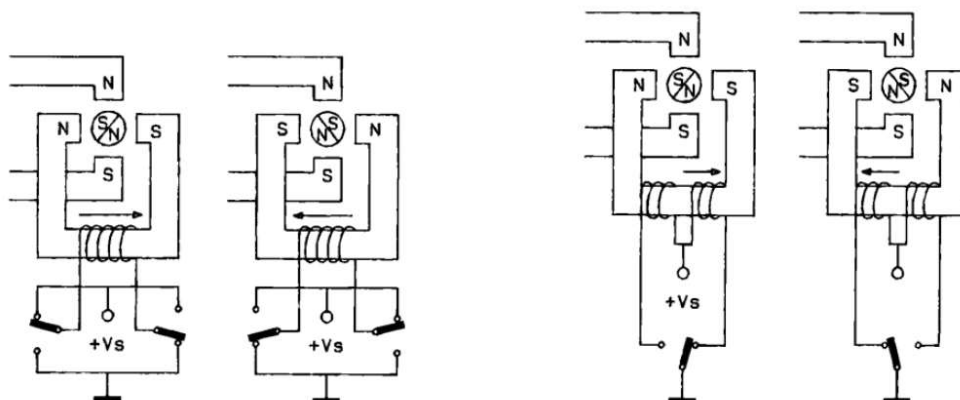
Hibrid léptetőmotorok

A léptetőmotorok típusai közül az iparban a hibrid léptetőmotorok terjedtek el legszélesebb körben. Előnye a többi típussal szemben, hogy magasabb felbontás érhető el velük ($0,9-3,6^\circ$), ebből következik, hogy több lépésre bontható az egyszeri körbefordulás (100-400 lépés/fordulat). A hibrid motorok megbízhatóbbak a többi típusnál, tartási nyomatékuk és az elérhető maximális forgási sebességük is nagyobb. Ezeknek a motoroknak a forgórésze állandó mágnesekből áll, viszont az állandómágneses kivitellel szemben a mágnesezésük axiálisan történik. A rotor két ellentétesen mágnesezett gyűrűből áll, ezek a motor tengelyén találhatók. A gyűrűkön réselt hornyok vannak, melyek a forgórész fogait képezik. [19]

Unipoláris és bipoláris motorok

A léptetőmotorok csoportosítása más megközelítésben az unipoláris és bipoláris motorokra különíthető el. Ezen osztályozás a kétfázisú motorok tekercselésén alapszik. A két típus különbsége a polaritásokban keresendő. Az unipoláris gépek egy árampolaritással működnek, a bipoláris gépek kettővel. Utóbbi esetben az áram iránya változtatható (1. Ábra). További különbség, hogy a motor tekercseinek kapcsolódniuk kell egymáshoz, hogy az áram az egyik

tekercs elejéről a másik végére kerülhessen. Az ily módon csatlakoztatott tekercsek esetén használható egy polaritású áram. [19] [22]



2.8. Gépi látás

Napjainkban nagy adatmennyiséget hordoznak magukban a képek és videók. Ezeknek a virtuális adatoknak a feldolgozását gyűjtőnéven gépi látásnak vagy képfeldolgozásnak nevezzük. A gépi látás olyan algoritmusokból áll, amelyek alkalmasak egy adott kép vagy videó megértésére, meg tudják állapítani mi látható egy képen, videóban. Ez a feladat egy szöveg értelmezéséhez hasonlóan bonyolult feladat. Az emberek számára a vizuális tartalom felismerése természetes, viszont a gépek nem rendelkeznek olyan általános tudással, mint az ember. A gépi látás tehát napjainkban is fejlesztés alatt áll a szövegek értelmezésével együtt.

Nagyon sok célalkalmazásban használható már a gépi látás, ahol egy adott képből kell valamilyen adatot a gép számára értelmezhetővé tenni. Két ilyen célalkalmazás:

Ipari minőségellenőrzés: Az iparban a gépi látást gyakran alkalmazzák a termékek minőségének ellenőrzésére. Kamerák segítségével vizsgálható a tárgyak színe, formája vagy más látható jellemzőjük. Az ellenőrzési folyamatok jól automatizálhatók, az emberekkel ellentétben a gépek nem alkalmazkodnak környezetükhöz (akkomodáció), így a vizsgálat eredménye minden esetben közel azonos. Ipari minőségellenőrzésre gyakorlati példa lehet a paradicsomok vagy szőlőszemek válogatása. Eladásra csak az érett, sérülésmentes zöldségek, gyümölcsök kerülhetnek, a vizsgálandó jellemzők a forma, a szín, valamint a felületi sérülések. Ezek a tulajdonságok automatizált folyamat keretein belül ellenőrizhetők számítógép segítségével. [24]

Képek szegmentálása: A képek gép által való feldolgozása esetén az első feladatok egyike a képek szegmentálása, vagyis a kép valamely részének különválasztása a többbitől. Jellemzően ez fontos és felesleges részek elkülönítését jelenti. A szegmentálás sokféleképpen történhet, szegmentálhatunk például szín, textúra, világosság alapján. Legtöbb esetben nem a képek bizonyos részei hordozzák magukban az értékes információt, hanem a különböző területek közötti élek (határvonalak). [24]

2.9. A gépi látás megvalósítása

Előfeldolgozás: Az előfeldolgozás lényege, hogy a feldolgozni kívánt képet előkészítse. Ehhez szűrőket használ. Ekkor történik a feldolgozandó kép felesleges részeinek eltávolítása, valamint a nem számottevő különbségek elhárítása. Példaként említhető a célpontdetektálás különböző időjárás vagy napszak esetén. Napos időben erős a kontraszt és világos a kép, felhős időben gyenge a kontraszt, éjszaka sötét a kép. Ezekben a különböző helyzetekben kell felismerni a célpontot. A kontraszt és a világosság azonos szintre állítását követően a művelet végrehajtható. A képek feldolgozásakor gyakran nem a színek, hanem az élek és a sarkok hordozzák az információt. Ezekben az esetekben az előfeldolgozás annyit jelent, hogy a beérkező színes képből szürkeárnyaltos képet készítenek, így megszabadulva a felesleges résztől. Azokban az esetekben, ahol viszont éppen a színeknek van jelentőségük ez a megoldás nem alkalmazható. Ilyen alkalmazás például a paradicsomok ellenőrzése. Ha a paradicsomok közül szeretnénk a sérülteket kiszűrni, a bemeneti adatnak csak a kék csatornáját vizsgálva egyértelműen azonosíthatók lesznek a foltok. A kép többi részét feleslegesnek minősíthetjük. [24]

Szegmentálás: Az előfeldolgozást követi a szegmentálás, ami a feldolgozandó kép kettéválasztását jelenti hasznos és haszontalan részekre. Az előfeldolgozással ellentétben nem csak a képpontok színe és világosság értéke alapján történik a szegmentáció, de a legtöbb esetben ezek alapján is szűr. [24]

Karakterisztikus jegyek: Az előfeldolgozáson és a szegmentáláson átesett képek már alkalmasak lehetnének arra, hogy különböző módszerekkel értelmezni tudják őket, de ez a gyakorlatban nem jellemző. Ennek oka, hogy a kép rengeteg képpontból áll, amely a legtöbb algoritmus számára problémát jelent, nem tudják kezelni. Az esetek túlnyomó többségében azonban erre nincs is szükség. Ha a problémát megfelelő irányból közelítjük meg,

meghatározható, melyek azok a vizuális tulajdonságok, amelyek felhasználásával elvégezhető a kívánt feladat. [24]

Felismerés és modellillesztés: Miután a karakterisztikus jegyek azonosításra kerülnek descriptorok (leírók) tömörítik magukba a feldolgozott kép hasznos információit. Ennek következtében ez az adat sokkal kezelhetőbbé válik a számtalan RGB képponttal szemben. [24]

2.10.A gépi látó-rendszer komponensei

Fényforrás: Ha a vizsgált objektum nem bocsát ki vagy nem esik rá fény, semmit sem lehet megfigyelni, feldolgozni. Ezért olyan tárgyak számára, amelyek nem bocsátanak ki fényt, elengedhetetlen a megfelelő megvilágítás.

Kamera: A kamera gyűjti össze a vizsgált tárgyról visszaverődő fénysugarakat. Ez a legegyszerűbb esetben egy optikai lencsével történik.

Érzékelő: A szenzor a fénysugarakat megfelelő jellé alakítja a további feldolgozáshoz. Ehhez képkalkotó rendszereknél egy kétdimenziós szenzortömbre van szükség a sugarak térbeli eloszlásának rögzítésére. Bizonyos esetekben elegendő egyetlen érzékelő vagy érzékelősor.

Feldolgozó egység: Feldolgozza a beérkező adatokat, kinyerve belőlük a megfelelő jellemzőket, amelyek alapján meghatározhatók a vizsgált objektum tulajdonságai. A feldolgozó egység másik fontos része a memória. Itt tárolódnak a vizsgálattal kapcsolatos információk, valamint a nem releváns adatok törlése is itt történik.

Végrehajtó: A végrehajtók reagálnak a gépi látó-rendszer eredményére és így ők is a rendszer részévé válnak. [25]

3. Anyag és módszer

Ebben a fejezetben bemutatásra kerül a szerkezet összeállítása, a hozzá tartozó elektronikai kapcsolás, valamint a működtető program.

3.1. Robotkar kiválasztás

A projekt elején a célkitűzés megfogalmazását követően kezdtem olyan, már meglévő robotkar konstrukciókat keresni, amelyek alkalmasak az általam definiált probléma megoldására. Az alábbi döntési mátrix alapján választottam ki a számomra legmegfelelőbb robotkar típust:

2. Táblázat Robotkartípus választási mátrix

	Csuklós robot	Gömbkoordinátás robot	Hengerkoordinátás robot	Delta robot	SCARA robot
Egyszerű szerkezet	1	4	4	2	4
Effektor orientációkezelése	1	1	5	5	5
Multifuncionalitás	5	4	4	4	4
Vezérlés egyszerűsége	1	2	5	2	5
Ipari elterjedés	5	2	2	4	5
Összesítés:	13	13	20	17	23

A mátrix eredménye a SCARA típusú robotkar lett, így tehát ilyen konstrukciókat vizsgáltam. A robot mechanika tervezésétől - az általam meghatározott feladat megfogalmazásakor - mentesítettem magam. Több megoldást is találtam, amely alkalmas a feladat ellátására, de a későbbi felhasználhatóság érdekében végül a howtomechatronics.com oldalán található robotot választottam. Ez volt az egyetlen olyan kész konstrukció, amely 4 tengellyel rendelkezik. Későbbi terveim között szerepel, hogy a dolgozat keretein belül használt robotkart lézer

gravírozóként, 3D nyomtatóként használjam, illetve párhuzamos megfogót is szeretnék hozzá végrehajtóként. Ezekhez elengedhetetlen a végrehajtót forgató negyedik tengely.

3.2. Robotkar nyomtatása

A howtomechatronics.com oldaláról letöltött 3D fájlokat szálhúzásos technológiával nyomtattam ki. A szálhúzásos (FDM) eljárás során egy felmelegített fúvókán keresztül nyomva a hőre lágyuló műanyag hozzátapad a nyomtatóasztalhoz (Bed). Ezt követően a következő rétegek mindig az előző rétegekre rakódnak, kihűlés után összetapadnak, így jön létre a 3D modell kézzel fogható mása. Ez az eljárás kiválóan alkalmazható egyedi gyártás, prototípusgyártás, valamint modellezésnél is. [26]

Az általam használt nyomtató berendezés egy Creality Ender 3 Pro (2. Ábra). Lényeges különbség az Ender 3-hoz képest, hogy az asztal lapja mágnessel van rögzítve, amely nyomtatás után levehető. Az asztallap minden irányban hajlítható, így a nyomtatott darab könnyen eltávolítható róla.

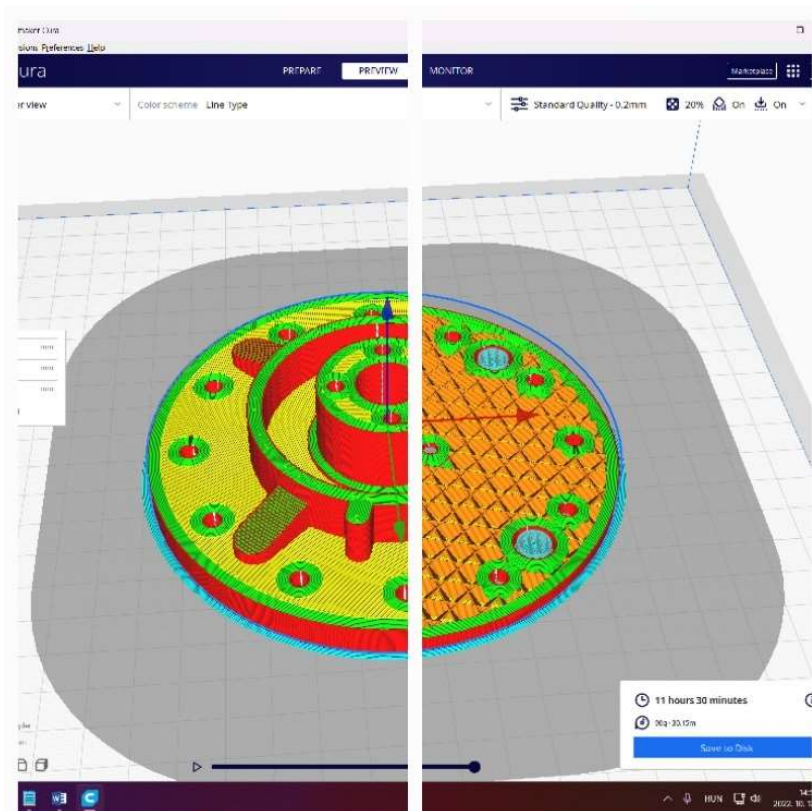


1. Ábra Creality Ender 3 Pro (forrás: 3djake.hu)

A nyomtatáshoz PLA (PolyLactic Acid) és PLA+ anyagokat használtam, mivel ezekkel az anyagokkal érhető el a legszebb nyomtatási felület, nem vetemednek és kellő szilárdsággal rendelkeznek a feladat végrehajtásához. Alternatív megoldás lehetett volna az ABS (akrilnitril-

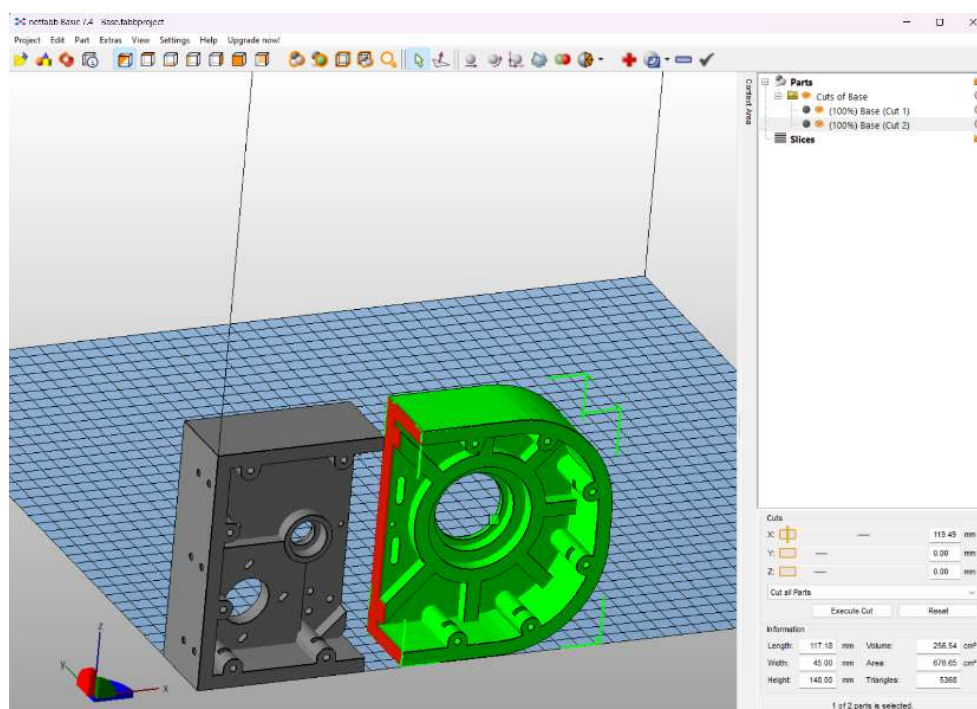
butadién-sztirol) valamint a PET-G (polietilén-tereftalát). Az ABS igen kemény, kopás és ütésálló anyag nagy szilárdsággal, hátránya viszont, hogy vetemedik, magas tálcahőmérsékletet igényel és zárt nyomtatási teret. A PET-G az ABS és a PLA tulajdonságai között képez átmenetet, nyomtatás közben a nagy hőmérséklet miatt hajlamos szárazodni, vagyis két pont között haladva szálakat hagy maga után.

A PLA nyomtatásánál minden alkatrész hasonló beállítással készült. A fűvóka hőmérséklete 210°C, a nyomtatóasztal 60°C, a nyomtatandó darabok falvastagsága 8 szál, kitöltésük 20%-os kocka alakú. A nyomtatáshoz 0,4mm-es fűvókát használtam. Ezeket a beállításokat a G-kódot generáló Ultimaker Cura nevű programban (3. Ábra) állítottam be, valamint figyelembe vettem a modellek asztalra helyezését követően, hogy a lehető legkevesebb támasztás generálása mellett készüljön el a G-kód az anyagpazarlás mellőzése okán.



2. Ábra Ultimaker Cura működés közben
(forrás: Cura, saját forrás)

A robotkar alkatrészeinek nyomtatása egy darab kivételével problémamentesen történt. A robot alapja befoglaló méretei miatt nem fért el a nyomtatóasztalon. Erre a nehézségre a Netfabb nevű program (4. Ábra) adott választ. Ebben a programban a 3D modellek x y és z tengely mentén szeletelhetők, a szeletelés után pedig nem hagyja nyitva a modellt, tömör testet ad vissza.



3. Ábra Netfabb képernyőkép
(forrás: Netfabb, saját forrás)

Nyomtatandó alkatrész volt még a mágnesstartó konzol, a ciklohajtómű és az infra érzékelőt tartó konzol, ezekről a következő fejezetekben található leírás.

3.3. Elektronikai elemek kiválasztása

Ez a fejezet rész tartalmazza az elektronikai alkatrészek kiválasztásának folyamatát.

3.3.1. Motorok és vezérlők

A robotkar mozgatásához léptetőmotorokat alkalmaztam, mivel nem igényelnek külső enkódert, könnyen vezérelhetők és a pozícióra állás is könnyen megvalósítható velük. Ahogyan a 3D-nyomtatóknál és kis méretű DIY CNC gépeknél, rajzgépeknél NEMA 17 típusú bipoláris léptetőmotorokat (5.Ábra) hasznának, úgy én is ezek mellett a motorok mellett döntöttem. Ezek a motorok kompatibilisek az Arduino fejlesztőplatformmal, tehát a célkitűzésben állított követelménynek megfelelnek.

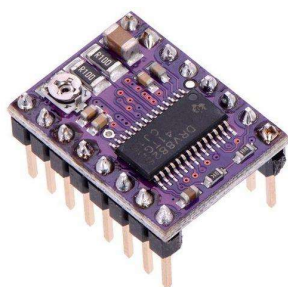


4. Ábra NEMA 17 Léptetőmotor
(forrás: Techfun.sk)

A léptetőmotorok megfelelő vezérléséhez motorvezérlőt használtam. A vezérléshez 3 motorvezérlőt vizsgáltam a létező megoldások közül:

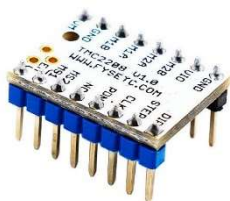
- DRV8825,
- TMC2208,
- A4988.

A DRV8825 (6. Ábra) a 3D nyomtatásban elterjedt kis méretű vezérlő modul. Társainál nagyobb feszültség mellett nagyobb a kimeneti áram, tehát nagyobb erő kifejtésre képes és 1/32-es mikrolépésre is képes. Esetemben semmi nem indokolja ezeknek a funkcióknak a használatát.



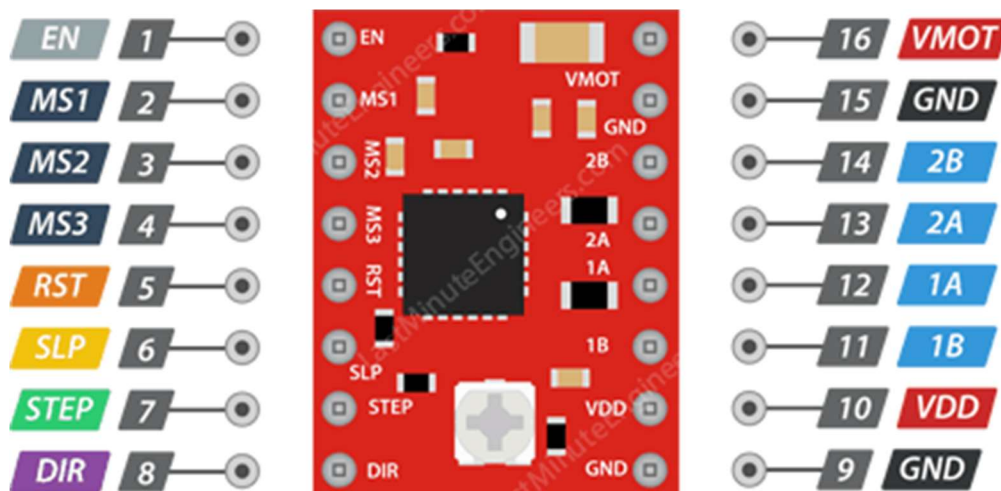
5. Ábra DRV8825 motorvezérlő (forrás: TavIR.hu)

A TMC2208 típusú vezérlőket (7. Ábra) olyan területeken használják, ahol elvárás a halk működés. Ez a vezérlő képes kiküszöbölni a motorzajt és a rezgést, ami a motor „szakaszos” lépkedéséből ered. Jellemzően magasabb felosztással rendelkezik és kisebb a kimeneti árama, mint társainak.



6. Ábra TMC2208 motorvezérlő (forrás: wiki.fysetc.com)

Az általam választott léptetőmotor vezérlő az A4988 (8. Ábra). Ezek a vezérlők 5-35V között működnek, maximális kimeneti árama 2A. Előnye, hogy mindössze 2 lábra van szükség a motor vezérléséhez. Ez a két láb felel a lépésközért (Step) és a forgásirányért (Dir). A maradék 14 lábra kell kötni a logikai 3-5,5V logikai jelszintet, a motorok tápellátásához szükséges 8-25V-ot, a motor tekercseit, valamint az MS1 MS2 MS3 lábakkal aktiválhatók a mikrolépések (Microstepping). A vezérlő hűtőbordát igényel.



7. Ábra A4988 motorvezérlő (forrás: lastminuteengineers.com)

A motorok tápellátásához 12V-ot használtam, az áramkorlátot a következő képlet szerint állítottam be 0,48 A-re.

$$\text{Áramkorlát} = V_{ref} \div (8 \cdot R_{cs}), \quad (1)$$

ahol:

V_{ref} – a referencia feszültség a potenciométer és a földpotenciál között, V

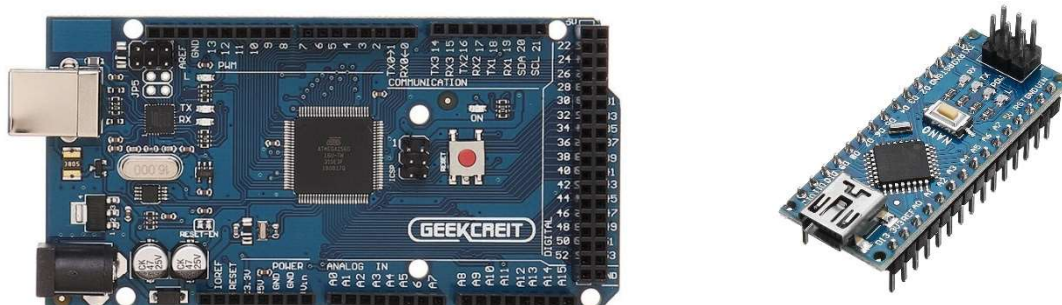
R_{cs} – az áramérzékelési ellenállás (current sense resistance), Ω

3.3.2. Arduino és a Shield

A célkitűzésben meghatározottak szerint a robotkar, valamint a munkadarabokat szállító szalag motorjainak mozgatásához Arduino vezérlőkártyákat használtam.

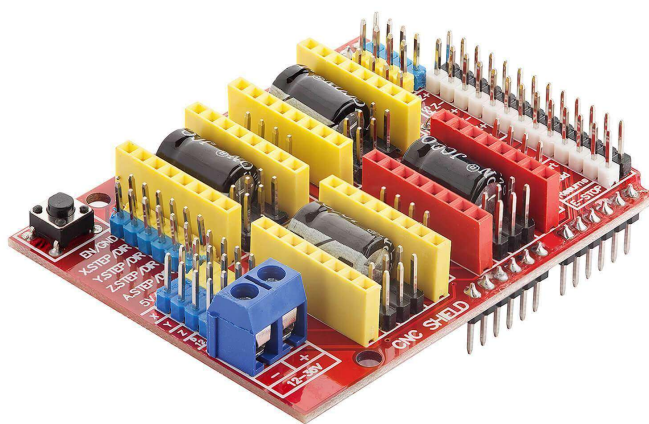
A robotkar mozgatásához Arduino UNO vezérlőkártyát szerettem volna használni. A robot tesztelése során ez elegendőnek bizonyult, de a Python kóddal való kommunikációt bevonva az ellátandó feladatai közé a munkafolyamat megghiúsult. Ennek okán váltottam Arduino Mega (9. Ábra) vezérlőkártyára. Ez az eszköz gond nélkül meg tudta valósítani a Python kódból érkező adatokra való mozgatást. Az Arduino lábainak kiosztására a következő fejezetekben térek ki.

A szállítószalag motorjának mozgatásához Arduino Nano-t (9. Ábra) alkalmaztam. Egy motor és egy szenzor kezeléséhez számítási kapacitása több, mint megfelelő. Az Arduino lábainak kiosztására a következő fejezetekben térek ki.



8. Ábra Arduino Mega és Nano (forrás: banggood.com)

A léptetőmotorok vezérléséhez használt A4988-as vezérlők több csatlakozást igényelnek logika és teljesítményoldalról egyaránt. Nem is lenne elegendő pin egy Arduino-n három motor vezérléséhez. Ennek a problémának a kiküszöbölésére készítettek az Arduino panelekhez úgynevezett bővítő paneleket, Shield-eket. Az általam meghatározott feladat végrehajtásához a CNC Shield V3.0 (10. Ábra) 4-tengelyes pajzsot választottam. Ez a bővítőkártyára négy A4988-as vagy hasonló 16 pin kiosztású motorvezérlőt képes vezérelni. A motorok tápellátását erre a panelra kell bekötni, a megengedett feszültség 12-36V. Ezt a pajzsot eredetileg az Arduino UNO panelhez gyártják, de problémamentesen csatlakoztatható nagyobb testvéréhez, az Arduino Megához. A bővítőpanel további funkciókra is alkalmas. Külön definiált lábakra csatlakoztathatók a végálláskapcsolók és mivel CNC panel, dedikált pin tartozik a marószerszám elindítására. Én ezt a lábat az elektromágnes ki- és bekapcsolására használtam.



9. Ábra CNC Shield V3 bővítőpanel
(forrás: megaeshop.pk)

3.3.3. Végálláskapcsolók

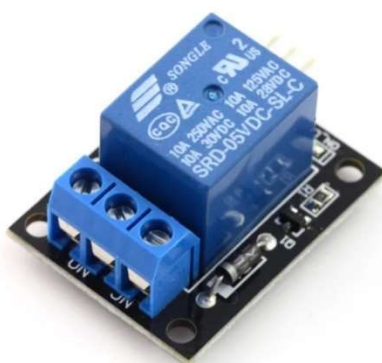
A robotkar négy tengelyéből hármat használtam, ezért három végálláskapcsolóra volt szükségem. A X-szel jelölt első forgó mozgást engedő tengelyhez, az Y-nal jelölt második forgó mozgást engedő tengelyhez és a lineáris mozgást engedő tengelyhez, ennek jelölése Z. Az általam használt görgővel ellátott kapcsolók (11. Ábra) 3 kivezetéssel rendelkeznek, a felhasználó dönthet arról, hogy NO (Normally Open / Alaphelyzetben nyitott) vagy NC (Normally Closed / Alaphelyzetben zárt) üzemben szeretné használni. Én az NO kötési módot választottam, de alkalmazható lett volna az NC megoldás is, hiszen a kapcsoló állapotváltozása hordozza magában az információt.



10. Ábra Végálláskapcsoló (forrás: gmelectronic.com)

3.3.4. A mágnes kapcsoló relé

Az elektromágnes be- és kikapcsolásához egy nyomtatott áramkörre szerelt fl-3ff-s-z típusú relét használtam (12. Ábra). A mágnes bekapcsolt állapotáról egy piros LED ad visszajelzést. A relé egyik felén a logikai egységhez tartozó tápellátás valamint a hozzá tartozó bemenet található (5V), a másikon a teljesítménykör. A végálláskapcsolóhoz hasonlóan itt is három csatlakozás található. A középső a közös csatlakozás, a két szélső közül a felhasználó választhatja ki milyen üzemet szeretne (NO/NC).



11. Ábra Relé (forrás: botland.store)

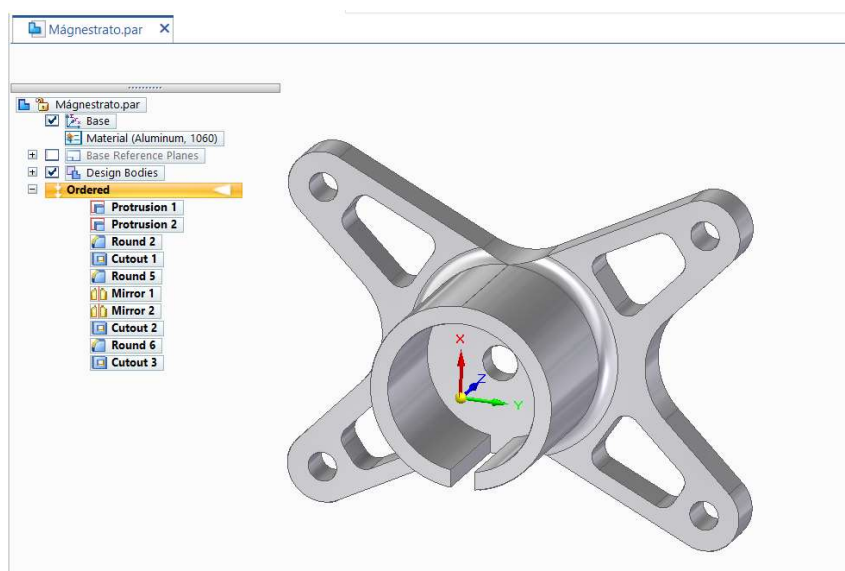
3.3.5. Elektromágnes

Az elektromágnes kiválasztásánál a teherbírás lehetett volna a fő szempont, azonban a hungarocellből készült munkadarabok súlya elhanyagolható, ezért a gyakorlatban található mágnesek közül a legkisebb teljesítménnyel rendelkezőt választottam. A mágnes a 13. ábrán látható, átmérője 20mm, teljesítménye 2,4W, tápfeszültsége 12V, teherbírása a leírás alapján 3kg.



12. Ábra Elektromágnes (forrás: hestore.hu)

A mágnes végrehajtóként való használatához valamilyen megoldást kellett találnom arra, hogy a mágnest a robotkar végére rögzítsem. A robotkar végén található karimán körkörös elhelyezkedő furatok azt a célt szolgálják, hogy a végrehajtó könnyen cserélhető legyen. Releváns volt egy ide rögzíthető elektromágnes-tartó konzolt tervezni. A tervezéshez a Solid Edge gépészeti tervezőrendszer diákok számára elérhető verzióját használtam. A tervezés során a finommechanikában használatos elvek szerint járhattam el, miszerint a terméket a méretek, a környezeti és a gyártási körülmények határozzák meg és nem a szilárdsági viszonyok. A tartókonzolt 3D nyomtatás technológiával gyártottam le, modellje a 14. ábrán látható.



13. Ábra A mágnes tartó konzol 3D modellje Solid Edge programban
(forrás: Solid Edge, saját forrás)

3.3.6. Munkadarab érzékelő

A futószalagon érkező munkadarab felvételi pozíciójára érkezését detektálni kell, és a szalag mozgását megállítani amíg a munkadarab ott van. Ehhez a feladathoz infravörös akadály érzékelő modult választottam. Az általam használt KY-032 (AD-032) típusú szenzor (15. Ábra) két potenciométerből két visszajelző LED-ből egy IR LED-ből és egy HS0038B infra érzékelőből áll. Működési elve, hogy az IR LED által kibocsátott infravörös fény a tárgyról visszaverődve (tárgyreflexió) érzékelhető az infra érzékelő számára. Előnye, hogy a két potenciométerrel az érzékelő látótávolsága nagyon pontosan beállítható, változtatható. A szenzor tápellátásáról a Pled (Power LED), az akadály jelenlétéről az Sled (Sense LED) ad információt. [27]



14. Ábra Infrared akadályérzékelő (forrás: hobbyking.com)

3.3.7. Tápellátás

A logikai elektronika tápellátását (5V) a programot futtató laptop biztosítja USB portokon keresztül. A teljesítményelektronika működtetéséhez szükséges 12V-os tápellátást egy számítógép tápegység szolgáltatja.

3.4. Elektronikus elemek kapcsolása

Az építő elemek kiválasztása után ebben a fejezetben bemutatásra kerül a kapcsolási körük. Két kapcsolási kör van. Az első a robotkarhoz tartozik, a második a munkadarabokat szállító szalaghoz.

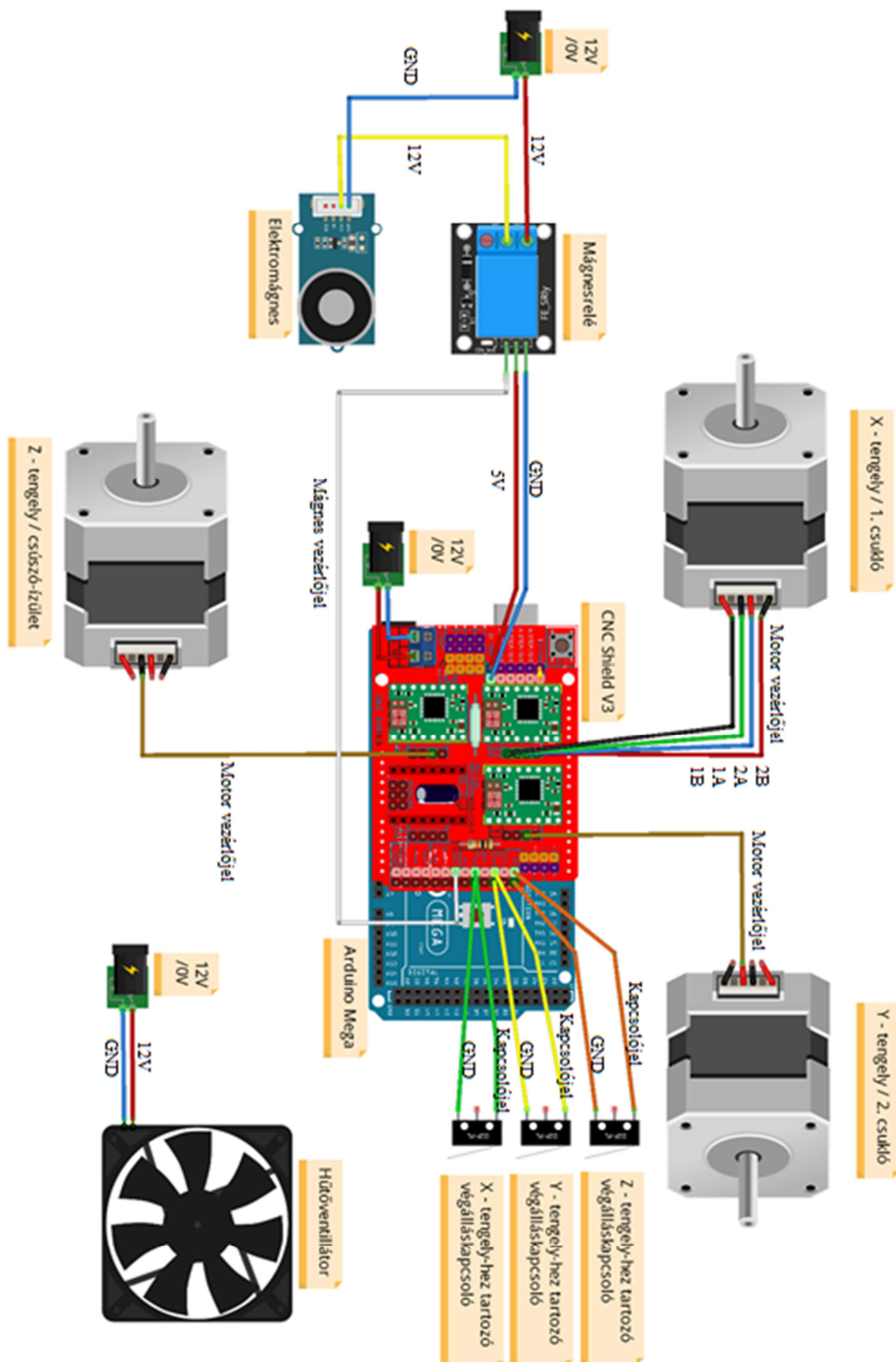
3.4.1. A robotkar áramköre

A robotkar áramköreiben 2 feszültség szint van, a logikai körhöz tartozó 5V-os jelszint, valamint a teljesítménykörhöz tartozó 12V.

Az Arduino Mega tápellátását a laptop USB portjáról kapja, a CNC Shield pedig az Arduino-tól. Az 5V-os jelszintet használja a mágneset kapcsoló relé, a végálláskapcsolók és a motorokat vezérlő A4988-asok is. 12V működteti a léptetőmotorokat és a mágnesrelét. A 12V-os tápegységről párhuzamosítóval kap tápot a CNC Shield és a relé. A CNC Shield aktiválásához össze kell kötni az EN (Enable) pinto a GND (Ground) pinnel. Ezt követően tudtam tesztelni a motorok és motorvezérlők működését. A motorvezérlők egyértelműen definiált, kialakított helyet kaptak a pajzson, minden pin helye meghatározott, feltüntetett. A felhelyezett motorvezérlők alatt helyezkedik el a simítókondenzátor. A léptetőmotorok négy kivezetésének sorrendje adott (piros, kék, zöld, fekete), azonban a fordított csatlakoztatás (fekete, zöld, kék, piros) forgásirányváltozást idéz elő. Az esetleges fordított bekötés a programban előjelváltással korrigálható.

A mágneset működtető relé a pajzs 5V/GND lábairól kapja tápellátását, a vezérlő jelet pedig a SpnEn pinről.

A végálláskapcsolók csatlakozási helye a bővítőpanelen jelölt, a könnyebb értelmezhetőség miatt érdemes ezeket a jelöléseket használni. A végálláskapcsolókat csak negatív irányban használtam, funkciójuk csak a nullázás (Homing – következő fejezetben részletezve). A túlfordulás védelme abban mutatkozik meg, hogy a vezérlőszoftver nem adhat ki a mozgásterén kívül eső értéket. A robot elektronikai összeállítását a 16. ábra tartalmazza.



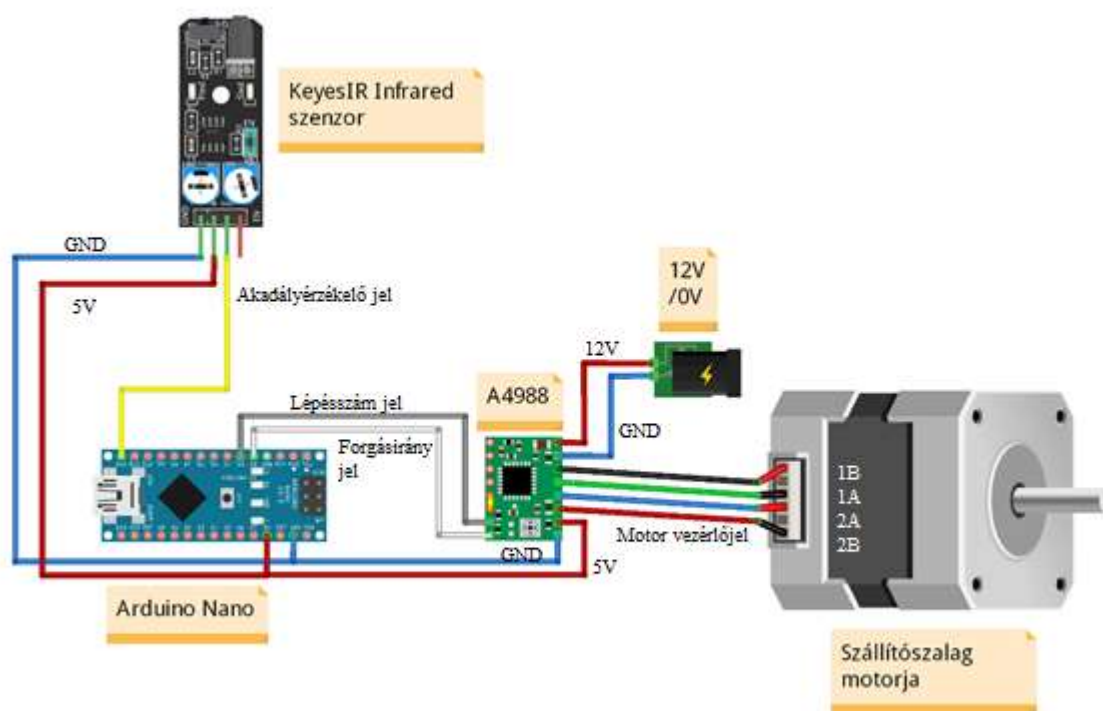
15. Ábra A robot elektronikai összeállítása (forrás: Fritzing, saját forrás)

3.4.2. A szállítószalag áramköre

A robotkar tápfeszültségigényéhez hasonlóan itt is 5V és 12V feszültség szint van.

Az Arduino Nano a Megával megegyezően laptop-ról kapja tápellátását, az Infravörös akadály érzékelő és a motort vezérlő A4988 az Arduino 5V/GND lábairól párhuzamosítóval, a szalagot mozgató motor pedig a vezérlőn keresztül 12V-os tápról kapja a működéséhez szükséges energiát.

A KeyesIR szenzor az Arduino Nano D12-es lábára küld jelet, ha akadályt érzékel. A D3-as lábról küldhető adat a vezérlőnek a motor lépésszámát illetően, a D2-es lábról a motor forgásiránya vezérelhető. A szállítószalag elektronikai összeállítása a 17. ábrán látható.



16. Ábra A szállítószalag elektronikai összeállítása (forrás: Fritzing, saját forrás)

3.4.3. A vezérlőeszközök hűtése

„Harmadik” körként értelmezhető az elektronikát hűtő ventilátor, aki a kapcsolásában egyedüli elemként van jelen. Tápellátása 12V-ot igényel.

Fontos megjegyezni, hogy az egész rendszer egy darab 12V-os tápegységről működik. A feszültségszint párhuzamosítón keresztül jut el az építő elemekhez.

3.5. Működtető program

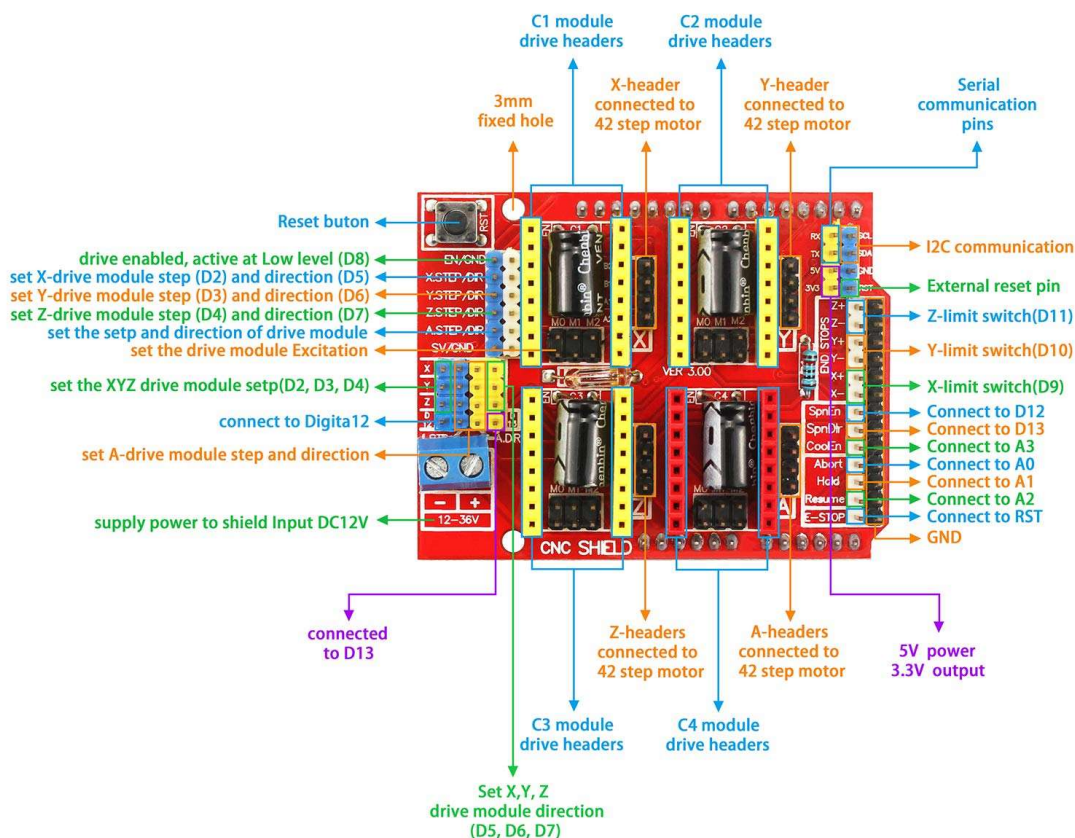
Az elektronikai elemek kiválasztása és kapcsolását követően ismertetem a működtető szoftvert.

3.5.1. Arduino Mega

Az Arduino Mega vezérli a robotkar motorjait, kapcsolja az elektromágneket és kezeli a végálláskapcsolókat. A léptetőmotorok egyszerű kezeléséhez az AccelStepper nevű könyvtárat használtam, ezen felül importáltam még a MultiStepper könyvtárat, amely több léptetőmotor egyidejű mozgatását teszi lehetővé (`#include <AccelStepper.h>; #include <MultiStepper.h>`).

A könyvtárak importálása után definiáltam a végálláskapcsolók csatlakozó lábait, valamint az elektromágneshez tartozó pin-t (`#define limitSwitchx y; #define magnetPin 12`).

A motorok deklarálásához segítségként a CNC Shield V3 pinout (lábkiosztás) ábráját használtam (18. Ábra). A motorok megadásához szükséges a könyvtár neve (AccelStepper), a motor neve, amivel később hivatkozni szeretnék rá, majd kerek zárójelben a motorvezérlés típusa, a lépésszám és a forgásirány pin. A motorvezérlés típusa esetenben motorvezérlővel történik, amelyhez 2 vezérlőpin tartozik. Ennek a típusnak az értéke 1.



17. Ábra CNC Shield V3 lábkiosztása (forrás: forum.arduino.cc)

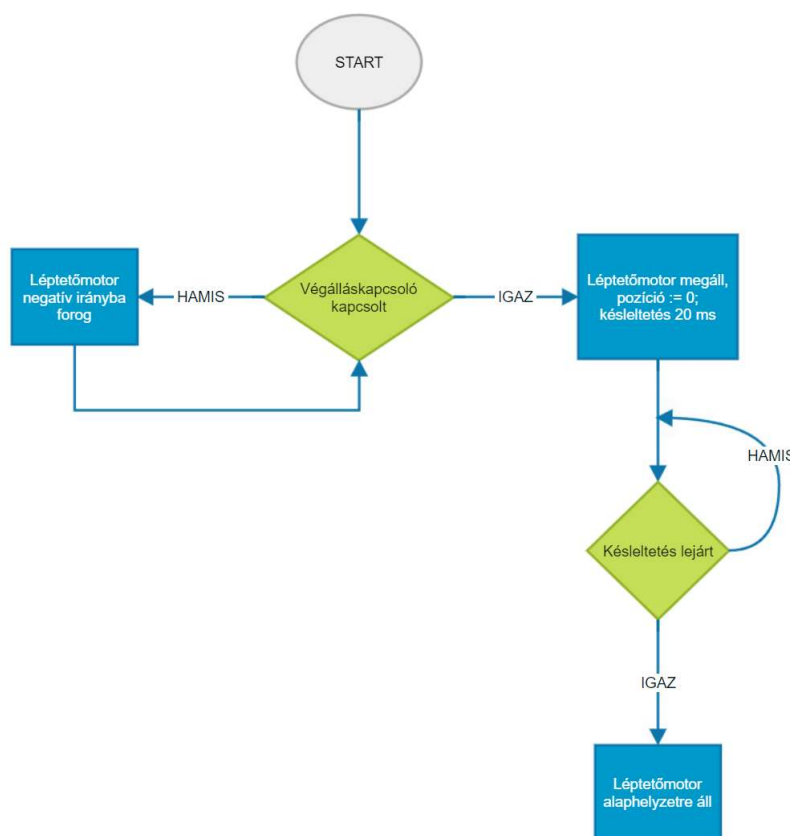
A motorok meghatározása után létrehoztam egy változót *steppers* néven, amit az 1-es és 2-es motor egyidejű mozgatásához használok fel (MultiStepper steppers;).

A program setup része előtt tettem meg a felhasználandó változók deklarálását. A soros porton érkező adatok mentéséhez hoztam létre az *msg* az *x* és *y* változókat String adattípusként, mivel a soros portra küldött adat is String típusú. Az AccelStepper könyvtárban szereplő parancsok azonban számtípust igényelnek, ezért szükség volt az *a* és *b* változók létrehozására int (egész szám) adattípussal. Ezekbe a változókba alakítottam át az *x* és *y* String típusú értékeket.

A setup programrészben megadtam a végállaskapcsolók és a mágnes vezérlőpinjének módját. A végállaskapcsolók jelét bemenetként, a mágnes vezérlőpinjét kimenetként definiáltam. A végállaskapcsolóknál aktiváltam a felhúzóellenállásokat, ezzel elkerülve a jel lebegését. A robot biztonsága érdekében minden motorhoz tartozóan megadtam a maximálisan

megengedhető sebességet és gyorsulást, így a motorok ezeket az értékeket nem lépik túl. Végül a *steppers* – korábban definiált – változóhoz hozzáadtam a *stepper1* és *stepper2* nevű motorokat és meghívtam a homing programrészt.

A homing programrész a setup-hoz hasonlóan egyszer fut le a setup rész végén, a programrész működését a 19. ábra tartalmazza. Feladata, hogy a robot bekapcsolását követően minden kart külön-külön a végálláskapcsolóig mozgasson, majd nullázza pozícióját. Ez a művelet után értelmezhetők az adott pozíciók. A motorok mozgatása while ciklusokkal történik. A ciklus feltétele, hogy amíg a végálláskapcsoló nem kapcsol, addig közelítsen hozzá a kar. Ha a kapcsoló kapcsol, a motor 0-ról az általam megadott alaphelyzet értékre áll be. A homing programrész lefutását követően a soros porton megjelenik az „OK” felirat, amely a Python kóddal való kommunikációhoz szükséges.

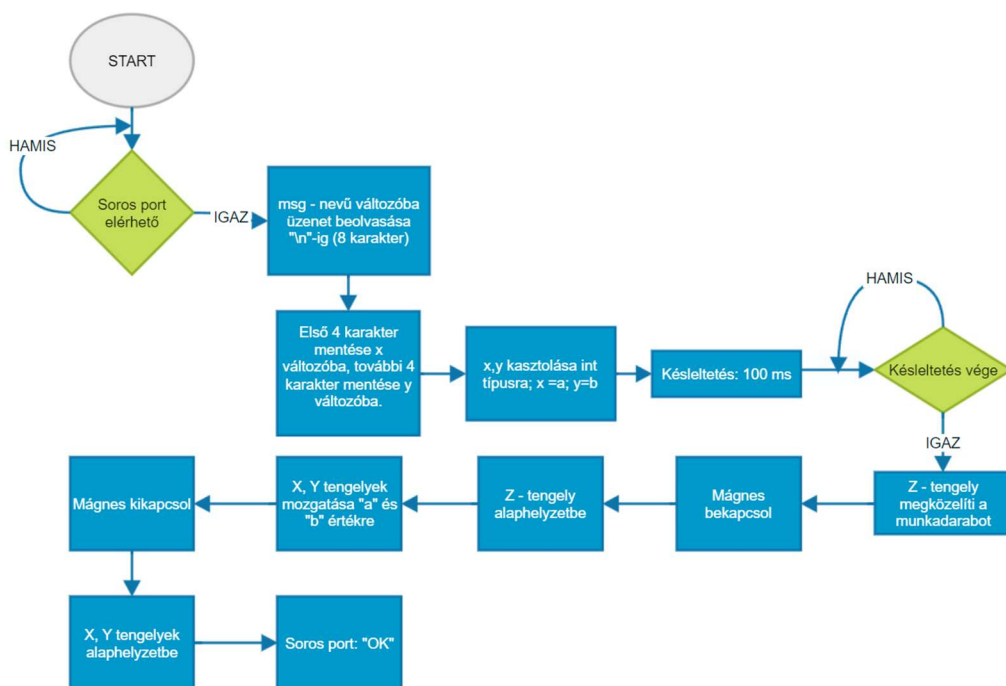


18. Ábra A homing függvény folyamatábrája
(forrás: gliffy.com, saját forrás)

A főciklusban végtelenítve fut a szelektáláshoz szükséges mozgást végző program és a kommunikáció. A loop elején létrehoztam egy kételemű int típusú tömböt *positions* néven, így hivatkozhatok egyszerre két pozícióra két motor egyidejű mozgásához.

A Pythonból soros porton érkező adatok kezeléséhez egy „if” utasítást használtam. Amennyiben a soros port elérhető az *msg* változóba megtörténik az adatsor rögzítése. Ezt követően a *substring* paranccsal szétválasztom a 8 karaktert 4-4 karakterre, ezeket az *x* és *y* változóknak tárolom el. Az értékek későbbi felhasználásához alakítom át a karakter típust egész számmá a *toInt* függvénnyel. Az egész szám értékek az *a* és *b* változóknak tárolódnak.

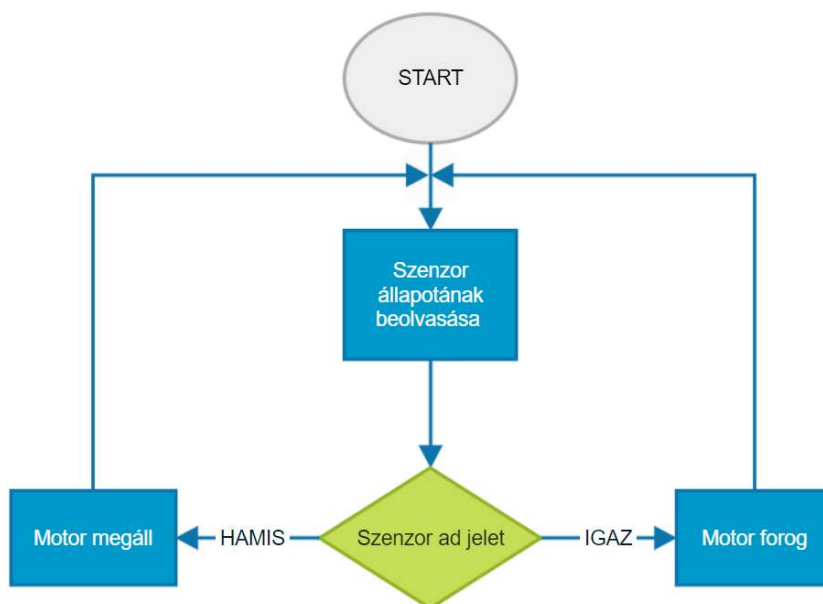
Ha az adat megérkezett, a robotkar megközelíti a munkadarabot és bekapcsol az elektromágnes, majd 1 másodperc késleltetéssel visszaáll a Z – tengelyen alaphelyzetbe. Ezt követően a két rotációs tengely egyidejű mozgással a munkadarabot a színe szerint kiválasztott helyre szállítja. Érkezés után a mágnes elenged, a tengelyek pedig visszatérnek alaphelyzetükbe és a soros porton megjelenik az „OK” felirat. A translációs tengely motorjának mozgásához a *stepper.runToNewPosition(0);* függvényt, a rotációs tengelyekhez A *steppers.runSpeedToPosition();* függvényt hívtam meg. Az utóbbi függvény hátránya, hogy a gyorsulást nem tartalmazza. A mágnes be és kikapcsolását a *digitalWrite()* függvénnyel végeztem. Az program folyamata a 20. ábrán látható.



19. Ábra A robotvezérlő folyamatábrája (forrás: gliffy.com, saját forrás)

3.5.2. Arduino Nano

Az Arduino Nano a szállítószalag mozgatásáért felel. A szalag hajtását GT2 bordás szíjjal egy NEMA 17 típusú léptetőmotor végzi. A motor mozgatásához ebben az esetben is az AccelStepper könyvtárat használtam. A motort *stepper* néven hoztam létre, mozgatása vezérlővel történik a D3 és D2 lábakon (*AccelStepper stepper (1, 3, 2)*). Definiáltam az IR akadályérzékelő szenzort a D12-es lábon, majd létrehoztam egy változót *sensorState* néven. Ez a változó fogja tárolni a szenzor állapotát. A setup programrészben a *sensorPin*-t bemenetként értelmeztem, valamint megadtam a motor legnagyobb sebességét és gyorsulását. A főciklusban megtörténik a szenzor állapotának beolvasása a D12-es *sensorPin*-ről, majd egy „if” elágazásban lekérdezem a szenzor állapotát (HIGH/LOW). Ha az érték magas, az érzékelő előtt nincs munkadarab. Ebben az esetben a szalag mozog, szállítja a munkadarabokat. Ellenkező esetben, ha a szenzor akadályt érzékel a motor sebessége 0-ra csökken és mindaddig áll, amíg a robotkar el nem veszi a szenzor elől a munkadarabot. A folyamat a 21. ábrán látható.



20. Ábra A szállítószalagvezérlő folyamatábrája
(forrás: gliffy.com, saját forrás)

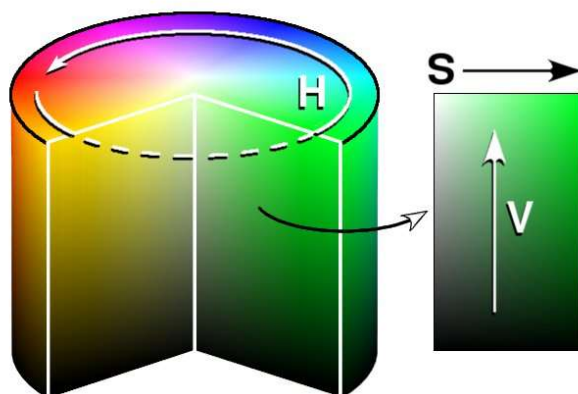
3.5.3. Python

A Python kód feladata egy kamera segítségével felismerni a munkadarabok színét és a színekhez tartozó pozíció értékeket kiadni az Arduino számára.

A színek detektálására az Open Computer Vision (OpenCV) könyvtárat használtam. A *cap* nevű változóba helyeztem a kamera képét a *VideoCapture(0)* függvényvel. Több kamera esetén a zárójelben lévő értékkel választhatjuk ki, melyiket szeretnénk használni. Ezt követően beállítottam a kamerakép felbontását 640x480 pixelre és létrehoztam a *send* nevű boolean típusú logikai változót True kezdetiértékkel, melynek a kommunikációban lesz szerepe.

A főciklusban létrehoztam egy keretet (frame), amelyben megjeleníthető a kamera képe. Fontos kiemelni, hogy amennyiben nem a ciklusban olvastatjuk be a kamera képét, élőkép helyett a program futtatásakor készült pillanatképet kapjuk vissza.

Következő lépésben létrehoztam egy új keretet azért, hogy az elsődleges BGR formátumú színteret HSV másodlagos színtérre konvertáljam. A BGR (blue, green, red) vagy RGB formátum esetén minden egyes képpont 3 értékkel írható le. A három érték a három szín különböző mértékű keverésével határozza meg a különböző színeket. Esetemben ennek a formátumnak a használata nagyban nehezítené a feladat teljesítését. A probléma BGR színtérnél a következő. Egy adott szín érzékelésekor három érték tér vissza, a kék, a zöld és a piros mértéke. Ha a detektálni kívánt test megvilágítása megváltozik, a három szín keverésének mértéke is változik igen nagy mértékben. Tehát BGR esetén három értéket kellene vizsgálni és viszonylag nagy intervallumokon belül egy objektum színének meghatározásához. Ezzel szemben a HSV másodlagos színtérnél (22. Ábra) elegendő egy érték változását vizsgálni egyszerű színek azonosításához és az intervallumok is könnyen behatárolhatók. A HSV (hue, saturation, value) színteret úgy hozták létre, hogy az ember színérzékeléséhez hasonló legyen. A Hue a színárnyalat, a Saturation a telítettség, a Value pedig az érték vagy fényerő. Egy adott szín felismeréséhez elegendő az első értéket, vagyis a színárnyalatot vizsgálni, hiszen ez egy skálán tartalmazza a piros, narancssárga, citromsárga, zöld, kék és lila színeket. [28]

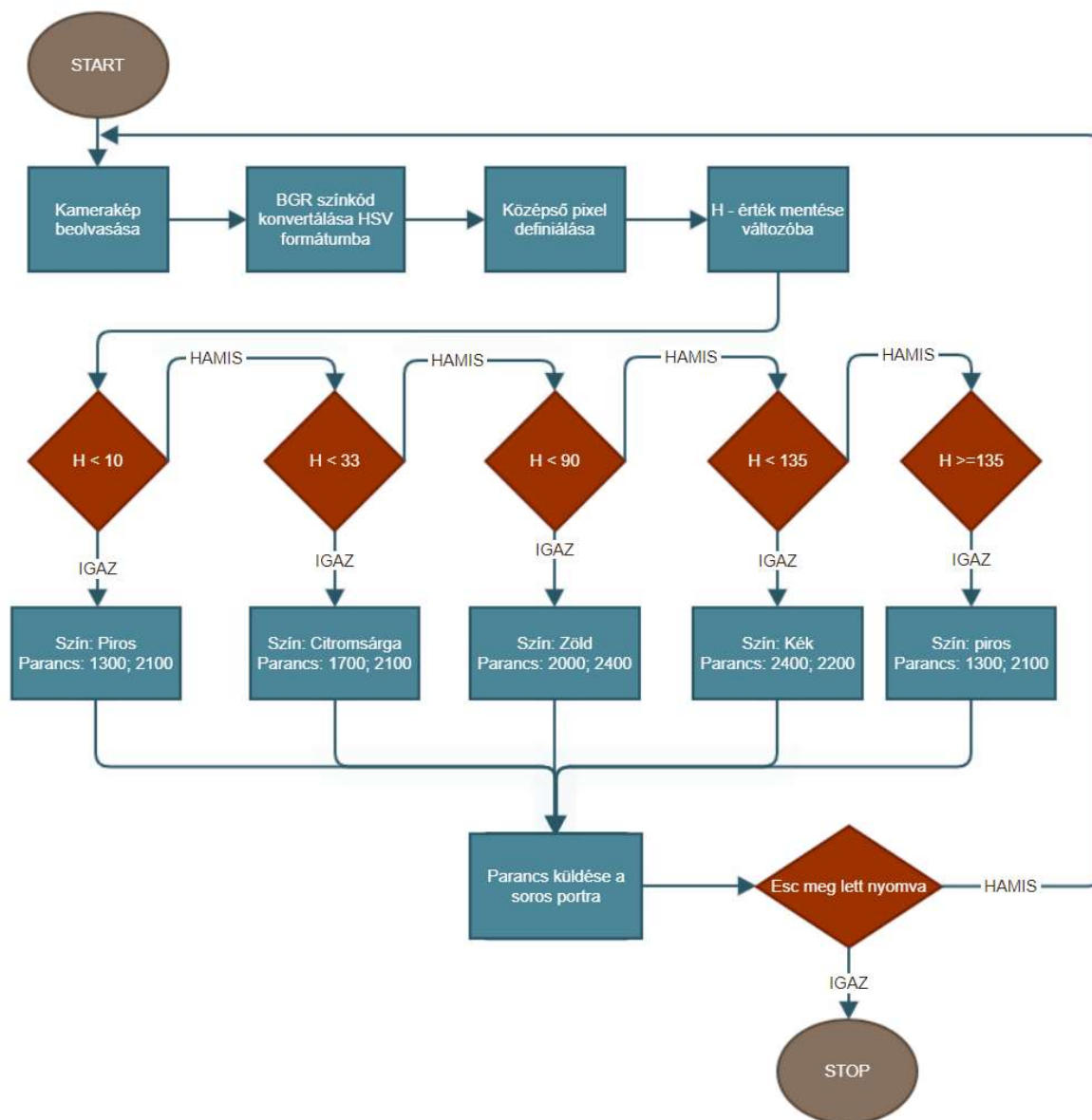


21. Ábra A HSV színtér (forrás: researchgate.net)

A színtér konvertálása után szükségem volt egy képpontra amelynek az értékeit vizsgálhatom. Kitüntetett pontként a keret közepét (center) választottam, mely a keret magasságának és szélességének felénél helyezkedik el. A *hue_value* változóba mentettem a középső pixel H értékét, majd egy elágazásban megadtam a különböző H értékekhez tartozó adatokat. A folyamatábra tartalmazza, mely H értékekhez milyen szín és pozíció tartozik.

A következőkben szín alapján történt pozíciókiválasztás eredményét el kell küldeni a robotkart vezérlő Arduino számára. Ehhez egy különálló kommunikációs kódot használtam, amely 2 függvényt tartalmaz, ezek a *write* és a *read* függvények. A *write* függvény feladata kiküldeni az adatot a soros portra, a *read* függvényé pedig beolvasni a soros porton érkező adatot. A program első lefutásakor a *send* változó értéke True, ezért a Python kódból nem jut ki adat a soros portra, viszont a robotkar homing művelete után érkezik egy „OK” felirat. A *KomScript read* függvénye ezt beolvasva a *send* változó értékét False-ra változtatja. Az ezt követő program-lefutások már nem hagyják figyelmen kívül az „if” utasításban szereplő *KomScript.write(cmd)* függvényt és megtörténik a kívánt pozíció közlése az Arduinoval. Az Arduino ciklusai végét az „OK” felirat jelzi.

A program végén található a keret közepét jelző kört, valamint a bal felső sarokban az aktuálisan detektált szín szerint változó feliratot megjelenítő parancs. Ezt követi a keretet megjelenítő, Esc billentyűre bezáró parancs. A program működése a 23. ábrán látható.



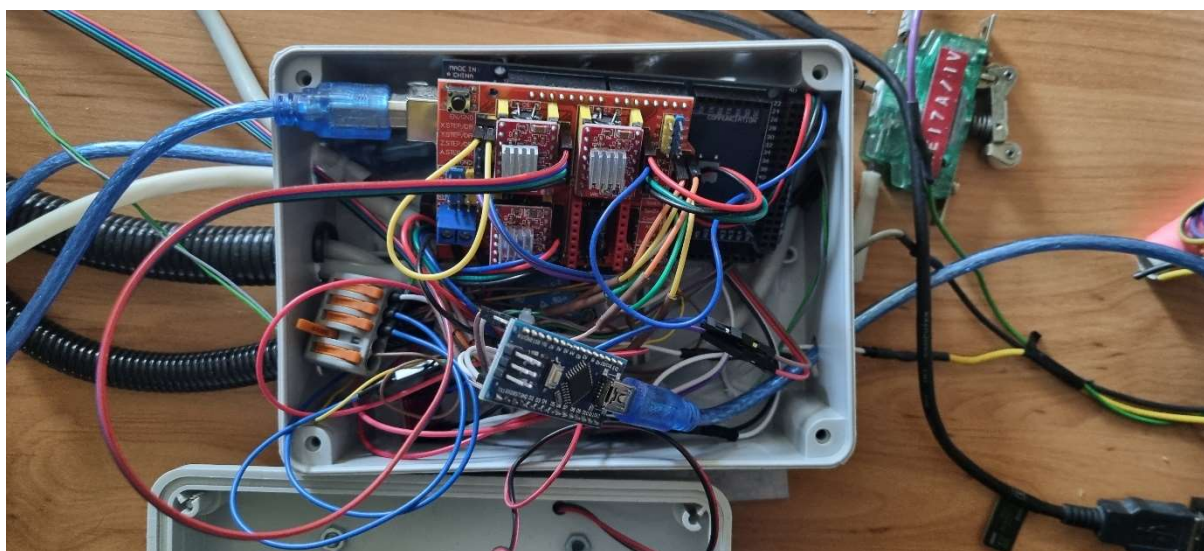
22. Ábra Python program folyamatábrája
(forrás: gliffy.com, saját forrás)

4. A megvalósult robot

Ez a fejezet tartalmazza a projektről készült képeket.



23. Ábra Az elkészült projekt



24. Ábra Az elkészült elektronika

5. Összefoglalás

A dolgozatom témája egy olyan robotkar vezérléstervezése volt, amely szín szerint objektumokat válogat. A robotkar építéséhez a howtomechatronics.com oldalán található SCARA robotkar konstrukcióját használtam. A kar 3D nyomtatását követően néhány kisebb módosítást kellett végrehajtanom a megbízhatóbb működésért, valamint kiegészítőket készítettem az általam definiált feladat ellátásához (mágnestartó-konzol, szállítószalag, IR tartókonzol).

Következő lépésben kiválasztottam a robotkar és a szállítószalag működtető elektronikájának elemeit. A célkitűzésben foglaltak szerint a robotkar működtetéséhez NEMA 17 típusú léptetőmotorokat használtam. A hajtáshoz alkalmazkodva több lehetőséget vizsgálva választottam az A4988-as motorvezérlőt. Az egyszerűbb kezeléshez a CNC Shield V3, Arduinohoz használatos bővíítőpanelt alkalmaztam. A motorok és vezérlők után meghatároztam a további szükséges elemeket, mint a végálláskapcsolók, az elektromágnes, a kapcsolórelé és az IR szenzor.

A robotkar után összeállításra került az elektronika is, a válogatást két különálló kapcsolási körrel valósítottam meg. Az egyik kör a robotkarhoz tartozik, a másik a munkadarabokat szállító szalaghoz. A két kör külön vezérlőkártyáról működik, tápellátásuk ezzel ellentétben logika és teljesítményoldalról is közös.

A robotkar hardveres összeállítása után megírtam a működtető programokat. A célkitűzésnek megfelelően - a robotkar mozgatásához - a motorokat Arduino Mega vezérlőkártya működteti, a szállítószalagot Arduino Nano, a színek detektálását Python kód végzi, ami egy számítógépen fut.

Az robotkar és a működtető program az elvárt feladatot sikeresen teljesítette, azonban számos fejlesztési irány helyezhető kilátásba. Amennyiben a végrehajtó mágneset megfogóra cserélem, lehetőségem nyílik nem mágnesezhető munkadarabok megfogására. A színmeghatározás pontosabb lehet, ha nem csak egy képpontot, hanem környezetének értékeit is figyelembe veszem, a pozíciók könnyebb felvételéhez pedig forward vagy inverz kinematika használata lenne releváns fejlesztési irány.

6. Summary

The topic of my thesis is the design of the control of a robot arm that can sort objects based on their color. I used the mechanical design of an already existing SCARA robot arm as the basis of my design that can be found at howtomechatronics.com. After the 3D printing of the arm, I had to make some minor modifications for more reliable operation as well as make accessories in order to perform the task I set (magnet support bracket, conveyor belt, and IR support bracket).

I selected the elements of the operating electronics of the robot arm and the conveyor belt as the next step of my design process. I used NEMA 17 stepper motors to move the robot arm. In order to meet the requirements of the motors, I chose the A4988 stepper motor driver after examining several options. I used the CNC Shield V3, which is an expansion panel for the Arduino, to control the drivers. Finally, I determined the other necessary elements, such as the limit switches, the electromagnet, the switching relay, and the IR sensor.

After assembling the robot arm, I connected the electronic elements to each other. The set task was accomplished with two separate circuits. One circuit belongs to the robot arm, the other to the belt that transports the workpieces. The two circuits operate from separate control cards, but their power supply is shared.

After the assembly of the robotic arm, I wrote the operating programs. In accordance with the objective—to move the robot arm—the motors are operated by an Arduino Mega control card, the conveyor belt by an Arduino Nano, and the colors are detected by a Python code that runs on a computer.

The robot arm and the operating program successfully fulfilled the set task. However, several development directions can be put into perspective. If the executive magnet is replaced with a gripper, the robot gains the ability to grip workpieces that cannot be magnetized. Color determination can be more accurate if not only one pixel is tested but also the values of its environment, and the use of forward or inverse kinematics would be a relevant development direction to allow more precise positioning.

7. Irodalomjegyzék

- [1] P. Gergely, „Felderítő Robotok,” Debrecen, 2009.
- [2] Z. Dr. Szabó, Budai Csaba, L. DR. Kovács és G. Dr. Lipovszki, Robotmechanizmusok, BME MOGI, 2014.
- [3] R. K. Miller, Industrial Robot Handbook, New York: Springer Science+Business Media ISBN 978-1-4684-6608-9, 1989.
- [4] S. Balla, P. Bánlaki, B. Göndöcs, G. Haidegger, T. Markovits, Z. Pál, J. Takács és Z. Weltsch, Gyártásautomatizálás, Budapest: ISBN 978-963-279-630-7, 2012.
- [5] „Netlife Robotics,” Netlife Robotics, 2022. [Online]. Available: <https://netliferobotics.hu/blog/robotok-alkalmazasa/>.
- [6] „Az ipari robotok nélkülözhetetlensége a gyártási folyamatokban,” ABB, 19 07 2022. [Online]. Available: <https://new.abb.com/news/hu/detail/93295/az-ipari-robotok-nelkulozhetetlensege-a-gyartasi-folyamatokban>.
- [7] H. H., Y. T. és N. Y., „Analysis and Control of Articulated Robot Arms with Redundancy,” Kyoto University, Kyoto, Japan, 1981.
- [8] K. Béla, Robottechnika I., Budapest: ISBN 978-963-279-625-3, 2012.
- [9] „Autopro,” 13 06 2019. [Online]. Available: <https://autopro.hu/gyartosor/a-robotok-negy-fajtajat-minden-gyartanak-ismernie-kell/216525>.
- [10] M. M. András, „Ipari robotok,” Békéscsaba.
- [11] S. Zoltán, Ipari Robotok Kézikönyve, Tatabánya: ISBN 978-615-00-0747-2, 2018.
- [12] R. Industry, „The Ultimate Guide: Everything need to know about 3D Printing Services,” Industry, Roche, [Online]. Available: https://www.rocheindustry.com/3d-printing-ultimate-guide/#What_is_3D_Printing.

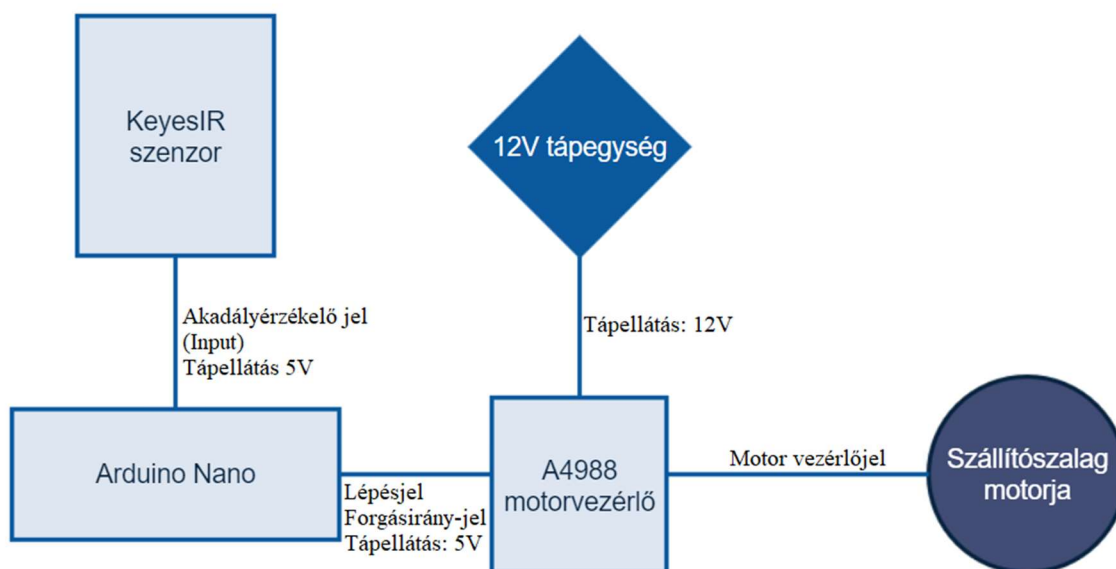
-
- [13] Robin, „Minden, amit a 3D-nyomtatásról tudni kell,” Dongguan Roche Industrial, [Online]. Available: <https://www.cnc.hu/2020/05/minden-amit-a-3d-nyomtatasrol-tudni-kell/>.
- [14] G. Makan, A. Somogyi, R. Dr. Mingesz és J. Mellár, „Tananyagok,” 2016. [Online]. Available: <http://www.inf.u-szeged.hu/~makan/tananyagok/elektronikai-alapok-programozoknak/az-arduino-bemutataa/>.
- [15] B. E. N., „What is an Arduino?,” Sparkfun, [Online]. Available: <https://learn.sparkfun.com/tutorials/what-is-an-arduino>.
- [16] R. Harsányi és M. A. Juhász, Fizikai számítástechnika: elektronikai alapok és Arduino, Budapest, 2014.
- [17] B. W. Evans és R. Cseh, Arduino Programozási Kézikönyv, Budapest: TavIR, 2011.
- [18] R. Gábor, „Arduino,” Raspberry Pi Magyarország, [Online]. Available: <https://malnapc.hu/yis/arduino-termekek>.
- [19] „Léptetőmotorok - Típusok és alkalmazások,” Components, TME Electronic, 08 09 2020. [Online]. Available: <https://www.tme.eu/hu/news/library-articles/page/41861/Leptetomotorok-tipusok-es-alkalmazasok/>.
- [20] „WIKI - VIGVÁRI,” Vígvári Rendszerház Számítástechnikai Kft., 03 07 2020. [Online]. Available: <https://wiki.vigvari.hu/doku.php?id=motorok>.
- [21] K. György, Mikrokontroller alapú léptetőmotor-vezérlő program fejlesztése, Debrecen: Debreceni Egyetem Informatikai Kar, 2010.
- [22] T. Hopkins, „ST - AN235 Application note,” 11 2012. [Online]. Available: https://www.st.com/resource/zh/application_note/cd00003774-stepper-motor-driving-stmicroelectronics.pdf.
- [23] F. Richárd, „GÉPI TANULÁS A KÉPFELDOLGOZÁSBAN,” SZTE Informatikai Intézet, [Online]. Available: <https://www.inf.u-szeged.hu/~rfarkas/ML20/image.html>.

-
- [24] V. Gergő, „VIDEOPRAKTIKA,” 04 2006. [Online]. Available: <http://www.vassg.hu/pdf/gepilatas1.pdf>.
- [25] B. Jähne és H. Haußecker, Computer Vision and Application, USA: Academic Press ISBN 0-12-379777-2, 2000.
- [26] H. István, Tárgyasult ötletek, Budapest: BBS-INFO KÖNYVK. ÉS INFORM. KFT. ISBN9786155477973, 2021.
- [27] M. H., S. Z., P. H. és M. Kastek, „Long-range passive IR sensor,” [Online]. Available: <http://qirt.org/archives/qirt2000/papers/057.pdf>.
- [28] J. H. Bear, „Mi a HSV színmodell?,” eYewated, [Online]. Available: <https://hu.eyewated.com/mi-a-hsv-szinmodell/#>.

8. Mellékletek

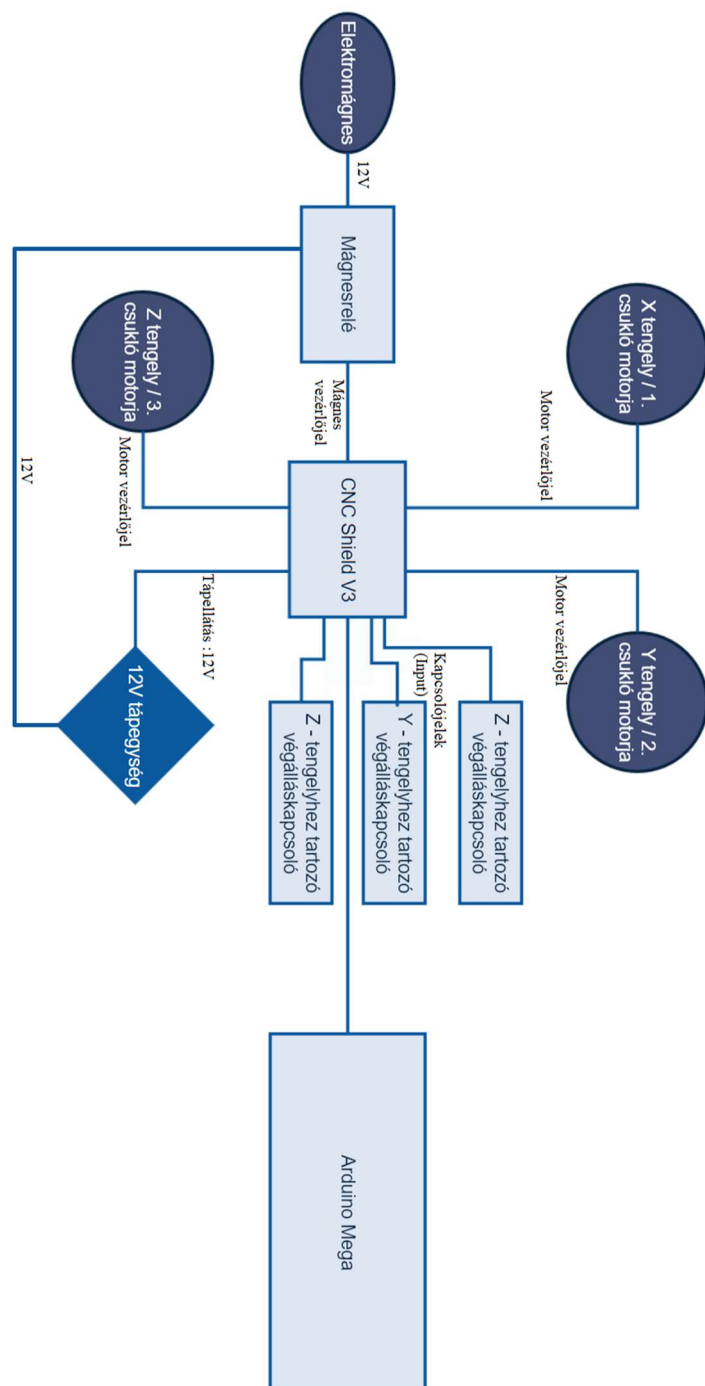
A Mellékletek című fejezet tartalmazza az elektronikai összeállítások blokkdiagramjait, a működtető szoftver program kódjait és a mágnesstartó konzol műszaki rajzát.

1. Melléklet: Szállítószalag elektronikai blokkdiagrammja



25. Ábra Szállítószalag elektronikai blokkdiagrammja
(forrás: gliffy.com, saját forrás)

2. Melléklet: A robot elektronikai blokkdiagrammja



26. Ábra A robot elektronikai blokkdiagrammja
(forrás: gliffy.com, saját forrás)

3. Melléklet: Arduino Mega szoftverének forráskódja

```
#include <AccelStepper.h>
#include <Servo.h>
#include <math.h>
#include <MultiStepper.h>
#define limitSwitch1 9
#define limitSwitch2 10
#define limitSwitch3 11
#define magnesPin 12
// Define the stepper motors and the pins the will use
AccelStepper stepper1(1, 2, 5); // (Type:driver, STEP, DIR)
AccelStepper stepper2(1, 3, 6);
AccelStepper stepper3(1, 4, 7);
AccelStepper stepper4(1, 12, 13);
MultiStepper steppers;
String x;
String y;
String msg;
int a = 750;
int b = 2000;
void setup() {
  Serial.begin(115200);
  //Serial.setTimeout(3);
  pinMode(limitSwitch1, INPUT_PULLUP);
  pinMode(limitSwitch2, INPUT_PULLUP);
  pinMode(limitSwitch3, INPUT_PULLUP);
  pinMode(magnesPin, OUTPUT);
  stepper1.setMaxSpeed(200);
  stepper1.setAcceleration(100);
  stepper2.setMaxSpeed(200);
  stepper2.setAcceleration(100);
  stepper3.setMaxSpeed(1000);
  stepper3.setAcceleration(200);
```

```
stepper4.setMaxSpeed(200);
stepper4.setAcceleration(100);
steppers.addStepper(stepper1);
steppers.addStepper(stepper2);
homing();
}
void loop() {
    long positions [2];
    if (Serial.available()) {
        msg = Serial.readStringUntil('\n');
        x = msg.substring(0, 4);
        y = msg.substring(4, 8);
        a = x.toInt();
        b = y.toInt();
    }
    else {
        return;
    }
    delay(100);
    stepper3.runToNewPosition(0);
    delay(100);
    digitalWrite(magnesPin, HIGH);
    delay(1000);
    stepper3.runToNewPosition(6000);
    delay(100);
    positions[0] = a;
    positions[1] = b;
    steppers.moveTo(positions);
    steppers.runSpeedToPosition();
    delay(100);
    digitalWrite(magnesPin, LOW);
    delay(1000);
    positions[0] = 750;
```

```
positions[1] = 2000;
steppers.moveTo(positions);
steppers.runSpeedToPosition();
delay(500);
Serial.println("OK");
delay(100);
}

void homing() {
  // Homing Stepper1
  while (digitalRead(limitSwitch1) != 1) {
    stepper1.setSpeed(-200);
    stepper1.runSpeed();
    stepper1.setCurrentPosition(0); // When limit switch pressed
    set position to 0 steps
  }
  delay(20);
  stepper1.moveTo(750);
  while (stepper1.currentPosition() != 750) {
    stepper1.run();
  }
  // Homing Stepper2
  while (digitalRead(limitSwitch2) != 1) {
    stepper2.setSpeed(-200);
    stepper2.runSpeed();
    stepper2.setCurrentPosition(0); // When limit switch pressed
    set position to 0 steps
  }
  delay(20);
  stepper2.moveTo(2000);
  while (stepper2.currentPosition() != 2000) {
    stepper2.run();
  }
  // Homing Stepper3
```

```
while (digitalRead(limitSwitch3) != 1) {  
  stepper3.setSpeed(700);  
  stepper3.runSpeed();  
  stepper3.setCurrentPosition(36000); // When limit switch  
pressed set position to 0 steps  
}  
delay(20);  
stepper3.moveTo(6000);  
while (stepper3.currentPosition() != 6000) {  
  stepper3.run();  
}  
Serial.println("OK");  
}
```

4. Melléklet: Arduino Nano szoftverének forráskódja

```
#include <AccelStepper.h>
AccelStepper stepper (1, 3, 2);
const int sensorPin = 12;
const int ledPin = 13;
int sensorState = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
  stepper.setMaxSpeed(1000);
  stepper.setAcceleration(200);
}
void loop() {
  sensorState = digitalRead(sensorPin);
  if (sensorState == HIGH)
  { Serial.println("Clear");
    stepper.setSpeed(200);
    stepper.runSpeed();
  }
  else {
    stepper.setSpeed(0);
    stepper.runSpeed(); }
}
```

5. Melléklet: Python vezérlőprogram forráskódja

```
from turtle import width
import cv2
import KomScript
import time
cap = cv2.VideoCapture(1)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
send = True
while True:
    _, frame = cap.read()
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    height, width, _ = frame.shape
    cx = int(width / 2)
    cy = int(height / 2)
    pixel_center = hsv_frame [cy, cx]
    hue_value = pixel_center[0]
    value_value = pixel_center[2]
    color = "Undefined"
    cmd = '13002100\n'
    if hue_value < 10:
        color = "RED"
        cmd = '13002100\n'
    elif hue_value < 33:
        color = "YELLOW"
        cmd = '17002100\n'
    elif hue_value < 90:
        color = "GREEN"
        cmd = '20002400\n'
    elif hue_value < 135:
        color = "BLUE"
        cmd = '24002200\n'
    else:
```

```
color = "RED"
cmd = '13002100\n'
if send == False:
    send = True
KomScript.write(cmd)

data = KomScript.read()
if len(data) != 0 :
    print(f"'{data}'")
    send = False
    #print(pixel_center)
    pixel_center_bgr = frame [cy, cx]
    b, g, r =
    int(pixel_center_bgr[0]),int(pixel_center_bgr[1]),int(pixel_c
enter_bgr[2])
    cv2.putText(frame, color, (10, 50), 0, 1, (b, g, r), 2)
    cv2.circle(frame, (cx, cy), 5, (0, 0, 0), 3)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

6. Melléklet: Python Kommunikációs modul forráskódja

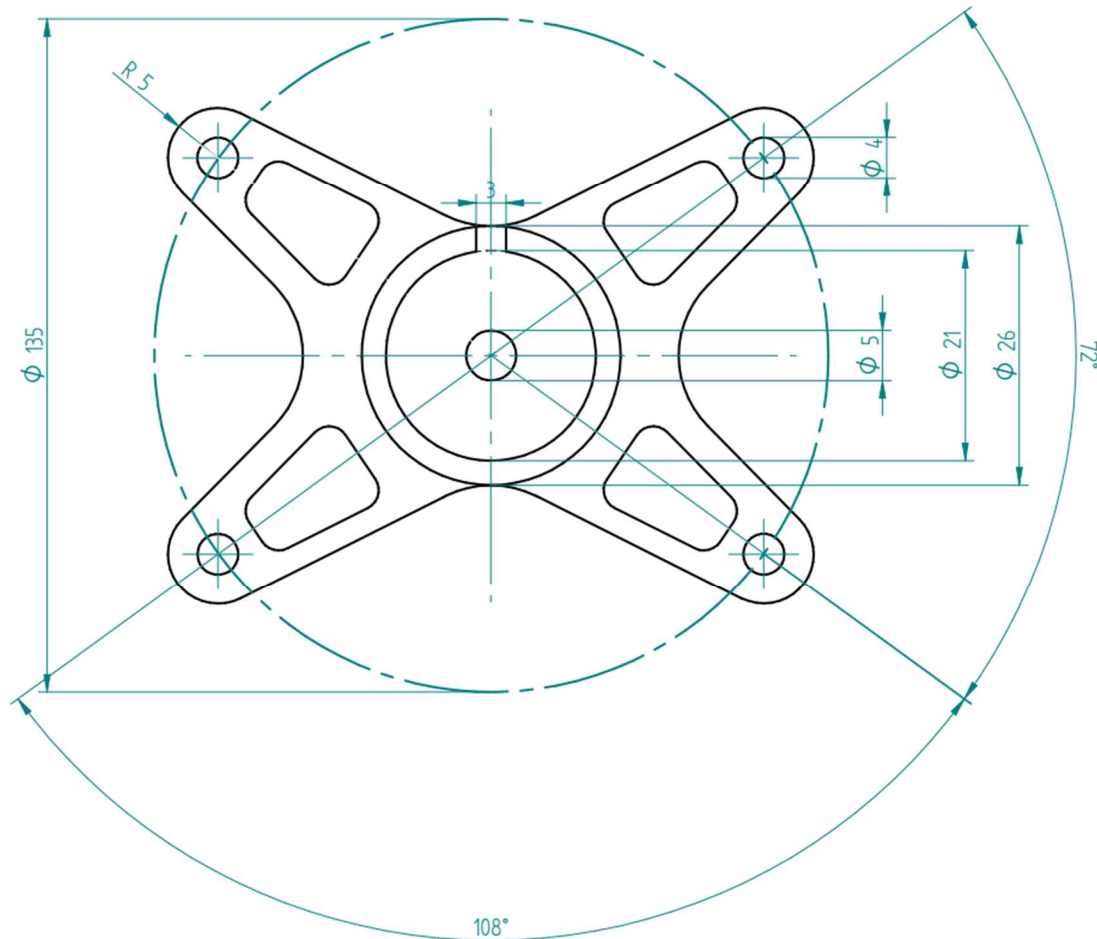
```
import serial
import time

arduino = serial.Serial(port='/dev/ttyUSB1', baudrate=115200,
timeout=0.1)

def write(x):
    arduino.write(bytes(x, 'utf-8'))
    print(f"'{x}'")
    #time.sleep(0.1)
    #data = arduino.readline().decode('utf-8').rstrip()
    #return data

def read():
    return arduino.readline().decode('utf-8')
```

7. Melléklet: Mágnesstartó konzol műszaki rajza



SOLID EDGE ACADEMIC COPY

Név		Szakdolgozat	
TERVEZŐ	Bagi Gergő		
Ellenőrizve		Megnevezés Elektromágnes tartókonzol	
Dátum	2022.10.17.		
Tantárgy	Szakdolgozat	Lap m. Anyag A4 PLA	
ELTÉRŐ RENDELKEZÉS HIÁNYABAN MÉRETEK MILLIMÉTERBEN ÉRTENDŐK SZÖGEK $\pm X^\circ$ 2 PL $\pm X$ XX 3 PL $\pm XXXX$		Fájlnev Magnestarto.dft	
		Mérete: 2 : 1	Tömeg: 13 g
		Lapok 1 / 1	