ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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

SZAKDOLGOZAT

OE-KVK
2023.

Szabó-Szűcs Tamás
T006877/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Szabó-Szűcs Tamás
Szakdolgozat száma: SZD2302271236497407[redacted]
Törzskönyvi száma: T006877/FI12904/K Neptun kódja: [redacted]

Szak: villamosmérnöki (Budapest)
Specializáció: Műszer-automatika specializáció
Automatizált gyártórendszerek modul

A dolgozat címe: Öt szabadságfokú robotkar vezérlése és alkalmazása
A dolgozat címe angolul: Control and application of five DOF robotic arm
A feladat részletezése: Csuklós anyagmozgató robot vezérlésének kivitelezése saját fejlesztésű szoftverrel, modern technológiák alkalmazásával.

Intézményi konzulens neve: Sándor Tamás

A kiadott téma elévülési határideje:2026. 02. 27.
Beadási határidő: 2023. 05. 15.

A szakdolgozat: Nem titkos.

Kiadva: Budapest, 2023. 03. 31. P.H.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Elektronikai és Kommunikációs Rendszerek Tanszék

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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

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, 2023.11.01.

.....
hallgató aláírása

Tartalomjegyzék

1. BEVEZETÉS

Az elmúlt évszázadokban hatalmas ipari változások könnyítették meg és gyorsították fel a különböző iparágak gyártási folyamatait és ezáltal nagymértékben megnőtt a maximálisan előállítható javak mennyisége. Emellett a gépesített munkafolyamatok kihatással voltak a termékek előállítási költségeire is, ami pozitív módon befolyásolta a piaci árakat. Az első ipari forradalom 1750-től a 19. század közepéig tartott. Ebben az időszakban a mezőgazdaságban bekövetkezett nagymértékű gépesítés következtében kevesebb munkaerőre volt szükség a földeken, ezért az agráriumból eltávozó munkaképes emberek általában a városokban található gyárakban kaptak munkát. A gyártói szektorban is jellemző volt, hogy a korábban kézi munkával végrehajtott, egyszerűbb folyamatokat gépiesítették, így az iparnak magasabb képzettségű, komplexebb munkafolyamatok ellátására képes dolgozókra volt szükségük. Az évek során az egyes munkafázisok elvégzésre egyre összetettebb mechanikai szerkezeteket hoztak létre, amiket kezdetleges gőzgépek segítségével hajtottak meg. Ezt a technológiát később Matthew Boulton közreműködésével James Watt tökéletesítette. [1] A második ipari forradalom során már nagyobb szerepet kapott a tömeggyártás, felgyorsult az ipari innováció. Az akkori változások fő mozgatórugója az acélipar volt, melynek következtében a vasútvonalak kiépítése is gyorsabb iramban folytatódhatott. Ez az időszak nagy szerepet játszott a tömegközlekedés fellendülésében, illetve megkönnyítette a nyersanyagok, valamint a késztermékek szállítását is. [2] Ebben a korszakban a legnagyobb mérnöki előrelépést a villamos energia felhasználása jelentette, aminek elméleti és gyakorlati alapjait Michael Faradaynek köszönhetjük. Ezen a területen a második ipari forradalom során számos áttörést értek el, de még nem beszélhetünk a gyártósorok teljes automatizálásáról. Az ehhez szükséges tudást és eszközöket csak a harmadik ipari forradalom során bekövetkezett elektronikai fejlesztéseknek köszönhetjük. Ekkor fektették le a digitális elektronikai alapjait és jelentek meg az első számítógépek. A harmadik ipari forradalom során jelentek meg az első, komplexebb feladatokat ellátó ipari robotok. Jelenleg a negyedik ipari forradalom korát éljük. A XXI. századi trendeket vizsgálva azt láthatjuk, hogy a céges világban, így a gyártási folyamatok terén is paradigmaváltás van folyamatban. [3] A fizikai munkaerő egyre kisebb szerepet tölt be a gyártási folyamatok, az anyagmozgatás és a raktározás terén is. A jelenlegi számítási kapacitásnak és az ezáltal elérhetővé vált gépi tanulásnak köszönhetően olyan munkafolyamatok automatizálására van lehetőség, ami a korábban elképzelhetetlennek tűnt. Az ipar 4.0

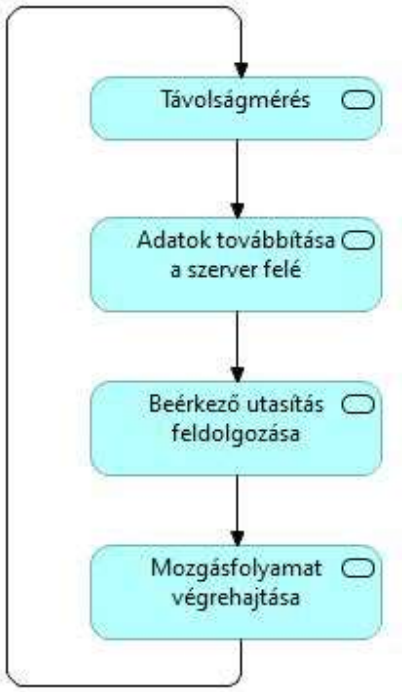
innovációinak köszönhetően lehetőség van arra, hogy az értékláncban részt vevő gyártóegységek között folyamatos adatáramlás jöjjön létre, így megoldhatóvá vált, hogy a fizikai folyamatokat számítógépes rendszerek felhasználásával akár valós időben nyomon kövessék. Az így kinyert hatalmas mennyiségű adatot pedig komplex matematikai modellek segítségével elemzik. A gyártóegységek önálló, decentralizált döntéseket hozhatnak a gyártási folyamatok különböző paraméterei alapján. A nagymértékű automatizálás előnye, hogy a karbantartástól eltekintve a gépek szünet nélkül képesek a termékek előállítására. Az emberi munkaerővel ellentétben a munkafolyamatok megváltozása esetén nem igényelnek betanítást. A robotok és egyéb eszközök újra konfigurálását követően rövid időn belül képesek a feladataik pontos ellátására. Ebből következik, hogy a gyártók gyorsan és költséghatékonyan tudnak alkalmazkodni a megrendelőik folyamatosan változó igényeihez.

2. MEGOLDANDÓ PROBLÉMA MEGFOGALMAZÁSA

Szakdolgozatom célja, hogy egy ipari anyagmozgató robot asztali modelljét létrehozzam. Fontos szempont, hogy a jelenlegi ipari trendeket követve, otthoni keretek közt, minél jobban megismerkedhessek a számítógépes képfeldolgozás használatával, illetve, hogy az eddigi tanulmányaim során megszerzett, a mikrokontrollerek és a valós idejű operációs rendszerek működésével kapcsolatos tudásomat tovább gyarapítsam. Elsősorban szeretnék egy, mikrokontroller alapú robotvezérlő egységet létrehozni, amely vezeték nélküli hálózaton, TCP/IP protokoll használatával képes a központi egységtől kapott utasítások végrehajtására. A központi egység feladatai közé tartozik, hogy képfeldolgozás által megállapítsa a munkaterületre beérkező tárgy színét, majd ezalapján megfelelő utasításokat adjon a vezérlőegység számára. Alapvetően a munkaállomás feladata, hogy a korábbi gyártósori részegységek által szolgáltatott tárgyakat válogasson szét színük alapján, szem előtt tartva a bővíthetőséget és a széleskörű alkalmazhatóságot. A rendszernek képesnek kell lennie arra, hogy emberi beavatkozás nélkül automatikusan hajtsa végre a munkafolyamatot abban az esetben, ha a rakodóterületre áru érkezik. A leírásból adódóan a feladat két fő egységre bontható, egy központi egység, amely képes a képfeldolgozási folyamat elvégzésére és a szerver futtatására. Emellett szükségünk van egy vezérlő egységre is, aminek alkalmasnak kell lennie arra, hogy a robot szervomotorjait vezérelje, valamint arra is, hogy a távolságmérő szenzor által szolgáltatott jelet feldolgozza, kiértékelje és az így kinyert adatok függvényében eldöntse, hogy a tárgy megfelelő pozícióban van a munkafolyamat megkezdéséhez.

3. SPECIFIKÁCIÓ KIDOLGOZÁSA

3.1. Logikai rendszerterv



1. Ábra A robot vezérlésének blokkdiagramja

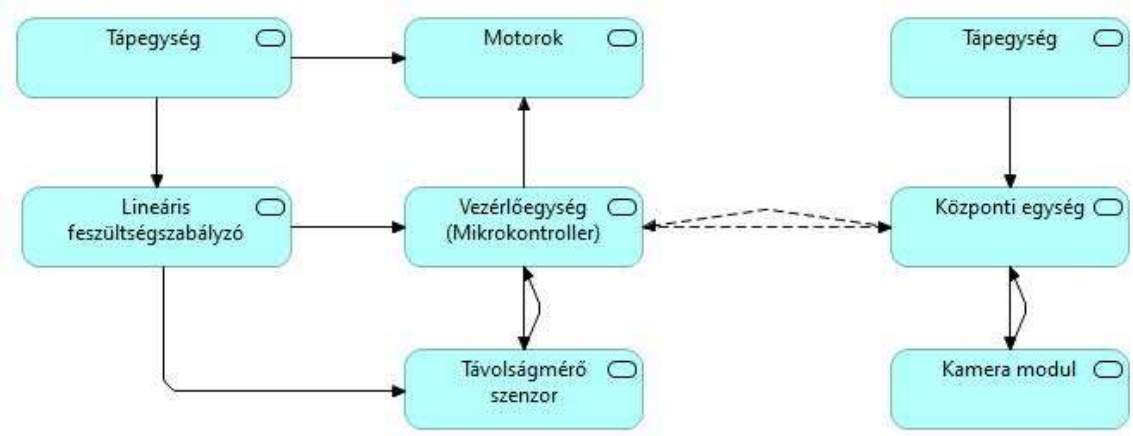
A feladat leírása alapján megállapíthatjuk, hogy a munkafolyamatot a vezérlőegység szempontjából három főbb egységre lehet bontani. Első lépésként meg kell győződnünk róla, hogy a munkaterületre beérkező tárgy megfelelő távolságban van ahhoz, hogy a robot megkezdhesse a tárgy mozgatását. Ebből következik, hogy a távolságmérő eszköznek nem csak a tárgy meglétét kell vizsgálnia, hanem meg kell tudnia állapítania a tárgy szenzortól mért távolságát is. A mérési tartománya függ a rendszer fizikai méretétől és a robot hatótávolságától. A szenzortól érkező jelek feldolgozását követően a mikrokontrollernek vezeték nélküli hálózaton keresztül kell adatokat továbbítania a szerver felé, majd fel kell dolgoznia az onnan kapott utasítást. Végül a robot motorjainak vezérlése által az adott tárgyat el kell juttatnia a munkaterületről az egyik rakodóterületre. A 1. ábrán látható a munkafolyamat blokkdiagramja.

A leírásból adódik, hogy a központi egység működését két külön eljárásra lehet bontani. Folyamatosan fogadnia kell a kliens csatlakozását és az innen származó kérés alapján képet kell készítenie a területre beérkezett tárgyról, majd képfeldolgozás során meg kell állapítania a színét. Végül a kinyert információk alapján utasítást kell küldenie a vezérlőegység számára. A szerver működésének blokkdiagramját a 2. ábrán figyelhetjük meg.

2. Ábra Központi egység működésének blokkdiagramja.

3.2. Fizikai rendszerterv

Célom, hogy egy olyan asztali modellt hozzak létre, amivel egy ipari anyagmozgató robot működését lehet szimulálni. Ebből következik, hogy a robotkarnak kis helyen is el kell férnie és fontos az is, hogy könnyen lehessen mozgatni. A feladat ellátásához szükséges, hogy a robotkar kinyújtott állapotban legalább 15 [cm] hosszú legyen. Mechanikailag alkalmasnak kell lennie 7[g] tömegű és 3,5[cm] átmérőjű tárgyak mozgatására. A teljes vezérlőegység áramfelvétele nem lehet több, mint 1,5[A]. Tápfeszültsége maximálisan 12[V]. A képfeldolgozásért felelős egységnek külön tápegységről kell működnie. Rendelkeznie kell beépített kamerával, vagy alkalmasnak kell lennie arra, hogy felszerelhető legyen a piacon elérhető kamerák egyikével. A távolságmérő modul maximális tápfeszültsége 5[V], illetve működés közben az áramfelvétele nem haladhatja meg az 500[mA]-t. Emellett megfelelő pontosságú mérési eredményeket kell szolgáltatnia 10 és 20[cm] közötti távolságokon. A fizikai rendszerterv blokkdiagramját a 3. ábrán láthatjuk.



3. Ábra Fizikai rendszerterv

3.3. Részegységekkel kapcsolatos főbb elvárások:

3.3.1. Központi egység

- Rendelkezik, vagy fel lehet szerelni kamerával.
- Alkalmas magasszintű programozási nyelvek futtatására.
- Számítási kapacitása lehetővé teszi, hogy több alegységet is vezéreljen.
- Wifi kapcsolattal rendelkezik.
- Alacsony energiafelhasználású.

3.3.2. Végrehajtó egység

- Wifi periféria megléte,
- Legalább 5 db PWM kimenettel rendelkezik,
- CPU órajel frekvenciája minimum 100MHz,
- Képes RTOS futtatására.

3.4. Anyagmozgató egység kiválasztása:

Életünk számos területén találkozhatunk robotokkal, még akkor is, ha első pillantásra nem is tűnnek annak. Gondoljunk csak a nappalinkban cirkáló porszívóra, amely képes önállóan ellátni feladatát vagy akár a játékok sokaságára, amik végtagjaik mozgatásával és különböző hangok kibocsátásával szórakoztatják a felhasználót. Léteznek még katasztrófák helyszínén, veszélyes terepen bevetett mentő robotok, katonai-, szórakoztatóipari-, orvostechnikai- és telekommunikációs robotok is. Viszont, az elvégzendő munkafolyamat leírása alapján csak az ipari robotok fajtáira térnék ki bővebben. Az ISO 8373:2021-es szabvány szerint ipari robotnak minősül minden „*automatikusan vezérelt, újra programozható, ipari környezetben, fix helyre telepített vagy mozgó platformra szerelt, automatizálási feladatokat ellátó, többcélú manipulátor, amely három- vagy több tengely mentén képes mozgást végezni*”. [4] Korábban az ipari robotkarok elkülönített munkatérben dolgoztak, melyeket általában védőráccsal vagy egyéb korláttal határoltak a munkavállalók sérülésének elkerülése érdekében. A robotok új generációja viszont képes az emberekkel való együttműködésre is. A modern ipari robotok különböző szenzorok segítségével képesek érzékelni a környezetükben lévő tárgyakat és embereket. A robotok mozgását leíró pontos dinamikus modellek ismeretében és a munkateret felügyelő szenzorok által szolgáltatott információk alapján a robot képes a munkáját balesetmentesen folytatni még akkor is, hogy ha idegen tárgy került a munkaterébe.

3.5. Ipari robotok főbb kategóriái

3.5.1. SCARA robotok

4. Ábra SCARA robot [Forrás: https://www.fanuc.eu/hu/hu/robotok/robotsz%C5%B1r%C5%91-lap/scara-series/sr-6ia-c]

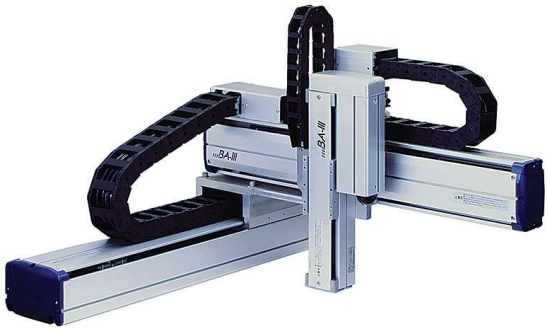
A SCARA (Selective Compliance Articulated Robot Arm) robotokat általában kis gyártási kapacitású, de nagy precizitású összeszerelő egységek részeként alkalmazzák. Kialakításuknak köszönhetően a munkaterük jellemzően henger alakú. Az első két kar teljes hossza a kör alakú munkaterület átmérőjét, a robotkar teljes magassága pedig a henger palástjának a hosszúságát adja meg. Ebből adódóan a SCARA robotok mozgása előre és oldal irányban korlátozott, viszont helyigényük jóval kevesebb, mint az azonos kinyúlású ortogonális vagy delta robotoké. Nagymennyiségű alkatrész mozgatása során fontos szempont, hogy milyen gyorsan képes elhelyezni őket a robot. A SCARA típusú robotok jellemzően a leggyorsabban közé tartoznak. Ezért általában akkor alkalmazzák őket, amikor nagymennyiségű alkatrész mozgatására van szükség. A többi típushoz képest kevesebb mozgó csuklójuk van, ezért leegyszerűsödik a fordított kinematika számítása, így a működtetésük kevesebb számítási kapacitást igényel. Szintén az egyszerűbb kialakításból adódik, hogy az ismételhetőségük is kimagasló. Hátrányuk, hogy a Z tengely mentén nem képesek forgó mozgást végezni. [5]

3.5.2. Delta robotok

5. Ábra Delta robot [Forrás: https://swisscham.hu/en/abb-launches-its-fastest-five-axis-delta-robot-for-picking-packaging-and-positioning-of-light-products/]

Három vagy négy karból álló párhuzamos robotok. Kialakításának és precizitásának köszönhetően a karok rendkívül gyors és precíz mozgásra képesek. Szerkezeti kialakításuk miatt viszont a maximális terhelhetőségük alacsonyabb, mint a többi robotnak. Ezért olyan környezetben szokták őket használni, ahol sok kisebb alkatrészt rövid idő alatt, nagy pontossággal kell elhelyezni az adott munkaterületen. Ilyen munkafolyamat például az SMT (Surface Mount Technology) alkatrészek beültetése nyomtatott áramköri lapokra.

3.5.3. Ortogonális robotok



6. Ábra Ortogonálisi robot [Forrás: https://www.assemblymag.com/articles/93615-cartesian-robot-doubles-output-of-solar-lego-modules]

Másnéven lineáris robotok. Három, egymásra kölcsönösen merőleges elsődleges vezérlőtengelyből állnak. A tengelyeken való elmozdulást csúszó ízületek biztosítják. Emiatt forgó mozgást nem végeznek. Előnyük, hogy háromdimenziós térben történő anyagmozgatás során nagyon precíz mozgások végrehajtására alkalmasak. Kialakításuknak köszönhetően azonban nem képesek arra, hogy a mozgatni kívánt tárgyakat függőlegesen elforgassák.

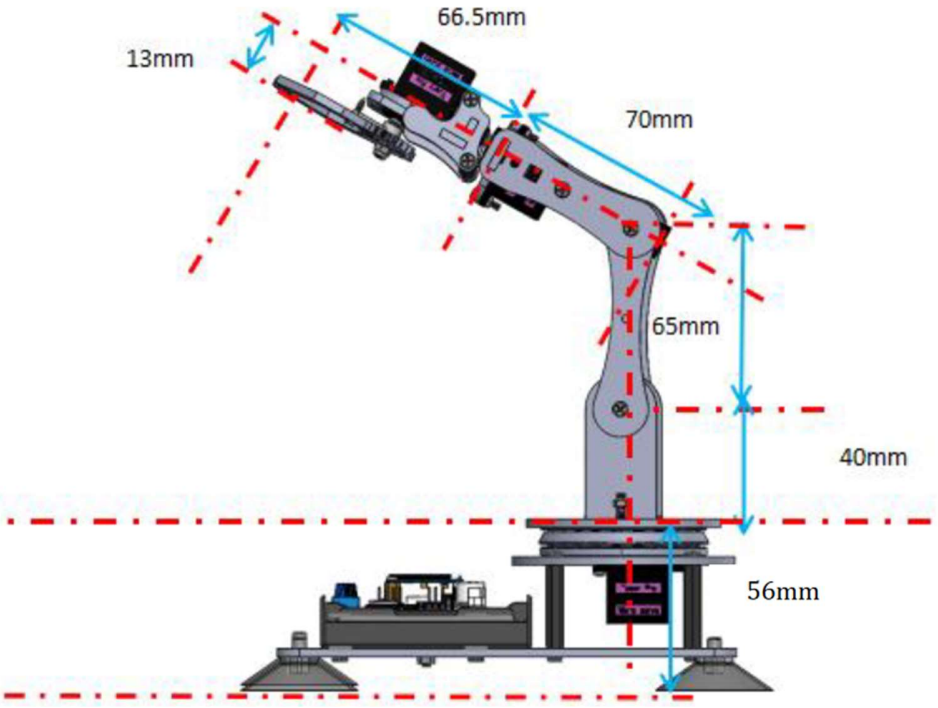
3.5.4. Csuklós robotok

A csuklós robotok működésüket tekintve az emberi karhoz hasonlítanak. Az emberi kar 7 szabadságfokú mozgásra képes. Mivel ahhoz, hogy a kézfejet egy adott pozícióba mozgassuk és megfelelő orientációba hozzuk hat szabadságfokra lenne szükség, ezért kijelenthetjük, hogy az emberi kar redundáns rendszernek minősül. Az iparban általában 4, 5, 6 vagy hét tengelyes csuklós robotokkal találkozhatunk, ezek közül leggyakrabban a hat szabadságfokú robotkarokat alkalmazzák. Sokoldalúságuknak köszönhetően ilyen robotokat használnak többek közt hegesztéshez, festéshez és bizonyos repetitív mozdulatokat igénylő tesztek elvégzéséhez is. Robotikában a csuklókat tengelyeknek nevezik. Ezek a tengelyek általában forgó mozgást végeznek, és szervomotorok gondoskodnak a meghajtásukról. A tengelyek láncolt rendszert alkotnak.

7. Ábra Csuklós robot [Forrás: https://robotsdoneright.com/ABB-Robots.html]

3.6. Megfelelő robot kiválasztása

A feladat leírásából következik, hogy az anyagmozgató egységnek képesnek kell lennie az adott tárgyat megfelelő orientációval elhelyezni a rakodóterületek egyikére. A robotot úgy kell megválasztanunk, hogy fizikai felépítéséből adódóan ne befolyásolja a távolságmérés és a képfeldolgozás menetét sem. Így megállapítható, hogy nem alkalmazhatunk olyan robotot, aminek a működése szempontjából elengedhetetlen, hogy a munkaterületre belógó szerkezeti elemekkel rendelkezzen. Fontos, hogy önálló talapzattal rendelkezzen, amit a rakodóterület mellett a talajhoz lehet rögzíteni. Ebből következik, hogy a feladat ellátására csuklós robotot kell alkalmaznunk. A piacon elérhető csuklós robotok közül végül az Adeept 5 szabadságfokú robotkar mellett döntöttem, aminek szerkezeti felépítését és az adott szegmensek hosszát a 8. ábrán láthatjuk. Az ábrán látszik, hogy a robotkar maximális kinyúlása a talapzat közepétől a megfogó szerkezetig 201,5 [mm], ami eleget tesz a specifikációban meghatározott értéknek. Szerkezeti elemei akrilból készültek a hajtott tengelyek mozgatásáért pedig egy-egy AD002-es szervomotor felel. Ezeknek a működését és a robotkar teherbírását egy későbbi fejezetben tárgyalom.



8. Ábra Adeept 5 szabadságfokú robotkar [Forrás: https://www.adeept.com/learn/detail-64.html]

Talapzatán található négy darab menetes távolságtartó hüvely, ami lehetővé teszi, hogy felfogassuk a vezérlés áramköri lapját.

3.7. Központi egység kiválasztása

A központi egység kiválasztása során három különböző SoM (System on Module) hardver paramétereit hasonlítottam össze. Az adatok összefoglalását az 1-es táblázatban láthatjuk.

1. táblázat Központi egységek paraméterei

Megnevezés	Raspberry Pi 3 model B	Raspberry Pi 4 model B	Jetson Nano
Kép	 *9. ábra Raspberry Pi 3 model B [Forrás: https://us.rs-online.com/m/d/4252b1ecd92888dbb9d8a39b536e7bf2.pdf]*	 *10. ábra Raspberry Pi 4 model B [Forrás: https://ipon.hu/shop/termek/raspberry-pi-4-model-b-4gb-memoriaval/1754869?gad_source=1]*	 *11. ábra Jetson Nano [Forrás: https://www.rpibolt.hu/NVIDIA-JETSON-NANO-2GB-DEVELOPER-KIT]*
Gyártó	Raspberry Pi foundation	Raspberry Pi foundation	NVIDIA
CPU	Broadcom BCM2387	Quad core 64-bit ARM-Cortex A72	ARM Cortex -A57 MPCore
Processzor órajele	1.2GHz	1.5GHz	1.43GHz
GPU	Dual Core VideoCore IV	VideoCore VI 3D Graphics	Maxwell 128-core GPU
RAM	1GB LPDDR2	4 Gigabyte LPDDR4	4GB LPDDR4
Wireless LAN	802.11 b/g/n	802.11 b/g/n/ac	-

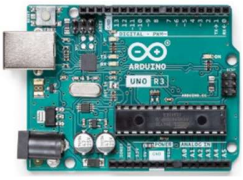
A táblázatban szereplő adatok, az eszközök piaci elérhetősége és az ár-érték arányuk alapján megállapíthatjuk, hogy a kijelölt feladat ellátására a Raspberry Pi 4 model B számítógép felel meg a legjobban.

3.8. Rapsberri Pi 4 Model B

A Raspberry Pi 4 model B az elődjeihez képest lényegesen nagyobb teljesítménnyel rendelkezik. A technikai fejlődésnek köszönhetően ez a változat képes akár két 4K felbontású monitor egyidejű meghajtására, a négymagos, 1,5GHz-es processzornak köszönhetően. Számítási kapacitása a korábbi modellekhez képest megtöbbszöröződött, az 1 [Gbit/s]-os Ethernet portjának köszönhetően akár szerverként is megállja a helyét, illetve a négy darab USB csatlakozók közül kettő USB 3.0 technológiával rendelkezik. Ahhoz, hogy az eszközt megfelelően tudjuk használni első lépésként az operációsrendszer lemezképét kell kiírnunk egy microSD kártyára, amit legegyszerűbben a gyártó által szolgáltatott Raspberry Pi imager szoftver segítségével oldhatunk meg. Ha ez megtörtént, akkor a telepítés utolsó lépéseit már a célhardveren tehetjük meg, végül pedig felhasználásának megfelelően konfigurálhatjuk az eszközünket.

3.9. Fejlesztői panel kiválasztása

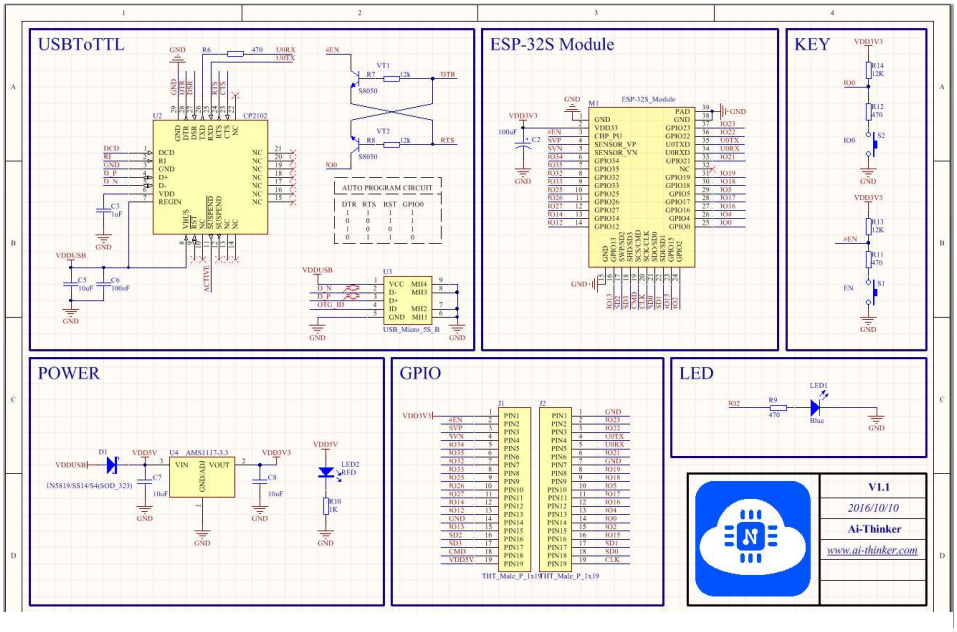
2. táblázat Fejlesztői panelek összehasonlítása

Megnevezés	Arduino UNO R3	NodeMCU ESP32S
Kép	 *12. Ábra Arduino UNO R3 [Forrás: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf]*	 *13. Ábra NodeMCU ESP32S [Forrás: https://www.rpibolt.hu/NodeMCU-32S]*
Gyártó	Arduino S.r.l	AI-Thinker
CPU	8 bit AVR RISC	Xtensa 32-bit LX6
Processzor órajele	16Mhz	240MHz
PWM csatornák száma	6	16
Wireless LAN	-	802.11b/g/n

Ugyan az Arduino UNO R3 fejlesztői panel a motorok vezérléséhez megfelelő számú PWM csatornát tartalmaz, viszont nem rendelkezik a vezeték nélküli hálózathoz való csatlakozást lehetővé tevő perifériával. Erre a célra a gyártó készített egy kiegészítő áramkört, ami lehetővé

teszi a WiFi használatát, viszont ez a megoldás további költségekkel járna. Emellett hátránya még, hogy a paraméterei nem teszik megfelelővé valós idejű operációs rendszerek futtatására, ugyanis nem rendelkezik elég számítási kapacitással és tárhellyel ahhoz, hogy megfelelően ki lehessen használni azok előnyeit. Ezzel szemben a dokumentáció alapján a NodeMCU ESP32S fejlesztői panel teljes mértékben megfelel a feladat megoldására. Illetve előnyt jelent még az is, hogy a fejlesztői környezet használata során egy, a gyártó által optimalizált, FreeRTOS alapú operációs rendszerrel dolgozhatunk. Így végül az utóbbi használata mellett döntöttem.

3.10. NodeMCU ESP-32S

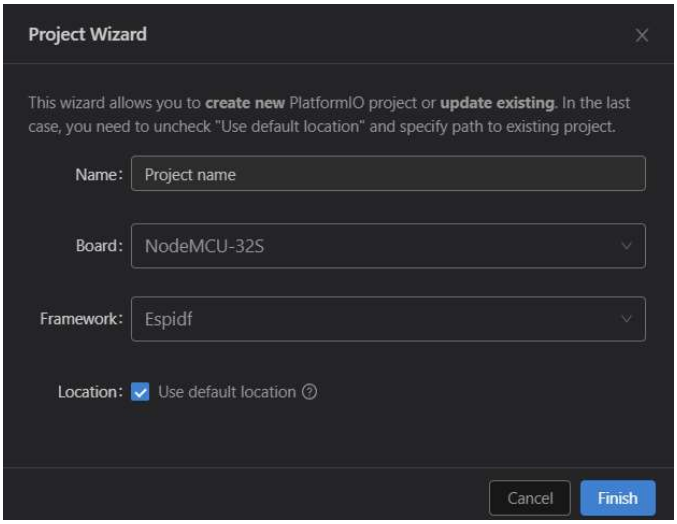


14. Ábra NodeMCU ESP-32S kapcsolási rajz [Forrás: https://docs.ai-thinker.com/en/esp32/boards/nodemcu_32s]

A NodeMCU ESP-32S fejlesztői panelt nagy teljesítménye és a sokoldalúsága miatt rendkívül széles körben alkalmazzák elsősorban IoT (Internet of Things) fejlesztés során. Alapjául egy ESP32 SoC (system-on-a-chip) szolgál. Központi egysége egy Harvard-architektúrájú, 32-bites Xtensa LX6 processzor. A beépített WiFi és Bluetooth interfészek mellett alkalmazásakor összesen 41 különböző perifériát vehetünk igénybe. [6] Működtetéséhez 3.3[V] tápfeszültség szükséges, melyet a fejlesztői panelen elhelyezett AMS1117-3.3 lineáris feszültség stabilizáló IC (Integrated Circuit) állít elő. Összesen 49 I/O (Input/Output) lábbal rendelkezik, amelyekből a nyomtatott áramköri lap tüskesorára 38 lett kivezetve. A projekt szempontjából fontos megemlítenünk, hogy

az adatlap szerint a bemenetekre 3.3[V]+0.3[V] maximális feszültséget lehet csatlakoztatni, illetve a kimenetek által szolgáltatott feszültség 3.3[V].

3.11. Fejlesztői környezet



15. Ábra PlatformIO projekt konfigurálása

Alapvetően a kiválasztott fejlesztői panel teljes mértékben kompatibilis az Arduino fejlesztői környezettel. Mivel azonban ennek a projektnek az egyik fő célja, hogy minél jobban megismerkedhessek az adott hardver nyújtotta adottságokkal, és lehetőleg saját implementációt hozzak létre az a feladat végrehajtására, ezért ezt a lehetőséget elvetettem. Így a gyártó által létrehozott és jól dokumentált ESP IDF (IoT Development Framework) használata mellett döntöttem. Az említett keretrendszer a PlatformIO kiegészítő által egyszerűen integrálható a Visual Studio Code fejlesztői környezetbe. A PlatformIO rengeteg SDK-t (Software Development Kit) és keretrendszert támogat, köztük az ESP IDF-et is. Emellett beépített unit tesztelési lehetőséggel, soros port olvasóval és automatizált kód analizátorral rendelkezik. [7] A telepítést követően a 15. ábrán látható beállításokkal hozható létre az új projekt. Utolsó lépésként már csak a Silicon Labs CP2102-es USB-UART átalakító illesztőprogramját kell letöltenünk a gyártó honlapjáról és feltelepítenünk a számítógépünkre.

4. MEGVALÓSÍTÁS

4.1. Munkaterületre kerülő tárgy érzékelése

A tárgyak érzékelésére két lehetséges megoldást vettem fontolóra. Az egyik megoldás az, hogy a szerver előre meghatározott időnként fényképet készít a területről, ezt képfeldolgozási algoritmusok segítségével kiértékeli és az így kapott információk alapján eldönti, hogy milyen beavatkozás szükséges. Ennek előnye, hogy nem szükséges plusz szenzorral ellátni a robotkart vezérlő elektronikát, valamint ezzel a megoldással elkerülhető a felesleges adatáramlás a két eszköz között. A projekt jövőbeni fejlesztésének szempontjából viszont fontos szem előtt tartani azt, hogy adott esetben a szerver nem csak egy klienst fog kiszolgálni. Ebből következik az, hogy amennyiben ezt a megoldást választanám, akkor későbbiekben előfordulhatna az az eset, mikor egy adott alegység kritikus információra vár a szerver részéről, ám azt csak időben késleltetve kapná meg, mert a szerver erőforrásait lekötné a képek kiértékelése. Ezért ezt a megközelítést elvetettem. Úgy döntöttem, hogy jelen helyzetben sokkal hasznosabb azt a megközelítést alkalmazni, miszerint minden alegység egy-egy szenzorral rendelkezik, lokálisan érzékeli, hogy mikor került új anyag a munkaterületére és csak ebben az esetben kéri a munkaterületről felvett kép kiértékelését a központi egységtől.

4.1.1. Szenzor típusának kiválasztása

A szenzor kiválasztása során létrehoztam az alábbi táblázatot, amely tartalmazza az egyes típusok főbb paramétereit:

3. táblázat Különböző szenzorok összehasonlítása

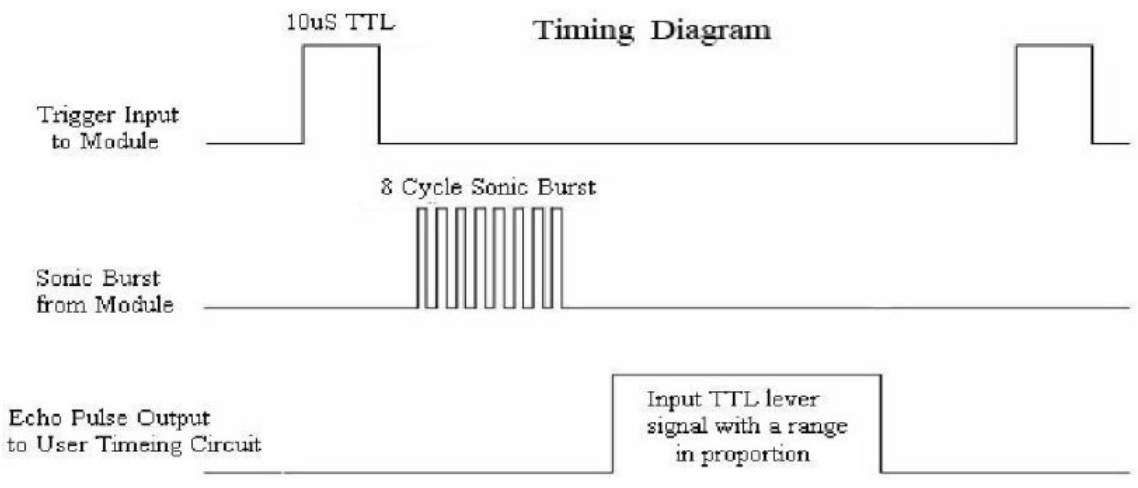
Szenzor típusa	Érzékelt tárgy anyaga	Maximális érzékelési távolság [mm]
Kapacitív	műanyag, üveg, fa, papír, porok, granulátumok, folyadékok	50
Induktív	fémek	60
Optikai	Bármely anyag érzékelésére alkalmas, ám bizonyos felülettípusok megnehezíthetik az alkalmazását.	1000
Ultrahangos	Bármely anyag, azonban a tárgy merevsége és elhelyezése befolyásolhatja a mérés eredményét.	6000

A feladat megoldásának szempontjából fontos, hogy a választott szenzor a munkaterület teljes egészét le tudja fedni. Emellett azt is szem előtt kell tartanunk, hogy az érzékelendő dobozok tartalmát és azok anyagát nem ismerjük, ezért anyag-specifikus érzékelési megoldást nem alkalmazhatunk. Valamint a táblázatban szereplő adatokból látszik, hogy az induktív és kapacitív érzékelők nem rendelkeznek megfelelő nagyságú mérési tartománnyal, így ezeknek a használatát elvethetjük. A specifikációnak megfelelő paraméterekkel rendelkező fotoelektromos távolságérzékelők pedig túl drágának bizonyultak. A korábban felsorolt indokok miatt végül egy ultrahangos távolságmérő modul alkalmazása mellett döntöttem. Az elvégzendő feladat szempontjából nincs szükség arra, hogy gyors mérési periódussal dolgozzunk, hiszen a tárgy munkaterületen való elhelyezése is időigényes folyamat, így a távolságmérést másodpercenként egyszer fogjuk elvégezni. A projekt szempontjából előnyt jelent, hogy a dobozok oldalai eléggé merevek és aránylag nagy felületet biztosítanak ahhoz, hogy a hanghullámok megfelelően visszaverődhessenek róluk. Emiatt a tárgy felületi minőségéből adódó csillapítást ultrahangos érzékelő esetén elhanyagolhatónak tekinthetjük.

4.1.2. Kiválasztott szenzor ismertetése és alkalmazása

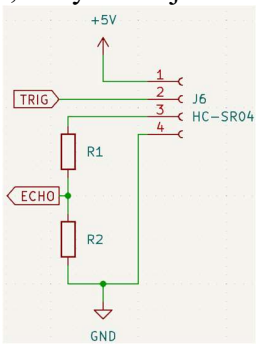
16. Ábra HC-SR04 [Forrás: https://www.electroschematics.com/hc-sr04-datasheet/]

Alapos keresés során, az eszköz ár-érték arányát is szem előtt tartva végül a hobbi elektronikai körökben jól ismert HC-SR04-es szenzort választottam. Ez a modul 15°-os szögben, 20[mm]-től 4000[mm]-ig terjedő távolságok mérésére alkalmas, mindössze ±3[mm]-es hibahatárral. Megfelelő szoftveres implementáció mellett az említett hiba mértéke elhanyagolhatónak bizonyul a projekt szempontjából. A modul működtetéséhez 5[V]-os egyenfeszültségű tápegység szükséges, maximális áramfelvétele 15[mA]. [8] A felvett áram nagysága alapján megállapítható, hogy amennyiben a szenzor által mért értéket egy mikrokontroller segítségével szeretnénk feldolgozni és nem rendelkezünk egyéb fogyasztókkal a kapcsolásunkon belül, akkor a működtetéshez nincs szükség külön tápegységre. Elegendő az USB (Universal Serial Bus) port által szolgáltatott 500[mA]-es áramkorlát, ami az eszköz működtetéséhez szükséges program megvalósítása és tesztelése során előnyösnek bizonyult. Méretét tekintve a szenzor 45*20*15[mm] nagyságú. A nyomtatott áramköri lapon található egy hangszóró, ami a mérés alapjául szolgáló hanghullám előállításáért felel. Ennek meghajtása egy 40[kHz] frekvenciájú kristály oszcillátorral történik. A hanghullám visszaverődésének érzékelését egy mikrofon biztosítja. Az eszköz felhasználásának megkönnyítése céljából a PCB-n (Printed Circuit Board) más integrált áramköri elemek is helyet kaptak.



17. Ábra HC-SR04 időzítési diagramja [Forrás: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf]

A 17. ábrán szereplő időzítési diagram alapján pontosabban megérthetjük a szenzor működését. Ahhoz, hogy megkezdjük a mérési folyamatot, a modul trigger bemenetének feszültségszintjét legalább 10[µS]-on keresztül magas jelszinten kell tartanunk. Ezt követően kerül kibocsátásra egy 40[kHz] periódusú impulzus sorozat, melynek teljes időtartama 200[µs].



18. Ábra HC-SR04 kpcsolási rajz

Az utolsó impulzus hatására a modul echo kivezetésének jelszintje megváltozik, és ebben az állapotban marad egészen addig, amíg az utolsó hanghullám visszaverődött az adott tárgyról, vagy le nem telik a 4 méter távolsághoz tartozó időkorlát, ami szobahőmérsékleten 23,5 milliszekundumnak felel meg. A megvalósítás során a trigger láb 3.3[V]-os jelszintre emelése megfelelőnek bizonyult ahhoz, hogy megkezdődjön a mérés. Ezzel ellentétben ahhoz, hogy az

Echo láb jelszintjét figyelni tudjuk a kiválasztott mikrokontroller segítségével, jelszint illesztést kell alkalmaznunk, ugyanis az ESP32 maximális bemeneti feszültsége 3.3[V] ±0.3[V]. Ennek a legegyszerűbb módja egy feszültségosztó áramkör alkalmazása. Az áramkör kapcsolási rajza az 18. ábrán látható. A szükséges ellenállás értékeket pedig a következő képlet alapján számíthatjuk ki:

$$U_{ki} = U_{be} \times \left(\frac{R_{ki}}{R_{er}}\right), ahol$$

U_{ki}: Az ESP32 maximális bemeneti feszültsége

U_{be}: A HC-SR04 által szolgáltatott kimeneti feszültség

A képlet átrendezése és a behelyettesítés után megkapjuk, hogy $\frac{R_{ki}}{R_{er}}$ értéke 0.66-nak felel meg. Így olyan ellenállás párt kell találnunk, amelyekre a korábbi megkötés igaz. Az IEC 60063:1963 szabvány szerinti E192-es sorból választva megállapíthatjuk, hogy a megfelelő ellenállás értékek: 1000[Ω] és 500[Ω]. Ezeket az értékeket behelyettesítve az eredeti feszültségosztó képletbe azt kapjuk, hogy a kapcsolás kimeneti feszültsége 3.333[V]. Ez pedig megfelel a V_{DD}±0.3[V]-os határértéknek. Természetesen az alkatrészek tűrése miatt sosem kapjuk meg pontosan ezt az értéket, ezért még az áramkör megépítése előtt megmértem az ellenállások valós értékét és az így kapott mennyiségekkel ismét elvégeztem a feszültségosztást. Így a következő értékeket kaptam: R1 = 519,6[Ω] és R2 = 999 [Ω] ezekkel az értékekkel számolva, 5[V]-os tápfeszültség mellett az echo láb aktív jelszintjének feszültség értéke 3,29[V].

4.1.3. HC-SR04 szoftveres implementáció

A szoftver megírása során abból indulhatunk ki, hogy egy olyan RTOS (Real-Time Operating System) feladatra van szükség, amely a robotkar alaphelyzetbe állását követően másodpercenként fut le egészen az első tárgy észleléséig. Ezt követően pedig felfüggesztésre kerül egészen addig, amíg az anyagmozgatási folyamat be nem fejeződött. Ezeket a feltételeket szem előtt tartva és a szenzor dokumentációját felhasználva a szoftver működését leíró állapotdiagram a 19. ábrán látható.

19. ábra Távolságmérés folyamatábra

A működéshez szükséges konstansokat a modul header fájljában kerülted definiálásra. Ennek köszönhetően, ha a későbbiekben változások bevezetésére van szükség, akkor az adott paramétert elég itt átírni, nem kell az egész kódot módosítani. A mérés megkezdéséhez szükséges impulzust az 20. ábrán látható módon állíthatjuk elő.

```
gpio_set_level(TRIG_PIN, HIGH);
ets_delay_us(TRIGGER_DURATION_US);
gpio_set_level(TRIG_PIN, LOW);
```

20. Ábra Impulzus implementáció.

A dokumentációban szereplő 10[µS]-os késleltetést a <rom/ets_sys.h> könyvtárban definiált ets_delay_us függvény meghívásával érhetjük el. Ahhoz, hogy a rendszer által szolgáltatott erőforrásokat minél jobban ki tudjuk használni ezt a megoldás a későbbiekben célszerű egy timer alapú időzítésre cseréni. A trigger lábat bemenetként konfiguráltuk fel- és lefutó élre érzékeny megszakítással. Mind a két megszakítás során az esp_timer_get_time függvény meghívásának

segítségével egy statikus változóban eltárolódik a mikrokontroller bekapcsolásától eltelt idő µs-os pontossággal. A két érték különbsége pedig megegyezik a hanghullámok kibocsátása és visszaverődése között eltelt idővel. Ebből adódóan a lefutó él hatására a program átlép a következő szakaszba, ahol az alábbi képlet segítségével kiszámításra kerül az érzékelt felület távolságát.

$$S = \frac{T \times c}{2}$$

A hang terjedési sebességének képlete pedig:

$$c = \left(331{,}5 + (0{,}6 \times \theta)\right)[\frac{m}{s}],$$

ahol Θ a környezeti hőmérséklet °C-ban.

A képlet alapján megállapítható, hogy a hang terjedési sebessége a környezeti hőmérséklettől függ. Behelyettesítve megkapjuk, hogy a hang terjedési sebessége szobahőmérsékleten közel 343,5 $[\frac{m}{s}]$. Így az implementáció során ezzel az értékkel számolhatunk. Az alkalmazás körülményeiből adódóan a hőmérsékleti tényezőből származó 0,6 $[\frac{m}{s}]$-os sebességkülönbség [°C]-onként elhanyagolhatónak bizonyult. Végül az így kiszámított távolság alapján megállapítható, hogy a szenzor elé került tárgy a megadott határtávolságokon belül helyezkedik el vagy sem.

4.2. Eszköz csatlakoztatása a vezeték nélküli hálózathoz

Ahhoz, hogy a WiFi perifériát megfelelően tudjuk használni, először a mikrokontroller NvM (Non-volatile Memory) területét kell beállítanunk, ugyanis a kommunikációhoz szükséges adatok itt tárolódnak. A TCP/IP stack inicializálását az ESP IDF könyvtár beépített függvényeivel végezzük, a mikrokontroller WiFi perifériája station módban működik. A vezeték nélküli hálózathoz való

```
wifi_config_t wifi_config = {
    .sta = {
        .ssid = SSID,
        .password = PASS,
        .threshold.authmode = WIFI_AUTH_WPA2_PSK,
        .sae_pwe_h2e = WPA3_SAE_PWE_UNSPECIFIED,
    },
};
```

21. Ábra WiFI konfiguráció struktúra

csatlakozáshoz szükséges erőforrások allokációját az esp_wifi_innit függvény meghívásával érhetjük el, ami többek közt az RX és TX bufferek beállításáért is felelős. Ezzel egyidőben a WiFi feladat futtatása is megkezdődik. A csatlakozáshoz szükséges struktúra a 21. ábrán látható. A jelszó és a hálózat neve egy külön header fájlban, makrókként lett definiálva. A hálózat titkosítása WPA (WiFi Protected Access)-PSK (Pre-Shared Key) protokollal történik.

4.2.1. TCP/IP kliens

A számítástechnikai korai szakaszában az internetet elsősorban katonai célokra használták, majd a tudósok munkáját tette hatékonyabbá. Végül az 1960-as években megnövekedett az emberek igénye az otthoni számítógépek iránt, így ebben az időszakban vált széleskörben elterjedtté. A hidegháború során az Amerikai Egyesült Államok számára elengedhetetlenné vált, hogy olyan kommunikációs csatornát hozzanak létre az egyetemek kutatóközpontjai és a hadsereg vezetése között, amit az ellenséges kémek által vagy egyéb támadások során nem tudnak használhatatlanná tenni. Emiatt az első, decentralizált csomagkapcsolt hálózati rendszert a DARPA (Defense Advanced Research Project Agency) fejlesztette ki. Az ARPANET (Advanced Research Project Agency Network) NCP (Network Control protocol) alapú kommunikációt tett lehetővé az egyes számítógépek között. Ekkor főként fájlok cseréjére, más számítógépek távoli elérésére és levelezésre használták. Később ezt a protokollt a TCP/IP váltotta le 1983-ban. [9] A TCP az OSI (Open Systems Interconnection Model) szállítási rétegében kapott helyet és működését tekintve három fő részre bonthatjuk. Első lépésként létrejön a kapcsolat két végpont között, ezt követően végbemegy az adatátvitel, majd a kapcsolat lezárul és megkezdődik a felhasznált erőforrások felszabadítása. Az UDP-vel (User Datagram Protocol) ellentétben, a TCP alapú kommunikáció kapcsolat orientál és hibamentes összeköttetést biztosít két eszköz között. A kapcsolat létrehozásának első lépése, hogy az adott program vagy alkalmazás jelzi az az operációs rendszer felé a kapcsolat létrehozási szándékát. Majd a csomópontok közötti kapcsolat kezdeményezése és elfogadása és a szükséges szinkronizáció után megkezdődik az adatok átvitele. Az adatátvitel során a két végpont közötti kapcsolat állandóan fennmarad. A kommunikáció során az egyes folyamatokhoz portokat kell rendelnünk. [10] Az implementáció során a 54010-es port került felhasználásra. Ezt a számot célszerű úgy megválasztani, hogy minél nagyobb értékű legyen,

hiszen az alacsonyabb értékű portok általában az operációs rendszer működése szempontjából kritikus feladatokat látnak el. Az egyes folyamatok csak a saját socketjeiken keresztül kezdeményezhetnek kommunikációt. A szerver és a kliens közötti adatcseréért felelős tömb az 4. táblázat alapján lett létrehozva.

4 táblázat Adatszerkezet

Megnevezés:	Küldött utasítás	1-es motor pozíciója [°]	2-es motor pozíciója [°]	3-as motor pozíciója [°]	4-es motor pozíciója [°]	5-ös motor pozíciója [°]
Adattípus:	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t
Határértékek:	0U - 11U	0U -180U	0U -180U	0U -180U	0U -180U	0U -180U

Az első 8 bites előjel nélküli szám tartalmazza a szervernek küldendő kéréseket. Az egyes kérések magyarázata és értéke a 5. táblázatban láthatóak. A motorok pozícióinak beállításához szükséges paraméterek a kommunikáció során jelenleg nem lettek felhasználva. A szerver által kiszámított egyes pozíciók számított értékeinek befogadására lettek létrehozva.

5. táblázat Szerver kérések

Érték:	Leírás:
0x10	Az automatikus mozgás végrehajtásához szükséges. Ekkor a szerver a válaszában megadja, hogy melyik lokációra kell helyezni az adott tárgyat.
0x11	Ez a kérés a projekt továbbfejlesztése érdekében lett létrehozva és a szerver által kiszámított pozíciók értékeinek lekérdezésére szolgál.

Beérkező tárgy érzékelése esetén, automatikus anyagmozgatási üzemmódban a parancsok lekérdezése a következőképpen zajlik:

1. Kapcsolat létrehozása a szerverrel.
2. Kérés küldése a szerver felé.
3. A beérkező parancs tárolása.
4. Kapcsolat bontása.

Ezt követően a robotkar megkezdi a mikrokontroller memóriájában előre definiált pozíciók felvételét és ezáltal áthelyezi a dobozt a beérkező platformról a jobb vagy baloldali lerakási helyre. A motorok működtetésének és a meghatározott pozíciók leírását a következő fejezet tartalmazza.

4.3. ROBOTKAR MOZGATÁSA

A robotkar mozgatását 5db AD002-es szervomotor végzi. A gyártó honlapján szereplő leírás alapján ezek a szervomotorok mind felépítésben, mind pedig szerkezetileg megegyeznek az MG90S típussal. Így a továbbiakban az MG90S adatlapjában szereplő adatokat használhatjuk. Maximális szögelfordulásuk 180[°]. Terhelhetőségük és működési sebességük függ a tápfeszültség nagyságától. 4.8[V]-os tápfeszültség esetén a motorok egyenként 1.8[kg/cm] forgatónyomatékot képesek kifejteni, míg 6[V]-os tápfeszültség mellett ugyanez az érték már 2.2[kg/cm]-nek felel meg. Ezek alapján lineáris közelítéssel 9[V]-os tápfeszültséggel 3.2[kg/cm], ami 0.314[Nm]-nek felel meg. A legnagyobb megterhelésnek a váll ízületnél lévő motor van kitéve. Abban az esetben, ha a robotkar összes többi motorja 90[°]-os szögben áll és a megfogott tárgy emelését csak ez a motor végzi, akkor az alábbi képlet segítségével kiszámítható, hogy 9[V]-os feszültség mellett mekkora a legnagyobb terhelhetőség.

$$W_{max}[N] = \frac{F_{9V}[Nm]}{D[m]}$$

, ahol W_{max} : a megemelni kívánt tárgy legnagyobb súlya,

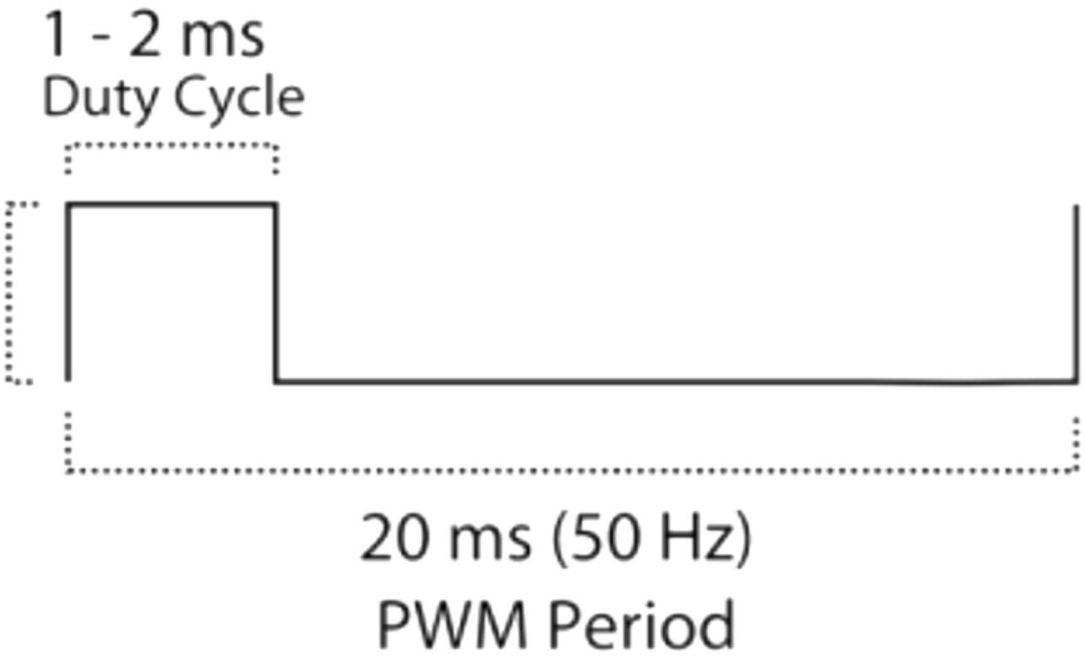
F_{9V} : a motor forgatónyomatéka 9[V]-os tápfeszültség esetén és

D: a tárgy távolsága a motor tengelyének középpontjától.

Így behelyettesítve:

$$W_{max} = \frac{0.314[Nm]}{0.2[m]} = 1.57[N]$$

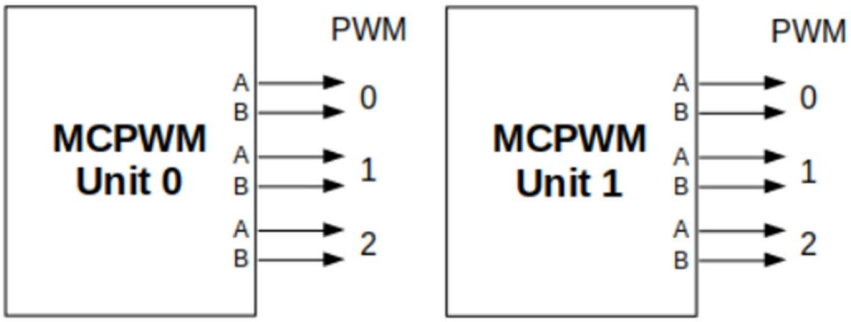
Ez a Földön 0.16[kg]-nak felel meg. Az elkészített színes dobozok tömege egyenként 0.007[kg] a robotkar teljes tömege a motorok kábeleivel együtt pedig 75[g]. Így megállapítható, hogy ezen körülmények mellett a robotkar képes a dobozok mozgatására.



22. Ábra PWM jel az AD002-es motorok működtetéséhez [Forrás: https://www.adeept.com/ad002-servo-motorx2_p0274.html]

Az AD002-es szervomotorok három kivezetéssel rendelkeznek. Ebből kettő a tápellátásért felelős egy pedig a motor vezérléséért. A motorok vezérléséhez szükséges PWM (Pulse-width Modulation) jelet a 22. ábrán szereplő időzítési diagram alapján kell előállítanunk.

A motorok szempontjából 5%-os kitöltési tényező 0[°]-nak, míg 10%-os kitöltési tényező 180[°]-nak felel meg. A PWM jel előállítását a mikrokontroller időzítő perifériájának segítségével



23. Ábra ESP32 Timer [Forrás: https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/mcpwm.html]

hozhatjuk létre. Az ESP32 mikrokontroller összesen négy darab 64-bites hardveres időzítővel rendelkezik, 16 bites előosztással. Az időzítők két főcsoportra lettek osztva. A 23. ábrán látható, hogy mindkét csoport 6-6 kimenetet tud vezérelni. A robotkaron összesen 5 db szervomotor található, így az összes PWM láb vezérlése megoldható egyetlen timer-rel. A periféria beállítását egy külön erre a célra írt servo_channel_setup függvény hajtja végre.

```
ledc_timer_config_t ledc_timer = {
        .speed_mode       = LEDC_LOW_SPEED_MODE,
        .timer_num        = LEDC_TIMER_0,
        .duty_resolution  = LEDC_TIMER_13_BIT,    /*Resolution is 8191*/
        .freq_hz          = PWM_FREQ,             /*50Hz -> 20ms*/
        .clk_cfg          = LEDC_AUTO_CLK
};
ledc_timer_config(&ledc_timer);
```

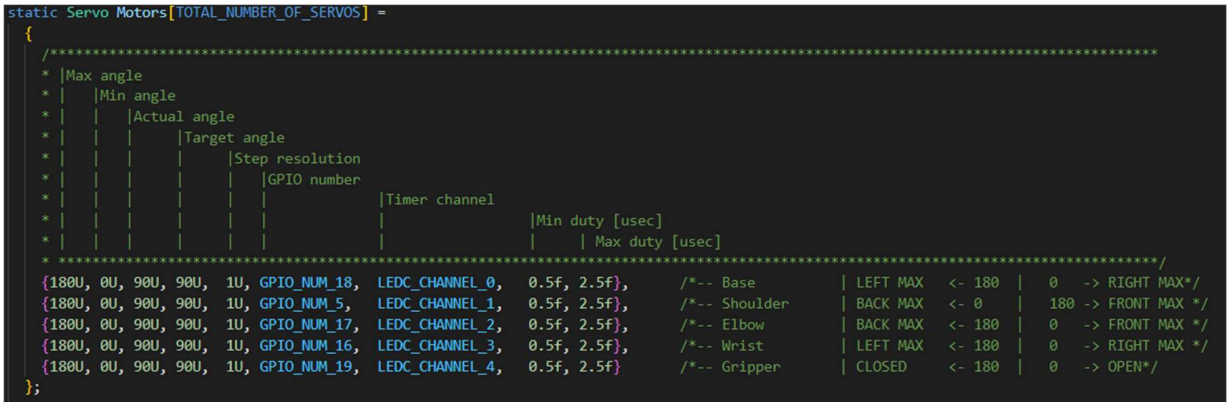
25. Ábra Timer inicializáció

```
typedef struct servo
{
    uint8_t maxAngle;
    uint8_t minAgnle;
    uint8_t actualAngle;
    uint8_t targetAngle;
    uint8_t stepResolution;
    uint8_t servoPin;
    uint8_t ledcChannel;
    float   minServoMs;
    float   maxServoMs;
} Servo;
```

24. Ábra Motorok paramétereinek struktúrája

Az időzítő beállítása a 25. ábrán szereplő kódrészlet segítségével történik. Ennek megfelelően a PWM jel periódusideje 20ms, 13 bites felbontással. 13 biten 0 és 8191 közötti értékeket állíthatunk be, ezek az értékek megfelelnek a PWM jel 0%-os és 100%-os kitöltési tényezőjének. A timer beállítását követően a motorok csatornáinak konfigurálására kerül sor. Jelenleg mind az öt szervomotor ugyanolyan típusú. Ennek ellenére a kódot érdemes úgy elkészíteni, hogy változtatások esetén a konfigurálás minél könnyebb legyen. Ezért először egy struktúrát definiálunk, ami a 24. ábrán felsorolt paramétereket képes befogadni. Ezt követően létrehozunk

egy olyan tömböt, ami a korábban említett struktúrákat képes befogadni a motorok mennyiségével megegyező számban.

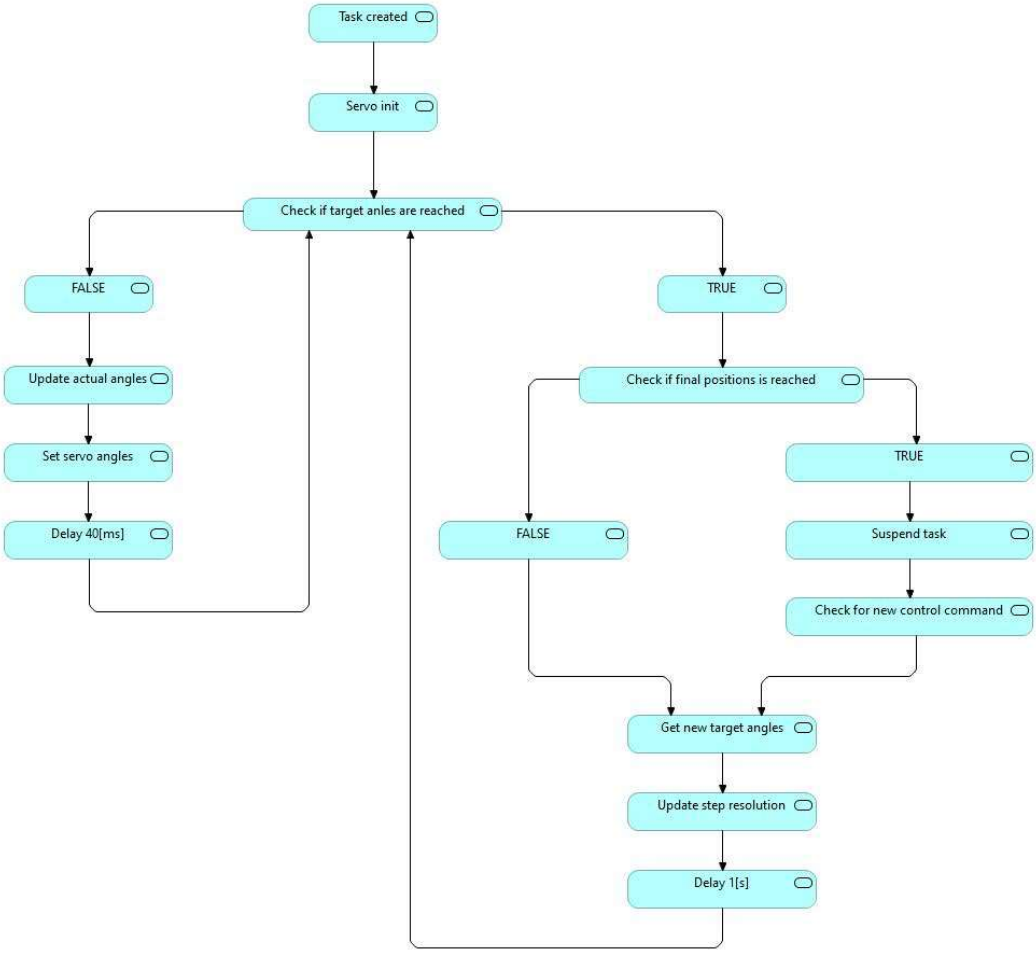


```
static Servo Motors[TOTAL_NUMBER_OF_SERVOS] =
  {
    /*****************************************************************************************************************************************
    * |Max angle
    * |   |Min angle
    * |   |   |Actual angle
    * |   |   |     |Target angle
    * |   |   |     |     |Step resolution
    * |   |   |     |     |   |GPIO number
    * |   |   |     |     |   |   |             |Timer channel
    * |   |   |     |     |   |   |             |                |Min duty [usec]
    * |   |   |     |     |   |   |             |                |     | Max duty [usec]
    * *****************************************************************************************************************************************/
    {180U, 0U, 90U, 90U,  1U, GPIO_NUM_18,  LEDC_CHANNEL_0,   0.5f, 2.5f},      /*-- Base          | LEFT MAX   <- 180  |   0   -> RIGHT MAX*/
    {180U, 0U, 90U, 90U,  1U, GPIO_NUM_5,   LEDC_CHANNEL_1,   0.5f, 2.5f},      /*-- Shoulder      | BACK MAX   <- 0    |   180 -> FRONT MAX */
    {180U, 0U, 90U, 90U,  1U, GPIO_NUM_17,  LEDC_CHANNEL_2,   0.5f, 2.5f},      /*-- Elbow         | BACK MAX   <- 180  |   0   -> FRONT MAX */
    {180U, 0U, 90U, 90U,  1U, GPIO_NUM_16,  LEDC_CHANNEL_3,   0.5f, 2.5f},      /*-- Wrist         | LEFT MAX   <- 180  |   0   -> RIGHT MAX */
    {180U, 0U, 90U, 90U,  1U, GPIO_NUM_19,  LEDC_CHANNEL_4,   0.5f, 2.5f}       /*-- Gripper       | CLOSED     <- 180  |   0   -> OPEN*/
  };
```

26. Ábra Motorok paraméterei

A 26. ábrán látható paraméterek közül az inicializáció során a motorkimenetek lábszámai és az egyes szervokhoz tartozó csatornák számai kerülnek felhasználása. Emellett egyéb, a motorok működéséhez szükséges paraméterek is definiálásra kerültek.

4.3.1. Motorvezérlés megvalósítása



27. Ábra Motorvezérlés állapotdiagramja

A motorvezérlő feladat első futtatásakor a PWM csatornák beállítása után az összes motor kilencven fokos alaphelyzetbe kerülnek. Ez azt jelenti, hogy a robotkar függőleges állásban, félig nyitott megfogó szerkezettel áll. Ezt követően a motorvezérlés belép a fő vezérlési ciklusba. Mivel a projekt működéséhez szükséges szoftver komponensek önállóan kerültek megvalósításra, így a robotkar egészen az integrációs fázisig olyan módon működött, hogy a beérkező platformról felemelte a tárgyat, lerakta azt az egyik oldalon található lokációra, majd oldalt cserélt, végül ezeket a lépéseket ismételte. Ezt a folyamatot úgy oldhatjuk meg a legegyszerűbben, ha a mozgást felosztjuk kilenc különböző pozícióra és a robotkar ezeket az előre definiált pozíciókat veszi fel egymás után. Az egyes állásokhoz tartozó szögértékek egy tömbben tárolódnak, ami elemenként öt darab előjel nélküli, 8 bites egész számot tartalmaz. Így a tömb egyes elemei 1-1 felvehető

pozíciónak felelnek meg. A mozgás megkezdése előtt az összes motor „targetAngle” változóját kell módosítanunk, amit a „servo_uptade_target_angles” függvény meghívásával érhetünk el. Amennyiben a „targetAnlge” és az „actualAngle” változók nem egyeznek meg, megkezdődik annak a kiszámítása, hogy egy iteráció alatt hány fokos szögeltéréssel működjenek az egyes motorok. Ezeket a számításokat minden pozíció felvétele előtt kiszámítjuk. Első lépésként meghatározásra kerül, hogy mekkora az egyes motorok aktuális és elérni kívánt szögének különbsége. Ezen különbségek összegéből ki lehet számítani, hogy mekkora a teljes rendszerre számított szögeltérés. Majd végül azt is, hogy az adott motor ebből mekkora résznek felel meg. Ennek nagyságától függ, hogy az adott motor a következő pozíció eléréséig hány fokos léptetéssel működik. Ez a mennyiség a MIN_STEP_RESOLUTION és MAX_STEP_RESOLUTION makrók értékének változtatásával állítható be. Az értékek módosítása során szem előtt kell tartani, hogy a maximási érték hatással van arra, hogy milyen időközönként lehet frissíteni a motorok PWM jelének kitöltési tényezőjét, ugyanis a motorok 0.11 másodperc alatt maximum 60[°]-os elfordulásra képesek.

```
static void servo_update_step_resolution(void)
{
  int16_t   delta = 0;                                          /*16 bit signed integer needed since int8_t is only from -128 to 127*/
  uint16_t  deltaSum = 0U;                                      /*5 * 180 = 900 -> uint16 = 2^16-1 = 65 535*/
  uint8_t   deltaDegrees[TOTAL_NUMBER_OF_SERVOS];

  for(uint8_t motorIndex = 0; motorIndex < TOTAL_NUMBER_OF_SERVOS; motorIndex++)
  {
    delta = Motors[motorIndex].targetAngle-Motors[motorIndex].actualAngle;
    (delta < 0) ? (deltaDegrees[motorIndex] = (uint8_t)(delta*(-1))):(deltaDegrees[motorIndex] =(uint8_t)delta);      /* ABS function*/
  }

  for(uint8_t deltaIndex = 0; deltaIndex < TOTAL_NUMBER_OF_SERVOS; deltaIndex++)
  {
    deltaSum += deltaDegrees[deltaIndex];
  }
  if(0U == deltaSum)
  {
    deltaSum = 1U;
  }

  for(uint8_t motorIndex = 0; motorIndex < TOTAL_NUMBER_OF_SERVOS; motorIndex++)
  {
    Motors[motorIndex].stepResolution = (uint8_t)((((float)deltaDegrees[motorIndex])/(float)deltaSum)*(MAX_STEP_RESOLUTION-MIN_STEP_RESOLUTION)+MIN_STEP_RESOLUTION);
  }

}
```

28. Ábra Motorok léptetési szögének kiszámítása

A fejlesztés során a léptetési szög maximumát 4[°]-nak megfelelő értékre állítottuk, ami azt jelenti, hogy ezzel az értékkel az elérhető leggyorsabb módosítás 7.33[ms]-onként következhet be. Ennek időzítését a vTaskDelay függvény meghívásával érhetjük el, aminek bemeneti értékként 40[ms]-ot kell megadnunk. Így MAX_STEP_RESOLUTION maximális értéke ezen beállítás mellett 21[°]-nak felel meg. Ezzel a módszerrel csökkenthető, hogy mennyi idő telik el a motorok mozgásainak befejezése között, viszont a fenti paraméterekkel nem érhető el a teljes szinkron mozgás. A motorok léptetési értékének kiszámítási módja a 28. ábrán figyelhető meg. A feladat megvalósításához használt motorok nem rendelkeznek visszacsatolással, így az aktuális szögek megállapításához egy másik függvényt hozunk létre, ami egy adott motor korábbi ismert pozíciójához előjelhelyesen hozzáadja a léptetés nagyságának értékét. Ez a függvény gondoskodik arról is, hogy a motorok ne lépjék túl a maximális és minimális értékeket, illetve, hogy ne forduljanak túl a célértéken. (29. Ábra)

```
static void servo_update_actual_angles(void)
{
  for(uint8_t motorIndex = 0; motorIndex < TOTAL_NUMBER_OF_SERVOS; motorIndex++)
    {
      if(Motors[motorIndex].actualAngle == Motors[motorIndex].targetAngle)
      {

      }
      else if(Motors[motorIndex].actualAngle > Motors[motorIndex].targetAngle)
      {
        if(Motors[motorIndex].stepResolution > (Motors[motorIndex].actualAngle-Motors[motorIndex].targetAngle))
        {
          Motors[motorIndex].actualAngle = Motors[motorIndex].targetAngle;
        }
        else
        {
          Motors[motorIndex].actualAngle -= Motors[motorIndex].stepResolution;
        }
      }
      else
      {
        if(Motors[motorIndex].stepResolution > (Motors[motorIndex].targetAngle-Motors[motorIndex].actualAngle))
        {
          Motors[motorIndex].actualAngle = Motors[motorIndex].targetAngle;
        }
        else
        {
          Motors[motorIndex].actualAngle += Motors[motorIndex].stepResolution;
        }
      }
    }
}
```

29. Ábra Aktuális pozíciók kiszámítása

Ahhoz, hogy a motorokat 0 és 180[°] között mozgathassuk, először meg kell állapítanunk, hogy mekkora érték felel meg az egyes PWM csatornák kitöltési tényezőinek. Az adatlap leírása alapján

ennek az értéknek 410 és 820 között kell lenni, annak érdekében, hogy az impulzusszélesség 1[ms] és 2[ms] között legyen. A számítás menetét a 30. ábrán tekinthetjük meg.

```
uint32_t minDuty = ((int)(((Motors[setIndex].minServoMs)/(MS_PER_PWM_DUTY))*(float)(PWM_MAX_DUTY)));
uint32_t maxDuty = ((int)(((Motors[setIndex].maxServoMs)/(MS_PER_PWM_DUTY))*(float)(PWM_MAX_DUTY)));
uint32_t duty = ((int)((((float)Motors[setIndex].actualAngle/(float)Motors[setIndex].maxAngle)*(maxDuty-minDuty))+minDuty)); /*Calculate duty cycle.*/
ledc_set_duty(LEDC_LOW_SPEED_MODE, Motors[setIndex].ledcChannel, duty);
ledc_update_duty(LEDC_LOW_SPEED_MODE, Motors[setIndex].ledcChannel);
```

30. Ábra Kitöltési érték számítása

A ledc_set_duty függvény meghívásakor a kitöltési tényező értéke nem frissül azonnal, így a motor nem fogja megkezdeni a mozgást egészen addig, amíg ezt nem kezdeményezzük a ledc_update_duty függvény meghívásával. Az ábrán látható lépések minden egyes mozgás során motoronként egyszer végrehajtódnak.

4.3.2. Előre definiált pozíciók és a mozgásfolyamat

```
static uint8_t defined_Positions[MAX_NUMBER_OF_POSITIONS][TOTAL_NUMBER_OF_SERVOS] =
{
  /*************************************************************
*   | base
*   |     | shoulder
*   |     |       | elbow
*   |     |       |       | wirst
*   |     |       |       |     | gripper
* *************************************************************/
/* AUTO MOVEMENT*/
  { 90U,  90U,    90U,    90U,  30U},     /*-- Base position. Robotic arm stand vertically with fully opened gripeper.*/
  { 90U,  152U,   70U,    90U,  30U},     /*-- Lower the arm to pick up position*/
  { 90U,  152U,   70U,    90U,  120U},    /*-- Grip the object.*/
  { 90U,  120U,   70U,    90U,  120U},    /*-- Lift the object*/
  { 0U,   120U,   70U,    90U,  120U},    /*-- Rotate to one of the drop off zones*/
  { 0U,   152U,   70U,    90U,  120U},    /*-- Lower the arm to drop off zone.*/
  { 0U,   152U,   70U,    90U,  30U},     /*-- Put the object down.*/
  { 0U,   90U,    70U,    90U,  30U},
  { 90U,  90U,    90U,    90U,  30U},
  /* SERVER CONTROLLED POSITIONS*/
  { 90U,  90U,    90U,    90U,  90U},
  { 90U,  90U,    90U,    90U,  90U}
};
```

31. Ábra Felvett pozíciók listája

A robotkar két különböző mozgásfolyamatot tud végrehajt annak megfelelően, hogy a dobozt színe szerint melyik oldalra kell mozgatni. Az egyes pozíciókhoz tartozó szögértékek a 31. ábrán láthatóak. Ez alapján megállapíthatjuk, hogy alap esetben az elforgatásért felelős szervomotor 0[°]-os értéket vesz fel. Ennek megfelelően az alapértelmezett végpozíció a robotkartól jobbra található. Annak érdekében, hogy a másik lokációt is el tudjuk érni egy külön függvény segítségével a 4. 5. 6. és 7. indexű pozíciók első elemeit módosítanunk kell 0-ról 180-ra. Ezt a célt szolgálja a 32. ábrán szereplő függvény.

```
void change_side(uint8_t desired_side)
{
  if(desired_side == LEFT_LOC)
  {
    for(uint8_t position_counter = 4U ; position_counter < END_OF_PREDEFINED_POSITIONS ; position_counter++)
      {
        defined_Positions[position_counter][0U] = 180U; /*180 degrees is left for the base motor.*/
      }
  }
  else if(desired_side == RIGHT_LOC)
  {
    for(uint8_t position_counter = 4U ; position_counter < END_OF_PREDEFINED_POSITIONS ; position_counter++)
      {
        defined_Positions[position_counter][0U] = 0U; /*0 degrees is right for the base motor.*/
      }
  }
  else
  {
    /*Do nothing. Wrong side given or the server is in charge.*/
  }
}
```

32. Ábra Iránymódosítás megvalósítása

A folyamat végrehajtását egymást követő, kisebb részfolyamatok képezik. A tömbben tárolt pozíciók funkciói fentről lefelé a végrehajtásuk sorrendje szerint a következők:

6. táblázat Robotkar pozícióinak listája

Index:	Pozíció leírása:
0	Alaphelyzet. A robotkar teljesen kinyújtott állapotban van, a megfogó szerkezet teljesen nyitott.
1	A robotkar teljesen nyitott megfogóval megfelelő pozícióba áll a doboz felemeléséhez.
2	A tárgy megfogásra kerül, a megfogó szerkezet bezárul.
3	A robotkar megemeli a tárgyat.
4	Elfordul a megfelelő platform irányába.
5	A robotkar megfelelő magasságba ereszkedik ahhoz, hogy a dobozt biztonságban le tudja helyezni.
6	A megfogó szerkezet alaphelyzetbe áll, ezáltal elengedi a dobozt.
7	A robotkar az adott platform felett félig kinyújtott helyzetbe áll.
8	Alaphelyzet felvétele.

4.4. TCP/IP szerver létrehozása

A szerveren futó program nem csak az adatok továbbításáért felelős, hanem a képfeldolgozási folyamat elvégzéséért is. Ezért célszerű olyan programozási nyelvet választani, amiben mindkét funkcionalitás könnyen megvalósítható. Ezen indokok miatt a Python programozási nyelv használata mellett döntöttem. A socket könyvtár használatával könnyen létrehozható egy egyszerű TCP/IP szerver. Annak érdekében, hogy a fejlesztés során ne kelljen gyakran átírni a kliens és szerver oldali kódot, a Raspberry Pi számára statikus IP címet állítottam be a router megfelelő menüpontjában. Ezt követően már csak definiálnunk kell a szerver specifikus adatokat, majd az IP cím és a megfelelő port hozzárendelését követően megkezdődhet az aktív kapcsolatok fogadása. Ezt a „listen” függvény meghívásával érhetjük el, aminek az argumentumában megadhatjuk azt is, hogy hány kliens kapcsolatát szeretnénk egymás után kezelni. Jelen esetben csak egy kliens kiszolgálása szükséges, így ezt az értéket ennek megfelelően állítottam be. Ezt követően a program belép a fő ciklusba. Amennyiben létrejött a kapcsolat a klienssel, első lépésként a kliensre vonatkozó adatok kiírása következik, majd a kérés feldolgozása, kiértékelése és a válasz küldése és végül a kapcsolat bontására kerül sor. Az integrációs fázisig a program olyan módon működött,

hogy a jobb és bal oldalhoz tartozó parancsokat csatlakozásonként felváltva küldte ki a kliens számára.

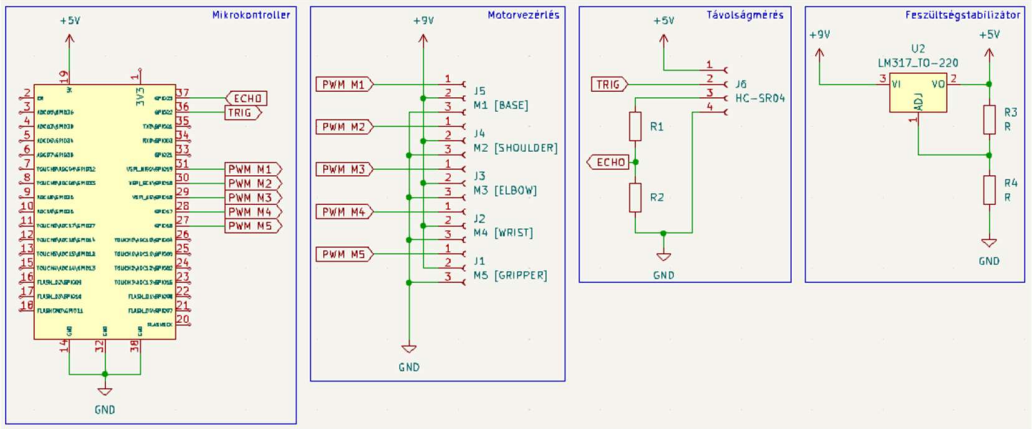
```
import socket

tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_address = ('192.168.0.220', 54010)
tcp_socket.bind(server_address)

tcp_socket.listen(1)
```

33. ábra TCP/IP szerver konfigurálása

4.5. Képfeldolgozás

Ahhoz, hogy a képfeldolgozáshoz szükséges képet el tudjuk készíteni a Raspberry Pi segítségével, először csatlakoztatnunk kell a kamera modult a CSI (Camera Serial Interface) csatlakozóhoz. Ezt követően pedig a „raspi-config” parancs megadásával megnyithatjuk a számítógép konfigurációs beállításait, ezen belül pedig az interfész beállítások menüpont alatt engedélyezhetjük a kamera használatát. Ahhoz pedig, hogy megkezdhessük a képfeldolgozás folyamatát először telepítenünk kell az ehhez szükséges Python modulokat. Az OpenCV (Open Source Computer Vision Library) több, mint 2500 algoritmust tartalmaz. Ez a könyvtár lehetővé teszi, hogy a számítógépes látást, vagy akár a gépi tanulást igénylő feladatokat könnyedén elvégezzük anélkül, hogy nekünk kéne megírni az ehhez szükséges függvényeket. A kamera 60 FPS (Frames Per Second) sebességgel készít képeket, ezek közül a feladat megvalósításához nekünk csak egy képre van szükségünk, amit elmentünk. első lépésként a felvételt szürkeárnyalatos képpé konvertáljuk. A képen előforduló zaj csökkentése és a fókusz beállítása érdekében Gauss-elmosást alkalmazunk. Végül Canny éldetektor függvény segítségével kinyerjük a képen fellelhető tárgyak kontúrjait. A dobozok fizikai adottságai miatt csak olyan tárgyakra vagyunk kíváncsiak, amiket négy egyenes vonalú él határol. Miután a képen megtaláltuk a doboz határoló vonalat, az ezek által körbekerített terület pixeleinek színéből átlagot vonunk. Ennek az értéknek megfelelően pedig megállapítjuk a helyes visszatérési értéket, ami piros tárgy esetén „0x01”, kék tárgy esetén „0x02” és zöld tárgy esetén pedig „0x03”-nak felel meg.

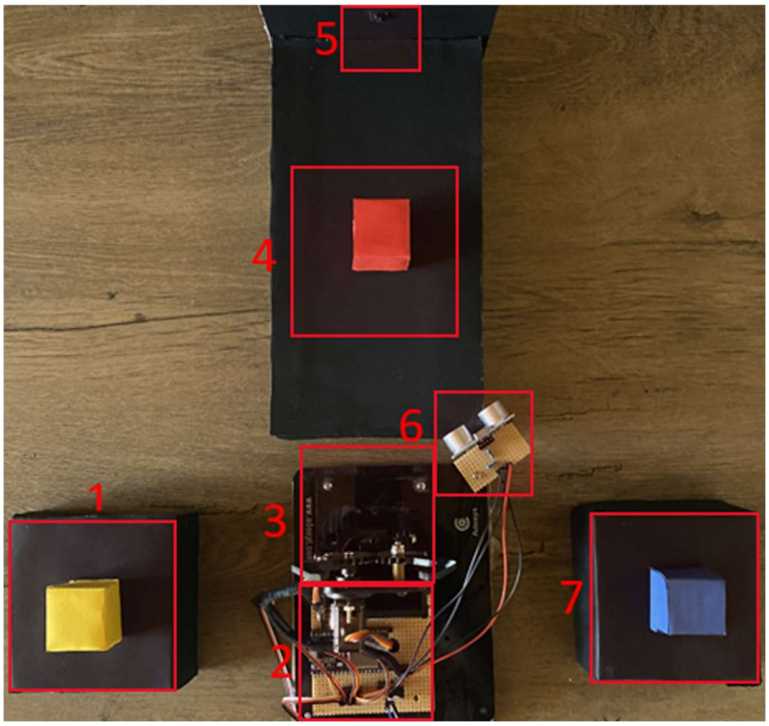
5. INTEGRÁCIÓ



34. Ábra Vezérlőegység kapcsolási rajza

Az integrációs fázis első lépéseként a teljes rendszer kapcsolási rajza került megtervezése, amit a 34. ábrán láthatunk. Ezt követően, a végleges kapcsolási rajz alapján az áramkör fizikai kivitelezése került sorra. A vezérlés fő áramköri lapján az LM317 feszültségstabilizátor IC, a működéséhez szükséges ellenállásokkal együtt, az ESP32 és a távolságmérő modul befogadására alkalmas tüskesor aljzatok, valamint a motorok tüskesora kaptak helyet. Annak érdekében, hogy a távolságmérő szenzort megfelelően be lehessen állítani, a működtetéséhez szükséges áramköri elemek és aljzat egy külön áramköri lapon kerültek összeszerelése. Az egyes modulok fizikai elrendezését a 35. ábrán láthatjuk, ahol az alábbi egységeket figyelhetjük meg:

1. Bal oldali rakodóterület
2. A robotkart vezérlő áramköre
3. Robotkar
4. Munkaterület
5. Kamera
6. Távolságmérő modul
7. Jobb oldali rakodóterület



35. Ábra Fizikai elrendezés

A megépítést követően az egyes, önálló folyamatok időzítésével és a közöttük lévő adatátvitel kivitelezésével foglalkoztam. A taszkok létrehozását követően a program belép a fő vezérlési ciklusba, amit egy állapotgép segítségével oldottam meg. Az egyes taszkok innen kapják meg a működésükhöz szükséges szemafort, amit a feladatuk ellátását követően visszaadnak a folyamatot irányító algoritmusnak. Első lépésként a távolságmérési feladat futtatása kerül sorra mindaddig, amíg meg nem történik a tárgy érzékelése. Ezt követően a távolságmérést felfüggesztjük egészen addig, amíg a doboz át nem került az egyik rakodóterületre és a robotkar vissza nem állt a kiinduló pozícióba. Majd a TCP/IP kliens lekérdezi az adatokat a szervertől. Ennek következtében a központi egység készít egy fényképet a munkaterületről, kiértékeli azt, majd visszaküldi a megfelelő parancsot a vezérlés számára. Végül a motorvezérlő algoritmuson a sor, amely által a robotkar felveszi az egyes pozíciókat és ezáltal áthelyezi az adott dobozt a megfelelő helyre, végül visszaáll az alaphelyzetbe. Innentől pedig az egész folyamat újraindul. Az egymást követő lépések leírásából következik, hogy jelenleg a vezérlés nem képes arra, hogy folyamatosan adatokat fogadjon a központi egységtől, ugyanis az egyes kérések csak akkor kerülnek küldésre, hogy ha bekövetkezett a doboz érzékelése. Egészen addig a kliens taszk futtatása fel van függesztve. Erre megoldást jelenthet, hogy ha a TCP/IP kliens taszk magas prioritással, folyamatosan futna, és a

szerver által szolgáltatott adatokat megadott időközönként lekérdezné, majd az időzítést egy queue használatával oldanánk meg. Ez a megközelítés lehetőséget adna arra is, hogy a robotkar ne csak előre beprogramozott pozíciókat vehessen fel, hanem a doboz pozíciója alapján dinamikusan tudjon alkalmazkodni az adott feladathoz. A szerver oldali integráció a megfelelő programozási nyel kiválasztásának köszönhetően egyszerű feladatnak bizonyult. Mindössze a korábban bemutatott képfeldolgozási algoritmus futtatására volt szükség abban az esetben, ha a kliens ezt kezdeményezte.

6. TESZTELÉS

A tesztelés a fejlesztési folyamattal szorosan összefügg, ám annak önálló és rendkívül fontos része. Megfelelő tesztelés során megbizonyosodhatunk arról, hogy a létrehozott rendszer megfelel a minőségi elvárásoknak, és valóban az előírt feladatot látja el. Emellett természetesen a tesztelés a minőségbiztosítás szerves része. Így minden cég számára elengedhetetlen lenne, hogy megfelelő mennyiségű erőforrást allokáljanak erre a területre. A teszteket két főbb kategóriára lehet osztani, annak függvényében, hogy az adott rendszer belső működése milyen mértékben ismert a tesztelő által. Fehérdobozos tesztelésről beszélhetünk abban az esetben, hogy ha teljes rálátásunk van az adott egység működésére, vagy akár a teljes forráskódot ismerjük. Ezzel ellentétben a feketedobozos tesztelés során a szoftver belső működése a tesztelő elől rejtve marad. [12] Mivel a teljes megvalósítást én végeztem, ezért a korábbi tesztelési eljárást alkalmaztam. Így az adott szoftverre optimalizált teszteket hajtottam végre. A feladat szempontjából azonban nem beszélhetünk tisztán szoftveres tesztelésről, hiszen beágyazott rendszerek esetén a mikrokontrolleren futó szoftver, illetve a megvalósításhoz szükséges hardver egy egységet alkot, így együtt kell őket tesztelnünk. Első lépésként a szervomotorok megfelelő működését teszteltem. Ehhez a 31. ábrán bemutatott tömbben szereplő értékeket úgy módosítottam, hogy minden motor egymást követően felvegye a szögelfordulásának minimum, illetve maximum értékét is. A teszt futtatása során az összes motor működése megfelelt az elvárásoknak. Ezzel az eljárással meggyőződtem arról is, hogy a vezérlés szoftveres implementációja és a motorok bekötése összhangban van, megfelelő sorrendben kezdik meg a működésüket. Ezt követően azt vizsgáltam, hogy mi történik akkor, hogy ha egy-egy motornak 180 foknál nagyobb értéket állítunk be. Ez az eset megfelelően lett kezelve az implementáció során. Túl magas érték esetén a motorok beállnak a maximum értékre, majd a hiba megszűnését követően probléma nélkül képesek folytatni a megfelelő működést. Végül a dobozok megfogását ellenőriztem. Jelenleg a rendszer előre definiált pozíciókkal dolgozik, így abban az esetben, hogy ha nem megfelelő értéket állítunk be a megfogó szerkezetnek, akkor a tárgy sérülhet, vagy ellenkező esetben karok nem zárnak össze kellő mértékben a tárgy megemeléséhez. Erre megfelelő megoldást jelentene, hogy ha a mozgásért felelős motor áramfelvételét vizsgálnánk a megfogási folyamat során. Ezzel a módszerrel a

korábban említett problémákat teljesen meg lehetne szüntetni és a robotkar több, különböző méretű tárgy befogadására is alkalmassá válna. A távolságmérő szenzor működésének vizsgálata során meggyőződtem arról, hogy a szenzortól mért távolságok megfelelnek a valóságnak és arról is, hogy a mérés valóban elvégzésre kerül másodpercenként egyszer. Majd a határértékek beállítását követően ellenőriztem az is, hogy a szoftver csak akkor lép tovább a következő fázisba, hogy ha a tárgy ezen értékek között helyezkedik el. Abban az esetben, hogy ha az echo láb és a mikrokontroller között megszakad a kapcsolat, a mért távolság értékének kiszámítása nem fog megtörténni. Így a további működtetés lehetetlenné válik. A hibát csak az eszköz újraindításával és a csatlakozási probléma megoldásával lehet megszűntetni. Erre megoldást jelenthet, hogy ha egy megfelelő nagyságú időkorlát alkalmazásával megakadályozzuk azt, hogy a program ebben a hurokban ragadjon.

7. ÖSSZEGZÉS

Szakdolgozatom célja egy olyan, önálló anyagmozgató egység modelljének létrehozása volt, ami képes önállóan elvégezni egy összetett anyagmozgatási feladatot, előre programozott lépések végrehajtása által. Ennek megfelelően megalkottam a teljes rendszer tervét, és létrehoztam a feladat megoldásához szükséges szoftveres és hardveres részegységeket is. Megismerkedtem az iparban leggyakrabban használt robotok működésével, a probléma megoldásához szükséges képfeldolgozási algoritmusokkal, illetve a TCP/IP alapú szerver és kliens közötti kommunikáció elméleti hátterével és a gyakorlati alkalmazásával is. A robotkart vezérlő áramkör kivitelezése során tapasztalatot szereztem a szervomotorok használatában és megtanulhattam, hogy hogyan kell a távolságmérő szenzort megfelelően használni egy mikrokontroller segítségével. A felhasznált eszközök csak arra adtak lehetőséget, hogy egy egyszerűbb, de működőképes egységet hozzak létre. A kivitelezés során igyekeztem a további fejlesztési lehetőségeket is szem előtt tartani. Így egyszerűen megvalósítható lenne például az, hogy a robotkar ne előre definiált pozíciókat vegyen fel, hanem a képfeldolgozás során kinyert információk alapján, a szükséges kinematikai számítások elvégzését követően a beérkező tárgyat a tér bármely pontjáról fel tudja venni. Ezen felül egy árammérő modul beszerelésével és a működtetéséhez szükséges algoritmus megírásával el lehetne érni azt is, hogy a megfogó szerkezet változó méretű tárgyak befogadására is alkalmas legyen. Emellett különböző szenzorok használata által elérhető lenne a munkafolyamat optimalizálása is.

8. SUMMARY

The aim of my thesis was to create a model of an autonomous material handling unit capable of independently performing a complex material handling task through the execution of pre-programmed steps. Accordingly, I developed the entire system plan and created the software and hardware components necessary to solve the task. I familiarized myself with the operation of robots most commonly used in the industry, image processing algorithms required to solve the problem, as well as the theoretical background and practical application of TCP/IP-based communication between server and client. During the implementation of the control circuit for the robotic arm, I gained experience in the use of servo motors and learned how to appropriately use a distance sensor with the help of a microcontroller. The tools used only allowed for the creation of a simpler but functional unit. Throughout the implementation, I aimed to keep potential future development opportunities in mind. For example, it could be easily realized that the robotic arm does not use predefined positions but rather picks up incoming objects from any point in space based on information obtained during image processing and following the necessary kinematic calculations. Additionally, by installing a current measurement module and writing the required algorithm for its operation, it could be achieved that the gripper structure is capable of accommodating objects of variable sizes. Furthermore, the use of various sensors could enable the optimization of the work process.

9. IRODALOMJEGYZÉK

[1] Marshall, Thomas H. James Watt., https://web.archive.org/web/20100615081524/http://www.history.rochester.edu/steam/marshall/chapter3.html , 2023.11.20.

[2] Richmond Vale Academy. Second Industrial Revolution: The Technological Revolution. 2021. December 27., https://richmondvale.org/blog/second-industrial-revolution/ , 2023.11.20.

[3] Saurabh Vaidya, Prashant Ambad, Santosh Bhosle, Industry 4.0 – A Glimpse (2018) ScienceDirect 234-238., https://www.sciencedirect.com/science/article/pii/S2351978918300672 , 2023.05.23.

[4] International Standard ISO 8373, https://www.iso.org/obp/ui#iso:std:iso:8373:ed-3:v1:en, 2023.05.17.

[5] Scara Robots, https://www.fanuc.eu/hu/hu/robotok/robotsz%C5%B1r%C5%91-lap/scara-series/selection-support, 2023.08.11.

[6] ESP 32 Technical Reference Manual, https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf, 2023.06.20.

[7] What is PlatformIO, https://docs.platformio.org/en/latest/what-is-platformio.html , 2023.06.20.

[8] Elec Freaks Ultrasonic Ranging Module HC-SR04, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf, 2023.06.20.

[9] Ted Nellen. HISTORY OF THE INTERNET, http://www.tnellen.com/cybereng/history.html, 2023.11.21.

[10] Vinton Cerf, Yogen Dalal, Carl Sunshine, SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM, 19974, https://datatracker.ietf.org/doc/html/rfc675 , 2023.11.21.

[11] Erico Guizzo Types of Robots, https://robots.ieee.org/learn/types-of-robots/, 2023.05.20.

[12] Fehér Krisztián, Automatizált és manuális szoftvertesztelési alapismeretek, szerzői magánkiadás, 2021. ISBN 978-615-6184-14-6

10.ÁBRAJEGYZÉK