



DIPLOMATERVEZÉSI FELADAT

Herőczi Sándor
Mechatronikai mérnök hallgató részére

Mikrovezérlő alapú autonóm fegyverrendszer tervezése és fejlesztése

Napjainkra a technológia fejlődés lehetővé tette a teljesen autonóm hadi eszközök széleskörű megjelenését, elég ha harci drónokra gondolunk. Bár az ilyen autonóm, tehát működésüket tekintve emberi beavatkozástól és felügyelettől mentes fegyverrendszerek használata számos jogi, morális és politikai kérdést vet fel, műszaki szempontból komoly kihívást jelentenek a fejlesztők számára. A diplomamunka célja egy kisméretű, hobbycélokra használható airsoft típusú fegyverrendszer tervezése és megvalósítása.

A hallgató feladatának a következőkre kell kiterjednie:

- Végezzen irodalomkutatást az autonóm fehgyverrendszer területén!
- Készítse el az eszköz koncepciótervét, válassza ki a fontosabb eszközöket!
- Tervezze meg az eszköz egyedi mechanikáját!
- Válassza ki a prototípushoz szükséges elektornikai elemeket, szükséges esetén készítse el az áramköri terveket!
- Valósítsa meg egy mikrokontrolleren az eszköz vezérlőkódját!
- Tesztelje az eszköz működését!
- Dokumentálja a kapott eredményeket!

Tanszéki konzulens: Dr. Stumpf Péter Pál, egyetemi docens

Külső konzulens:

Budapest, 2024. február 18.

Dr. Charaf Hassan
egyetemi tanár
tanszékvezető



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Mikrovezérlő alapú autonóm fegyverrendszer tervezése és fejlesztése

DIPLOMATERV

Készítette
Herőczi Sándor

Konzulens
Dr. Stumpf Péter Pál

2024. december 10.

Tartalomjegyzék

Kivonat	i
Abstract	ii
1. Bevezetés	1
1.1. Motiváció és háttér	1
1.2. Kihívások és célkitűzések	3
1.3. Korlátozások	4
1.4. Erkölcsi nyilatkozat	5
2. Irodalomkutatás	6
2.1. Megvalósult rendszerek	6
2.2. Tüzelési mechanizmus	9
3. Rendszertervezés és követelmények	11
3.1. Rendszeráttekintés	11
3.2. Követelmények	14
4. Mechanikai tervezés	17
4.1. Kinematika	17
4.2. Mechanikai alkatrészek	19
4.3. 3D tervezés, modellezés	21
4.3.1. Skeleton modell	21
4.3.2. Torony	22
4.3.3. Keret	23
4.3.4. Függőleges tengely elemei	25
4.3.5. Fogaskerekek	26
4.4. Gyártás és összeszerelés	28
4.4.1. Gyártás	28
4.4.2. Összeszerelés	30
5. Hardvertervezés	32
5.1. Áramköri tervezés	32
5.2. Elektronikai alkatrészek	34
6. Szoftverfejlesztés	37
6.1. Használt Python modulok, eszközök	37
6.2. Gépi látás	39

6.2.1. Template matching	39
6.2.2. Feature Matching	40
6.2.3. Target Tracking	40
6.2.4. Haar cascade	41
6.3. A szoftver működése	42
6.3.1. A PC-n futó program működése	42
6.3.2. A Raspberry Pi-n futó program működése	46
7. Tesztelés	52
7.1. Elektronikai alkatrészek tesztje	52
7.2. Képfelismerő algoritmusok tesztje	53
7.3. Éles teszt	56
8. Hibák, észrevételek	57
9. Továbbfejlesztési lehetőségek	58
10. Konklúziók	59
Köszönetnyilvánítás	60
Irodalomjegyzék	61
Függelék	63
F.1. A Bambu Slicer beállításai	63

HALLGATÓI NYILATKOZAT

Alulírott *Herőczi Sándor*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Ki-jelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelt után válik hozzáférhetővé.

Budapest, 2024. december 10.

Herőczi Sándor
hallgató

Kivonat

A diplomamunkám célja egy autonóm fegyverrendszer fejlesztése és elkészítése, amely funkciójában hasonlít a valós, éles helyzetben alkalmazott rendszerekhez. Ez annyit takar, hogy a fegyvernek képesnek kell egy bizonyos méretű területet belátni, ebben *felismerni* és *azonosítani* a célpontokat, majd tüzelni, vagy engedélyt kérni tüzelésre. Ezentúl szükséges funkciója a manuális vezérlés, amely segítségével az operátor valós időben tudja távolról irányítani az eszközt, egy asztali számítógép segítségével.

Az eszköz alkatrészei 3D nyomtatási technológiával készültek, a kötőelemeket, csapágynakat és elektronikai elemeket kivéve, első feladatom ezek nyomtatás-helyes megtervezése volt. A következő lépés a hardver és elektronikai rendszer tervezése volt, majd a szükséges modulok, szenzorok, mikrovezérlő kiválasztása és megrendelése. Az utolsó alfeladat pedig a szoftver fejlesztése, amely jelentette minden a beágyazott vezérlő szoftvert, mind a számítógépen futó felhasználói felületet.

A projekt megvalósítása során végig sok időt emészttet fel a rendszer tesztelése, amelyet párhuzamosan kellett végezni a későbbi alfeladatok tervezésével. Kihívást jelentett a megfelelően részletes szakirodalom felkutatása is, ugyanis a valós megoldások paramétereit gyakran nem teszik elérhetővé civilek számára, illetve a jelenleg folyó fejlesztések is titkosítottak.

Úgy gondolom, a téma interdiszciplináris jellegét tekintve jól illik a mechatronikai tanulmányaimba, ugyanis egyaránt érinti a *műszaki mechanika*, a *gyártástechnológia*, a *szoftverfejlesztés* és a *jelfeldolgozás* területeit.

Diplomamunkám eredményeként megvalósítok egy működő prototípust, amit előadás keretében fogok bemutatni. Végül pedig kitérek az esetleges hibákra amiket elkövettem, a tanulságokra és a továbbfejlesztési lehetőségekre.

A projekt teljes egészében, minden modellel és kódossal együtt hozzáférhető az alábbi linken:

https://github.com/heroczi/Diplomamunka_2024

Abstract

The aim of my thesis is to develop and build an autonomous weapon system that functions similarly to real-world systems used in live scenarios. This means that the weapon must be able to *monitor* a designated area, *recognize and identify* targets, then *fire* or *request permission* to fire. Additionally, a necessary feature is manual control, allowing the operator to remotely control the device in real-time using a desktop computer.

The components of the device were created using 3D printing technology, except for the fasteners, bearings, and electronic elements. My first task was to design these components to be suitable for printing. The next step was to design the hardware and electronic system, followed by selecting and ordering the necessary modules, sensors, and microcontroller. The final subtask was the development of the software, which included both the embedded controller software and the user interface running on a computer.

Throughout the project, significant time was consumed by system testing, which had to be carried out simultaneously with the planning of subsequent subtasks. A challenge was also posed by finding sufficiently detailed technical literature, as the parameters of real-world solutions are often not made available to civilians, and current developments are often classified.

I believe that the interdisciplinary nature of this topic fits well with my mechatronics studies, as it encompasses areas such as *mechanical engineering*, *manufacturing technology*, *software development*, and *signal processing*.

As a result of my thesis, I will produce a working prototype, which I will present during my final defense. Lastly, I will address any mistakes I made, lessons learned, and possibilities for future development.

The whole project, along with all its model and code files is openly available at the following URL:

https://github.com/heroczi/Diplomamunka_2024

1. fejezet

Bevezetés

1.1. Motiváció és háttér

Ezt a projektet nem a tanszéki diplomamunkatémák listáján találtam, hanem én kerestem hozzá konzulest. Szeretném ezért először megemlíteni a személyes motivációtam, és érdekeltségeimet vele kapcsolatban. Mint a legtöbb reál beállítottságú fiú, engem is fiatal korom óta érdekel a technológia, a járművek, a gépek, vagy a fegyverek. Ezek közös metszéspontja a haditechnológia, ami általában jóval fejlettebb, mint amivel a civil életben találkozhatunk. Ez az érdeklődés megmaradt kamaszkoromra is, amikor a videojátékok által új ingerként értek a *Team Fortress 2*-ben és a *Portal*-ban lévő ún. **Sentry Gun**-ok.



1.1. ábra. TF2 Sentry Gun [1]



1.2. ábra. Portal Sentry Gun [2]

Ezek olyan lövegek, amelyek a lábukon állva képesek voltak forogni, és ha az ellenség a látóterükbe jutott, akkor arra tüzet nyitottak. Nyilván játékokról van szó, tehát mindig azt hittem hogy ez csak a jövő technológiája, de nem sokkal később rá kellett jönnöm, hogy nagyon is valódiak. Ezután, mikor már egyetemre jártunk, egy barátommal egyre komolyabban kezdtünk beszélgetni arról, hogy esetleg mi is tudnánk egyet építeni. Végül az egyetemi éveim végére érve el is jött a tökéletes alkalom, hogy megvalósítsam régi álmomat.

A szentimentalitást félretéve, természetesen nem választottam volna ezt a témát, ha nem lenne a tanulmányaim szempontjából is releváns. Úgy gondolom, ez a projekt a mechatronika oktatás sok fontos elemét magában hordozza. A feladat a mechanikai konstrukcióval kezdődik és a CAD modellezéssel, amely a gépe-szeti oldalát hasznosítja a képzésünknek. Foglakozni kell a hardverrel, amihez elektronikai ismeretek szükségesek. Végül pedig a szoftvert kell lefejleszteni, ami *informatikai* szempontból érdekes. Ráadásul az egész beágyazott környezetben történik, ami miatt még relevánsabb az *intelligens beágyazott rendszerek* specializációmhoz. Véleményem szerint a projekt nehézsége és kihívása nem az egyes alfeladatokban rejlik, hanem az egész rendszer integrálásában, egymáshoz illesztésében.

Szintén fontosnak tartom megemlíteni, hogy a feladat lehetőséget ad megismerni és használni pár igen modern technológiát. A közelmúltban elérhetővé (és megfizethetővé) váltak civil felhasználásra a *3D nyomtatók*, valamint a fejlődés a *Deep Learning* algoritmusok terén nagyban befolyásolta a gépi látás szakterületét is. Többek között a célom, hogy jobban megismerkedjek ezekkel a módszerekkel, és alkalmazzam őket.



1.3. ábra. Phalanx CIWS rendszer [3]

Mint manapság az ipar minden területén, így a fegyveriparban is egyre nagyobb mértékű a *digitalizáció* és az *automatizáció*. Ennek ékes példája a távolról irányítható fegyverrendszerek térnyerése, amelyeknek nagy előnye a fegyver és a tüzér egymástól való elkülönítése, aminek több előnye is van. Természetesen a legfontosabb és leginkább szembetűnő, hogy ezzel a módszerrel minimalizálható, vagy akár megszüntethető a saját embereink életének kockáztatása. Ezután olyan helyen is tudjuk használni ezeket az eszközöket, ahova egy tradicionális géppuska fészek telepítése nehézkes lenne, például mostoha természeti körülmények közé, egy torony tetejére, vagy akár egy hadihajó oldalára. Szintén egy nagy előny, hogy ezek az eszközök felszerelhetők több kezeléssegítő alegységgel, például hőkamerával vagy éjjellátóval. Majdnem minden, számottevő hadsereggel rendelkező országnak van saját fejlesztésű távirányított fegyverrendszere.

A következő lépés az *automatizálás*. Hiszen egyre erősebb hardverekkel rendelkezünk, egyre jobb algoritmusokat tudunk implementálni, és elérünk az a szintet, hogy bizonyos helyzetekben a "gép" jobb munkát tud végezni, mint egy ember. Az első automatikus célzórendszerrel rendelkező légvédelmi gépágyú az amerikai *Phalanx CIWS* [3] az 1970-es években került kifejlesztésre, ezzel megszületett a "Lethal autonomous weapon (LAW)" kifejezés.

A technológiát érthető módon leggyakrabban védelmi célokra használják, sokszor légvédelemre. A gyakorlatban nagy szerepe van Dél-Korea és Izrael védelmében, ahol a rakétatámadások minden napot veszélyt jelentenek. Offenzív célokra a gyakorlatban még csak rakéták célzására használnak automatikát, a "terminátor" jellegű gyilkos robotok még csak fejlesztési fázisban vannak.

1.2. Kihívások és célkitűzések

A hagyományos, ember által irányított biztonsági rendszerek, illetve manuálisan vezérelt fegyverek hatékonysága korlátozott. Az ember reakcióideje lassú lehet krízishelyzetben, különösen ha nagy sebességű, gyors reagálású, esetleg automatizált a célpont. További gyengesége az embernek, hogy hajlamos hibázni. A fáradtság, stressz, éhség, és rengeteg egyéb tényező kényszerítheti hibára. Ez szükségessé teszi egy olyan automata rendszer kifejlesztését, amely képes azonnal reagálni, felismerni a fenyegetéseket és pontosan célozni.

Az automatikus gépágyúk fejlesztése során számos technikai kihívás merül fel:

Célpontok felismerése és követése: A gépágyúnak gyorsan és pontosan kell érzékelnie és nyomon követnie mozgó célpontokat. Ez kihívást jelenthet változó fényviszonyok, mozgássebességek és környezeti zavaró tényezők mellett.

Pontosság és reakcioidő: A rendszernek gyorsan kell döntéseket hoznia, ugyanakkor elengedhetetlen a pontos célzás. A mechanikai mozgásvezérlés, a számítógépes látás és az elektronikai rendszerek szinkronizációja mind kritikus tényező a rendszer hatékonysága szempontjából.

Környezetérzékenység: Külső környezeti tényezők (pl. időjárás, akadályok, fényviszonyok) befolyásolhatják a rendszer működését, ezért a szoftvernek és a hardvernek *rugalmasnak* és *robustusnak* kell lennie. A dolgozatom célja egy innovatív megoldás kidolgozása, amely eleget tesz a megszabott követelményeknek és megoldást nyújt a fenti problémákra.

Kiemelt célok:

- Egy megbízható célpontfelismerési rendszer fejlesztése, amely változó környezeti feltételek mellett is képes hatékonyan működni.
- Egy stabil és erős mechanikai konstrukció tervezése, amely képes követni a szoftver utasításait.
- Egy olyan valós idejű szoftvervezérlési rendszer megalkotása, amely gyorsan reagál a célpontok mozgására és változására.

Összegezve tehát:

Egy olyan automata gépágyú fejlesztése és megépítése, amely magabiztosan képes felismerni egy célpontot, követni azt, célozni, és tüzelni.

1.3. Korlátozások

Természetesen tisztában kell lenni bizonyos korlátozásokkal a projekt elkészítése közben. Csupán két félévem van lefejleszteni, egyedül vagyok, és nincsen százmilliós bűdzsém, mint az iparban hasonló kutatással foglalkozó csapatoknak. Ebből kifolyólag reálisan kell látni a helyzetet, és úgy meghatározni a követelményeket, hogy egy egyetemi hallgató számára is elérhető legyen.

A kamerarendszer például, amit a valós megoldásokban használnak, önmagában tízmilliós tétel szokott lenni. Sajnos az én megvalósításom valószínűleg nem fog működni sötétben, nagy távolságokban, vagy esőben, hiszen a költségvetésbe nem fér bele az éjjellátó, a hőkamera, az optikai zoom vagy több kamerából álló rendszer.

Nem áll rendelkezésemre korlátlanul se CNC gép, se fém 3D nyomtató, így a legtöbb alkatrészt műanyagból kell elkészítenem. Ez befolyásolhatja a szerkezet stabilitását, amit figyelembe kell venni a későbbiekbén.

És végül fontos megemlíteni, hogy mivel valamilyen játékfegyvert kell majd használnom, ez befolyásolni fogja a gépágyú pontosságát, amit szintén figyelembe kell venni a tervezéskor és értékeléskor.

1.4. Erkölcsi nyilatkozat

Az automata fegyverrendszer morális megítélése vitatott téma, ezért szeretném kijelenteni, hogy diplomamunkám, melynek címe „*Mikrovezérlő alapú autonóm fegyverrendszer tervezése és fejlesztése*”, kizárolag tanulmányi és mérnöki célokát szolgál. A dolgozatom keretében fejlesztett prototípus semmilyen formában nem szándékozik ösztönzi az erőszakos, káros vagy törvénysértő tevékenységeket. Fejlesztésem célja a technológiai kutatás, a mechatronikai és automatizálási rendszerek megismerése, illetve a gépi látás alkalmazásainak tanulmányozása.

Határozottan elhatárolódom bármilyen rosszindulatú felhasználástól, és hangsúlyozom, hogy a projekt eredményei nem használhatók fel ártalmas vagy illegális célokra. A munka során mindenkor tiszteletben tartottam az etikai irányelveket, és felelős mérnöki magatartást tanúsítottam. Az általam fejlesztett rendszer kizárolag oktatási, kutatási és technológiai demonstráció célját szolgálja.

2. fejezet

Irodalomkutatás

A legjelentősebb katonai hatalmak mindegyike rendelkezik távvezérelt fegyverrendserekkel, leggyakrabban valamilyen távolról irányított gépfegyver formájában. Azonban se az egyes országok nemzetbiztonságának, se a fegyveripari partnereknek nem áll érdekében a szükségesnél több információt kiadni. Ez egy kicsit megnehezítette az irodalomkutatást, de a képek alapján azért sok információt ki lehet nyerni.

2.1. Megvalósult rendszerek

CROWS

Az egyik legnagyobb darabszámban gyártott távirányított fegyverrendszer az amerikai CROWS [4] rendszer, amely a NATO-országokban, köztük Magyarországon is rendszeresített. Ennek értelmében telepíthető sok NATO által használt páncélozott járműre, köztük a *Humvee*-ra, a *Stryker*-re, és a *Buffalo MRAP*-re. Több verziója létezik több kaliberrel, a 2.1. ábrán egy M240B géppuskával látható.



2.1. ábra. Az amerikai CROWS rendszer [4]

A szerelék a fegyverrel együtt 360°-ban képes elfordulni, és -20°-tól +60°-ig tud billenni. A fegyvercső giroszkóppal stabilizált. A kezelő egy 15 hüvelykes kijelzőn tud célozni a fegyverrel. A rendszer a sima kamera mellett rendelkezik hőkamerával is, így éjszaka is használható. Mind a két kamera el van látva lézeres távolságmérővel, amivel rá lehet állni a célpontra, és a jármű mozgása közben is lehet azt követni. A kamerát és a fegyvert lehet külön is mozgatni, ami azért hasznos, mert anélkül lehet követni a gyanús alakok mozgását, hogy félelmet keltenénk az emberekben.

Arbalet-DM

A CROWS rendszer orosz megfelelője a hasonló kialakítású *Arbalet-DM* [5] (2.2.ábra). Ennek a rendszernek az alapja a 12.7 mm-es *KORD* nehézgéppuska. Rendelkezik 4 gránátvetővel is, amelyek füstfüggöny felhúzására használhatók. A kamera és a fegyver elhelyezése, de még a lőszer pozíciója is teljesen hasonló az amerikai párrához.



2.2. ábra. Az orosz Arbalet-DM rendszer [5]

A rendszer 360°-ban képes elfordulni, és -20°-tól +70°-ig tud billenni, tehát egy kicsit nagyobb részt tud lefedni, mint a CROWS. A hatótáv nappal 2000 m, éjszaka 1500 m. Ez a rendszer is el van látva hőkamerával és lézeres távolságmérővel.

DeFNder

A következő megoldás a belga *DeFNder* [6] termékcsalád, amelynek két tagja a *Light* és a *Medium*(2.3. ábra). Értelem szerűen a kettő közül az utóbbi az, amelyre nehezebb fegyverzetet lehet telepíteni. A függőleges tengelyen 360°-ban képes elfordulni 90 fok/másodperc sebességgel, a vízszintes tengelyen -45°-tól +75°-ig tud billenni, 60 fok/másodperc sebességgel. Opcionálisan ellátható infravörös- és hőkamerával a rossz látási körülmények esetére, valamint lézeres távolságmérővel a ballisztikai kompenzációhoz.

Elég sok közös vonást találtam a fentebb említett rendszerekben. Az elrendezésük nagyon hasonló, van egy függőleges forgástengely nagyjából a teljes rendszer



2.3. ábra. A belga DeFNder rendszer [6]

súlypontján keresztül, valamint egy vízszintes forgástengely, nagyjából a fegyver csövével egy síkban. Erre valószínűleg azért van szükség, hogy a tüzelés során keletkező erők ne ébresszenek csavarónyomatékot a mozgató mechanizmuson. Az én esetben nagy erők nem fognak ébredni, de a tervezési elvet érdemes követni.

Samsung SGR-A1

Az egyetlen, valós harci helyzetben használt, teljesen autonóm gépágyú a koreai fejlesztésű *Samsung SGR-A1* [7]. A két Korea között húzódó demilitarizált övezet (DMZ) egy 250 km hosszú, 4 km széles sáv, amelyet mind a két oldalon szigorúan ellenőriznek. A határ folyamatos felügyelete rengeteg ember munkájába kerül, ami egy demográfiai válságban lévő ország csökkenő hadseregeben egyre értékelősebb. Főleg annak tekintetében, hogy csupán járőrözni és figyelni a határt nem feltétlenül igényli egy ember jelenlétéét. Ennek tudatában fejlesztették ki a Samsung és a Korea Egyetem mérnökei az SGR-A1 fegyverrendszerét. Hőkamerával és éjjellátóval felszerelve napközben 4 km-ről, éjszaka 2 km-ről képes azonosítani potenciális célpontokat, tehát tulajdonképpen a DMZ teljes szélességében. Képes felismerni az embereket, követni őket, és megkülönböztetni más élőlényektől. Hangfelismeréssel képes azonosítani a közeledő személyeket: amennyiben valaki 10 m-nél közelebb kerül és nem azonosítja magát, a rendszer riaszthat, gumilövedéket lőhet, vagy használhatja a K-3 gépfegyverét.

Civil szakértők számára nem egyértelmű, hogy a rendszer lőhet-e emberi engedély nélkül, és a hivatalos koreai álláspont az, hogy nem. Azonban ha már bemérte és ellenséggént azonosította a célpontot, akkor nehéz elképzelni, hogy pont a ravaszt ne tudná meghúzni.



2.4. ábra. Samsung SGR-A1 [7]

2.2. Tüzelési mechanizmus

A korábban említett rendszerek éles fegyverekkel vannak felszerelve, amit terméshetesen nem fogok követni. Így valamilyen olyan megoldást kellett találni, ami legális, elegendően pontos és megfelelően be lehet vele mutatni a célfelismerés és célzás működését.

Paintball

Több, a diplomamunkámhoz hasonló projektet is találtam, amelyek *paintball* [8] puskákat használnak. Ezek a puskák sűrített levegőt alkalmaznak, hogy egy festékkel töltött golyót lőjenek ki. A torkolati sebességük nagyjából 280 fps, maximális hatékony távolságuk kb 25-30 m. Mivel a lövedék alakja gömb, és a cső sincs huzagolva, ezért nem túl pontos, főleg hosszú távon.



2.5. ábra. Paintball puska alapú rendszer [9]

További problémák, hogy a legolcsóbb paintball puska is 70000 Ft. fölött van, valamint a lövedék is viszonylag drága, és nem lehet újrahasznosítani. Ezentúl a fegyverek nagyok, nehezek, és ezért nehéz a beépítésük.

Nerf

A *Nerf* [10] fegyverek a legelterjedtebb játékfegyverek. Sűrített levegővel lőnek ki egy hosszúkás szivacs lövedéket. A sűrített levegőt egy megfeszített rugó elengedésével érik el, amelyet vagy kézzel, vagy valamilyen átteles villanymotorral húznak fel. A legjobb modellek torkolati sebessége 70 fps körül van, és kb. 15 m a hatótávolságuk. Ezen a távon viszonylag pontosak a lövedék kialakításából adódóan, de jelentős az esés, így a ballisztikai pályát komolyan kell venni.

Ezek a fegyverek modelltől függően 10000 Ft. - 40000 Ft. között mozognak, de a lövedékeket újra lehet használni. Az a probléma itt is fennáll, hogy nagy a fegyver teste, így nehezebb beépíteni.

Airsoft

Az *airsoft* [11] fegyverek hasonló módon működnek, mint a *Nerf* puskák, de egy kisebb, műanyag golyót lőnek. Az elektromos airsoft fegyverek torkolati sebessége általában 300 fps és 400 fps között van, amit befolyásol a golyók tömege, a rugó minősége és rengeteg egyéb alkatrész. A hatótávjuk az átlagos fegyvereknek kb. 50 m, de fejlesztésekkel elérheti a 90 m-t is. A lövedék itt is golyó és a cső sincs huzagolva, mint a paintballnál, azonban az airsoft esetében használnak ún. hop-up kamrákat, amelyek perdületet adnak a golyónak.

Az airsoft fegyverek legnagyobb előnye a többi lehetőséggel szemben, hogy van egy kompakt egység, a "gearbox", ami felelős az elsütésért. Ezt ki lehet szedni egy fegyverből, vagy akár külön is meg lehet venni. Ez sokkal nagyobb szabadságot ad a beépítéshez, és a végeredmény is sokkal kompaktabb lesz. Az elsütés is csak az áramkör zárását jelenti a gearboxban, ami egyszerűen vezérelhető a mikrokontrollerrel.

3. fejezet

Rendszertervezés és követelmények

3.1. Rendszeráttekintés

Az autonóm fegyverrendszer fejlesztése során egy komplex mechatronikai rendszert kellett megvalósítani, amely több különálló modul együttműködését biztosítja. A rendszer fő komponensei közé tartozik a *mechanikai szerkezet*, az *elektronikai hardver*, az *érzékelők*, a *vezérlő egység*, valamint a *szoftveres háttér*. Ezek együttesen biztosítják a rendszer autonóm és manuális működését. Az alábbiakban részletes áttekintést adok a rendszer főbb elemeiről és azok feladatáról.

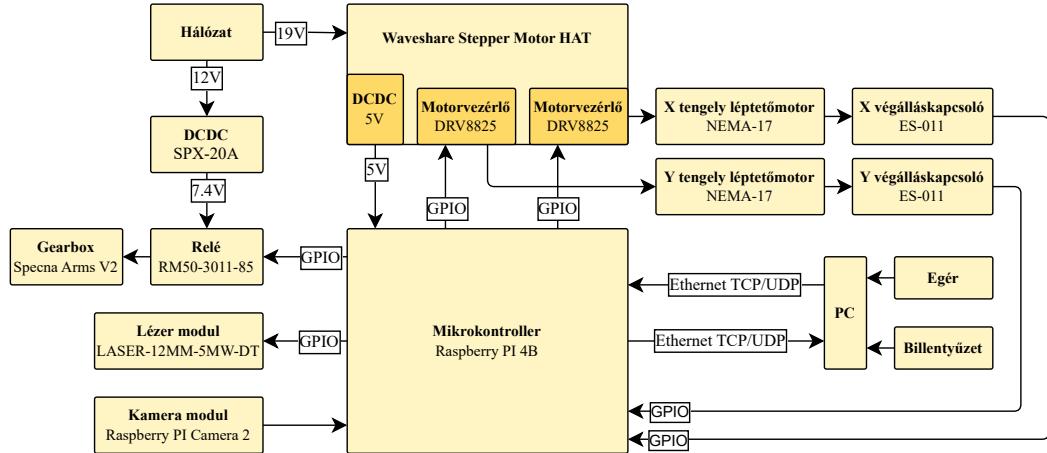
Mechanikai konstrukció

A váz gyanánt tulajdonképpen egy kéttengelyes *pan-tilt* mechanizmust kellett megvalósítanom, ami felelős a fegyver stabilan tartásáért és precíz mozgásáért. A váz építőelemei zömében 3D nyomtatási technológiával készültek, ami lehetővé tette a problémák gyors kiküszöbölését, illetve a nagyfokú szabadságot a tervezés során. Ahol tudtam, kereskedelemből beszerezhető alkatrészeket használtam, például a csapágyakat és a kötőelemeket.

A modellt több alegységre lehet bontani a funkciójuk szerint. Van a *torony* nevető alösszeállítás, amely feladata a gearbox stabil befogása, illetve erre épül rá a kamera konzolja, a tár összeállítása, a fogaskerék és a lézer is. A következő a *keret*, amelybe kerülnek a torony csapágyai, illetve az elektronika egy része. Erre az elemre van erősítve a Raspberry Pi konzolja, a relé, illetve az egyik motor is. A torony alatt található a *nagy csapágyhoz* tartozó elemek, amelyeknek feladata a stabil összeköttetés biztosítása a keret és a talp között. Legalul helyet kapott a *talp*, amely magába foglalja a lábakat, illetve az azokat összefogó elem is. Ezt úgy terveztem, hogy a lábak szükség esetén cserélhetők legyenek.

Elektronikai rendszer

Az elektronikai rendszer blokkdiagrammja a 3.1. ábrán látható.



3.1. ábra. Az elektronikai rendszer blokkdiagrammja

Az elektronikai rendszer két fő eleme a *Raspberry Pi*, illetve a *gearbox* volt. Mivel a *gearbox* igen nagy áramot képes felvenni, külön tápegységgel kellett ellátnom a mikrovezérlőt illetve a golyó kilövéséért felelős elektronikát. A motorok vezérléséért egy kereskedelemben kapható, *Raspberry*vel kompatibilis shieldet használtam, amely jelentősen megkönnyítette a NEMA-17 típusú léptetőmotorok csatlakoztatását. A lézer diódát és szükséges szenzorokat a *Raspberry Pi* lábairól vezéreltem, ahogy a *gearbox* aktiválásáért felelős relét is. A kamera modul egy *Raspberry Pi Camera V2* volt, amit a mikrovezérlőn található foglalaton keresztül tudtam elérni.

Az elektronika központi egysége természetesen a *Raspberry Pi* mikrokontroller volt, amely felelős volt a fő képfelismerő algoritmus futtatásáért, a szenzorok jeleinek feldolgozásáért, valamint a PC-vel való kommunikációért. A mikrokontroller kiegészült egy *Stepper Motor HAT*-el, amely egy kereskedelemben kapható áramkör. Megtalálható rajta két db. *DRV8825* motorvezérlő chip, illetve kiegészítő áramkörök, pl. microstep állító kapcsolók, valamint áramkorlátozó potméterek. Ezentúl helyet kapott egy 5V-os DCDC konverter, amin keresztül a *Raspberry Pi*-t is elláthatjuk árammal. A léptetőmotorok *NEMA-17* típusúak, és 4 tűskés csatlakozóval rendelkeznek.

Az elektronikai rendszerben szerepel még a *gearbox*, ami a tüzelésért felel. Mivel igen nagy áramot képes felvenni, ezért külön áramellátást terveztem a *gearbox*-nak. Egy 12V-os szervertáp működteti, amiből egy *SPX-20A* típusú, nagy teljesítményű DCDC konverter csinál a *gearbox* számára optimális 7.4V-ot. Ez a kiemelő feszültség állítható, a *gearbox* egészen 11V-ig képes működni. A kiemelő feszültséggel tulajdonképpen a tüzelés sebességét tudjuk szabályozni. A *gearbox* áramellátását egy relével lehet vezérelni, amely a *Raspberry Pi* lábára csatlakozik.

A rendszerben helyet kapott minden két tengelyre 1-1 végálláskapcsoló, amelyek a fegyver kalibrálásáért felelősek. Ezek a GPIO tüske soron csatlakoznak a mikrovezérlőhöz. Hasonlóképpen a *lézer dióda* is, amely a fegyver célzásával hivatott segíteni a felhasználót. A Raspberry Pi kamera csatlakozóján keresztül pedig beépítésre került egy *Raspberry Pi Camera 2* típusú kamera. A mikrovezérlő a rajta található *Ethernet* porton keresztül kommunikál a PC-vel.

Számítógépes vezérlés és szoftveres háttér

A szoftver rendszere két alegységből áll, a Raspberry Pi-n **beágyazott vezérlő szoftverből**, illetve a PC-n futó **felhasználói alkalmazásból**. Mivel magát a prototípust szeretném minél jobban közelíteni a valósidejű működéshez, ezért a két alegység közötti kommunikáció sebessége kiemelkedően fontos. Ezért döntöttem úgy, hogy a Raspberry Pi az Ethernet portján keresztül csatlakozzon a PC-hez. Ez ugyan kissé megköti a fegyver mozgásterét, de nem annyira, hogy feláldozzam a kapcsolat gyorsaságát.

A Raspberry Pi-n futó szoftver felelős a léptetőmotorok mozgatásáért, a felhasználó parancsainak fogadásáért, a célpont felismeréséért, illetve a kamera képének továbbításáért a PC felé. Ezeket a folyamatokat jól elkülönülő modulokra bontottam, amelyek egyszerre futnak külön szálakon. A különböző folyamatok közötti kommunikációra használtam megosztott erőforrásokat, állapotjelzőket és queue-kat. A kézi vezérlés és az automata működés között is tettek különbösséget, a szoftver bizonyos részei között is megosztott változókkal lehet választani.

A PC-n futó szoftver felelős a Raspberry által küldött videó dekódolásáért és megjelenítéséért a *HUD*-on. A *HUD*-on szerepelnek még fontos állapotjelzők, pl. hogy éppen melyik működési módban van a rendszer, vagy hogy be van-e biztosítva a fegyver. Emellett a szoftvernek fel kell dolgozni a felhasználótól kapott parancsokat is, és tovább kell küldenie a Raspberry Pi felé. Ez a program is két külön szálon futó alegységből áll, amelyek megosztott erőforrásokkal kommunikálnak egymással.

3.2. Követelmények

Az autonóm fegyverrendszer fejlesztése során számos követelményt kellett figyelembe venni annak érdekében, hogy a rendszer megbízhatóan, hatékonyan és biztonságosan működjön. Ezek a követelmények a rendszer mechanikai, elektronikai és szoftveres elemeire egyaránt kiterjedtek. A következő szakaszokban részletezem a legfontosabb műszaki és funkcionális követelményeket.

Mechanikai követelmények

A mechanikai komponensek tervezése során az alábbi követelményeknek kellett megfelelni:

- **Stabilitás és pontosság:** A fegyverrendszer mechanikai szerkezetének stabilnak és tartónak kell lennie annak érdekében, hogy a lövések közben ne mozduljon el, és ne veszítse el a célpontot. Ugyanakkor elegendő pontosságot kell biztosítania a célpont precíz követéséhez.
- **Fürgeség:** A rendszernek képesnek kell lennie a fegyver gyors és pontos mozgatására a pan-tilt mechanizmus segítségével. Ennek megfelelően a szervomotoroknak kellően gyorsnak és erősnek kell lenniük ahhoz, hogy valós időben tudják követni a mozgó célpontokat.
- **Strapabíró konstrukció:** Habár nagy terhelés nem fogja érni, a gearboxból jöhetnek rezgések, rángások, amik esetleg problémát jelenthetnek egy alul-méretezett alkatrész esetében.

Elektronikai követelmények

Az elektronikai rendszer megbízható működése érdekében az alábbi követelményeknek kellett eleget tenni:

- **Megfelelő teljesítmény:** A szervomotoroknak és szenzoroknak megfelelő tápegységre van szükségük, amely stabil energiaellátást biztosít. A rendszer energiaigényét előzetesen fel kellett mérni, hogy a tápegység terhelés alatt is megfelelően működjön.
- **Szenzorok pontossága:** A kamerának és egyéb szenzoroknak elegendő felbontással és érzékenységgel kell rendelkezniük ahhoz, hogy képesek legyenek a célpontokat megfelelően azonosítani. A valós idejű képfeldolgozás nagy adatsebességet és megbízható szenzorjeleket igényel.
- **Vezérlés:** A mikrokontrollernek kellően gyorsnak kell lennie, hogy a valós idejű adatokat folyamatosan feldolgozza és a vezérlési parancsokat késlekedés nélkül végrehajtsa.

Szoftveres követelmények

A rendszer működéséhez szükséges szoftverfejlesztés során a következő követelményeknek kellett megfelelni:

- **Valós idejű feldolgozás:** A számítógépes látás szoftverének valós időben kell elemeznie a kamerák által közvetített adatokat, felismerve a célpontokat és kiszámítva a mozgás irányát. A célzási és lövési döntések gyors és hatékony adatfeldolgozást igényelnek.
- **Biztonságos működés:** A rendszernek rendelkeznie kell olyan szoftveres biztonsági funkciókkal, amelyek megakadályozzák a véletlen tüzelést. Ez magában foglalja a lövési engedélykérés mechanizmusát és a manuális vészleállítás lehetőségét.
- **Felhasználói interfész:** A felhasználónak egyszerű és intuitív felhasználói felületet kellett biztosítani, amelyen keresztül könnyedén vezérelheti a rendszert, illetve áttekintheti a célpontok adatait és a kamera képét. A felületnek támogatnia kell a kézi irányítást és a lövési parancsok kiadását.

Funkcionális követelmények

A rendszer teljes funkcionalitásának biztosítása érdekében a következő kritériumoknak kellett megfelelni:

- **Autonóm működés:** A rendszernek képesnek kell lennie arra, hogy teljesen önállóan felismerje és kövesse a célpontokat, valamint meghozza a tüzelési döntéseket az előre meghatározott paraméterek alapján.
- **Manuális vezérlés:** Az autonóm működés mellett manuális vezérlési lehetőséget is kellett biztosítani az operátor számára. Ezen keresztül a felhasználó közvetlenül irányíthatja a fegyvert és manuálisan adhat lövési parancsot.

Biztonsági követelmények

Az autonóm fegyverrendszerek használatával kapcsolatban különösen fontos a biztonsági követelmények teljesítése:

- **Vészleállítás:** A rendszernek rendelkeznie kell egy vészleállító gombbal, amely azonnal megszakítja a fegyver működését, ha bármilyen hiba vagy vészhelyzet lép fel.
- **Engedélyezési mechanizmus:** A tüzelési parancs kiadása előtt a rendszernek engedélyt kell kérnie az operátortól, ezzel minimalizálva a véletlen tüzelés kockázatát.
- **Adatbiztonság:** A vezérlő szoftver és az operátor közötti kommunikáció titkosítva kell, hogy legyen, hogy megakadályozza a külső hozzáférést és a rendszer kompromittálását.

Környezeti követelmények

A rendszernek különféle környezeti feltételek között is megbízhatóan kell működnie:

- **Hőmérsékleti tűréshatár:** A rendszernek képesnek kell lennie normál működésre különböző hőmérsékleti körülmények között, amelyek tipikusan a beltéri használat során fordulnak elő.
- **Nedvesség és porállóság:** A rendszert úgy kell megtervezni, hogy ellenálljon a kisebb por- és nedvességterhelésnek, különösen, ha kültéri használatra is szükség van.

Azokhoz a követelményeket, amelyekhez tudok konkrét értéket rendelni, az alábbi táblázatban gyűjtöttem össze:

Nr.	Követelmény	Érték
1.	Szögsebesség	45°/ s
2.	Valósidejúság kézi vezérlésen	10 ms
3.	A célpont felismerése a képbe kerülés után	0.5 s
4.	Vízszintes tengely mozgási tartomány	-10°<x<75°
5.	Függőleges tengely mozgási tartomány	-135°<y<135°
6.	Tüzelés pontossága 5m-en, 20cm x 20cm	80%
7.	Felhasználói felület felbontása	640x480
8.	Hőállóság	-10°C<T<50°C

3.1. táblázat. Az órajel-generátor chip órajel-kimenetei.

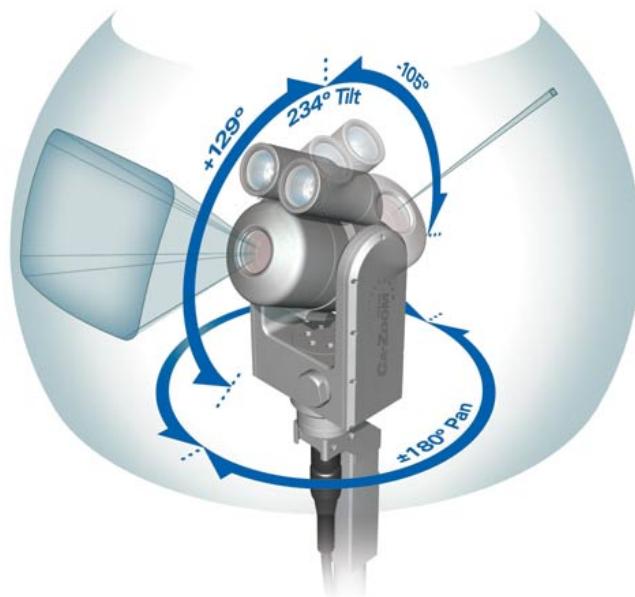
4. fejezet

Mechanikai tervezés

A mechanikai tervezést egy kinematikai modell kidolgozásával kezdtem, majd a meglévő, kereskedelemben kapható alkatrészek méreteihez igazítva megterveztem a 3D CAD modellt. Ezután finomhangoltam a 3D nyomtatási technológiához megfelelően.

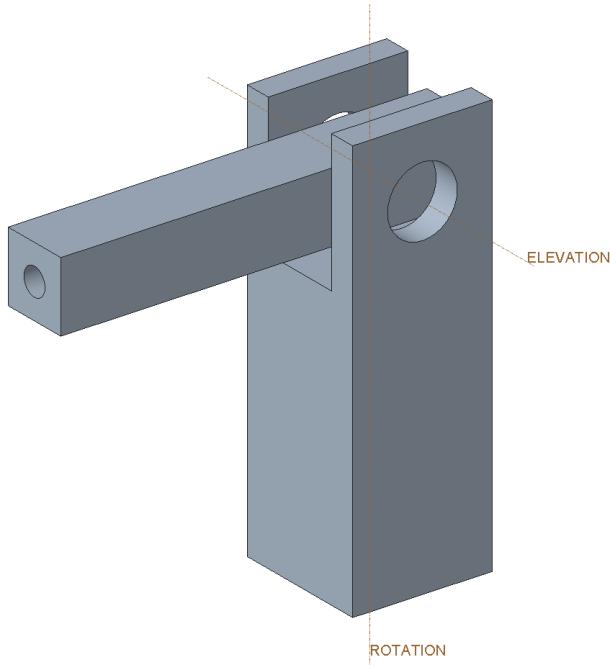
4.1. Kinematika

A rendszer kinematikai modellje hasonlatos a biztonsági kamerákhoz, aminek elnevezése *Pan-Tilt-Zoom*, röviden *PTZ*. Ennek lényege a 360 fokban elforgatható függőleges tengely, és egy általában korlátoltabb vízszintes tengely. Előnyük, hogy nagy sebességgel tudnak irányt változtatni, és nagy területet képesek belátni. A zoom aspektus az én esetemben nem fontos, hiszen nem a kamera a fő funkció. A 4.1. ábrán erről látható illusztráció.



4.1. ábra. PTZ kamera

A 2.1. bekezdésben vizsgált rendszerek is hasonlóképpen mozognak. Ezek közelebbi vizsgálata után elkezdtem kidolgozni a saját koncepciómat. Szemléltetésképpen készítettem a 4.2. ábrát. Az egyszerűség kedvéért a fegyver csövét egy síkba terveztem a pan és tilt tengelyekkel, ez megkönnyíti a későbbi számításokat.



4.2. ábra. Egyszerűsített kinematikai ábra

Ki kellett számolnom bizonyos geometriai megkötéseket, amelyek szükségesek a tervezéshez, illetve az alkatrészek kiválasztásához.

A torony szükséges fordulatszámát a következőképpen lehet kiszámolni:

$$rpm_{min} = \frac{v_t}{2 \cdot \pi \cdot r} = \frac{2.778 \text{ m/s}}{2 \cdot \pi \cdot \text{m}} = 0.442 \text{ 1/s} = 26.526 \text{ 1/min} \quad (4.1)$$

ahol:

v_t A célpont sebessége a fegyvercsőre merőlegesen,

r A távolság a célpont és a rendszer között

Meg lehet állapítani a torony mozgásának felbontását is, tehát hogy hány fokonként lehet állítani a mozgását.

$$\alpha_{min} = \arcsin\left(\frac{a}{2 \cdot r}\right) \cdot 2 = \arcsin\left(\frac{0.3 \text{ m}}{2 \cdot 10 \text{ m}}\right) \cdot 2 = 1.719^\circ \quad (4.2)$$

ahol:

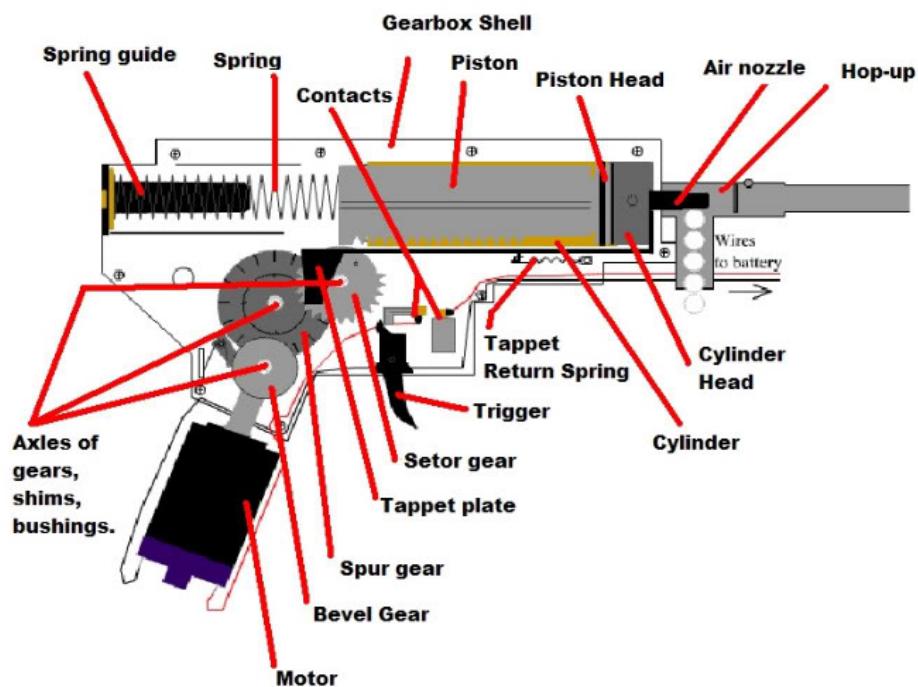
a A célpont mérete

r A távolság a célpont és a rendszer között

4.2. Mechanikai alkatrészek

Elsütő mechanizmus

Az elsütő mechanizmusnak egy Specna Arms M4-ből kiszerelt gearbox-ot, illetve annak csövét és hop-up kamráját használtam. A gearbox működése során egy villanymotor több áttételen keresztül hátrahúz egy dugattyút, és azzal együtt a mögötte lévő rugót. Mindeközben a hop-up kamrában betöltődik egy golyó, és megáll a csőben. Ahogy a részlegesen fogazott fogaskérék elengedi a dugattyút, azt a rugó előrelöki, ezáltal a dugattyúkamrában nagy légnyomás keletkezik, ami a fegyver csövéén keresztül tud kiegyenlítődni. Folyamatos működés során a motor egymás után húzza fel és engedi el a dugattyút, így amíg van golyó a tárban képes tüzelni. A gearbox belső alkatrészei a 4.3. ábrán láthatóak.



4.3. ábra. V2 gearbox alkatrészei [11]

A hop-up kamrán belül még található egy gumi csúszófelület, amivel a kilőtt golyó perdületét lehet állítani, ezáltal pedig a fegyver effektív távolságát növelni.

A gearbox-on belül alakítanom kellett a működésen, hogy az előbb leírt folyamatos működést biztosítani tudjam. Az elsütőbillentyűt, a tűzkapcsolót és minden ehhez tartozó mechanikai elemet kiszereltem, illetve áthuzaloztam. Erre azért volt szükség, mert különben csak a ravasz meghúzásával lehetett volna tüzelni, ami bonyolít a rendszer megvalósításán. A gearbox belső kialakítása a 4.4. ábrán látható.

A fegyver gearbox-ot körülvevő alkatrészeiről le tudtam venni méreteket, ez alapján alakítottam ki később a 3D nyomtatott alkatrészeket. Így végeredményben egy magába zárt alkatrészem lett, amin habár mechanikus nem lehet állítani



4.4. ábra. A gearbox belseje[11]

a tüzelés módját, de csupán két vezetékkel csatlakozik az elektronikához.

A gearbox áramfogyasztását illetően találtam méréseket az interneten, ahol kifejezetten az én modelemet tesztelték. Az itteni mérések alapján arra következtettem, hogy 15 A-ra méretezni az áramkört megfelelő lehet. [12]

Motorok

A projekthez kettő NEMA-17 léptetőmotort használlok[13]. Ezek a léptetőmotoroknak egy széles körben alkalmazott típusa, amelyeket főként precíziós mozgatási feladatokhoz használnak, például CNC gépekben, 3D nyomtatókban, robotikai alkalmazásokban és automatizálási rendszerekben. A NEMA-17 esetében a szám a motor elülső oldalának névleges méretét jelenti, amely 1.7 hüvelyk, azaz körülbelül 42,3 mm.

A léptetőmotorok a mozgásukat apró, egyenlő lépésekre osztják, így lehetővé téve a precíz pozícionálást és sebességszabályozást. A NEMA-17 léptetőmotor tipikusan kétfázisú bipoláris motor, amely négy vezetékes tekercseléssel rendelkezik. minden lépés során a motor egy adott szöggel fordul el, ami az adott motor típusától és felépítésétől függően tipikusan 1.8 fok, így teljes fordulat esetén 200 lépésre van szükség.

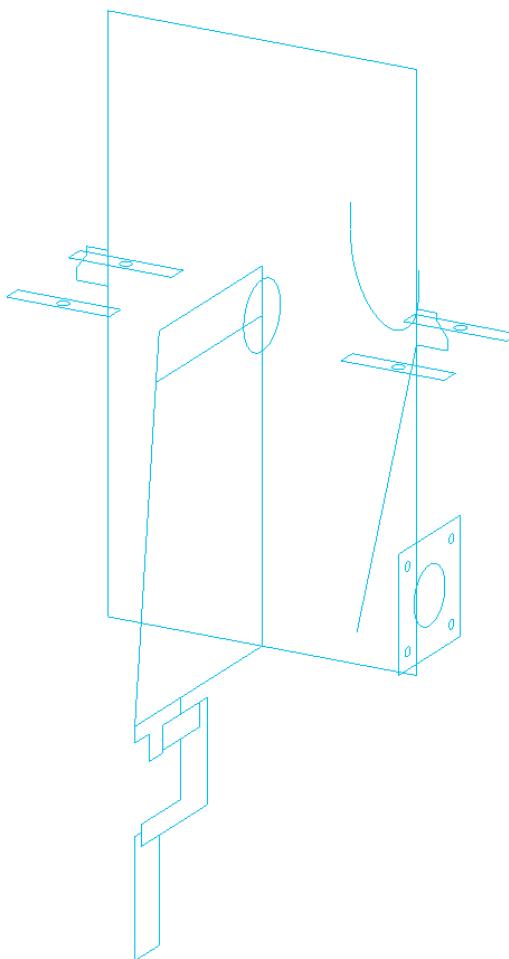
Az én esetemben használt léptetőmotorok lépésszöge 1.8 fok, bár ezt a motorvezérlőn lehet tovább osztani. A tartónyomatéka 0.4 Nm. Ezek a paraméterek 10-es áttételű fogaskerék-kapcsolattal megfelelőnek kell lenniük egy ilyen kis teljesítményű alkalmazásnál.

4.3. 3D tervezés, modellezés

A 3D tervezést Top-Down módszerrel végeztem, ez azt jelenti, hogy először az összeállítás szintjéről kezdem a tervezést, és egy úgynevezett skeleton modellbe veszem fel az egyes alkatrészek méreteit. Ezáltal tudom garantálni az egymáshoz való illeszkedést, valamint a változtatások könnyű implementálását.

4.3.1. Skeleton modell

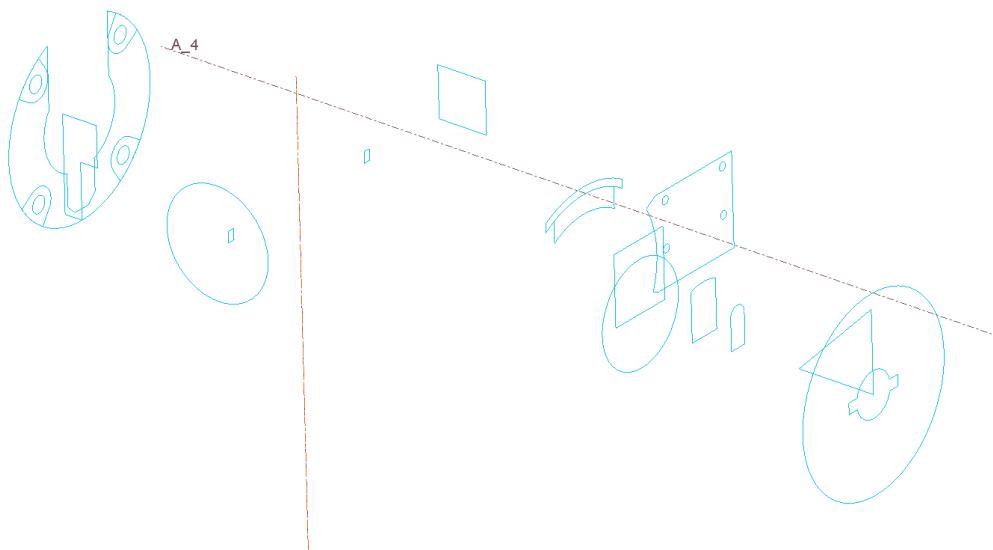
A modellezést a skeleton modellek megalkotásával kezdtem. A fő összeállításon belül két skeleton modellt hoztam létre, mivel maga a modell két alösszeállítása jól elkülöníthető egymástól.



4.5. ábra. DT-9999 skeleton modell

Az egyik (4.5.ábra) alkatrészei rendszerint a függőleges tengely körül szimmetrikusak, a másiké (4.6.ábra) pedig a fegyver csöve körül.

Az ábrákon láthatóak a vázlatok és tengelyek, amelyek alapján kialakítottam az egyes alkatrészeket. Sok méretet először csak hozzávetőlegesen vettem fel, pl. a torony magasságát. A Top-Down módszer előnye, hogy később ezeket könnye-



4.6. ábra. DT-4999 skeleton modell

dén módosíthatom, és az alkatrészek frissítés után ugyanúgy fognak egymáshoz illeszkedni.

4.3.2. Torony

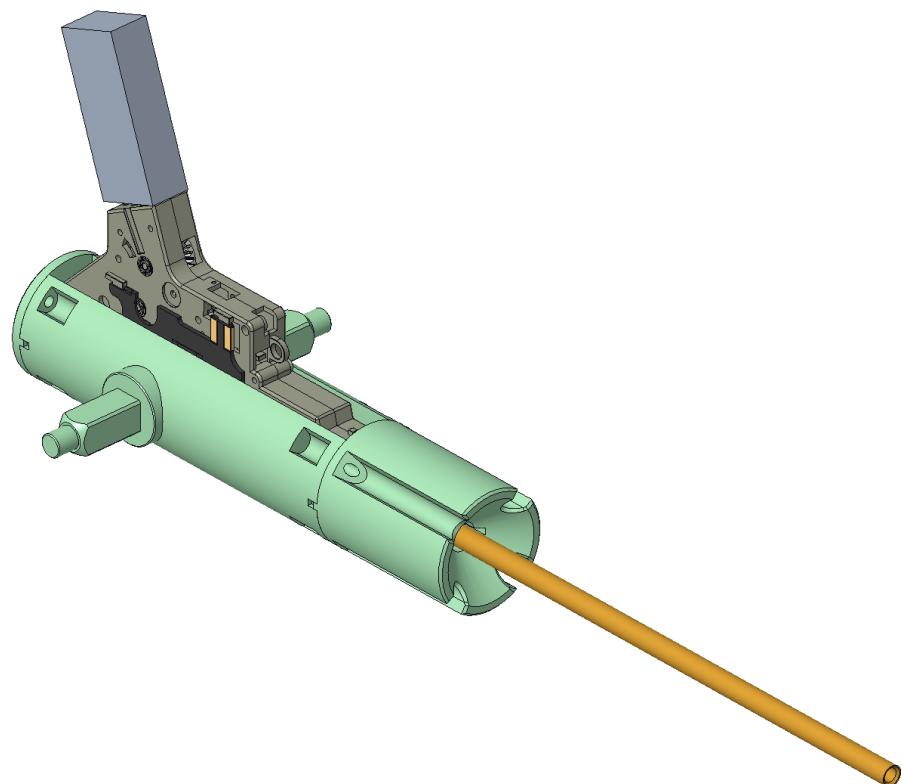
Elsőként a **gearbox tartó elemet** kezdtem el tervezni, mert maga a gearbox volt a tervezés elején az egyetlen elem, amiből tudtam következtetni a szükséges méretekre. Erről kép a 4.7. ábrán látható.

A gearbox ház 3 alkatrészből áll, egy központi elemből, valamint két fedélből a végein. A kialakítást a teljes airsoft fegyver alapján terveztem, hogy ugyanúgy álljon a gearbox mint az eredeti felhasználása során. A kritikusabb rész ebben az elemben a hop-up kamra körüli kialakítás volt, itt elég komplex volt a geometria, de a 3D nyomtatás lehetővé tette a megvalósítást. Ezt a 4.8. ábrán lehet látni.

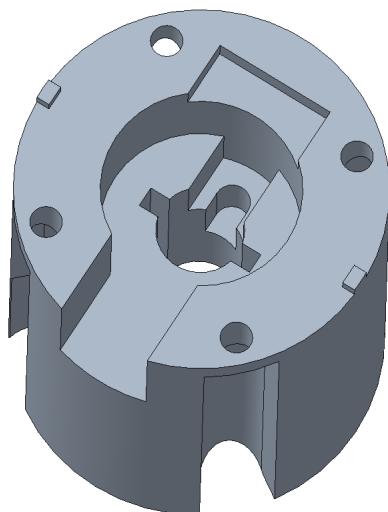
A következő alösszeállítás a **tár** volt, amely a gearbox ház bal oldali tengelyére csatlakozik. Ennek kialakítása a 4.9. ábrán látható. A 6 mm-es golyók a zöld kupak nyílásán keresztül tölthetőek a lila tárba. Alul és felül 1-1 gumigyűrű feszíti előre a zöld dugattyút, ami a tár kúpos végén keresztül nyomja ki a golyókat. Így mechanikailag, plusz elektronika nélkül biztosítható a fegyver lövedékkal való ellátása, és szemmel is látható a tár töltöttségi szintje.

Ez a kialakítás végül nem volt sikeres, ugyanis a golyók nagyon könnyedén elakadtak, gyakorlatilag egyszer sem sikerült lőni vele. Újragondolás után egy hasonló megoldást használtam, ám a golyók egyesével sorakoznak a tárban, ez látható a 4.10. ábrán.

Végül a toronyra kerültek az elektronikai alkatrészek is, ezek a 4.11. ábrán láthatóak. A lila elem a lézer modul, a piros pedig a kamera konzol. Mind a kettő ragasztással kerül rögzítésre. Fontosnak tartottam, hogy ezek az elemek közvetlenül aholhoz az elemhez legyenek rögzítve, ami a fegyver csövét is pozicionálja, azonban a kamera esetén a 3D nyomtathatóság miatt egy konzolt szükségesnek éreztem.



4.7. ábra. Gearbox tartó ház



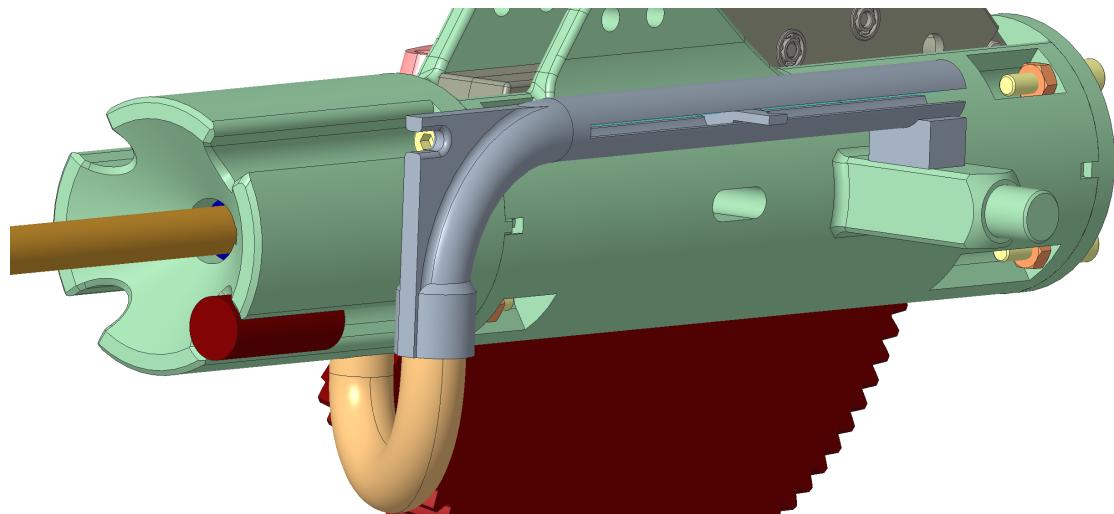
4.8. ábra. Hop-Up kamra körüli elem

4.3.3. Keret

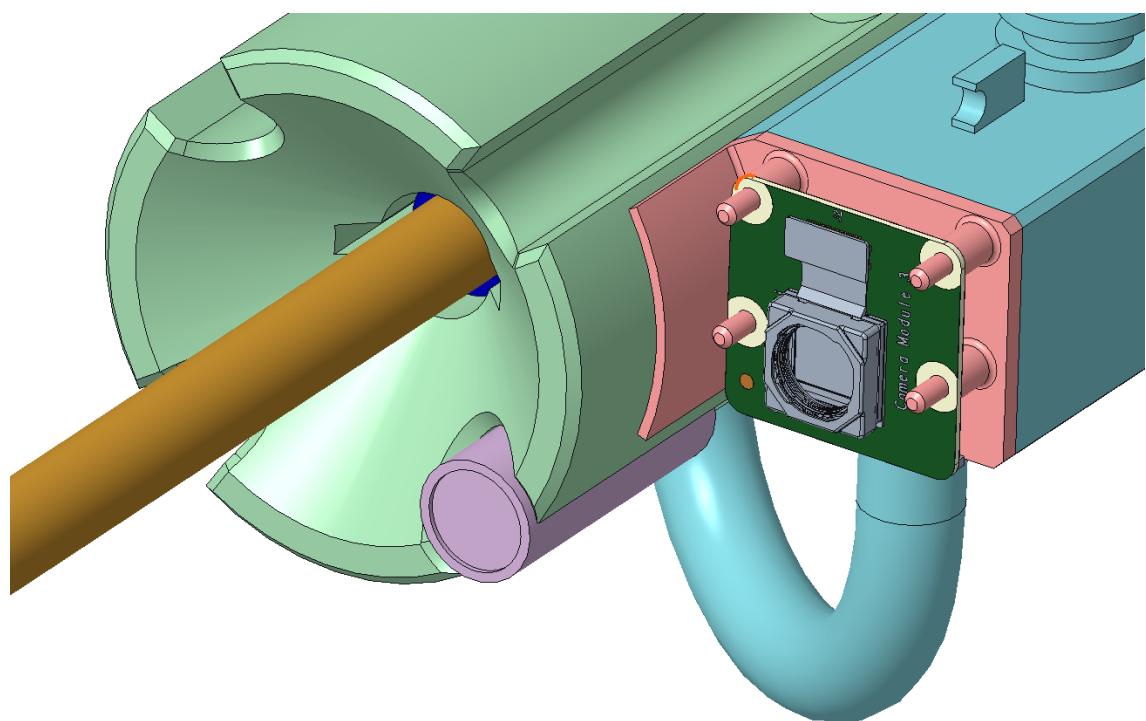
Az úgynevezett **keret** volt a konstrukció legösszetettebb eleme, és egyben a legnagyobb térfogatú is. Fő funkciója a torony stabil tartása a csapágyakkal együtt. Ezentúl erre az elemre van rögzítve a vezérlőelektronika nagy része, a végállás-kapcsolók és a vízszintes tengelyhez tartozó motor is. Ez az elem az alatta lévő alkatrészhez ragasztással lett rögzítve, hogy a szerelést egyszerűsítse. A csap-



4.9. ábra. Tár

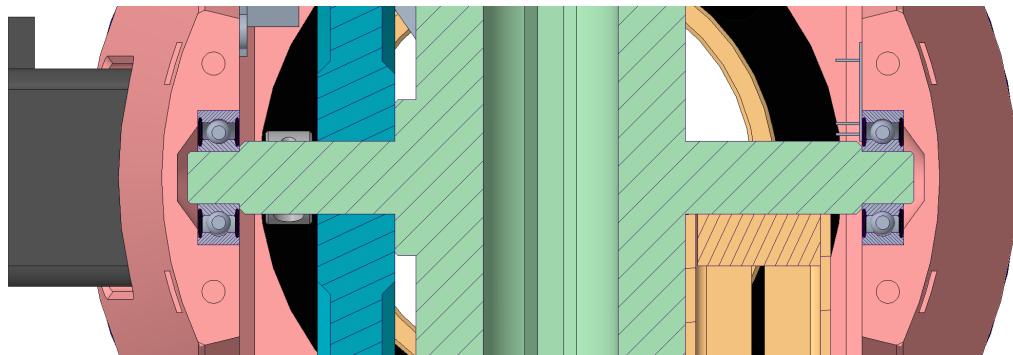


4.10. ábra. Új tár



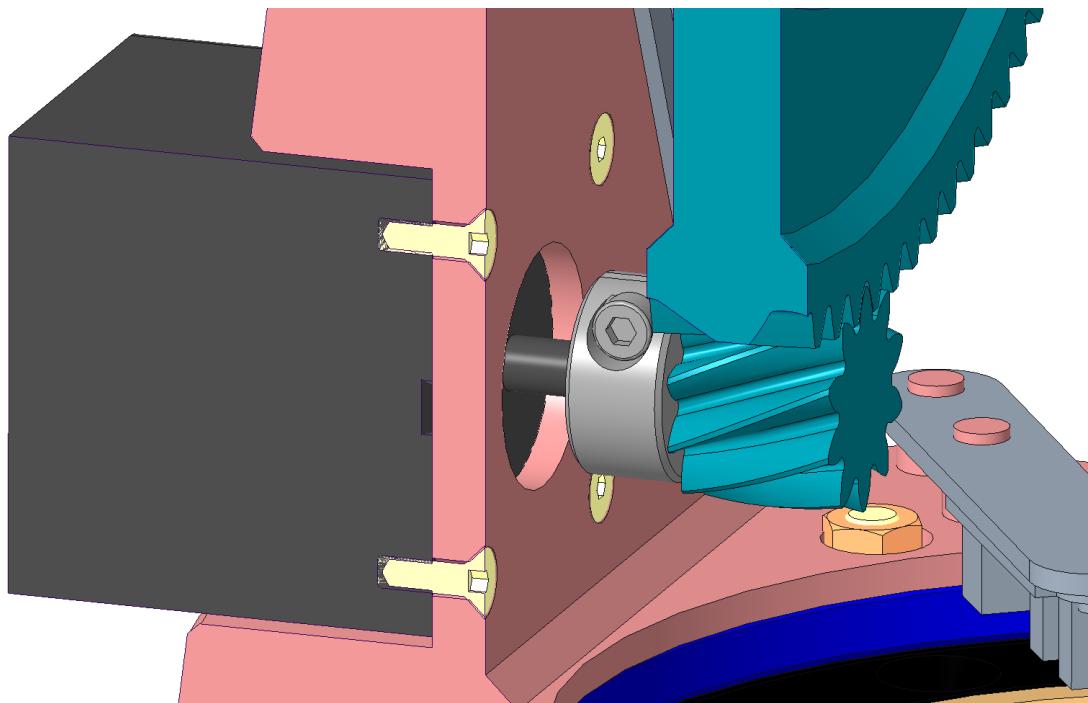
4.11. ábra. Kamera és lézer modul beépítés

ágyak támasztása X elrendezésű, és az osztott "csapágyház" miatt könnyen szérelhetőek. A csapágy elrendezése a 4.12. ábrán látható.



4.12. ábra. Vízszintes tengely csapágyainak elrendezése

Kétséges rész volt a **motor rögzítése**, ugyanis nagy szükség van a pontos illeszkedésre. A nyomtatás irányára azonban a motor központosítására szánt furat pont merőleges, de szerencsére a nyomtató így is elég nagy pontosságot tudott elérni.

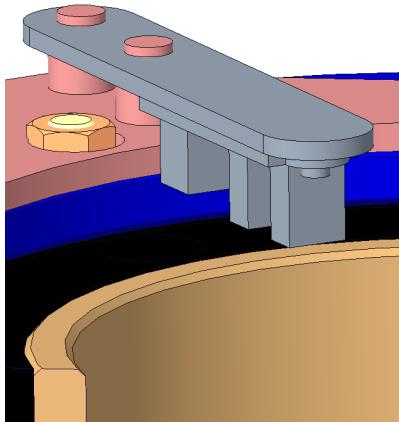


4.13. ábra. Vízszintes tengely motor beépítése

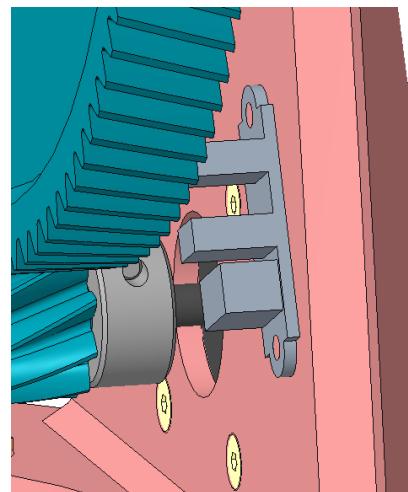
A kereten kaptak helyet a végálláskapcsolók is, amivel a rendszer indításkor kalibrálható. A egyik a vízszintes tengely nagy fogaskerekéhez, a másik pedig a függőleges tengely központi eleméhez viszonyít.

4.3.4. Függőleges tengely elemei

Utolsó lépésként megterveztem a függőleges tengely mozgatásáért felelős elemeket. ezek kialakítása a 4.16. ábrán látható. A narancssárga alakrész a központi elem, amelynek feladata a csapágy támasztása és a motor rögzítése. A motor



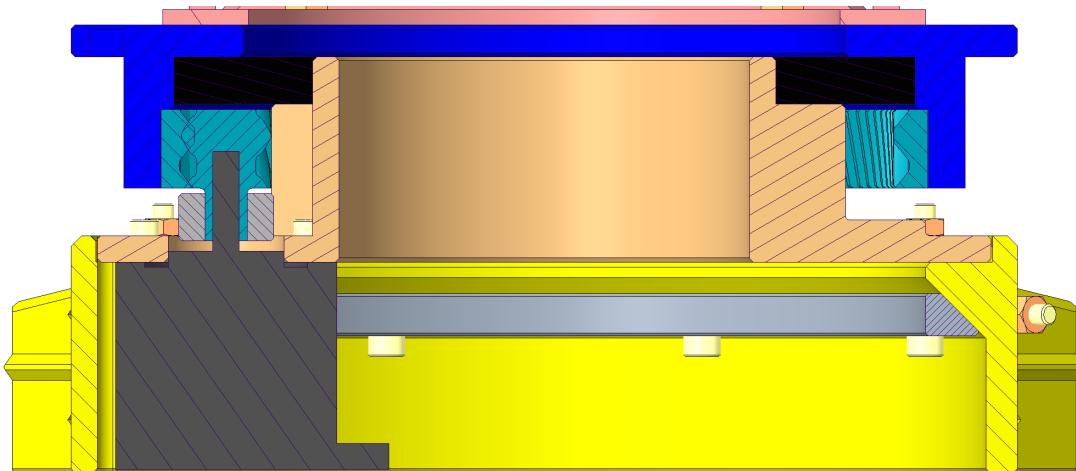
4.14. ábra. Pan végálláskapcsoló



4.15. ábra. Tilt végálláskapcsoló

vezetékei, valamint az elektronikából kilógó többi kábel ennek a közepén fut keresztül. A csapágy egyik fele ehhez az elemhez van rögzítve, a másik pedig az ábrán kékkel ábrázolt alkatrészhez. Erre a kék elemre került rögzítésre a belsőfogazású fogaskerék is.

A sárga alkatrész a talapzat része. Ez a legnagyobb kiterjedésű elem az egész modellben. Hogy elkerüljem a támaszanyag használatát, a csavarok fejéinek felfekvő felületeit egy külön alkatrész ragasztásával oldottam meg. Ez a talp 45 fokos belső felületére illeszkedik, és ragasztással kerül rögzítésre. A 4 oldalán kialakításra kerültek csatlakozó felületek, ahol különböző lábak helyezhetők az eszközre.

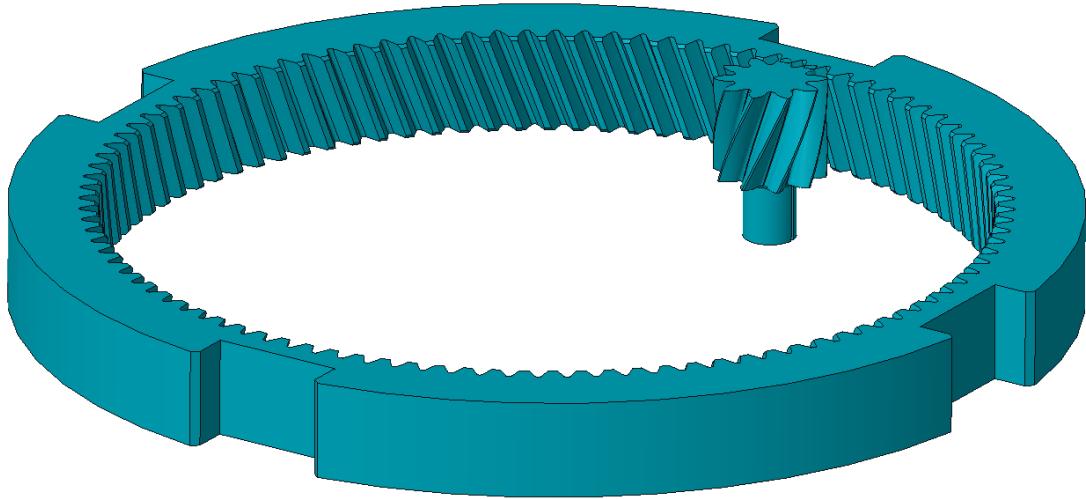


4.16. ábra. Függőleges tengely elrendezése

4.3.5. Fogaskerekek

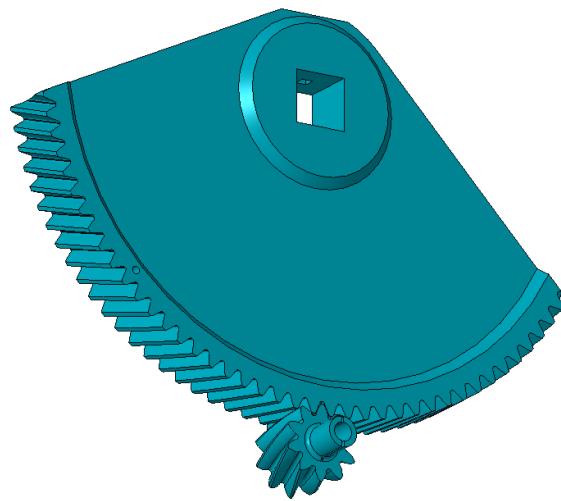
A prototípusban két fogaskerék kapcsolat kapott helyet. A függőleges tengely nagy fogaskereke belső fogazású, ezáltal el tudtam rejteni a motort a váz belsejében és kompaktabb lett a kialakítás, valamint valamivel stabilabb is. A kapcsolat

áttétele 10, és ferde fogazású, ezáltal a futás csendesebb és egyenletesebb lett. A kapcsolat illusztrációja a 4.17. ábrán látható.



4.17. ábra. Függőleges tengely fogaskerekek

A vízszintes tengely nagy fogaskereke a gearbox ház tengelyére lett erősítve, és csak részlegesen lett kinyomtatva. Ennek egyszerű oka, hogy más geometria is blokkolja a mozgást ezekben a tartományokban, illetve a Pan-Tilt kinematika mozgása redundáns lenne, ha teljesen körbe tudna forogni. A kapcsolat a 4.17. ábrán látható.



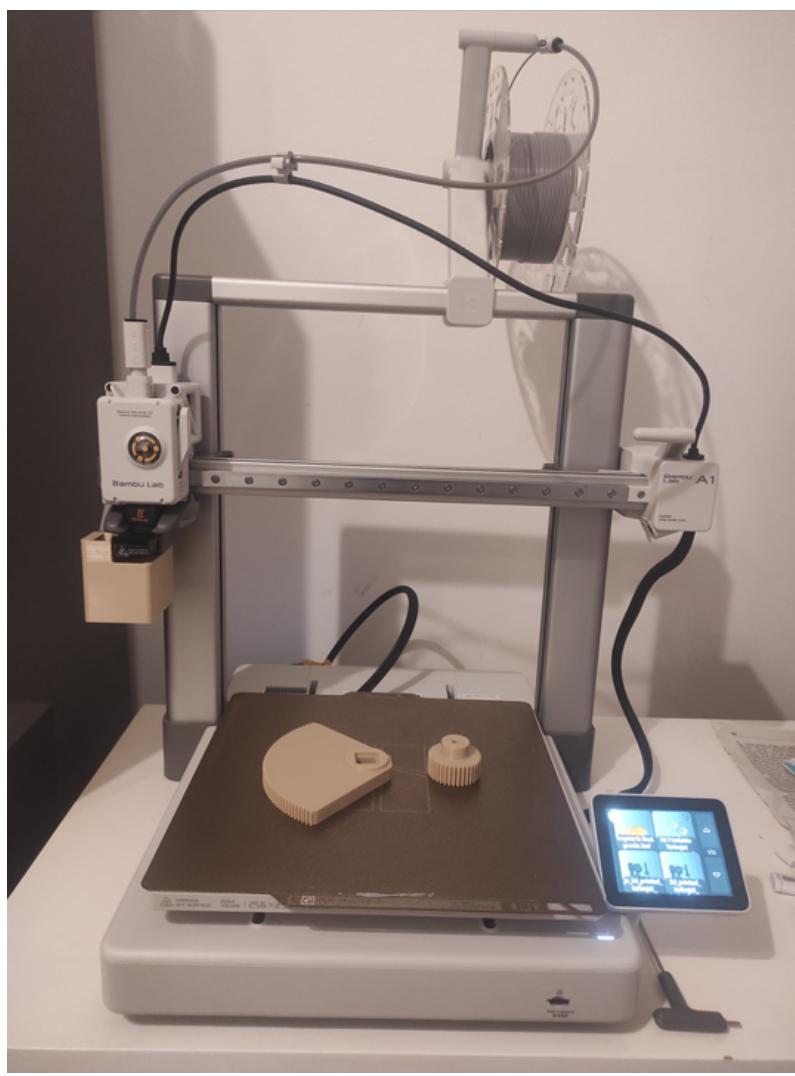
4.18. ábra. Vízszintes tengely fogaskerekek

Mind a két fogaskerékpár kisebb tagja közvetlenül a motor tengelyére lett rögzítve, egy acél rögzítőgyűrűvel. Ennek kialakítása a 4.16. ábra bal oldalán látható.

4.4. Gyártás és összeszerelés

4.4.1. Gyártás

Minden egyedi alkatrészt, amit a projekthez kellett gyártanom, 3D nyomtatással készítettem el. Ehhez egy **Bambu Lab A1** [14] típusú hobbinyomtatót használtam, amely a 4.19. ábrán látható. Mint a legtöbb ilyen árkategóriában lévő gép, ez is *FDM* technológiát használ. Az *FDM* lényege, hogy a nyomtató egy előre megadott terv alapján vékony műanyagszálat (filamentet) olvaszt meg egy forró extruderfejben (hotend), amelyet pontosan irányítanak a nyomtató XYZ tengelyei mentén. A műanyag szál olvasztott állapotban kerül a nyomtatóágyra, ahol gyorsan megszilárdul, így rétegenként építi fel az adott objektumot.

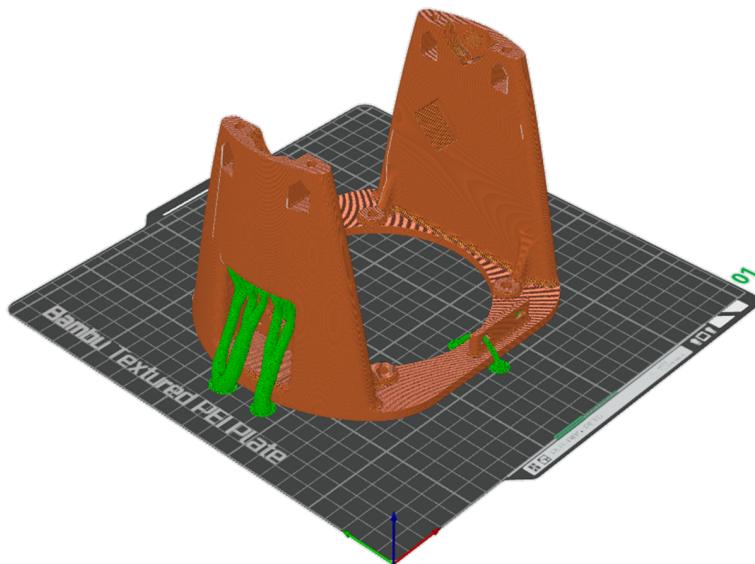


4.19. ábra. Bambu Lab A1

A használt nyomtató bizonyos tulajdonságai meghatározták a tervezett alkatrések tulajdonságait:

- **Nyomtatási térfogat:** $256 \times 256 \times 256$ mm, amely elég nagy volt a legtöbb általam tervezett elem számára.
- **Nyomtatási pontosság:** A nyomtató rétegvastagsága 0,1–0,4 mm között állítható. Főleg a nyomtatott fogaskereknél volt fontos a megfelelő pontoság, de kielégítő eredmény született.
- **Használható filamentek:** A nyomtató különféle anyagokkal kompatibilis, beleérte a PLA-t, ABS-t, PETG-t és TPU-t. Én a PLA mellett döntöttem, ugyanis nem szükséges a pl. ABS által nyújtott minőség, ellenben a költséghatékonyság igen.

A modelleket **PTC Creo** szoftverben készítettem. Ez egy magas szintű CAD tervező program, amely képes a modelleket exportálni STEP és STL fájlként is. Az STL fájlokat ezután a Bambu Studio slicer programba betölve tudtam a nyomtató által követhető G-kódra konvertálni. Itt kellett megtervezni a támasztást (4.20. ábra), illetve megadnia a nyomtatási paramétereiket, amelyek a függelék F.1. fejezetében láthatóak. Ezután már csak meg kellett várni, amíg a gép elkészíti az adott alkatrészt, ami a legnagyobb elem esetén kb. 10 óra volt, a teljes prototípus kb. 50 óra volt.



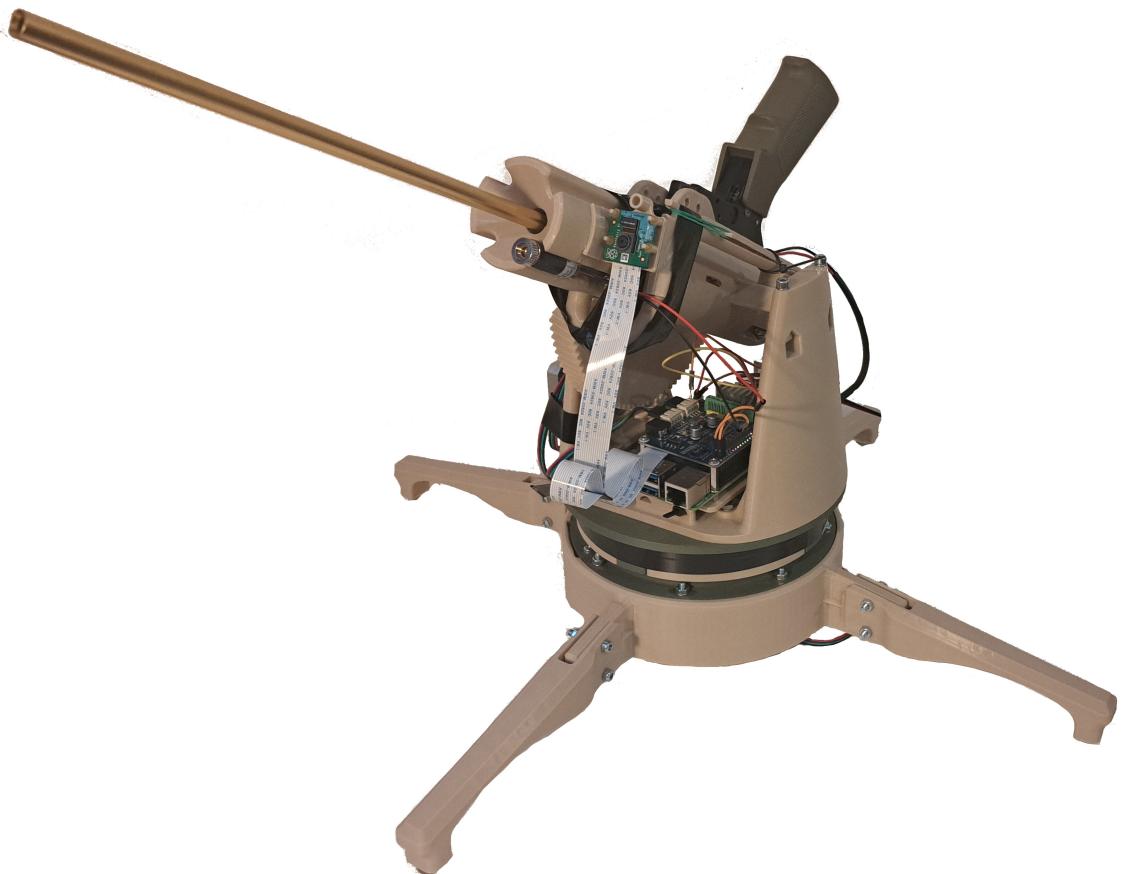
4.20. ábra. Bambu Studio Slicer

4.4.2. Összeszerelés

Ebben a fejezetben leírom az összeszerelés menetét. Először az elektronikát tartó konzollal kezdtem.

1. Ráhúztam a motorok tengelyére a kisfogaskerekeket.
2. Rácsavaroztam a távtartókat a Raspberry Pi-ra, a Stepper Motor HAT-ra és a DCDC konverterre, majd minden hármat rácsavaroztam a 3D nyomtatott konzolra. Félretettem az összeállítást.
3. Összeraktam a torony elemeit, a gearboxot, a tartó elemét és a két végén lévő elemeket. Összecsavaroztam a tervezett furatokon, majd ezt az összeállítást is félretettem.
4. Összeállítottam a prototípus 4 lábat és az ezeket tartó hengert. A lábak tal-pára gumi elemeket ragasztottam, hogy minél stabilabb legyen a rendszer. Félretettem száradni.
5. Ezután összecsavaroztam a nagy csapágyhoz csatlakozó alkatrészeket, illetve a DT-1000 elemhez rögzítettem az egyik motort. A DT-2000 elemhez ragasztottam a hozzá tartozó fogaskereket.
6. A prototípus talpát összecsavaroztam az előző pontban lévő elemmel úgy, hogy a keretet is tartsák a csavarok.
7. A kerethez rögzítettem a megfelelő motort, beragasztottam a relét és a vég-álláskapcsolókat is. Kábelkötözővel rögzítettem az elektronikai elemek konzolját a kerethez.
8. A korábban összerakott gearbox összeállítás tengelyeire illesztettem az oda tartozó fogaskereket és a csapágyakat. Így ráhelyeztem a keretre, és a csapágházak felső felét hozzácsavaroztam a keretre.
9. Összeillesztettem a kamera konzolt, és felragasztottam a helyére. Ugyanígy tettem a tárral is. Beillesztettem a lézer modult a helyére.
10. Összekötöttem a kábeleket, csatlakozókat.
11. Csatlakoztattam a tápokat és az ethernet-kábelt.

Az elkészült eszközt a 4.21. ábrán lehet látni.



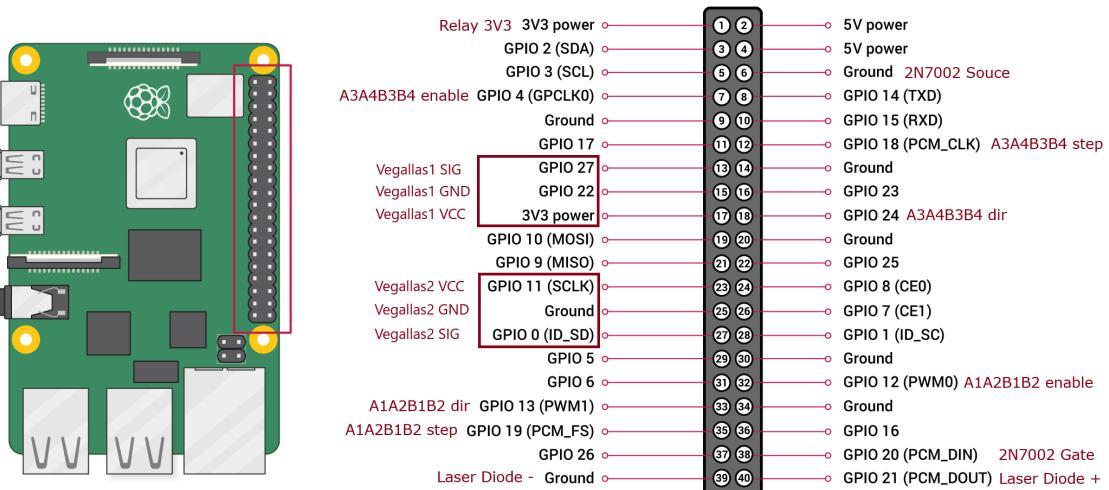
4.21. ábra. Összeállított prototípus

5. fejezet

Hardvertervezés

5.1. Áramköri tervezés

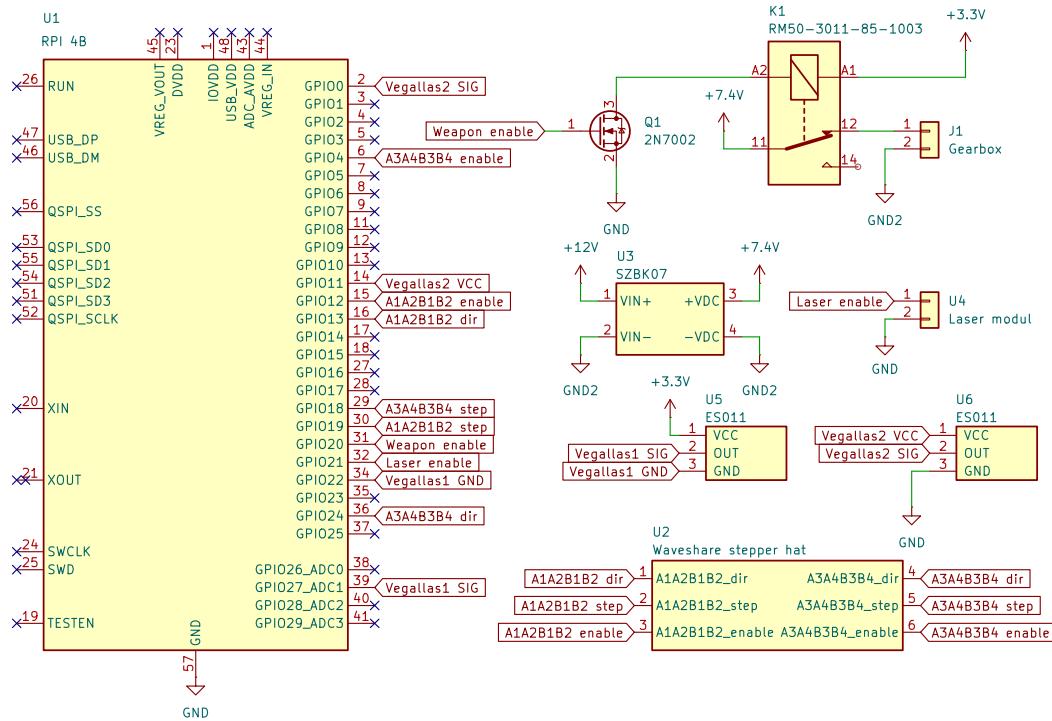
A prototípus áramkörét próbáltam minél egyszerűbben megvalósítani. Alapvetően két részből áll, a Raspberry PI központú vezérlőáramkörből, ami felelős a szenzorokért és a Raspberry PI-ért, illetve egy nagyobb teljesítményű áramkör egy külön tápról, ami a gearbox működtetéséért felelős.



5.1. ábra. A Raspberry PI lábkiosztása [15]

A prototípus áramköre a 5.2 ábrán látható. A kereskedelemben kapható *Stepper Motor HAT* nagyban leegyszerűsítette a tervezés folyamatát, a segítségével nem kellett megterveznem a motorvezérlők áramkörét, illetve ezeknek az áramellátását. A 5.1.ábrán látható, motorokra vonatkozó jelölések szerinti GPIO lábakat a *Stepper Motor HAT* ugyan lefoglalja, de a kártya tetején lévő tüskesor ugyanúgy elérhető marad, így oda kell figyelnem, hogy ezeket a lábakat ne használjam.

Ahogy az ábrán látható, a végálláskapcsolók is a GPIO tüskesorra lettek csatlakoztatva. Az egyiknél a földet helyettesíti GPIO22 láb, a másiknál a 3.3 V-t



5.2. ábra. A prototípus sémarajza

helyettesíti a GPIO11. Azért kellett ezt a megoldást alkalmaznom, mert a kábel csatlakozója egyben van, nem különálló jumper tüskék.

A lézer közvetlenül a GPIO21 lábról működik. Eleinte aggódtam, hogy nem tud elegendő áramot szolgáltatni a dióda működtetéséhez, de szerencsére mégis. Ellenkező esetben ki kellett volna egészítenem az áramkört egy tranzisztorral.

A relé kapcsolásához viszont nem volt elegendő a GPIO láb által szolgáltatott áram, így azt egy tranzisztorral vezérlem. Amennyiben logikai magas jelet kap, a relé zárja a gearbox áramkörét, és az elkezd lőni.

Az eszköz sémarajza az alábbi ábrán látható:

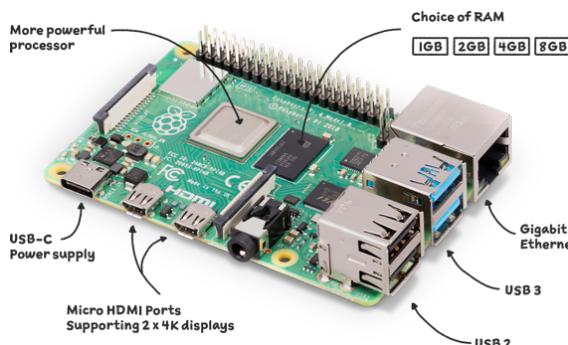
5.2. Elektronikai alkatrészek

Mikrovezérlő

A mikrokontroller kiválasztásánál több opciót vizsgáltam, többek között az Arduino, az STM-32 és a *Raspberry PI* modellek. A követelmények a következők voltak:

- Legyen könnyen beszerezhető, mind anyagilag, mind elérhetőség szerint
- Megfelelő teljesítmény gépi látás algoritmusokhoz
- Elegendő számú GPIO kimenet
- Python programozási nyelve támogatása
- Ethernet port

Ahogy összehasonlítottam a különböző opciókat, körvonalazódott, hogy a *Raspberry PI* [15] lesz a megfelelő megoldás. Az eszköz az 5.3. ábrán látható. Először is egy gyakorlati szempont szerint, mégpedig hogy egy Raspberry PI 4B modell eredetileg is volt a tulajdonban 4GB rammal. A gyártó hivatalos fórumán több felhasználó szerint ez már elegendő összetettebb gépi látás algoritmusok futtatására is. [16]



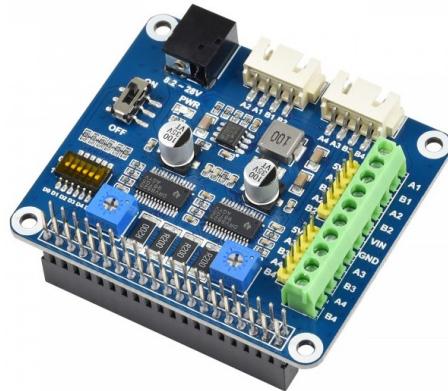
5.3. ábra. Raspberry 4 model B [15]

A kamera illesztése szintén különösen fontos a végtermék működése szempontjából, és ezen a területen magasan kiemelkedik a Raspberry a többi mikrovezérlő közül, mivel a gyártó magán a termékcsaládon belül ajánl több jó minőségű, könnyen beszerezhető kamera modult, amik illesztése már kiforrott az összes Raspberry-hez.

Ezentúl a *Raspberry PI* erősebb, mint a fentebb említett mikrokontrollerek. Linux rendszert futtat, és lehet rajta Python nyelven programozni, ami gépi látás, automatizálás projekteknél hatalmas előny. Számos ki- és bemenete van, ráadásul támogatja a *Bluetooth*, *Wi-Fi*, és gigabites *Ethernet* kapcsolatokat is, nem beszélve a rengeteg bővítő "kabátról", amiket lehet kapni hozzá. Ezek a termék továbbfejlesztését nagyban könnyítik, ráadásul kevesebb munkát kell a nyomtatott áramkör tervezésébe tenni.

Stepper Motor HAT [17]

A NEMA-17 léptetőmotorok vezérlésére több megoldás is létezik, elterjedtek a A4988, TMC5160T és DRV8825 típusú vezérlők. Ezeket azonban vagy külön modulként lehet megvásárolni, vagy pedig SMD alkatrészként, ami mellé mindenkorban kell tervezni kiegészítő áramkört.



5.4. ábra. Waveshare Stepper HAT [17]

Azonban a *Waveshare* cég gyárt egy Raspberry Pi-kompatibilis modult, ami számomra rengeteg segítséget nyújtott. Ez egy külön áramkör, amit a Raspberry GPIO tüskesorára lehet illeszteni. Képes két DRV8825 vezérlővel egyszerre kettő léptetőmotort vezérelni. A kártyán lévő kapcsolókkal könnyen lehet állítani külön motoronként a microsteppelést is, amennyiben szükség van rá egészen 1/32-ig. Szoftveresen is lehet állítani a microsteppelést, azonban ehhez forrasztani kellene a kártyára plusz ellenállásokat. A potméterekekkel motoronként tudjuk állítani a maximális felvehető áramot, maximum 2.5 A-ig. A kártyán helyet kapott egy standard 5.5 mm-es csatlakozó is, amivel 8.2 V és 28 V közötti feszültséggel lehet táplálni az áramkört. Egy belső regulator chip segítségével a Raspberry PI-t is el lehet látni árammal, így egy tápegységgel tudom működtetni a teljes áramkört. A modulon több lehetőség is van csatlakoztatni a motorokat, akár sorkapuccsal, akár a léptetőmotorok szabvány csatlakozójával. Ezentúl lehetővé teszi a Raspberry PI GPIO lábainak elérését, csupán arra kell odafigyelni, melyikeket használja. Az eszkösről a 5.4. ábrán látható illusztráció. Az eszköz el lett látva különböző biztonsági funkciókkal, pl. túláram védelemmel, magas hőmérséklet elleni védelemmel, illetve alulfeszültség lockout-tal.

A gyártó szintén elérhetővé tett illesztőszoftvereket több eszközre, több motortípusra, amit a későbbiekben én is tudtam használni.

Táp

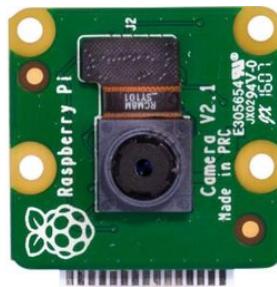
A prototípust két tápegységről működtettem. Az egyik a Raspberry PI, a léptetőmotorok és a szenzorok áramellátásáért, ez egy standard 5.5 mm-es csatlakozóval ellátott laptop töltő. 14.5 V tápfeszültséget és maximum 5 A áramot képes biztosítani, amely bőven elegendő a vezérlőelektronika ellátására. A másik táp egy HP ps-3701-1 12 V-os redundáns szervertáp 725 W teljesítménnyel.

DCDC konverter

A gearbox gyárilag 7.4 V-ról tud működni, tehát a kapott 12V-ot át kellett alakítani. Erre egy **SPX-20A** típusú DC-DC step-down konverted modult alkalmaztam. Ennek a maximum kimeneti árama 20A, maximum teljesítménye 300W. Elméletileg a gearbox felvett árama tüskeszerűen fellőhet 15A fölé, ami fölött már ventilátoros hűtés javasolt, de mivel ez nem lenne tartós, így én úgy döntöttem, ezt kihagyom. A kimeneti feszültséget és az áramkorlátot lehet állítani a kártyán található potméterekkel, de gyakorlatban csak a feszültséget állítottam. Minél nagyobb feszültséggel hajtom a gearboxot, annál gyorsabban pörög a motor, tehát annál nagyobb gyakorisággal lövi ki a golyókat. A gearbox és a táp kábeleit a kártyán található csavaros terminállal csatlakoztattam, maga a modul pedig a kerten kapott helyet, a Raspberry PI konzolján.

Kamera

A rendszer által használt kamera egy *Raspberry Pi Camera Module 2* volt.[18] Ez a modul 8 MP-es *Sony IMX219* szenzorral rendelkezik, amivel 3266 x 2450 felbontású állóképet tud készíteni. Képes Full HD videót felvenni 30 fps-el, HD-t 60 fps-el, vagy 640x480 felbontást 90 fps-el. A modul a 5.5. ábrán látható.



5.5. ábra. Raspberry Pi Camera Module 2 [18]

A kamera befoglaló mérete szerelési furatai és CSI portja megegyezik a többi Raspberry kamerával, így a továbbfejlesztés esetén könnyedén lehet őket cserálni.

Végálláskapcsoló

A prototípus bekapcsolásánál fontos, hogy beálljon egy kezdőpozícióba, és ahhoz képest tudjuk viszonyítani a mozgását. Ezt én a két tengelyen 1-1 végálláskapcsolóval oldottam meg. Ezek a modulok könnyen szerelhetők a 3D nyomtatott vázra, és az alapján adnak jelet, valami van-e az optokapu között. A Raspberry GPIO lábaira könnyedén lehet őket bekötni jumper kábelek segítségével.

6. fejezet

Szoftverfejlesztés

A prototípus szoftverét kettéválasztottam a két működési mód szerint, az egyik a manuális vezérlést valósítja meg, a másik az automatikusat. A kettő között lényeges különbségek vannak a prioritásokban. A kézi vezérlésnél a legfontosabb a valós idejű megjelenítés, és a minél gyorsabb reagálás a felhasználói parancsokra. Az automata működésnél a megjelenítés kevésbé fontos, viszont a képfelismerés és ez által a célpont követése kiemelkedő fontosságú.

6.1. Használt Python modulok, eszközök

OpenCV könyvtár [19]

Az **OpenCV** (Open Source Computer Vision Library) egy nyílt forráskódú könyvtár, amelyet eredetileg az Intel fejlesztett ki, és amelyet széles körben használnak a számítógépes látás (computer vision) és képfeldolgozás területén. Az OpenCV rendkívül sokoldalú eszközökkel biztosít a képekkel és videókkal kapcsolatos különféle feladatok megoldásához, amelyet elsősorban C++-ban fejlesztettek ki, de támogat Python, Java, és MATLAB interfésekkel is. A Python interfész különösen népszerű, mivel egyszerűsíti az OpenCV használatát gépi tanulási és mesterséges intelligencia-alkalmazásokban. A fő funkciói közé tartozik a képfeldolgozás, téglalap- és arcérzékelés, mozgáskövetés, valamint a 3D-s modellezés és képanalízis.

Az OpenCV architektúrája moduláris felépítésű, ami lehetővé teszi a különböző funkciók rugalmas használatát. A modulok között megtalálhatóak a képfeldolgozási, objektumfelismerési, gépi tanulási és 3D modellezési könyvtárak. A Pythonban a `cv2` könyvtárként importálható OpenCV API biztosítja a különböző funkciók egyszerű elérését, így akár néhány sor kóddal is gyors eredmény érhető el.

Többféle kéatformátumot támogat, például JPEG, PNG és BMP, és rendelkezik valós idejű video- és képadatok betöltésére szolgáló funkciókkal is, például a `VideoCapture` objektummal. Ez utóbbi lehetővé teszi, hogy közvetlenül kamerák-ból vagy videofájlok ból dolgozzunk.

Az **OpenCV** használata igen elterjedt az iparban és a kutatásban is, többek között autonóm járművek, biztonsági rendszerek, gyártósori ellenőrző rendszerek, orvosi képalkotás és mobilalkalmazások terén. A könyvtár rugalmassága és széleskörű támogatása lehetővé teszi, hogy számos programozási és mérnöki területen is alkalmazható legyen, ahol képfeldolgozásra és gépi látásra van szükség.

Socket modul [20]

Python `socket` modulja alacsony szintű hozzáférést biztosít a hálózati kommunikációhoz, lehetővé téve különböző típusú hálózati kapcsolatok létrehozását, beleértve az általam használt TCP-t és az UDP-t.

A **TCP** kapcsolat-orientált protokoll, amely biztosítja az adatok érkezési sorrendjét és helyességét. A `socket.SOCK_STREAM` típust használjuk, ha TCP kapcsolatot szeretnénk létrehozni. A TCP kapcsolatok esetében a `connect()` függvény a kapcsolat kezdeti lépése, és a kliens-szerver kommunikáció folyamatosan fennmarad.

Az **UDP** kapcsolatnélküli protokoll, amely nem garantálja az adatok sorrendjét és integritását, viszont gyorsabb, mivel nincs szükség kapcsolatra. Az UDP kapcsolathoz a `socket.SOCK_DGRAM` típust használjuk, így a kliens és a szerver bármikor küldhet és fogadhat adatokat. Ideális valós idejű alkalmazásokhoz, ahol nem kritikus minden csomag megérkezése (pl. video streaming).

Mindkét típusnál a `send()`, `sendto()`, `recv()`, és `recvfrom()` függvények használhatók az adatok küldésére és fogadására.

Multiprocessing modul [21]

A **multiprocessing** modul lehetővé teszi a Python számára, hogy párhuzamosan több folyamatot indítson el, ami segít a CPU-erőforrások jobb kihasználásában. A modul az operációs rendszer által kezelt különálló folyamatokat hoz létre, így azok külön memóriaterülettel rendelkeznek, és jobban teljesítenek többmagos processzorokon, mint a *threading* modul.

A `Process` osztály segítségével indíthatunk el új folyamatokat, amelyek egymástól függetlenül végrehajtanak egy adott funkciót. A modul támogatja az eseménykezelést, kommunikációs csatornákat (pl. `Queue`, `Pipe`) és szinkronizálást (pl. `Event`, `Lock`), amelyek segítségével a folyamatok egymással adatokat cserélhetnek és szinkronizálhatják működésüket.

Pygame modul [22]

A **pygame** egy multimédiás modul, amely elsősorban 2D játékok fejlesztésére alkalmas, de egyéb alkalmazásokkor is lehetőséget ad bemenetek kezeléséhez.

A `pygame.event.get()` függvény segítségével olvashatjuk le az egér- és billentyűzet-eseményeket, így például kattintások, gombnyomások és egérmozgások lekérdezése egyszerű. Az interaktív elemek kirajzolása és mozgatása

egyszerű a `pygame.display.update()` és `pygame.draw` függvényekkel, amelyek lehetővé teszik grafikus objektumok megjelenítését a képernyőn. Magában foglalja a számítógépes grafikákat, a hang- és programkönyvtárakat, amiket a Python programozási nyelvre fejlesztettek ki.

Pickle modul [23]

A `pickle` modul sorosítást és deserializációt tesz lehetővé, azaz a Python-objektumok bináris formátumba alakítását és visszaalakítását.

A `pickle.dumps()` segítségével a Python-objektumokat bináris formátumba alakíthatjuk, amely lehetővé teszi azok egyszerű tárolását vagy átvitelét, például hálózaton keresztül. A `pickle.loads()` a bináris formátumú adatokat visszaalakítja eredeti Python-objektummá. A pickle használata előnyös lehet akkor, ha komplex adatszerkezeteket szeretnénk fájlban tárolni vagy hálózaton keresztül továbbítani.

6.2. Gépi látás

A célpontfelismerés módszere volt az automata működés legfontosabb eleme. Elsősorban az OpenCV könyvtár különböző lehetőségeit vizsgáltam, ugyanis ez a legjobban kiforrott modul, amely könnyedén használható python programokban.

6.2.1. Template matching

A **Template Matching** [24] (sablonillesztés) egy egyszerű, de hatékony módszer, amelyet statikus képekben keresett minták (sablonok) azonosítására használnak. Ez a módszer összehasonlít egy kisebb, előre meghatározott képrészletet (a sablont) a nagyobb képen található területekkel, és megkeresi a legjobban illeszkedő pozíciót.

Működés: Az OpenCV `cv2.matchTemplate()` függvényével kivitelezhető, amely végigfuttatja a sablont a célképen, és minden pixelpozíciónál egy hasonlósági pontszámot generál. A legmagasabb értékű pontszám mutatja az illeszkedés helyét.

Előnyei: Gyors és könnyen implementálható, különösen ismert méretű és orientációjú objektumok azonosításához.

Korlátozások: Gyengén teljesít forgatott, méretarányosan eltérő vagy eltorzult objektumoknál, valamint változó fényviszonyok mellett.

6.2.2. Feature Matching

A *Feature Matching* [25] (jellemzőpont-illesztés) egy kifinomultabb módszer a képjellemzők összehasonlítására és egyeztetésére. A módszer a két képben található pontokat (feature-ket) párosítja, ezáltal lehetővé teszi forgatásokkal, elforgatásokkal és méretarányokkal szembeni robusztus egyeztetést.

Működés: Először meghatározzuk a képen lévő fontos jellemzőpontokat (például élek, sarkok), amelyeket különböző detektorok (például SIFT, SURF vagy ORB) segítségével találhatunk meg. Ezután az `cv2.BFMatcher()` (Brute Force Matcher) vagy a `cv2.FlannBasedMatcher()` függvényel azonosítjuk a jellemzőpontokat.

Előnyei: Robusztus a forgatásokkal, skálázással és elmozdulásokkal szemben, így alkalmas tárgyak nyomon követésére és objektumok felismerésére különböző nézetekből.

Korlátozások: A nagy számítási igényű algoritmusok miatt lassabb lehet, különösen nagy méretű képeken.

6.2.3. Target Tracking

A *Target Tracking* [26] (célkövetés) feladata egy mozgó objektum követése egy videófolyamon vagy valós idejű képforráson keresztül. Az OpenCV több nyomon követési algoritmust is biztosít, amelyek segítenek abban, hogy a kiválasztott objektumot stabilan követni tudjuk, még akkor is, ha változik a pozíciója vagy a környezeti fény.

Algoritmusok:

- Mean Shift: A `cv2.meanShift()` függvényel elérhető algoritmus egy adott objektum körüli eloszlás csúcsértékeit keresi, és folyamatosan frissíti a helyét.
- CamShift (Continuously Adaptive Mean Shift): A `cv2.CamShift()` algoritmus a Mean Shift továbbfejlesztett változata, amely figyelembe veszi az objektum méretének és alakjának változásait.
- Modern nyomkövetési algoritmusok: Az OpenCV 3.x-es és újabb verziói lehetővé teszik algoritmusok használatát, mint a KLT (Kanade-Lucas-Tomasi) nyomkövető, a CSRT és a MedianFlow, amelyek kifejezetten valós idejű és pontos követést biztosítanak.

Előnyei: Alkalmas valós idejű alkalmazásokhoz, mint például a videó megfigyerés vagy az autonóm rendszerek, ahol a cél folyamatosan mozgásban van.

Korlátozások: Bizonyos algoritmusok hajlamosak elveszteni a célpontot gyors mozgásoknál vagy hirtelen pozícióváltásoknál.

6.2.4. Haar cascade

A *Haar Cascade* [27] módszer az egyik legelterjedtebb technika az objektumfelismerésre, különösen az arcfelismerésben. A gépi tanuláson alapuló módszer képes egy képben vagy videófolyamon objektumokat megtalálni azáltal, hogy mintákat ismer fel az előképzett adatok alapján.

Működés: A Haar-cascade egy többlépcsős osztályozó, amelyet sok pozitív és negatív mintából tanítanak be. A `cv2.CascadeClassifier()` osztály segítségével előre tanított modelleket, például arcokat, szemeket vagy különböző objektumokat kereshetünk. módszer előnye, hogy nagyon hatékonyan képes felismerni az objektumokat. A gépi tanulás során a kimeneti modell jellemzőit úgy optimalizálják, hogy gyorsan tudja felismerni a keresett objektumokat még zajos vagy eltérő körülmények között is.

Előnyei: A Haar-cascade módszer jól használható olyan alkalmazásokhoz, ahol előre meghatározott objektumokat (például arcokat) kell azonosítani. Nagy előnye, hogy valós időben is képes futni.

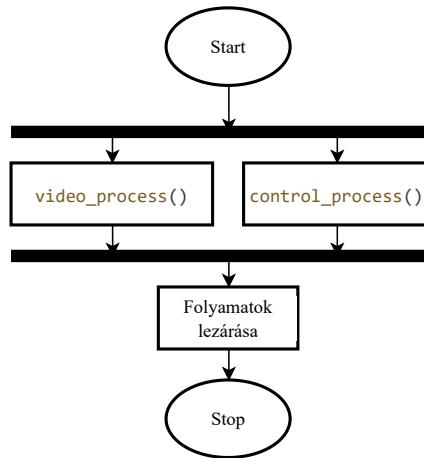
Korlátozások: Hajlamos a pontatlan felismerésre, ha az objektumok perspektívája, megvilágítása vagy pozíciója jelentősen eltér az előképzett mintáktól. Alternatív megoldásként mélytanulási módszerek (például konvolúciós neurális hálózatok) használhatók az ilyen problémák javítására.

6.3. A szoftver működése

A rendszer két alapvető részből áll: egy számítógépből (PC), amely felhasználói interfésként szolgál az operátor számára, és egy Raspberry Pi-ből, amely a fegyverrendszer vezérlését és képátvitelét végzi. A két eszközön futó program között két folyamat kommunikál, két különböző porton: egy UDP protokoll alapú adatátvitel a videó küldéséhez, illetve egy TCP alapú a vezérlőparancsok fogadásához.

6.3.1. A PC-n futó program működése

A PC-s kód `main()` függvénye két külön folyamatot indít: egy videó-fogadó folyamatot és egy vezérlő-küldő folyamatot, melyek egy multiprocessing-eseményjelző (`stop_event`) segítségével állnak meg szükség esetén. A két folyamat használ közös erőforrást, a `frame_queue`-t, amelybe a videó-fogadó betölti a Raspberry-ből kapott képkockákat, a Pygame pedig felhasználja azt a másik folyamatban. A felső szint folyamatábrája a 6.1. ábrán látható.



6.1. ábra. A PC-n futó program felső szintű folyamatábrája

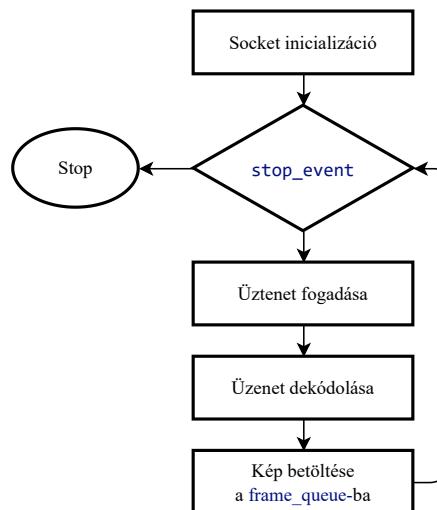
Videó-fogadó folyamat

A videó-fogadó folyamat (`video_receiver`) UDP-n keresztül kapja a képkockákat, amelyek a Raspberry Pi-ról érkeznek csomagokra bontva. A folyamat összegyűjt ezeket a csomagokat, majd a kép adatokat újraegyesíti és betölti a `frame_queue`-ba. Az UDP protokoll nagy sebességet tud elérni, de nem garantált a célba érés. A videó esetében azonban ez nem probléma, ha elveszik egy képkocka attól még értelmezhető marad a videó. A folyamat a következőképpen működik:

- A folyamat egy UDP socketet (`video_socket`) hoz létre a megadott porton.
- Egy ciklusban fogadja a képkockák csomagjait, minden egyes csomag egy fejlécet és a kép adatrészleteit tartalmazza.

- A fejléc alapján a folyamat azonosítja a csomagokhoz tartozó képkockát és azok sorrendjét. A bufferben gyűjtött csomagokat az összes részlet megérkezése után egyesíti.
- Az egyesített képkockát visszafejti (unpickling és JPEG dekódolás), majd a képre helyez egy célkeresztet a `cv2.line()` függvény alkalmazásával.
- A teljes képkockát behelyezi a `frame_queue` sorba, a `.put()` metódussal.

A folyamatábra a 6.2. ábrán látható.

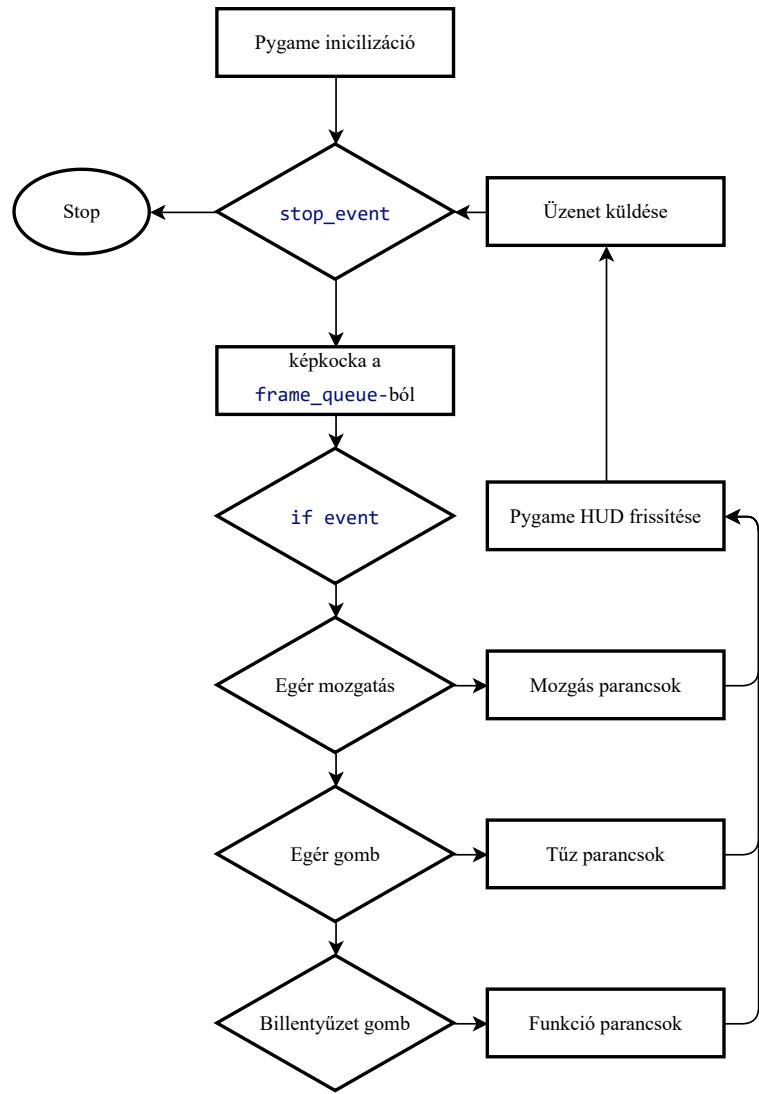


6.2. ábra. A `video_receiver` függvény folyamatábrája

Vezérlés-küldő folyamat

A vezérlés-küldő folyamat (`control_sender`) a felhasználó irányítási parancsait továbbítja a Raspberry Pi felé. A folyamat a Pygame könyvtár segítségével észleli az egér- és billentyűeseményeket, majd a megfelelő parancsokat TCP csomag formájában küldi a Raspberry Pi irányába. Ezentúl felelős a HUD megjelenítéséért, amiről a felhasználó le tudja olvasni a prototípus állapotát. A folyamat illusztrációja a 6.3. ábrán látható. A felhasználó számára fontos, hogy minden információt le tudjon egyből olvasni a fegyver működéséről, így ezeket a PyGame ablakon megjelenítettem (6.4. ábra). Itt látható:

- Használható parancsok és a hozzájuk rendelt gombok
- Milyen módban működik a rendszer
- Biztosítva van-e a rendszer
- Működik-e a lézer



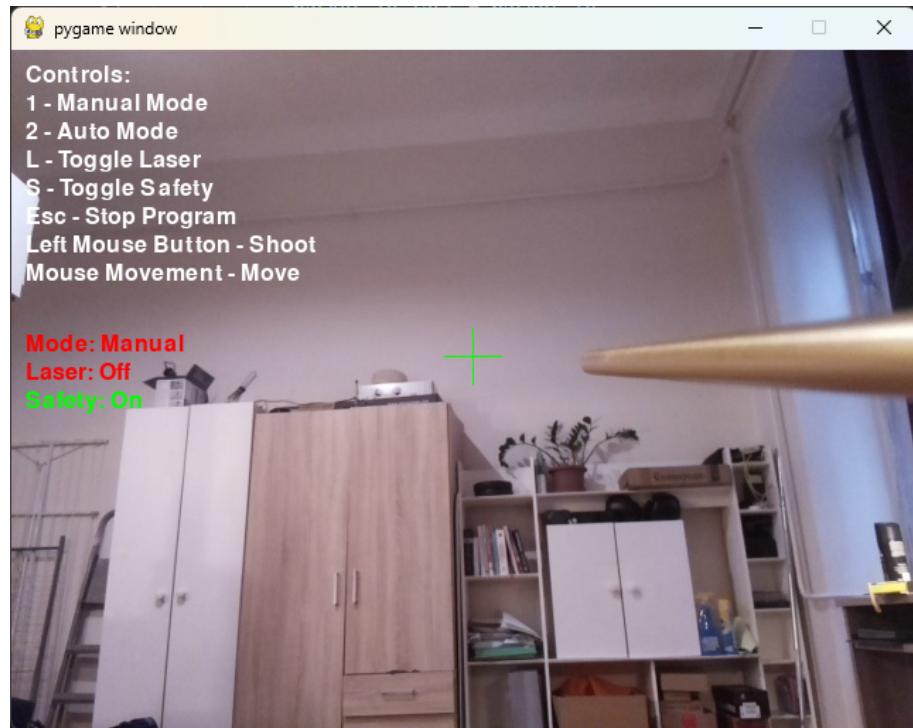
6.3. ábra. A `control_sender` függvény folyamatábrája

A PyGame ablak háttere pedig az éppen utolsó képkocka, amit a videó-fogadó folyamat a `frame_queue`-ba tett. A vizuális megjelenítés mellett a folyamat feladata a parancsok küldése, ezek a következők lehetnek:

- Az egér mozgása esetén `MOVE` parancs az X és Y elmozdulási adatokkal, így irányítva a fegyver mechanikai elmozdulását.
- A bal egérgomb lenyomásakor (`SHOOT_START`, a felengedéskor `SHOOT_STOP`) parancsot adhatunk ki, értelemszerűen a tüzelés megkezdésére és befejezésére.
- A billentyűzet gombjaival a lézert, (`LASERTOGGLE`) a biztosítást (`SAFETY`), illetve a működési módokat lehet állítani(`MANUALMODE` és `AUTOMODE`).

- Amikor a felhasználó az ESCAPE gombot megnyomja, a folyamat küld egy leállítási parancsot (STOP), majd befejezi működését.

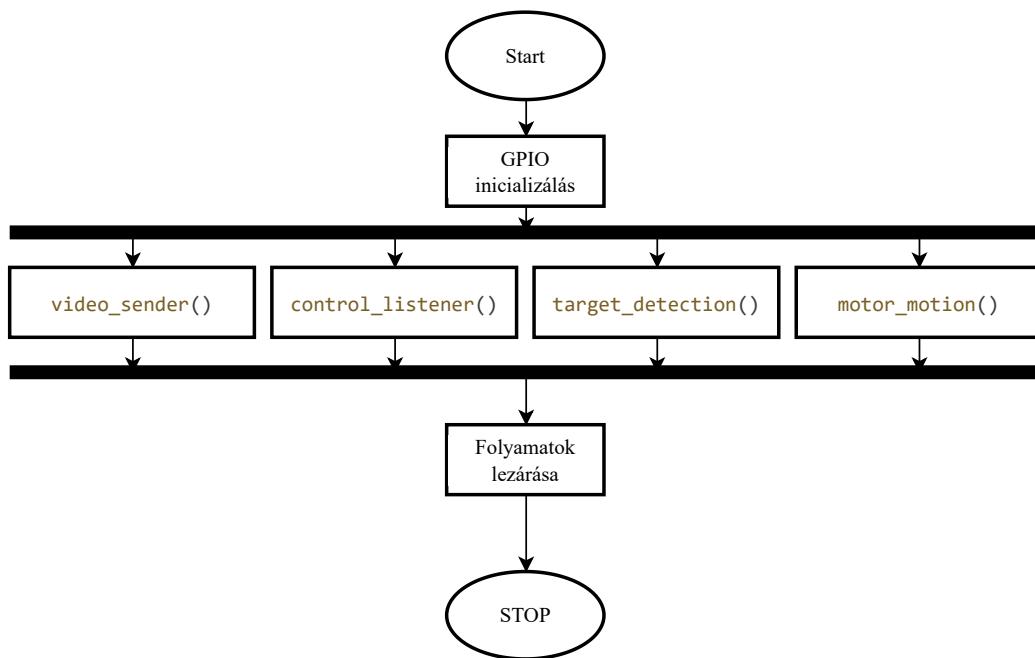
A program a működési módtól függetlenül működik, mindenkor ugyanazokat az üzeneteket küldi el. A kliens programtól függ, hogyan értelmezi azokat.



6.4. ábra. Pygame HUD

6.3.2. A Raspberry Pi-n futó program működése

A Raspberry Pi-n több folyamat is fut: Egy videó-küldő, egy vezérlés-fogadó, egy célpontfelismerő, illetve egy motormozgató. A közös erőforrások között szintén van egy `stop_event` eseményjelző, amely minden folyamat megállításáért felelős. Ezután van a `frame_queue` és a `pos_queue`, amelyek tartalmazzák a képkockákat és a felismert célpont pozíóját. A `main()` függvény illusztrációja látható a 6.5. ábrán.



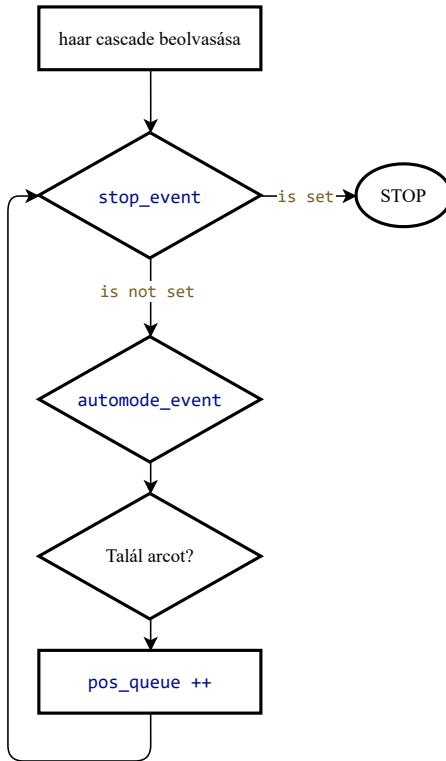
6.5. ábra. A Raspberry PI-n futó program felső szintű folyamatábrája

Célfelismerő folyamat

A `target_detection()` folyamat felelős az arcok felismeréséért, és a helyzetük meghatározásáért.

- A `cv2.CascadeClassifier()` függvénnyel betölti az előre betanított `haarcascade_frontalface_default.xml` modellt
- A `frame_queue.get()` metódussal kiveszi az utolsó képkockát a sorból
- A `cv2.detectMultiScale()` megtalálja az arcokat a képkockán
- Méret szerint sorba rendezi az arcokat, és kiszámolja a legnagyobb pozíóját
- Betölti a pozíciót a sorba a `pos_queue.put` metódussal

Ez a folyamat csak akkor fut, hogyha az `automode_event()` flag be van állítva, tehát az eszköz automatikusan működik. A folyamat illusztrációja a 6.6. ábrán látható



6.6. ábra. A `target_detection` függvény folyamatábrája

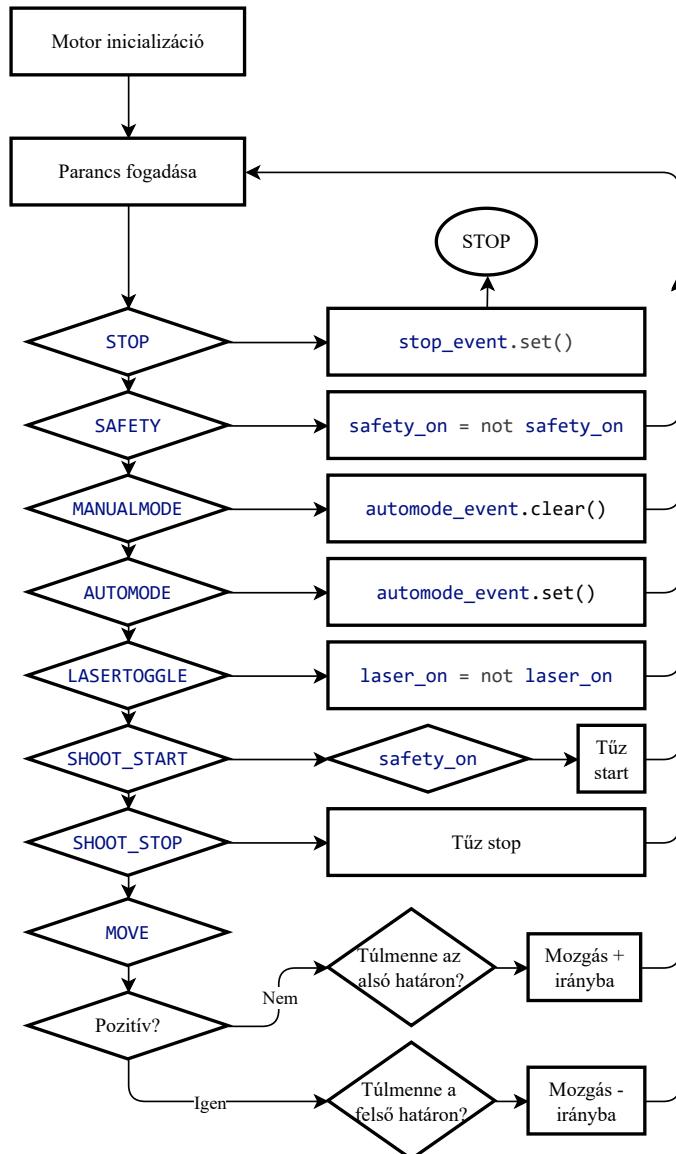
Vezérlés-fogadó folyamat

A `control_listener` folyamat először, a rendszer indításakor elvégzi a fegyver pozíciójának beállítását. A motorokat addig forgatja, amíg a végálláskapcsolók nem adnak jelet. Ezután pedig, miután tudjuk a pontos pozíció, visszaforgatja a fegyvert középre, és nullázza a `POSX` és `POSY` globális változókat. Ezután pedig TCP-n keresztül fogadja a PC-ről érkező irányítási parancsokat, és a megfelelő fizikai komponenseket működteti, majd a rendszer leállításakor újra középre állítja a fegyvert. A következő parancsokat tudja fogadni, attól függően, hogy melyik módban működik a prototípus.

- A parancs típusa alapján (`MOVE`, `SHOOT_START`, `SHOOT_STOP`, `LASERTOGGLE`, `STOP`) különböző műveleteket hajt végre.
- `MOVE` parancs esetén az X és Y elmozdulási értékek alapján irányítja a két DRV8825 motorvezérlő áramkört, így mozgatva a fegyvert az elvárt irányba.
- A lövésvezérlés (`SHOOT_START` és `SHOOT_STOP`) esetén aktiválja vagy deaktiválja a fegyver LED-jét.
- A lézerkapcsolás (`LASERTOGGLE`) a lézerdiódát ki- vagy bekapcsolja.

- Amennyiben `STOP` parancs érkezik, a folyamat leállítja a működését.

A függvény folyamatábrája a 6.7. ábrán látható.



6.7. ábra. A `control_listener` függvény folyamatábrája

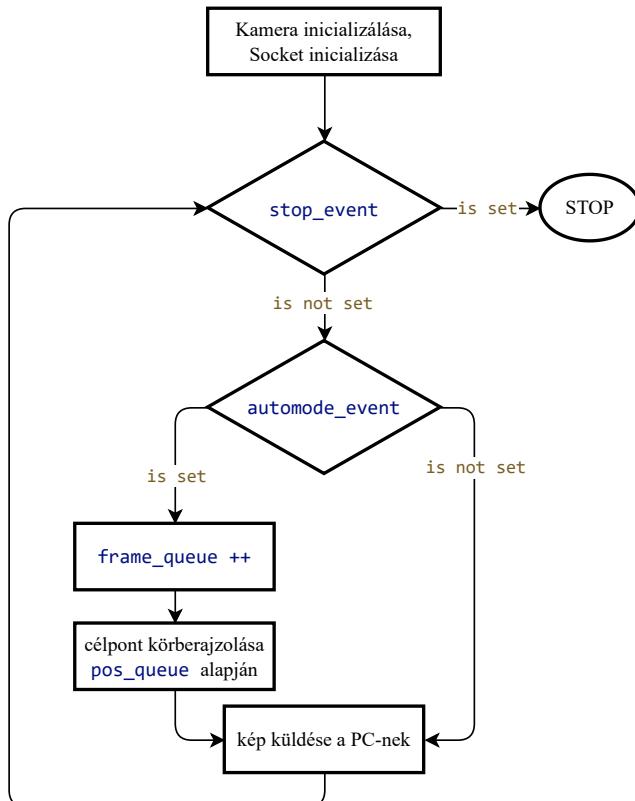
- `STOP` parancs esetén az `stop_event`-et beállítja, ezzel leállítva az összes folyamatot
- `AUTOMODE` parancs esetén az `automode_event`-et igazra állítja
- `MANUALMODE` parancs esetén az `automode_event`-et hamisra állítja

- `MOVE` parancs esetén az X és Y elmozdulási értékek alapján irányítja a két motorvezérlőt, így mozgatva a fegyvert a megfelelő irányba.
- A (`SHOOT_START` és `SHOOT_STOP`) esetén aktiválja vagy deaktiválja a fegyvert. Aktiválni csak akkor lehet, ha a `safety_on` hamis.
- A (`LASERTOGGLE`) a lézerdiódát ki- vagy bekapcsolja.
- Amennyiben `SAFETY` parancs érkezik, beállítja a `safety_on` flag-et.

A végtelen ciklus ellenőrzi a `stop_event` flag-et, és ha igaz lesz, akkor kilép belőle. Ekkor a `TurnStep()` fügvénnyel addig lép, amíg középen nem lesz a fegyver, valamint a lézert is lekapcsolja.

Videó-küldő folyamat

A videó-küldő folyamat (`video_sender`) a Raspberry Pi kamerájából érkező képet szerzi meg és osztja meg a PC-vel UDP kapcsolaton keresztül. A folyamat illusztrációja a 6.8. ábrán látható.



6.8. ábra. A `video_sender` függvény folyamatábrája

- A kamerakép feldolgozása során a kód rögzíti az egyes képkockákat, amelyeket JPEG formátumba kódol és pickle segítségével tömörít.

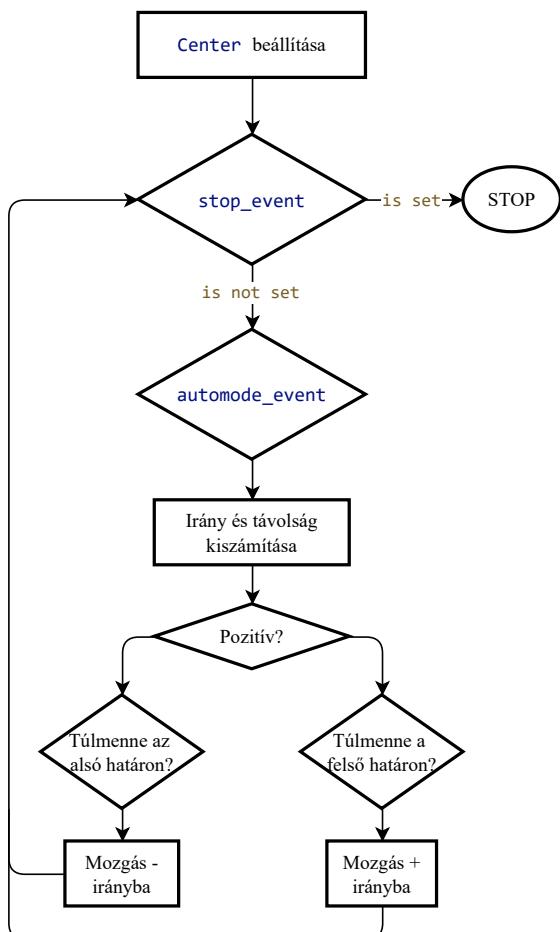
- Amennyiben az `automode_event` flag igaz, akkor beteszí a legutolsó képkockát a `frame_queue`-ba, és a `pos_queue` legutolsó elemei alapján egy téglalapot illeszt a képkockára, ezzel jelezve a talált arcot.
- Az adatok nagy mennyisége figyelembe véve a képkockákat kisebb csomagokra bontja, amelyekhez megfelelő fejlécek tartoznak.
- A fejlécekben megtalálható a csomag azonosítója, amely segít a PC-n futó videó-fogadó folyamatnak a csomagok megfelelő sorrendbe állításában és összerakásában.
- A csomagokat a megadott IP-címre és portra küldi, és minden új képkockához új azonosítót rendel.

Motormozgató folyamat

A `motor_motion()` folyamat azért felelős, hogy automata működés során mozgassa a motorokat, a kamera képe alapján. Azért gondoltam, hogy jobb így külön kezelni, mert az automata és kézi vezérlés mozgásának nagyon különböző a logikája. A folyamat működése a következőképpen működik:

- Kiolvassa a legutolsó pozíciót a `pos_queue`-ből, majd a pozíóból és a középpont különbségéből kiszámítja a távolságot.
- Amennyiben a fegyver pozíciója a mozgásterületen belül van, a megfelelő irányba mozdul a távolsággal arányosan.

A motor csak akkor mozog, ha egy bizonyos értéknél nagyobb a távolság, így kerültem el, hogy folyamatosan rátengelyezzen az eszköz. Ez a folyamat is csak akkor fut, hogyha a `automode_event` igaz. A folyamat illusztrációja a 6.9. ábrán látható.



6.9. ábra. A `motor_motion` függvény folyamatábrája

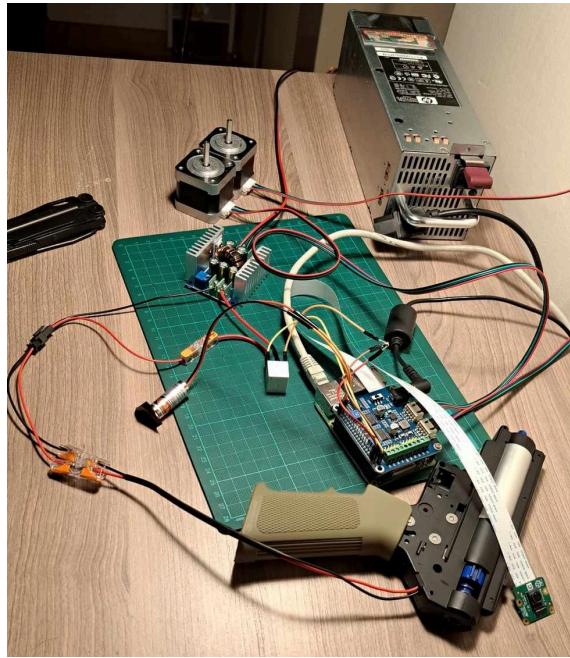
7. fejezet

Tesztelés

A prototípus tesztelése több lépcsőben zajlott le, minden alkalommal több egységet hozzáadva. Az első a **hardver** tesztje volt, ahol a *motorok illesztése*, a *gearbox*, a *kamera* és a *szenzorok* megfelelő működését vizsgáltam, illetve a *kommunikációt* a Raspberry PI és a PC között. Miután a teszt alapján beállítottam a kamera optimális paramétereit és működésének algoritmusát, a következő teszt a különböző **cél felismerő algoritmusok** összevetése volt. Itt először egy statikus állványra került a kamera, és a képfelismerés sebességét és megbízhatóságát vizsgáltam. Miután kiválasztódott az optimális megoldás és elkészültek a 3D nyomtatott alkatrészek, kipróbáltam a cél felismerést a motorok mozgásával együtt is. Végezetül pedig **éles helyzetben** próbáltam ki a prototípust, ami tulajdonképpen céltáblára lövést jelentett. Ekkor vizsgáltam a szerkezet stabilitását, a cél felismerést nagyobb távolságokban, a tüzelési mechanizmus megbízhatóságát és pontosságát, illetve a *biztonsági funkciókat*.

7.1. Elektronikai alkatrészek tesztje

Ezen teszt során csatlakoztattam a kamerát a Raspberry PI megfelelő csatlakozójához, a NEMA-17 léptetőmotorokat pedig a Stepper Motor HAT csatlakozóihoz. Ezután a GPIO tüskesorhoz csatlakoztattam a lézer diódát, a relé megfelelő lábait, és a végálláskapcsolókat. A gearbox kábeleit összekötöttem a tápegységgel, legvégül pedig a Raspberry PI-t az ethernet porton keresztül a PC-hez csatlakoztattam, majd áram alá helyeztem. Először a Stepper Motor HAT gyártója által szolgáltatott illesztőprogramot próbáltam ki, átkonfigurálva a motor típusának és a microsteppelésnek megfelelően. Ezután a kamera gyári driver-ét teszteltem az *OpenCV* könyvtár használatával. Végül pedig konfiguráltam a GPIO lábakat, és teszteltem a lézer diódát, a végálláskapcsolókat és a relét. Ezzel együtt teszteltem a kezdetleges szoftvert és az eszközök közötti kommunikációt, ez magába foglalta a *Socket* és *Pygame* modulok különböző lehetőségeinek összevetését. A teszt összeállítása a 7.1. ábrán látható. A teszt nem egy alkalommal zajlott le, inkább egy húzódó folyamatként, aminek során megtaláltam az optimális beállításokat a szoftverben.



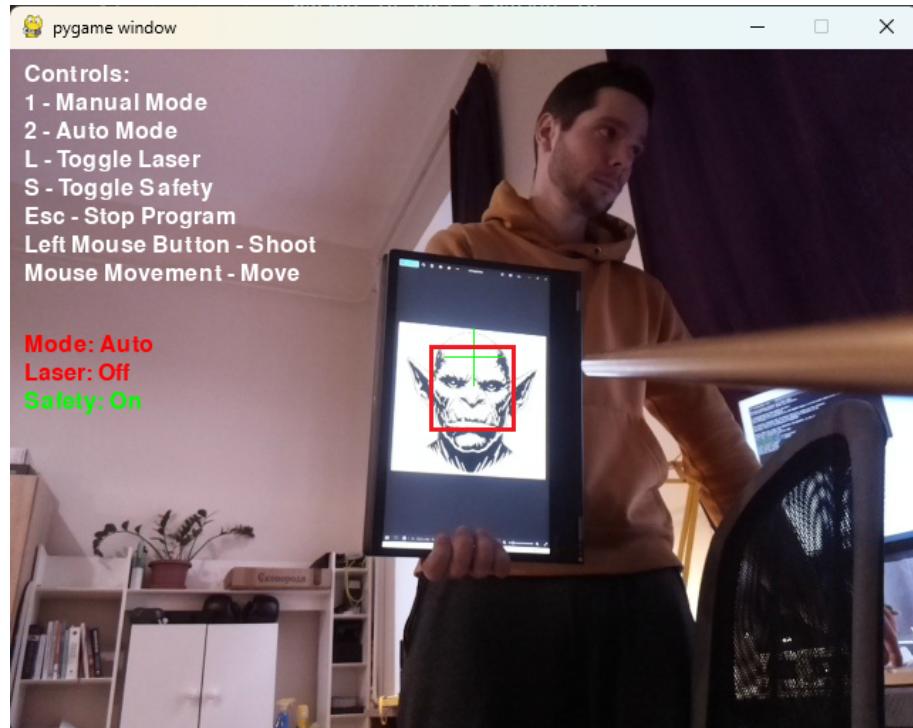
7.1. ábra. Elektronikai alkatrészek teszt összeállítása

7.2. Képfelismerő algoritmusok tesztje

A teszt összeállítása itt is ugyanaz volt, mint az előző bekezdésben, annyi különbséggel, hogy a kamerát felerősítettem egy állványra. A teszt során a kamera előtt mozgattam az aktuális célpontot, és a célfelismerés megbízhatóságát és sebességét vetettem össze. Először két olyan módszert vizsgáltam, ahol a célpont nem általános, hanem egy PNG képet vesz mintának, és ezt keresi a kamera által szolgáltatott képen, végül pedig egy betanított hálót.

Template Matching

Erről a módserről korábban a 6.2.1. bekezdésben volt szó bővebben. A `cv2.matchTemplate()` függvény megkeresi a képen a sablont, a `cv2.minMaxLoc()` pedig visszaadja a helyzetét, valamint az eredmény pontosságát. Az eredmény pontosságának megszabtam egy alsó határt, és azt változtattam a teszt során. Azt tapasztaltam, hogy ez a módszer nagyon érzékeny a célpont dőléssére és elfordulására, valamint a távolságára is, ezért nem tartottam megfelelő megoldásnak. Emellett minél nagyobb felbontású kép a sablon, a folyamat annál lassabb lesz.



7.2. ábra. Felismerő algoritmus tesztje

Feature Matching és Target Tracking

A feature matching elviekben jól kezeli a célpont torzulását, ezért ezzel kísérleteztem a későbbiekben. Az elképzelés az volt, hogy a feature matching megtalálja megbízhatóan a célpontot, majd a target tracking képes lesz gyorsan követni. A gyakorlatban, amennyiben egyszer megtalálta az algoritmus a célpontot, onnan tényleg pontosan tudta követni, azonban magával a felismeréssel voltak itt is a problémák. Nagyon megbízhatatlan volt, ha az érzékenységet lentebb állítottam, akkor fals pozitívot is mutatott, ha fentebb, akkor pedig nem találta meg a célpontot. A target tracking is csak akkor működött, ha a kamera egy helyen maradt. Ha már a motorok is mozogtak, akkor könnyedén elvesztette a célpontot.

Haar Cascade

Az utolsó módszer, amit teszteltem, az a Haar Cascade és tanított neurális hálók használata volt. Ez az előzőekhez képest meglepően megbízható volt, a célpontot felismerte viszonylag távolról is, akár rossz fényviszonyok között is. A célpont kb 15 fokos dőlése esetén szintén felismerte azt, amit én megfelelőnek találtam. Ezzel a módszerrel folytattam a munkát.

Miután eldöntöttem, melyik módszerrel haladok tovább, kipróbáltam az algoritmust a fegyver mozgásával összhangban. Ennek módja az volt, hogy a kamera előtt egy laptopon mozgattam a sablon. Erről kép látható a 7.2. ábrán.



7.3. ábra. Az éles teszt környezete



7.4. ábra. Az éles teszt környezete

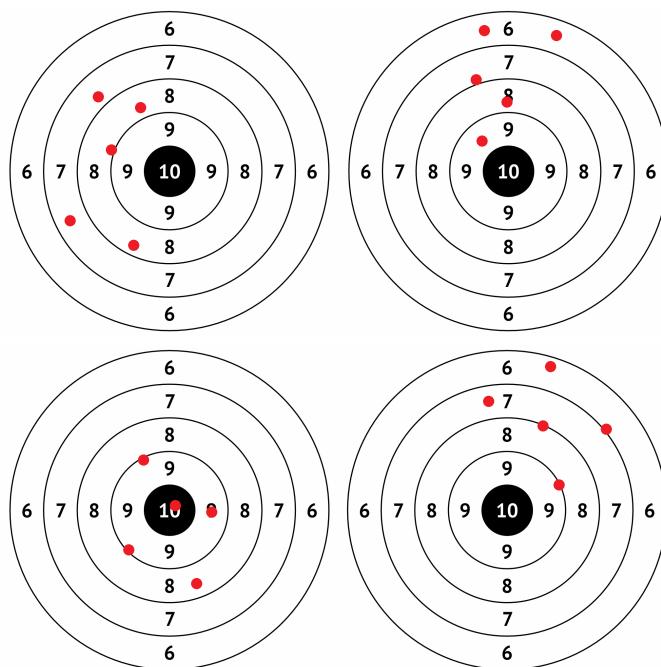
7.3. Éles teszt

A tesztet egy jól megvilágított tanteremben végeztem. Az eszköz egy stabil padon állt, a célpont pedig 5 méterrel előtte, és kb. fél méterrel fölötté. A célpont a 7.4. ábrán látható.

A teszt során a manuális működési móddal kezdtem, azon belül a fegyver kalibrálásával. Először azt tapasztaltam, hogy a kamera képének a közepe nem ott van, ahova a fegyver csöve mutat. Ezért a célkeresztet addig mozgattam a képen, amíg oda nem mutatott, ahova a fegyver ténylegesen lőtt. Ezután a rendszer nagy pontossággal tudott lőni, tulajdonképpen az összes leadott lövés a hibahatáron belül volt. A lézer irányzékot azonban nem tudtam pontosan beállítani, 5 méteren nagyjából 10-15 cm-t téved. A következő ábrákon látható a teszt eredménye. 5-ös sorozatokban tüzeltem, minden a cél közepére irányítva a kamera célkeresztjét.

A következő teszt az arcfelismerés, és a követés funkció tesztje volt. A körülmények ugyanazok maradtak, és szintén 5 m-re volt a célpont. A teszt során egy kolléga tartotta a kinyomtatott arcképet, és mozgatta, változtatva a sebességet és irányt. Természetesen a gearbox nem volt áram alatt, és a kolléga is viselt védőszemüveget. Azt tapasztaltam, hogy a fegyver körülbelül a sétáló ember sebességeit tudja követni 5 m távolságban, és igen pontosan be tudja céloznai. A tesztkörnyezet a 7.3. és 7.4. ábrákon látható.

Összességében elmondható, hogy a beállított célkereszttel, 5 méterről igen pontos eredmények születtek. 4 leadott sorozat során 5-5 golyót lőttem ki, az eredmények az alábbi ábrán láthatóak:



7.5. ábra. Az éles teszt eredményei

8. fejezet

Hibák, észrevételek

A prototípus fejlesztés során számos helyen vétettem kisebb-nagyobb hibákat, amiket súlyosságuktól függően vagy javítani kellett, vagy valahogyan kikerülni. Az első, amit meg szeretném említeni, az a **tár kialakítása** volt. Nem voltam elég körültekintő, és anélkül terveztem meg a tárat, hogy megnéztem volna, az eredeti fegyverekben hogyan működik. Ennek eredménye lett a nagy kapacitású tár, aminek elméletben folyamatosan kéne adagolni a golyókat, de a gyakorlatban a szűkítésnél minden esetben elakad. Ezzel a kialakítással egyet sem sikerült tüzelni, így újra kellett tervezni a teljes tárat. Miután jobban megvizsgáltam az eredeti fegyvert, láttam, hogy a golyók egyesével sorakoznak minden esetben, ezzel gátolva az elakadást. Miután hasonlóan megterveztem az új tárat, már megbízhatóan működött, habár kisebb kapacitással, így ezt a hibát sikerült kijavítani.

A következő, egy szintén mechanikai természetű hiba, a **lézer modul és a kamera rögzítése**. Egyik sem állítható, és ha eredetileg nem egy pontra néznek a fegyver csövével, akkor tulajdonképpen a hiba maradandó lesz. Ez így is lett, de a kamera eltérését még a célkereszt állításával tudtam orvosolni. A lézer modul maradandóan pár fokkal félre áll, ezt sajnos el kellett fogadnom.

A projekt elektronikai részénél is van, amit máshogy csinálnék, ha újrakezdeném. Az első koncepcióterveknél még nem néztem utána, mekkora áramot képes felvenni a gearbox. Úgy gondoltam, hogy egy átlagos 12V-os laptoptöltőről is tud majd működni a rendszer. Így igen meglepődtem, mikor utáranéztem, mekkora teljesítményre van szüksége. Az eredmény így az lett, hogy egy külön tápot kell adnom csak a gearbox működtetésére, ami igen kényelmetlen. Valószínűleg az lenne helyette egy jó megoldás, ha tervezek egy külön nyomtatott áramkört, amely képes arra, hogy töltsön egy akkumulátort a fegyveren. Így a gearbox aróról működne, akár csak egy módosítatlan fegyveren, működésen kívül pedig a Raspberry PI tápjáról töltődne.

9. fejezet

Továbbfejlesztési lehetőségek

Erősebb szerkezet: Az egyik egyértelmű fejlesztési lehetőség a jobb minőségű alkatrészek használata. Elsősorban a fogaskerekekre gondolok, amelyeket ha nem 3D nyomtatással gyártottam volna, hanem valamilyen nagy pontos-ságú folyamattal, akkor a szerkezet nem csak csendesebb, de precízebb is lehetne.

Jobb kamera: A Raspberry PI Camera 2 megfelelő minőséggel rendelkezik arra, hogy ezt a prototípust használni tudjam vele. Azonban ha távolabbi célpontokat kellene keresni, akkor egy optikai zoom mindenkorban szükséges lenne.

Vezeték nélküli működés: A tápellátás és a kommunikáció vezeték nélkülivé tétele magában hordozna új kihívásokat, például be kellene építeni egy nagyobb teljesítmény akkumulátort, illetve a vezeték nélküli kapcsolatok rendszerint lassabbak, mint a vezetékesek. Ellenben ha az eszköz célját tekintem, a távolsági operálás lenne a természetes fejlesztési irány, ahol a felhasználó akár egy másik épületből is képes lenne irányítani a fegyvert.

Mozgás: És ha már vezetékek nélkül képes működni az eszköz, akkor a következő lépés az lenne, hogy mozogni tudjon, például lánctalpak segítségével. Ez is szintén új kihívásokat jelent, de az eszköz többi részéhez képest már nem lenne nagyon bonyolult megvalósítani.

Radar, távolságérzékelő: Az eszköz következő verziójába kerülhetne akár radar is. A hasonló eszközök modern felhasználása leginkább légvédelmet jelent, azon belül drónok lelövését. Ezt vizuálisan nehézkes lehet érzékelni, hiszen igen távol is lehetnek, és kicsi a keresztmetszetük. Egy radar beépítése azonban megkönnyítené az észlelést. A távolságérzékelő pedig egyszerű több információt szolgáltat a célpontról, másrészt pedig nagy távolságoknál már számolni kell a lövedék bal-lisztikájával is, tehát hogy esik.

10. fejezet

Konklúziók

Összességében véve magabiztosan jelentem ki, hogy a projekt sikerrel járt. A két félév alatt, amíg ezzel a feladattal foglalkoztam, amellett, hogy az elméleti ismereteimet a gyakorlatban is kipróbálhattam, rengeteg hasznos új tudást is szereztem. Közelebbről megismerkedtem a **3D nyomtatás** technológiájával, a **gépi látás** eszközeivel, illetve számos hasznos **Python** könyvtárral is. Most már van gyakorlatom a **Raspberry PI** alapú rendszerek fejlesztésében, ami a későbbi karrierem mellett a hobbi projektjeimben is előny lesz.

Elkészítettem a prototípust, amire vállalkoztam: az eszköz magabiztosan és stabilan képes mozogni, fürgén követi a felhasználó által adott parancsokat, és önálló célfelismerésre is alkalmas. A tüzelés is zökkenőmentesen működik, a lövedék nem akad el a tárban. Mindemellett meglepően pontos. A biztonsági funkciók, amikkel elláttam az eszközt, biztosítják, hogy olyan célpont ne legyen lelőve, akit az operátor nem szándékozott eltalálni.

A projekt teljes egészében bárki által elérhető a *GitHubon*, ugyanis szeretném, ha más is reprodukálni tudná, ha hasonló érdeklődéssel rendelkezik. Ezzel a gondolattal írtam a dokumentációt is, amiben próbáltam részletesen kitérni a megtett lépésekre, gondolatok hátterére.

A projektet valamilyen formában mindenféle szeretném folytatni, akár egyedül, akár többedmagammal. Számos fejlesztési lehetőség maradt a prototípusban, és úgy gondolom több munkával, a most megszerzett tapasztalatokat bevetve, egy egészen kiforrott terméket készíthetünk el, amire talán mások is felfigyelnek.

Ha egy fontos konklúziót le szeretnék vonni így a feladatom végén, az az, hogy mindenféle érdemes volt olyan feladatot választani amiben a technikai ki-hívás mellett van személyes motiváció is. Nem csak egy elvégzendő feladatként tekintettem a dolgozatra, hanem őszintén érdekelt a sorsa, és ezért a rengeteg tanulás mellett kijelenthetem, hogy élveztem is foglalkozni a projekttel. Úgy gondolom, a diplomamunkám egy méltó lezárása lesz az egyetemi tanulmányaimnak.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani konzulensemnek, Dr. Stumpf Péter Pál tanár úrnak, amiért elfogadta a téma kidolgozása iránti igényemet, és a dolgozat írása közbeni folyamatos segítségéért, támogatásáért.

Külön szeretném megköszönni közeli barátomnak és kollégámnak, Zrupkó Zalánnak, amiért tanácsokkal látott el, segített a tesztelésnél és a saját idejét áldozta rá a prototípus helyes 3D nyomtatására.

Végezetül pedig szeretnék köszönetet mondani a családomnak, amiért lehetővé tették számomra, hogy elkezdjem és végigjárjam azt az utat, amelynek ez a dolgozat a végállomása. Tudom, hogy támogatni fognak továbbra is, bárhol is folytatódjon az életem az egyetem után.

Irodalomjegyzék

- [1] 1. Leírás a team fortress 2 játékban lévő sentry gun-ról (hozzáférés: 2024.11.19.). https://wiki.teamfortress.com/wiki/Sentry_Gun.
- [2] 2. Leírás a portal játékban lévő sentry gun-ról (hozzáférés: 2024.11.19.). https://theportalwiki.com/wiki/Sentry_Turret.
- [3] 3. Leírás a phalanx ciws rendszerről (hozzáférés: 2024.11.19.). http://www.navweaps.com/Weapons/WNUS_Phalanx.php.
- [4] 4. Leírás a crows rendszerről (hozzáférés: 2024.01.22.). <https://www.globalsecurity.org/military/systems/ground/m101-crows.htm>.
- [5] 5. Leírás az arbalet-dm rendszerről (hozzáférés: 2024.01.22.). <http://roe.ru/eng/catalog/land-forces/RWS/arbalet-dm/>.
- [6] 6. Leírás az defnder medium rendszerről (hozzáférés: 2024.01.22.). <https://fnamerica.com/products/weapon-systems/defnder-medium/>.
- [7] 7. Leírás a samsung sgr-a1 rendszerről (hozzáférés: 2024.02.10.). <https://www.globalsecurity.org/military/world/rok/sgr-a1.htm>.
- [8] 8. Információ a paintball fegyverekről (hozzáférés: 2024.10.22.). <https://steeltownpaintball.com/what-you-should-know-about-paintball-buying-upgrades-and-more/>.
- [9] 9. Információ a paintball robotról (hozzáférés: 2024.10.22.). <https://community.robotshop.com/robots/show/robotic-paintball-sentry>.
- [10] 10. Információ a nerf fegyverekről (hozzáférés: 2024.10.22.). <https://shop.hasbro.com/en-us/nerf>.
- [11] 11. Információ az airsoft fegyverekről (hozzáférés: 2024.10.22.). <https://highspeedbbs.com/all-about-airsoft-guns-types-styles-facts-science/>.
- [12] 12. Teszt az airsoft gearbox áramfogyasztásáról (hozzáférés: 2024.10.22.). http://www.airsoftlab.eu/docs/experiments/motor_current/.
- [13] 13. A nema szabvány (hozzáférés: 2024.10.22.). <https://www.cncitalia.net/file/pdf/nemastandard.pdf>.
- [14] 14. Bambu lab a1 (hozzáférés: 2024.10.22.). <https://bambulab.com/en/a1>.

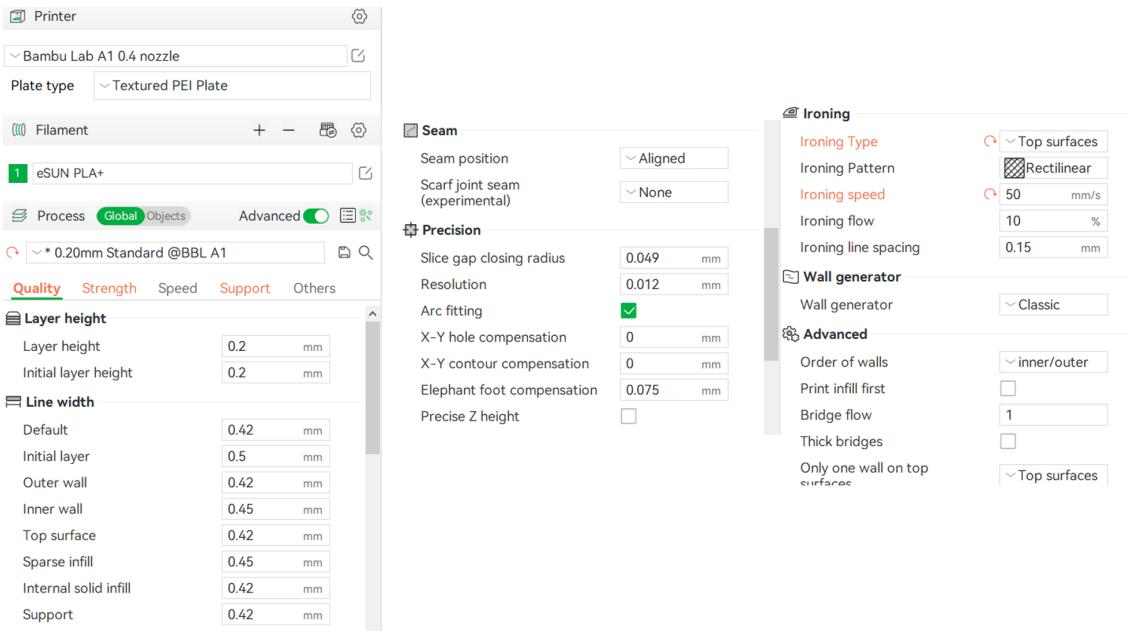
- [15] 16. Raspberry 4b specifikációk (hozzáférés: 2024.10.22.). <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>.
- [16] 15. Felhasználói diskurzus a raspberry pi gépi látás kapacitásáról (hozzáférés: 2024.10.22.). <https://forums.raspberrypi.com/viewtopic.php?t=366431>.
- [17] 17. Waveshare stepper motor hat spevifikációk (hozzáférés: 2024.10.22.). https://www.waveshare.com/wiki/Stepper_Motor_HAT.
- [18] 18. Raspberry 2 kamera specifikációk (hozzáférés: 2024.10.22.). <https://www.raspberrypi.com/products/camera-module-v2/>.
- [19] 19. Dokumentáció az opencv modulról (hozzáférés: 2024.10.22.). <https://www.raspberrypi.com/products/camera-module-v2/>.
- [20] 20. Dokumentáció a socket modulról (hozzáférés: 2024.10.22.). <https://docs.python.org/3/library/socket.html>.
- [21] 21. Dokumentáció a multiprocessing modulról (hozzáférés: 2024.10.22.). <https://docs.python.org/3/library/multiprocessing.html>.
- [22] 22. Dokumentáció a pygame modulról (hozzáférés: 2024.10.22.). <https://www.pygame.org/docs/>.
- [23] 23. Dokumentáció a pickle modulról (hozzáférés: 2024.10.22.). <https://docs.python.org/3/library/pickle.html>.
- [24] 24. Dokumentáció a cv2 template matching-ről (hozzáférés: 2024.10.22.). https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html.
- [25] 25. Dokumentáció a cv2 feature matching-ről (hozzáférés: 2024.10.22.). https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html.
- [26] 26. Dokumentáció a cv2 target tracking-ről (hozzáférés: 2024.10.22.). https://docs.opencv.org/4.x/d2/d0a/tutorial_introduction_to_tracker.html.
- [27] 27. Dokumentáció a cv2 haar cascade-ról (hozzáférés: 2024.10.22.). https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

Függelék

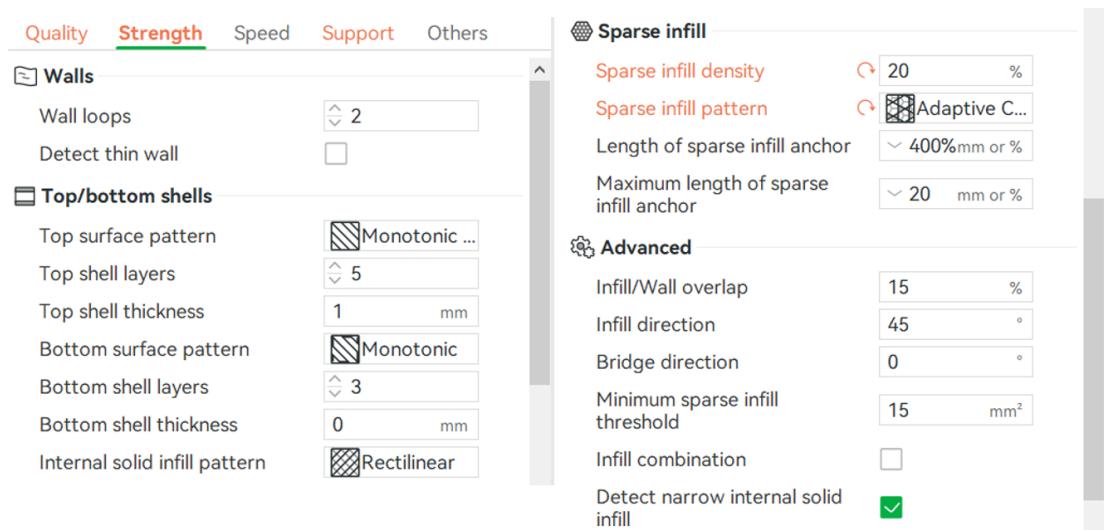
F.1. A Bambu Slicer beállításai

Line Type	Time	Percent	Used filament	Display
Inner wall	30m56s	10.1%	9.46 m	28.45 g
Outer wall	36m12s	11.8%	8.92 m	26.82 g
Overhang wall	18s	0.1%	0.01 m	0.04 g
Sparse infill	1h31m	29.8%	22.29 m	67.03 g
Internal solid infill	23m12s	7.6%	5.14 m	15.44 g
Top surface	5m9s	1.7%	0.70 m	2.10 g
Bottom surface	6m37s	2.2%	0.96 m	2.90 g
Ironing	20m0s	6.5%	0.07 m	0.20 g
Bridge	8m49s	2.9%	1.26 m	3.78 g
Gap infill	24s	0.1%	0.01 m	0.02 g
Support	17m36s	5.8%	1.86 m	5.58 g
Support interface	18s	<0.1%	0.04 m	0.12 g
Custom	5m57s	1.9%	0.02 m	0.07 g
Travel	59m24s	19.4%		
Retract				
Unretract				
Wipe				
Seams				✓
Total Estimation				
Total Filament:	50.73 m	152.54 g		
Model Filament:	48.84 m	146.84 g		
Cost:	3.51			
Prepare time:	5m42s			
Model printing time:	5h0m			
Total time:	5h6m			

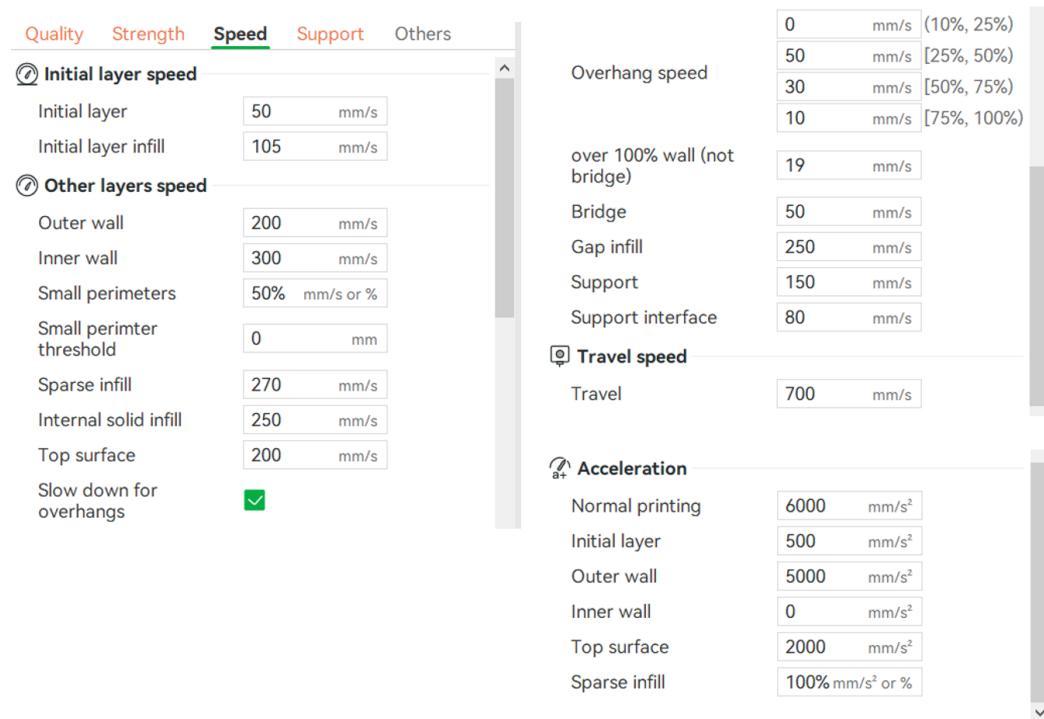
F.1.1. ábra. A Bambu Studio által szolgáltatott adatok



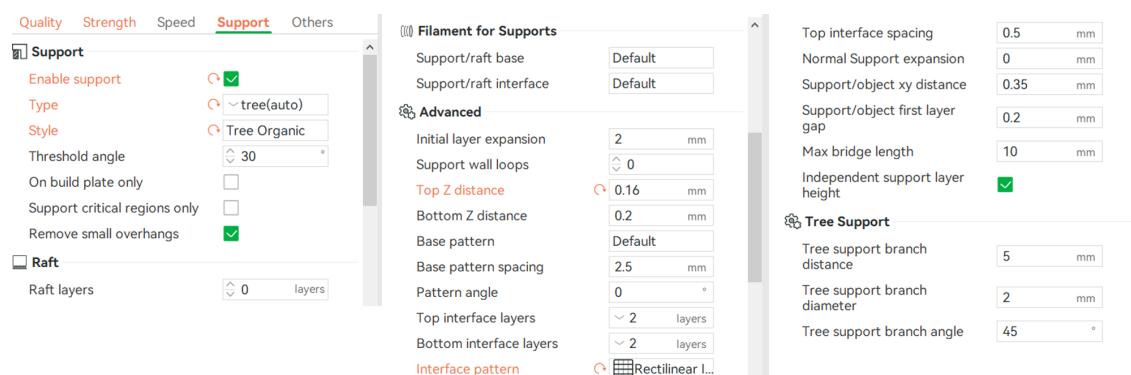
F1.2. ábra. A Bambu Studio minőségi beállításai



F1.3. ábra. A Bambu Studio erősségi beállításai



F.1.4. ábra. A Bambu Studio sebesség beállításai



F.1.5. ábra. A Bambu Studio támasz beállításai