C. I. W. S. RENDSZEREK BEÁGYAZOTT ALAPOKON

Ambrus Attila, Zisis Christoforos
Óbudai Egyetem
Neumann János Informatikai Kar, BSc III. évfolyam, BSc III. évfolyam
Konzulens: Dr. habil. Molnár András, egyetemi docens
Dr. Stojcsics Dániel Zoltán, adjunktus

Célunk egy olyan autonóm fegyverrendszer megvalósítása beágyazott rendszer alapokon, melynek segítségével szemléltethetjük a Close In Weapon System (CIWS) típusú rendszerek jelentőségét, működési mechanizmusait és megvalósításuk nehézségeit.

Dolgozatunk ennek megfelelően egy általunk elképzelt CIWS rendszer tervezetét és megvalósítását részletezi, melynek során a rendelkezésünkre álló szenzorokból, egy Arduino Mega és egy Raspberry PI 2-es fejlesztői panelből, valamint egy mobiltelefonból egy önműködő, de távolról manuálisan is vezérelhető fegyverrendszert állítunk elő.

Komplex rendszerünket öt, funkcionálisan külön álló alrendszerre bontottuk. A fegyverzet alrendszer feladata a kapott paraméterek alapján a lőfegyver célra állítása, a kilövés mechanizmusának megvalósítása (a lövedékek adagolása, újratöltése valamint azok célba juttatása). Az érzékelő alrendszer a célpontok érzékeléséért és beméréséért felelős alrendszer. Harmadik alrendszerünk a kamera alrendszer, mely a manuális célzást hivatott segíteni egy élőkép segítségével, amely a felhasználó tájékozódását képes megkönnyíteni. A sérülést detektáló alrendszer érzékeli a fegyverzet sérülését és erről értesíti a fegyver felhasználóját, melynek taktikai jelentősége lehet egy éles helyzetben. És végezetül a kommunikációs alrendszer, amely egy Androidos mobiltelefon, egy Arduino és a Raspberry közötti információáramlást teszi lehetővé. A TDK dolgozat, a fenti öt alrendszerből álló Close In Weapon rendszert, mint egységet hivatott ismertetni.

A megvalósítás során szükséges bizonyos egyedi alkatrészek elkészítése, melyeket 3D nyomtató segítségével valósítunk meg. Ezek az alkatrészek jellemzően a szerkezet vázát alkotják, az egyes szenzorokat rögzítik, illetve mechanikai szerepük van. Dolgozatunk egy rövidebb szegmense a nyomtatás kapcsán felgyülemlett tapasztalatainkat összegzi, kitér az alkatrészek megtervezésére, lemodellezésére.

A jelen szakdolgozat tárgyát képező CIWSduino álnévre keresztelt fegyverrendszer egy jóval kisebb, költséghatékonyabb megvalósítása a katonai célokra felhasznált nagyobb testvéreinek. Ezek elsődleges célkitűzése – ahogyan dolgozatunké is – a témakör ismertetése és egy olyan modell kialakítása, amelyen könnyedén szemléltethetőek a Close In Weapon rendszerek sajátosságai.

TARTALOMJEGYZÉK

Tartalom

TARTALOMJEGYZÉK

1. ABSTRACT

Célunk egy olyan autonóm fegyverrendszer megvalósítása beágyazott rendszer alapokon, melynek segítségével szemléltethetjük a Close In Weapon System (CIWS) típusú rendszerek jelentőségét, működési mechanizmusait és megvalósításuk nehézségeit.

Dolgozatunk ennek megfelelően egy általunk elképzelt CIWS rendszer tervezetét és megvalósítását részletezi, melynek során a rendelkezésünkre álló szenzorokból, egy Arduino Mega és egy Raspberry PI 2-es fejlesztői panelből, valamint egy mobiltelefonból egy önműködő, de távolról manuálisan is vezérelhető fegyverrendszert állítunk elő.

Komplex rendszerünket öt, funkcionálisan külön álló alrendszerre bontottuk. A fegyverzet alrendszer feladata a kapott paraméterek alapján a lőfegyver célra állítása, a kilövés mechanizmusának megvalósítása (a lövedékek adagolása, újratöltése valamint azok célba juttatása). Az érzékelő alrendszer a célpontok érzékeléséért és beméréséért felelős alrendszer. Harmadik alrendszerünk a kamera alrendszer, mely a manuális célzást hivatott segíteni egy élőkép segítségével, amely a felhasználó tájékozódását képes megkönnyíteni. A sérülést detektáló alrendszer érzékeli a fegyverzet sérülését és erről értesíti a fegyver felhasználóját, melynek taktikai jelentősége lehet egy éles helyzetben. És végezetül a kommunikációs alrendszer, amely egy Androidos mobiltelefon, egy Arduino és a Raspberry közötti információáramlást teszi lehetővé. A TDK dolgozat, a fenti öt alrendszerből álló Close In Weapon rendszert, mint egységet hivatott ismertetni.

A megvalósítás során szükséges bizonyos egyedi alkatrészek elkészítése, melyeket 3D nyomtató segítségével valósítunk meg. Ezek az alkatrészek jellemzően a szerkezet vázát alkotják, az egyes szenzorokat rögzítik, illetve mechanikai szerepük van. Dolgozatunk egy rövidebb szegmense a nyomtatás kapcsán felgyülemlett tapasztalatainkat összegzi, kitér az alkatrészek megtervezésére, lemodellezésére.

A jelen szakdolgozat tárgyát képező CIWSduino álnévre keresztelt fegyverrendszer egy jóval kisebb, költséghatékonyabb megvalósítása a katonai célokra felhasznált nagyobb testvéreinek. Ezek elsődleges célkitűzése – ahogyan dolgozatunké is – a témakör ismertetése és egy olyan modell kialakítása, amelyen könnyedén szemléltethetőek a Close In Weapon rendszerek sajátosságai.

2. BEVEZETÉS

Célunk egy olyan autonóm fegyverrendszer megvalósítása beágyazott rendszer alapokon, melynek segítségével szemléltethetjük a CIWS rendszerek működését, buktatóit és jelentőségeit. Az általunk elképzelt rendszer szenzorok segítségével érzékeli a környezetét és beavatkozás nélkül cselekszik, megsemmisíti az elé kerülő ellenséges objektumokat. Szükség esetén lehetőségünk van okos telefonról beavatkozni, illetve manuálisan irányítani a fegyverzetet, melyben egy ultrahangos radar és egy élő kamerakép lesz a segítségünkre.

3. HASONLÓ RENDSZEREK

3.1. Autonóm fegyverek

Autonóm fegyvereknek nevezzük azokat a fegyvereket, melyek képesek emberi beavatkozás nélkül célpontot választani és támadni. Az ilyen fegyverek különböző szenzorok és a csatatérről nyert egyéb információk segítségével eldöntik, hogy melyik célpont megsemmisítésével érhető el a legnagyobb eredmény. A jelenleg használt autonóm fegyverek ugyan képesek teljesen autonóm működésre, de még mindig emberi felügyelet mellett dolgoznak.

3.1.1. SAMSUNG SGR-A1 SENTRY GUN

A Samsung SGR-A1 egy Dél-Koreában gyártott teljesen autonóm fegyver. Az Észak- és Dél-Korea közötti demilitarizált zóna védelmére fejlesztették ki, hogy leválthassák vele az embereket. A fegyver optikai és infravörös érzékelőkkel érzékeli a mozgó célpontokat és egy lézeres távolságmérő segítségével pontosítja a célzást.

1. ábra - A Samsung SGR-A1[1]

3.2. Pont védelmi fegyverek

A pont védelmi (point-defence) fegyverek olyan fegyverek melyeket egy kisebb objektum védelmére használnak. Általában relatív kicsi a hatótávolságuk és a védeni kívánt objektum közvetlen közelében, vagy közvetlenül az objektumon (tank, csatahajó) helyezik el.

[1] http://i-cdn.phonearena.com/images/articles/203990-image/Samsung-SGR-A1-sentry-robot.jpg

3.3. Close in weapon systems (CIWS)

Az olyan pont védelmi fegyver rendszereket, melyeket elsősorban csak repülő célpontok ellen használnak, close in weapon system-nek nevezik. Ilyen célpontok lehetnek például: drónok, fix- és forgószárnyas repülők, rakéták, tüzérségi gránátok. Az ilyen CIWS rendszereknek két altípusa van.

3.3.1. GÉPÁGYÚ ALAPÚ

Az ilyen rendszerekben a célpontok megsemmisítéséért egy gépágyú felelős (általában gatling gun). A célpont bemérése után a cél, hogy a fegyver annyira meg tudja sebesíteni a beérkező célpontokat, hogy azok megsemmisüljenek, de legalább irányváltásra kényszerüljenek. A fejlettebb rendszerek most már „okos lövedékeket" is használhatnak, melyek sok kisebb lövedéket hordoznak magukban. Egy időzített robbanófej segítségével becsapódás előtt robbannak föl, így szétszórva a kis lövedékeket a célpont előtt.

3.3.2. IRÁNYÍTOTT RAKÉTA ALAPÚ

Ezeknél a rendszereknél a célpontok megsemmisítését irányított rakéták a felelősek. Ez sokkal drágább megoldás, mint a gépágyú alapú társa, cserébe viszont a hatótávolsága és a találati aránya sokszorosa elődjének.

3.4. Civil felhasználásuk

Civil felhasználásukban nem lehet halálos fegyvereket használni, így védelmi berendezésként nem alkalmazhatóak. Ez eléggé leszűkíti a felhasználási lehetőségeiket, de ettől függetlenül használják például taktikai játékoknál (paintball, airsoft) kiképzéshez és magához a játékhoz. Jogi okok miatt nem elterjedt az ilyen rendszerek használata.

3.4.1. PAINTBALL SENTRY

Ezek a rendszerek optikai mozgás felismerést alkalmaznak a célpont bemérésére. Mivel csak egy kamerát használnak, ez erősen limitálja a lefedhető területet.

2. ábra - paintball sentry-k[2]

[2] http://www.paintballsentry.com/images/sentrygunkit.JPG

3.5. Katonai felhasználásuk

Katonai felhasználásuknál első sorban hajókon rakétavédelemre, illetve kisebb bázisokon használják őket.

3.5.1. PHALANX (USA)[3]

Az General Dynamics (később Raytheon) által fejlesztet teljesen automata gépágyú vagy irányított rakéta alapú CIWS. Képes repülő, földi és tengeri célpontok bemérésére és megsemmisítésére egyaránt. A repülő célpontok bemérésére 2 különböző radart használ. Egy kereső radar, ami nagy távolságból képes felismerni a potenciális célpontok sebességét, irányát, magasságát és méretét beazonosítja a célpontot. Amint a rendszer észlelte a célpontot a kereső radar segítségével, akkor a célpont irányába fordul és bekapcsolja a célzó radart, ami sokkal kisebb szögben érzékeli, viszont nagyon nagy pontossággal képes követni azt. Ezt az információt felhasználva kiszámolja a lehetséges röppályákat. Ezen kívül a Phanax még föl van szerelve infravörös és optikai érzékelővel is földi és tengeri célpontok, esetleg nagyobb méretű lassan mozgó repülő célpontok bemérésére.

3. ábra - A Phalanx rendszer

3.5.2. GOALKEEPER (HOLLANDIA)[4]

Ugyanazt a radarrendszert használja a célpontok felismeréséhez, mint a Phalanx, viszont nincs felszerelve infravörös érzékelővel. Egyetlen optikai érzékelővel látták el, ami csak azt a célt szolgálja, hogy manuálisan irányítható maradjon a rendszer a radarok meghibásodása esetén.

4. ábra - A Goalkeeper rendszer

3.5.3. AK-630 (OROSZORSZÁG)[5]

Az orosz haditengerészet által használt CIWS. Különlegessége, hogy a már megszokott kereső és célzó radarrendszeren kívül elektrooptikai szenzorokat is használ a pontosabb és gyorsabb követésre. Így jelenleg ez az egyik legpontosabb close in weapon system.

5. ábra - Az AK-630-as rendszer

[3] http://vignette2.wikia.nocookie.net/battlefield/images/b/b1/Phalanx_Turret.jpg/revision/latest?cb=20100209132108
[4] http://media.defenceindustrydaily.com/images/ORD_CIWS_30mm_Goalkeeper_ROKN_lg.jpg
[5] https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Duetak630m2.jpg/324px-Duetak630m2.jpg

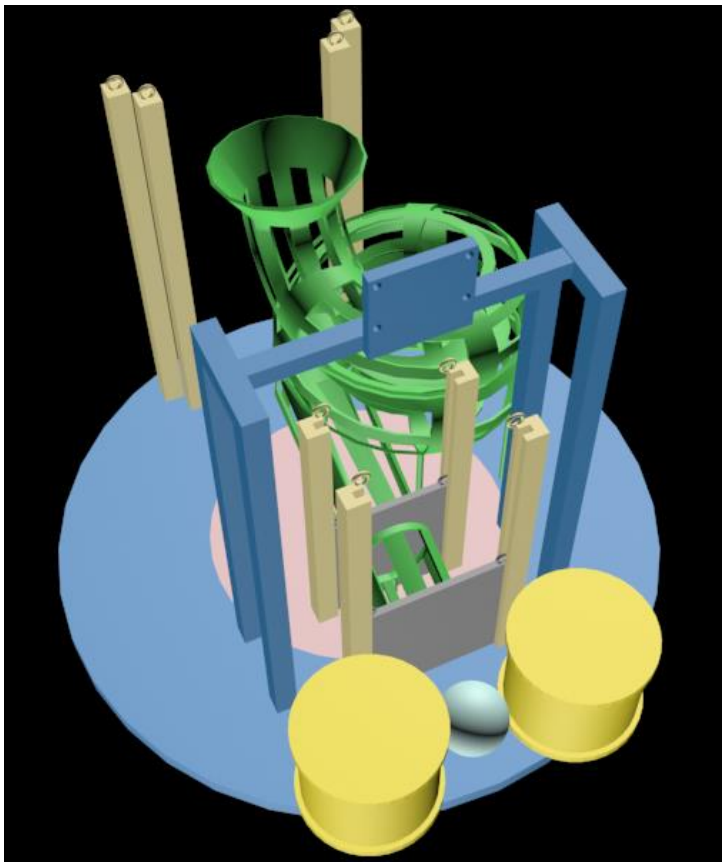
4. RENDSZER ISMERTETŐ

4.1. Rövid ismertető

Az általunk elképzelt architektúrát 5 kisebb alrendszerre bontottuk szét, melyek sorra: érzékelő alrendszer, kamera alrendszer, kommunikációs alrendszer, sérülés detektáló alrendszer és fegyverzet alrendszer. Az általunk megtervezett close in weapon system három fő modulból áll. Egy Arduino Megából, amely magáért az érzékelésért és a fegyverzet kezeléséért felelős. Egy Raspberry Pi 2-ből, amely egy kamerakép stream-elését látja el. Valamint egy okos telefonból, amely lehetővé teszi a fegyver távolról történő manuális vezérlését a kamerakép segítségével és egyéb statisztikai adatokat jelenít meg. Mint például ultrahangos radarkép, a célpont észlelés ténye, korábbi aktivitások, tárban lévő golyók száma, a fegyver sérülésének ténye és még sok más.

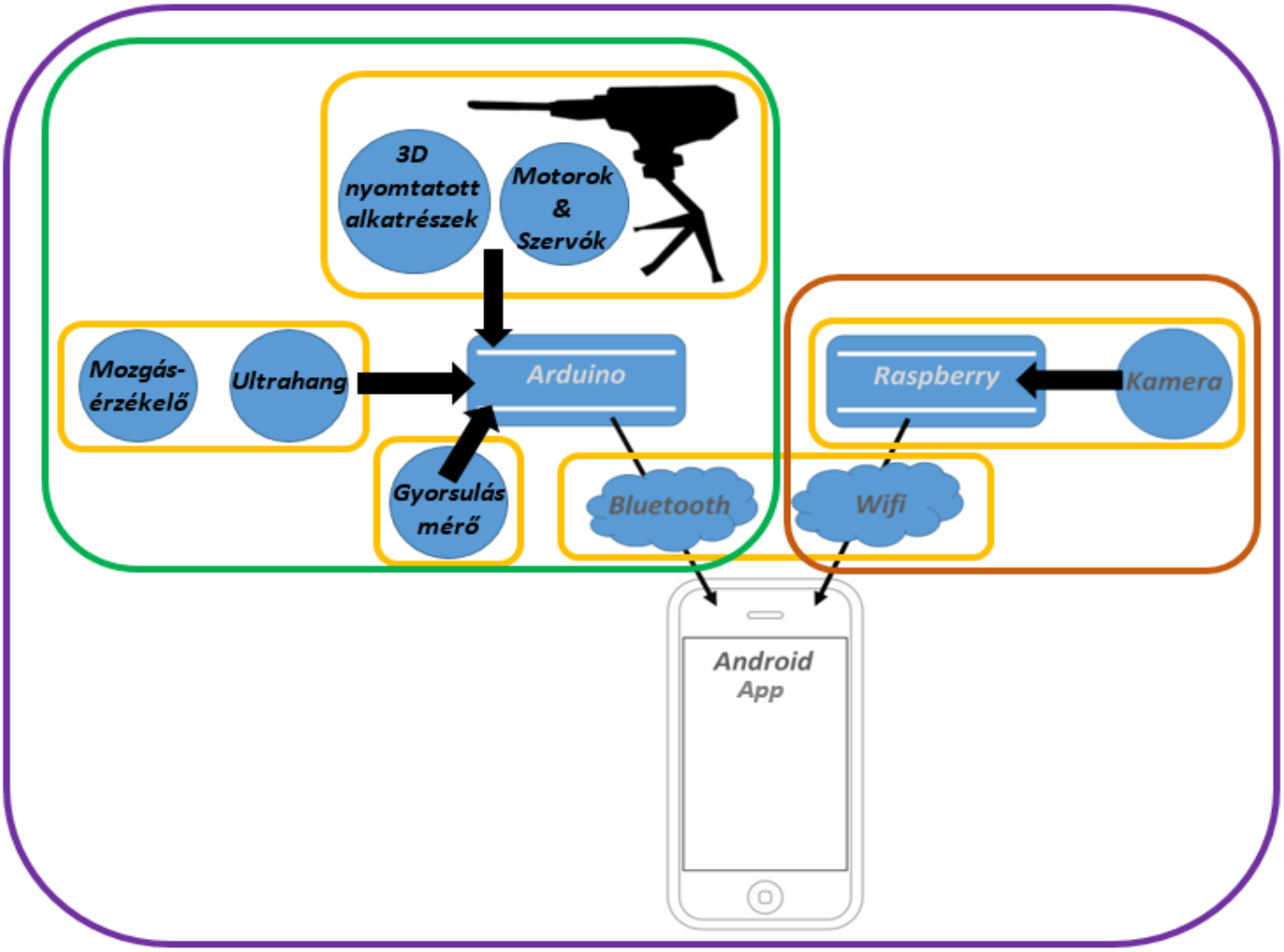
4.2. Bővebb ismertető

A célpontot egy 7m-ig ellátó infrás mozgásérzékelő modul érzékeli, amely érzékelés esetén aktivál egy 4,5m-ig ellátó ultrahangos távolságmérőt, ami pontosítja a cél helyzetét. A fegyvert egy motor segítségével ráirányítja a célpontra és megsemmisíti azt. A vezérlést bármikor átvehetjük egy telefon segítségével, amely bluetooth-on keresztül fog kommunikálni a fegyver Arduino alapú vezérlőjével. A manuális vezérlést egy kamerakép segíti, melyet a mobil, wifi-n keresztül egy Raspberry-től tud átvenni. A telefon azon kívül, hogy döntögetésre és egyéb interakciókra manuálisan is irányíthatóvá teszi a fegyverzetet, statisztikát is vezet arról, hogy mikor érzékelt aktivitást, valamint egyéb környezeti adatokat gyűjt és jelenít meg. Többek között az érzékelés tényét és radarképen a célpont helyzetét. Találat esetén rezgéssel jelzi, ha megsemmisítettük a célpontot, vagy ha sebzés érte a fegyvertartó állványt, azaz elmozdult a helyzetéből, vagy felborult. Ezt a tényt egy gyroscope és accelerometer modul érzékeli majd. Továbbá a telefon segítségével állíthatjuk a fegyver vezérlő beállításait, az érzékelés szögét és maximális távolságát, hogy jelezze-e vibrálással a feldőlést, vagy csak egy szolid értesítésként jelezze ki.



6. ábra - kezdetleges koncepcionális ábra a fegyverrendszerről

5. ALRENDSZEREK



7. ábra - a CIWSduino alrendszerei

5.1. Fegyverzet alrendszer

Viszonylag nagy, üveggolyó és pingpong labda közötti méretű műanyag golyókat szeretnénk kilőni, melynek kilövő szerkezete a pingpong / teniszlabda kilövő gépekhez hasonlít, azaz a két oldalról ellentétes irányba forgó hengerek közé beszorított labdát a hengerek kerületi sebessége fogja kilőni. A hengereket módosított szervo motorok fogják forgatni, melyek megállás nélkül 360°-ban tudnak forogni. Ehhez a fizikai biztosító pöcök eltávolítására van szükségünk az egyik fogaskerékről, illetve egy a motor belsejében lévő apró változtatásra, mely során 2db 2,2kOhm-os ellenállásokból egy feszültségosztót ültetünk a kapcsolásba. A golyókat egy spirál alakú tárba szeretnénk betölteni a helytakarékosság miatt. Ebből a tárból a betöltést 1-1 kapu végzi, melyeket szervo motorra kötünk. A két kapu között pontosan egy labdának van hely, így amikor a hátsó kaput kinyitjuk, akkor csak egy labda töltődik a csőbe. Amikor ezt a kaput betöltés után bezárjuk, a tárban lévő többi golyó el lesz különítve a kilövendő golyótól, így amikor az első kaput tüzeléskor kinyitjuk, csak ez az egyetlen golyó gurul ki a hengerek közé, így csak egy lesz kilőve és számolható lesz a kilőtt golyók és a tárban lévők száma. Természetesen, ha mindkét kaput nyitva tartjuk, akkor sorozatlövés is elérhető, de ekkor másként kell megoldani a golyók számlálását, így ezzel a szituációval jelenleg nem számolunk. A fegyverzet és a tár egy kör alakú panelen lesz rajta, melynek vízszintes szögét egy léptető motorral állítjuk, így lehetővé téve a fegyver célpontra állását. Ennek a panelnek a mozgatása független lesz az érzékelő alrendszer paneljétől, így külön tudnak majd mozogni. A golyókat, a tárat, a hengereket, a kapukat, a paneleket és a mozgatásukhoz szükséges fogaskerekeket az iskolai 3D nyomtatóval terveztük megvalósítani. A printer egy MakerBot Mini és maximum 10.0 L X 10.0 W X 12.5 H cm-es alkatrészeket lehet kinyomtatni a segítségével, így az esetlegesen nagyobb dimenziójú elemeket, összelegózható darabokból próbáljuk megvalósítani.

5.2. Érzékelő alrendszer

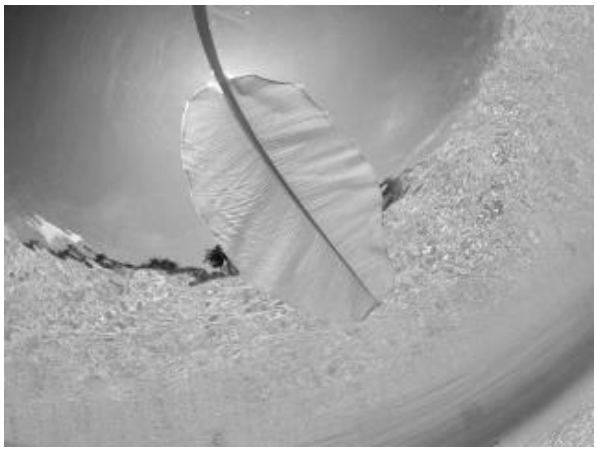
A mozgásérzékelő modul egy fix egység, amely előre fele néz és ha mozgás van 7m-es távolságon és 120° belül, akkor jelez. Jelzés esetén működésbe lép az ultrahang modul. Az ultrahang modul egy szervo motor tetején helyezkedik el. Így fogjuk tudni megoldani a pásztázást. A választott szervo motor 180°-ban képes mozogni, ez nekünk megfelelő. Induláskor a motor beáll az alappozícióba, azaz előre fele néz a 0°-ba. Innen indul el x fokonként balra, amíg a széléig ki nem ér. Ha elért, akkor vissza jobbra, amíg a másik széléig vissza nem ér. A lépések között a modul egy ping jelsorozatot küld ki és ennek a visszatértét várja. Ha végzett, csak akkor léphet tovább a léptető motor a következő pozícióba, így a rendszerben lesz egy minimális késleltetés. Leállításkor a motor szintén alappozícióba áll vissza. A szenzor csak 4,5m-ig mér és csak a legközelebbi távolságot mutatja, tehát hogy a legelső érzékelt jel mögött milyen egyéb objektumok vannak, azt nem tudjuk kijelezni a segítségével. A radar képét a könnyebb tájékozódás érdekében 30°-ént érdemes beosztani körcikkekre, valamint 50cm-enként körívekre.

5.3. Kamera alrendszer

A kamera tervezési okokból nem az ultrahangot tartó egységen van rajta, hanem külön mozgatható modulon a fegyver felett, hiszen a manuális célzást segíti és nem az érzékelést. Hogy pontosan tudjuk a fegyverzet szögét állítani, egy léptető motort fogunk alkalmazni, amely lépésenként 5,625°/64--edet fordul. Mivel 8 állapota van ezért a 180°-on belül 180/(8*5,625/64)=256 lépésközös felosztással tudjuk vezérelni.

Az Arduino maximális 115200Bps-es baud rátája mellet egy 320x240 pixel méretű 5-6kb-os jpg képet 0,3-0,5s alatt tudunk feldolgozni a kameráról. Így másodpercenként 2-3 képet tudunk megjeleníteni, ami 2-3fps. Ez nagyon darabos mozgást eredményezne és igen csak elmarad a filmeknél használt 22-26fps-hez képest. Ugyanilyen felbontású tömörítetlen bmp képnél ez 150kb és 13s is lehet, ami eléggé ellehetetlenítené a manuális vezérlést. A 640x480-as 20-30kb-os képeknél 1,7-2,6 másodperc, 1600x1200-as képeknél pedig 130kb 11 másodperc. A sebességet természetesen javíthatjuk, ha 8bites szürkeárnyalatos képet küldünk a 24bites színes kép helyett. Ekkor 1 pixel nem (255,255,255) értéket tárol, tehát 3 byte-ot, hanem csak (255)-öt, azaz 1byte-ot, ami 3-adára csökkenti a képméretet, azaz 3-szorozza a sebességet. De 320x240-es szürke jpg-ből még így is csak 6-9-et tudnánk küldeni másodpercenként, ami nem elég a célzáshoz.

*8. ábra - 320*240pixel méretű színes kép*[6]



*9. ábra - 320*240pixel méretű szürkeárnyalatos kép*

A Raspberry is tudja az UART csatlakozóján ezt a 115200-os baud rátát, ami körülbelül 14.4 kB/s, de itt van lehetőség egy kis tuningolásra. 1milliós baud rátánál még tökéletesen működik, ami körülbelül 125kb/s sebességet jelent.1,5milliónál már vannak benne kisebb szünetek, így már nem túl jó minőségű. 125kb/s már arra elég, hogy körülbelül 20 képkockát jelenítsünk meg másodpercenként. Ez teljesen megfelelne a célnak, mégsem ezt választjuk, hiszen a Raspberry-nek van 2 magos Video Core IV multimédia co-processora és CSI kamera csatlakozóján, sokkal jobb minőséget nyújt, mint az UART. Így akár 5Gbit/sec sebességgel dolgozhatunk fel képeket, valamint hangokat 8 különböző csatornán. A Raspberry FullHD-ra is képes, tehát 1080p-s videók szállítására is alkalmazhatjuk.

[6] http://www.ewallpapers.eu/w_show/leaf-on-sea-water-320-240-7215.jpg

5.4. Kommunikációs alrendszerek

5.4.1. ARDUINO ÉS ANDROID

Az Arduino a mobil telefonnal egy alacsony fogyasztású bluetooth modulon keresztül kommunikál, melynek az elméleti adatátviteli sebessége 25Mbit/sec, és nyílt területen 60m a hatótávolsága. Egy csomagon belül 20byte adatot lehet vele küldeni. A kamera kivételével ebbe a szűk keresztmetszetbe kell beleférjünk.

5.4.2. RASPBERRY ÉS ANDROID

A Raspberry egy wifi modulon keresztül fog kommunikálni, melynek elméleti sebessége 150Mbps. és USB 2.0-án csatlakoztatható a Raspberry-re. Ezen a kommunikációs csatornán csak a kamera képét fogjuk továbbítani.

Mindkét modul képes elfogadható biztonságú authentikációra és titkosításra. A bluetooth AES 128-at, a wifi modul pedig wpa2-psk AES 256-ot használ.

5.5. Sérülés detektáló alrendszer

A rendszer sérülését, a feldőlésével vagy elmozdulásával fogjuk vizsgálni. Ehhez egy 3 tengelyes gyroscope-ot (és gyorsulásmérőt) fogunk felhasználni, amellyel szabványos I2C-n keresztül kommunikálhatunk és kimenetéről egy 16bites adatcsomagot olvashatunk le. A szenzor érzékenységének felosztása állítható, így egyszerűen az igényeinkre szabhatjuk.
Gyroscope: 250/500/1000/2000 °/sec.
Gyorsulásmérő: 2/4/8/16g.

6. FELHASZNÁLT ESZKÖZÖK

6.1. 3D nyomtatott alkatrészek

Fegyverrendszerünk tervezésénél egy olyan egyszerűbb kilövőrendszert kialakítását tartottuk szem előtt, melynek alkatrészei könnyedén kinyomtathatóak egy hétköznapi használatra szánt 3D nyomtató segítségével is. Azonban a 3D nyomtatott alkatrészek helyes megtervezése nem egy egyszerű folyamat. A tervezésnél határt szabnak a fizikai korlátok. Ilyen például a nyomtató belső terének mérete, hiszen ez meghatározza a kinyomtatandó tárgyaink maximális méretét. Gyakran emiatt a korlát miatt az objektumokat kisebbre kell terveznünk, vagy több elemre szétszedve kell megvalósítanunk, melyeket majd a későbbiekben összeragasztunk. A másik nagy korlát a nyomtatóban felhasznált anyag minősége. Például az anyag sűrűsége meghatározza a kilövendő műanyag golyóink tömegét, melyekkel számításokat kell majd végeznünk a későbbiek folyamán. Ettől a fizikai számítástól függ a kiválasztandó motorjaink erőssége, és a rájuk csatlakoztatott kilövő hengerek sugarának nagysága is, hiszen ezek függvényében határozható meg a golyó röppályája. Ha a rendszerben bármi pontatlan, akkor a golyó nem száll elég messzire és túl nagy lesz a bemért célpont tényleges helyzete és a golyó becsapódásának a helye közötti távolság. Számolnunk kell a nyomtató nyomtatási felbontásával is, hiszen ez határozza meg, hogy milyen pontosan tudunk a számításainknál a kinyomtatandó tárgyaink tényleges méretével kalkulálni. Nem mindegy, hogy 3D tervezésnél a tervező programban tizedmilliméterre határoztuk meg a tárgyaink méreteit és számításainknál ezeket az adatokat használtuk fel, ha a nyomtatás után a tárgyak csak milliméterre pontosan felelnek meg ezeknek az értékeknek. Az ilyen és ezekhez hasonló pontatlanságok végzetes kimenetelűek lehetnek a projektünkre nézve. Végezetül, de nem utolsó sorban figyelembe kell vennünk, hogy a printer a tárgyakat rétegenként nyomtatja, alulról felfelé haladva, melynek több következménye is van. A 3D nyomtatás a tárgy méretétől, elhelyezkedésétől és részletességétől függően több órát is igénybe vehet. A nyomtatófej először minden nyomtatási pontot kinyomtat az aktuális rétegen, majd ezután lép egy réteggel felfelé a rétegek között. És így tovább, amíg be nem fejezi a teljes objektum kinyomtatását. A nyomtatandó tárgyainkat célszerű fektetve szervezni, hogy a fejnek minél kevesebb utat kelljen megtennie felfelé. Erre azért van szükség, mert a nyomtatófej a rétegen belül sokkal gyorsabban tud mozogni, mint a rétegek között, így számottevő időt spórolhatunk meg egy jól elhelyezett objektummal. Ilyenkor azonban ügyelnünk kell az elforgatott objektumok segéd alátámasztásokkal való ellátásáról, hiszen az objektumok súlypontja nem az elforgatott nézethez lett tervezve. Nyomtatás közben folyamatosan mozog a tárgy súlypontja, és ha nincs megfelelően alátámasztva, akkor elmozdulhat jelenlegi helyzetéből, ami meghiúsítja a nyomtatást. Hasonló okokból a nagyobb vízszintesen elhelyezett hidakat is alá szokták támasztan, hiszen a műanyag nem köt olyan gyorsan, hogy a nyomtatvány ne tudjon kicsit elfolyni, vagy eldeformálódni a gravitáció hatására. A legtöbb nyomtató rendelkezik automatikus algoritmussal az ilyen alátámasztások megtervezéséhez. Speciális esetekben azonban, ahol az ilyen alátámasztások utólagos letördelését a fő tárgyunkról, vagy kitördelését annak belsejéből lehetetlen lenne megoldani, ott magunknak kell gondoskodnunk a támogató nyomtatványok megtervezéséről is. Látható tehát, hogy a 3D nyomtatványok megtervezése egy komplex, körültekintést igénylő folyamat.

6.1.1. A NYOMTATÓ[7]

Az általunk használt nyomtató egy MakerBot Replicator Mini, mely árához képest remek nyomtatási minőséget biztosít. Ez a jelenleg 1500€-s 3D nyomtató, egyetlen nyomtatófejjel rendelkezik és maximum 10*10*12,5cm méretű objektum kinyomtatására képes. Azaz a tervezésnél a tárgyaink maximálisan 10cm szélesek, 10cm hosszúak és 12,5cm magasak lehetnek. Az ennél nagyobb elemeket, kisebb összeilleszthető darabokra kell szétosztanunk. A nyomtató által használt nyomtató segédszoftver .stl, .obj, .thing és . makerbot típusú 3D modellekkel képes együtt dolgozni, és ezeket nyomtatáskor 200 mikron-os rétegekben viszi fel a nyomtatási területre. A szoftver szerencsére minden főbb operációsrendszeren üzemel, így egyaránt alkalmazhatják a Windows-os, Linux-os és Mac OS-es felhasználók is. A beüzemelése nagyon egyszerű, csak USB-n keresztül fel kell csatlakoztassuk egy számítógépre, vagy Wi-fi kapcsolaton keresztül rá kell csatlakoztassuk egy nyomtatószoftverrel rendelkező eszközre és már meg is kezdhetjük a nyomtatást.

10. ábra - a MakerBot Replicator Mini 3D nyomtató

6.1.2. A NYOMTATÁSI SEGÉDANYAG[8]

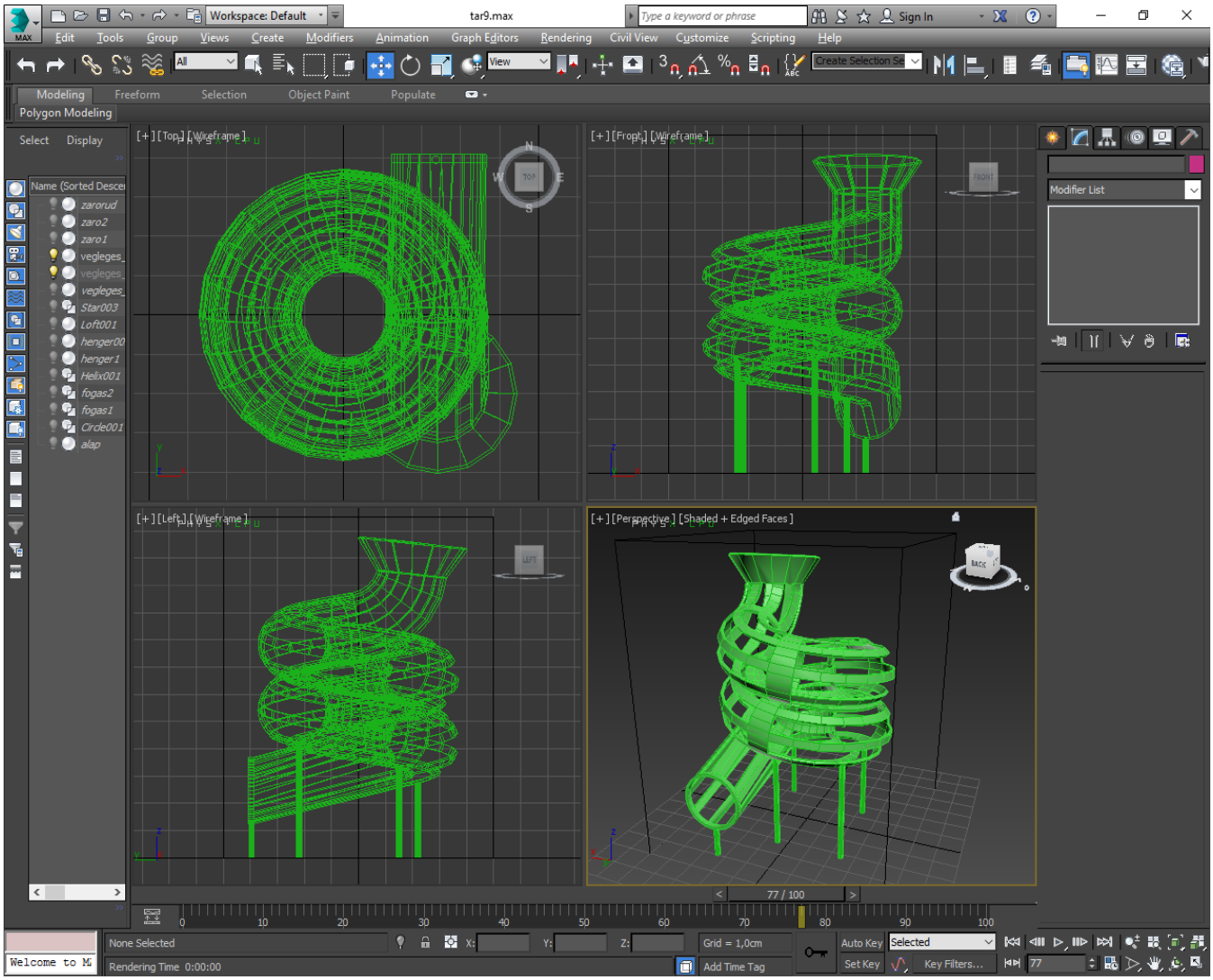
A MakerBot Mini által használható nyomtatási segédanyag 1,75mm átmérőjű szál átmérővel rendelkezik, melyeket orsókra feltekert kiszerelésekben vásárolhatjuk meg. Ezek többféle színben is a rendelkezésünkre állnak. Az alapszínű és a limitált színű tekercsek $18-ba, a speciális színűek $25-ba kerülnek jelenleg a gyártó oldalán. A Mini kizárólag kis tekercsméretű 1,25g/cm^3 sűrűségű, úgynevezett PLA anyaggal üzemel, így számításainknál ezekre az adatokra fogjuk alapozni a kilövendő lövedékeink méretezését.

11. ábra – 1,25g/cm^3 sűrűségű színes PLA filament

[7] https://eu.makerbot.com/shop/media/image/4a/ff/81/MP05925-Mini-Angled-Right-Shark-Jaw5581434b89b27_600x600.png
[8] http://store.makerbot.com/mb-images/store/filament/landing/spools.png

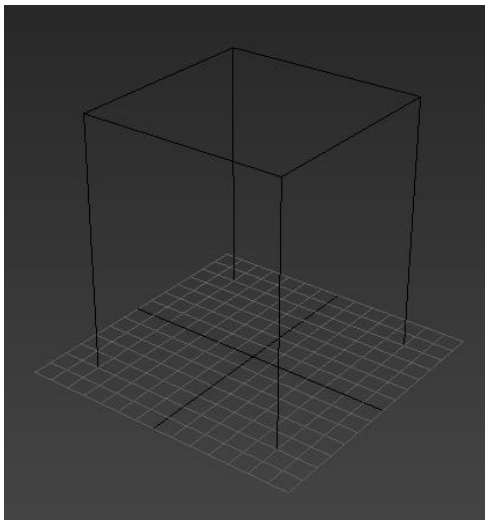
6.1.3. A TERVEZŐSZOFTVER ÉS A TERVEZÉS MENETE

A modellezés során a 3D Studio Max nevű 3D modellező, animációs és renderelő szoftvert használtuk, amely diákok számára ingyenesen elérhető az Autodesk által nyújtott Education license alatt. A modellező alapértelmezetten nem metrikus mértékegységekkel dolgozik. Hogy a későbbiekben a 3D nyomtató szoftver által kinyomtatandó modell és az ebben a fejezetben megtervezett modellező szoftverbeli elemek méretarányai azonosak maradjanak, át kell állítanunk a szoftver által kezelt mértékegységeket metrikus, centiméterben vett egységekre. Ezek után már minden készen áll a tervezés megkezdéséhez.



12. ábra - kép a 3D Studio Max környezetről

Már korábban említést tettünk a nyomtató fizikai korlátairól, melyeket figyelembe kell vegyünk. Az egyik ilyen legfontosabb tényező a nyomtatandó tárgy maximális 10*10*12cm-es mérete. Azért, hogy ez mindig a szemünk előtt legyen, és hogy megkönnyítsük a tervezést, létrehoztunk egy ilyen méret paraméterekkel rendelkező négyzet alapú hasábot, melyet a tervezési terület közepére helyeztünk. Ez a téglatest nem fog minket zavarni a tervezés során, hiszen átlátszóvá tettük a felszínét és csak az éleit hagytuk meg, így gyakorlatilag teljesen transzparens. Azért, hogy véletlenül se tudjuk elmozdítani pozíciójából, miközben a többi objektumot szerkesztjük, lefagyasztottuk. A lefagyasztott tárgyak pedig elmozdíthatatlanok és semmilyen méretezés nem hat rájuk. Ez a segédobjektum fogja számunkra arányaiban szemléltetni, hogy az éppen tervezett egység fizikailag még bele fér-e a nyomtatóba, vagy sem.

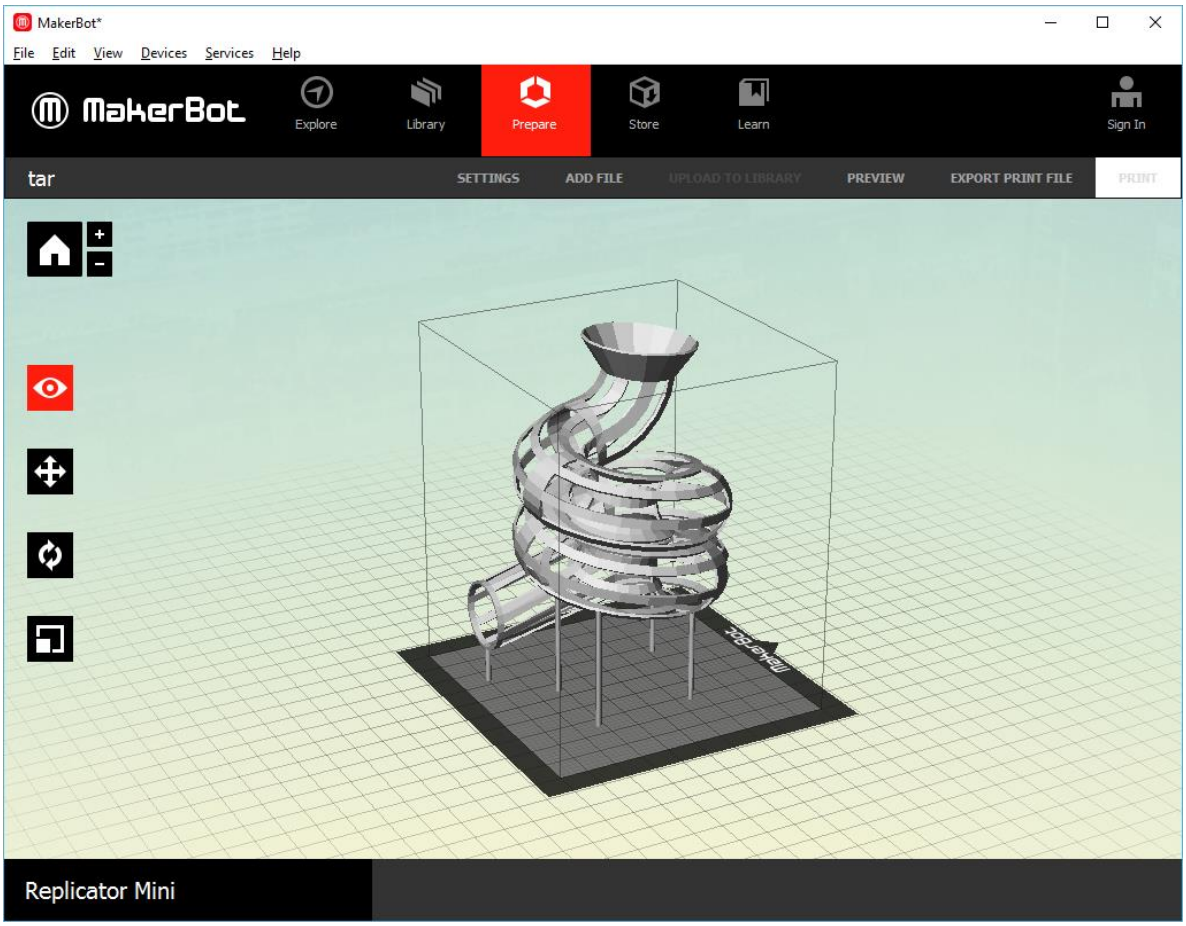


13. ábra - a tervezéshez felhasznált segédobjektum

A modellezővel elkészített objektumokat a folyamat végén STL, vagy OBJ formátumban exportálhatjuk ki, hiszen a nyomtató segédszoftvere kizárólag ilyen fájlformátumot képes beimportálni.

6.1.4. A NYOMTATÁS

A nyomtatáshoz a MakerBot Desktop nevű segédszoftvert fogjuk használni, melyet ingyenesen beszerezhetünk a gyártó weboldaláról. Egyszerűen csak hozzáadjuk a projekthez a nyomtatni kívánt STL állományt, elhelyezzük a nyomtatási területen, majd a kívánt beállítások megadásával pár kattintással már küldhetjük is nyomtatásra.



14. ábra - a MakerBot Desktop segédszoftver

6.2. Arduino

6.2.1. ARDUINO MEGA

Az érzékelők és a fegyverzet vezérlésére egy Arduino Megát választottunk. Azért erre a panelre esett a választásunk, mert teljesítményben lefedi az igényeinket. Mivel itt viszonylag sok érzékelőről van szó és minden érzékelőnek egyedi port szükséglete van, így a választásnál az egyik fontos tényező a portok száma volt. A Mega 54db digitális I/O porttal rendelkezik, melyből 15db PWM képes port, továbbá megtalálható még rajta 16db analóg bemenet is. A második fontos szempont, hogy beleférjünk a program memóriába, hiszen a komplex működés miatt a lefordított kódunk viszonylag igen nagyra is nyúlhat. A Mega 256kb flash memóriával rendelkezik, melyből már 8kb-ot elfoglal a bootloader, így ezzel a területtel gazdálkodhatunk. A Mega további előnyökkel is rendelkezik, ezekből ugyan nem mindegyikre van szükségünk, de a projekt esetleges bővítése során még jól jöhetnek. A Mega ATmega 2560-as 8bites, 16MHz-es AVR mikroprocesszorral van ellátva. Az UNO-hoz hasonlóan 5V-on működik, amire az Arduino-khoz készített modulok nagy része fel van készítve. Külső jack csatlakozós tápról is üzemeltethető az USB port mellett, ami nagyban megkönnyíti a hordozhatóságot. A külső táp ajánlott feszültsége 7-12V között van, de minimum 6V, maximum 20V amit tápként rákapcsolhatunk. I/O portjainak árama 20mA, a 3,3V-os lábán pedig 30mA áram folyik. Az UNO-ban csak 1db UART lehetőség van, ezzel szemben a Mega egy jobb választás a maga 4db UART-jával. Az alappanelt USB-n keresztül programozhatjuk. Egyéb kommunikációs lehetőségei közül még kiemelném az ICSP, SPI, és az I2C-t. Szintén nagy előnye az UNO-hoz képest, hogy 82kb statikus memóriával és 14kb eeprom-al rendelkezik.

15. ábra - az Arduino Mega alappanel[9]

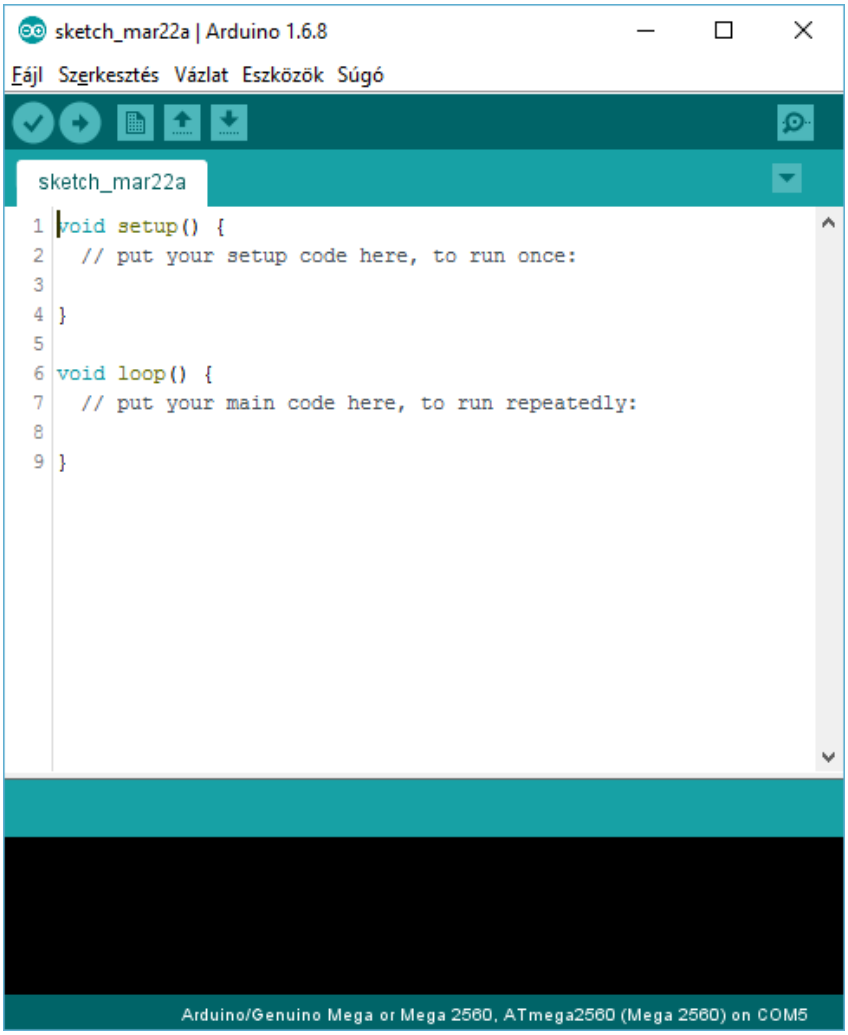
6.2.2. FEJLESZTŐI KÖRNYEZET

A fejlesztés során a legfrissebb rendelkezésünkre álló fejlesztői környezettel fogunk dolgozni. Az Arduino weboldaláról ingyenesen letölthető Arduino IDE jelenleg az 1.6.8-as verziószámnál tart. Az IDE grafikus felülete egyszerű használatra lett tervezve és szinte egy szimpla szövegszerkesztő program szintjére le van butítva. A program tartalmaz színes kulcsszó kiemelést, de a modern fejlesztői környezetektől eltérően nem tartalmaz sem automatikus kódkiegészítést, sem debug-golási lehetőségeket. Első használatkor, miután rácsatlakoztattuk alappanelünket a számítógépre USB kábel segítségével, még be kell állítanunk a szoftvert, hogy felismerje rácsatlakoztatott eszközünket. Ki kell választanunk a board alaplapját, processzorát és a COM portot, amire az eszköz felcsatlakozott. Ezek után meg is kezdhetjük a fejlesztést.

[9] https://www.arduino.cc/en/uploads/Main/ArduinoMega.jpg

6.2.3. FEJLESZTÉS AZ ARDUINO-RA

Az objektum orientált nyelvekkel ellentétben, itt nem gondolkodhatunk objektumokban. A hardver programozáshoz használt nyelvek általában rendszer-közeli nyelvek. Ezeknek a nyelveknek a használatakor elkerülhetetlen annak a célhardvernek a mély ismerete, amelyre fejleszteni szeretnénk. Az Arduino egy jól kiforrott termék, amely alaposan elfedi a hardver specifikus dolgokat ahhoz, hogy kezdők is alkalmazhassák, ugyanakkor meghagyja a lehetőséget az expertebb, a hardvereket jobban ismerő felhasználók számára. A nyelv elterjedtsége miatt számos harmadik fél által készített, ingyenesen felhasználható könyvtár áll rendelkezésünkre a fejlesztés során. Ezeket nem kötelező használnunk, de időt spórolhatunk azzal, ha valaki által már korábban megírt, többször is leteszteltt, megbízhatóan jól működő szoftvert használunk fel projektünknél ahelyett, hogy a nulláról kezdjük azt megvalósítani.

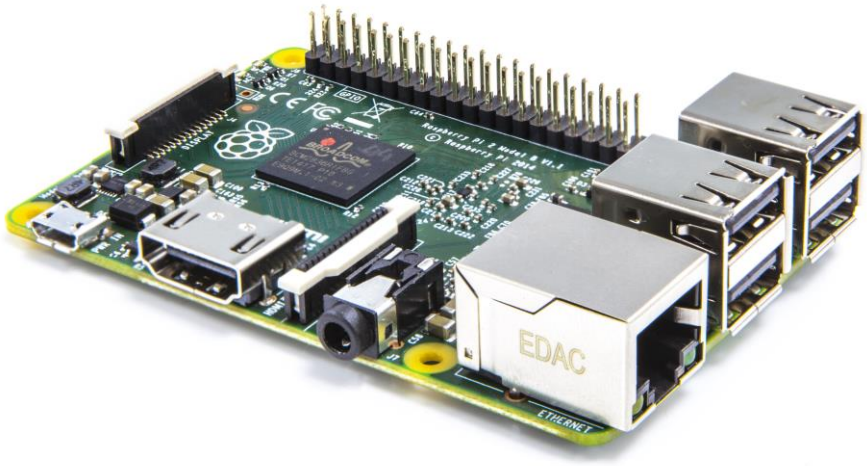


16. ábra - az Arduino Software IDE 1.6.8

6.3. Raspberry

6.3.1. RASPBERRY PI 2

A kamerakép lekezeléséhez egy komolyabb eszközre van szükségünk és a PI 2 több szempontból is jó megoldást jelent. Viszonylag gyors 900MHz-es 4 magos Boardcom BCM2836 Cortex-A7-es processzora van és 1Gb 450MHz-es statikus RAM-al rendelkezik, ami bőven meghaladja a kamerakép stream-eléséhez szükséges teljesítményt. Emellett egy beépített 2 magos Video Core IV multimedia co-processzorral rendelkezik, több csatornás HD audio-t kezel a HDMI felett és stereo audiót is a 3,5mm-es jack-en. Támogatja az operációs rendszerek használatát, mint például a Raspbian RaspBMC, Arch Linux, Rise OS, OpenELEC, Pidora, melyek megkönnyítik a stream-elést. Segítségükkel viszonylag magas szinten is megoldhatjuk a kamerakép feldolgozását és a wifi-n keresztül történő kommunikációt, az alacsony szintű hardver közeli kódolás helyett. Ezek mellett a PI szükség esetén MicroSD kártyával is bővíthető, 4db USB 2.0-ás porttal és 40db GPIO porttal rendelkezik. Van benne beépített 10/100Mbit-es RJ45-ös csatlakozású Ethernet modem. A táplálását pedig az 5V-os 800mA-es MicroUSB-porton keresztül oldhatjuk meg. Egyébként rendelkezik HDMI kimenettel, amely képes 640*350 / 1920*1200 és 1080p-s felbontásra is.



17. ábra - Raspberry Pi 2 alappanel[10]

6.3.2. OPERÁCIÓSRENDSZER

A Raspberry PI 2-höz számos operációsrendszer áll rendelkezésünkre. Ezek közül a legelterjedtebbek a Raspbian, RaspBMC, Arch Linux, Rise OS, OpenELEC és a Pidora. Választásunknál igyekeztünk olyan operációs rendszert választani, amelyik a legjobban dokumentált és nagy felhasználói bázissal rendelkezik. Ennek köszönhetően, ha bármilyen problémába ütköznénk a fejlesztés során, úgy véljük, hogy, könnyebben megoldásra lelhetünk. Éppen ezért választásunk egy Linux alapú operációs rendszerre esett, amelyik egy Debian Linux-ról lett átportolva a Raspberry architektúrájára. A portolásra azért volt szükség, mert a Raspberry ARMv6 és v7 architektúrája nem támogatott a Debian alatt, hiszen a Debian ARM kiadása csak egy későbbi verziószámtól támogatja hivatalosan az ARM processzorokat. Az általunk használni tervezett Raspbian egyetlen hátránya, hogy bizonyos speciális processzor utasításokat nem támogat, illetve a lebegő pontos számításokat intenzíven alkalmazó programoknál a PI teljesítménye gyengébb.

18. ábra - a Raspbian operációs rendszer emblémája[11]

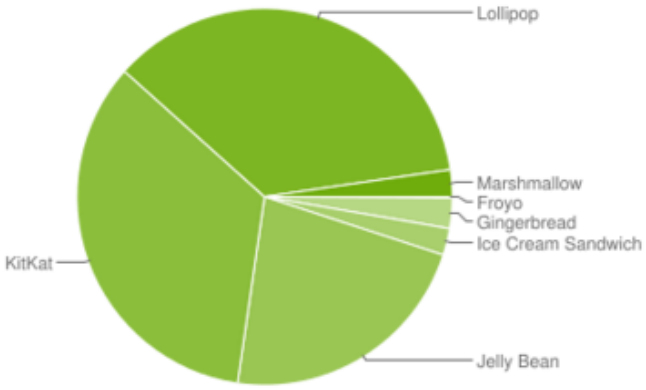
[10] https://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg
[11] https://www.raspbian.org/static/common/raspbian_logo.png

6.4. Android

6.4.1. ANDROID 4.0-6.0

A fegyverrendszert úgy terveztük, hogy bizonyos funkcióit távolról is el tudjuk érni. Ilyen funkciók például a kamerakép és a radarkép megjelenítése, a fegyverzet távolról való irányítása és az észlelésekkel kapcsolatos értesítések megjelenítése. Mivel napjainkban minden modern ország majdnem minden lakosa rendelkezik valamilyen okos eszközzel, és mivel a leggyakrabban ezek közül használatos eszközök a mobiltelefonok és tabletek, ezért nekünk is az okos mobiltelefonokra esett a választásunk. Ezeknek a telefonoknak nagy százaléka Wifi-képes és rendelkezik beépített Bluetooth modullal, melyek megkönnyítik számunkra az eszközök közötti kommunikáció megvalósítását. A rendszerünk prototípusának megalkotásához nem tartjuk fontosnak az összes főbb mobil platform lefedését, így az egyik legelterjedtebb rendszerre esett a választásunk, az Androidra. Az Android rendszer folyamatosan fejlődik, újabb és újabb verziószámú kiadásai jelennek meg sorra. Ezek a rendszerek egymástól lényeges eltéréseket mutatnak, melyeket kompatibilitási megoldásokkal oldhatunk meg. Minden verziót lefedni azonban nagyon bonyolult feladat és nem is éri meg az energia és idő befektetést. A rendszert használók teljesen szétszórva, más és más arányban használják ezeket a mobil operációs rendszereket. Nekünk fejlesztőként ennek a képzeletbeli tortának olyan szeletét kell kiválasztanunk, amelyik a legtöbb potencionális felhasználót magában foglalja és ehhez képest a lehető legkevesebb időt veszi el tőlünk a benne található rendszerváltozatok kompatibilitási problémáinak feloldása. Tervezésnél mi az Android 4.0-ás operációs rendszerekig szeretnénk támogatni a platform felhasználóit, így a minimum API levelként a 14-eset választottuk ki. Napjaink legnagyobb teljes értékű Android verziója a 6.0, melyhez a 23-as API level tartozik. Kijelenthetjük, hogy terveink szerint a 14-es szinttől a 23-as szintig szeretnénk minden Android felhasználót támogatni.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

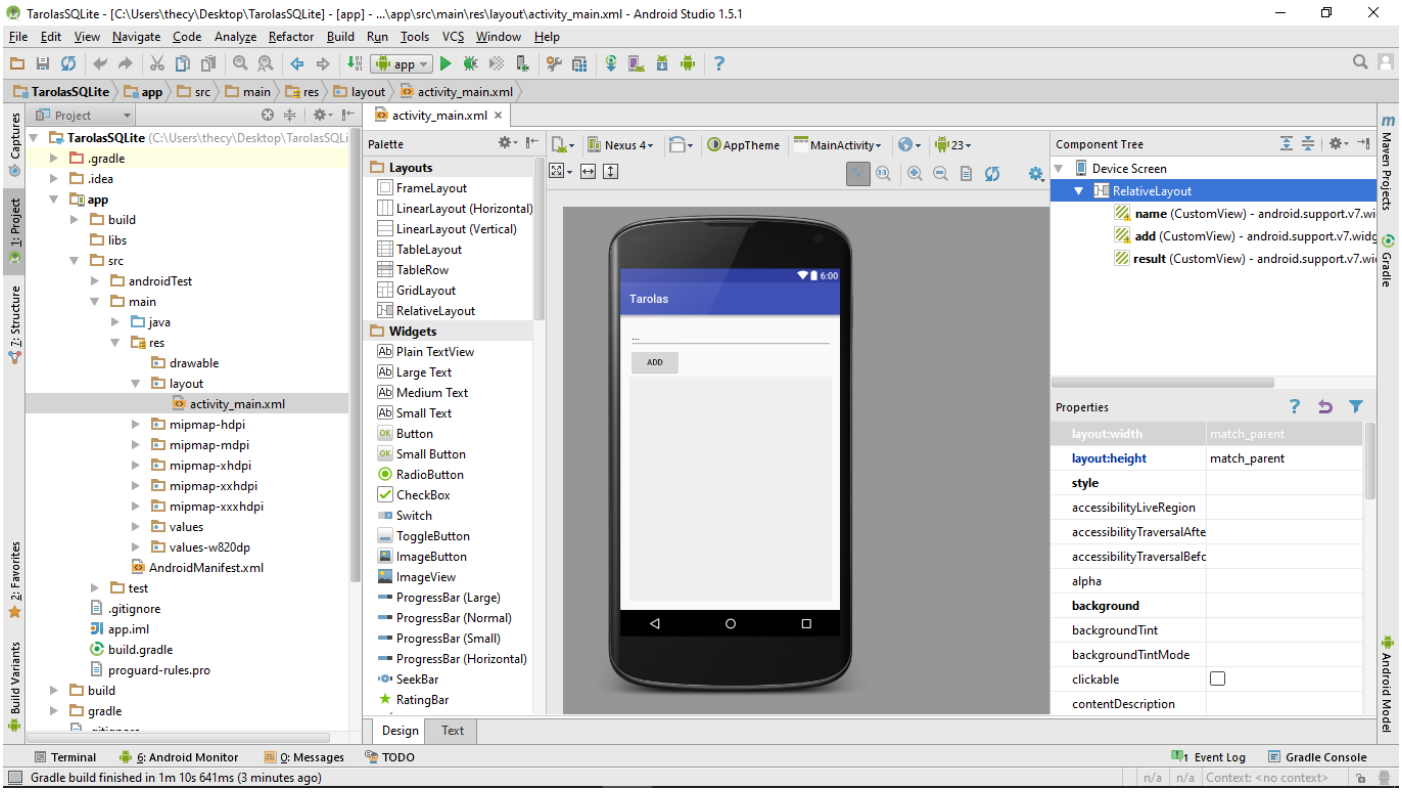


19. ábra - az Android verziók használatának százalékos eloszlása[12]

[12] http://developer.android.com/about/dashboards/index.html

6.4.2. FEJLESZTŐI KÖRNYEZET

Az Androidra való fejlesztéshez a Google egy remek fejlesztői környezetet tesz elérhetővé számunkra, melynek a neve Android Studio. A szoftver általunk használt változatának verziószáma az 1.5.1. Ez a fejlesztőkörnyezet az IntelliJ IDEA java-s fejlesztőknek szánt szoftverének Androidra való fejlesztéshez átalakított változata. Nagyon megkönnyíti számunkra a fejlesztést. Támogatja az automatikus kódgenerálást, az automatikus kódkiegészítést, az automatikus kódformázást, a különböző emulált eszközökkel való tesztelést, debug-olási lehetőséget és még számtalan hasznos funkciót. A fejlesztői környezet könnyedén összekapcsolható verziókezelő rendszerekkel, amely megkönnyíti számunkra a csapatmunkát és a szoftverünk verzióinak menedzselését. Projektünknél mi a GIT nevű szétosztott rendszert fogjuk használni, amely talán a világ egyik legelterjedtebb verziókezelő szoftvere. Az Android Studio egy másik fő előnye a beépített automatikus projektépítő eszköz, amely lekezeli a projektünk függőségeit, a Gradle. Ez jelentősen megkönnyíti számunkra az olyan folyamatok futtatását melyek a projektünk felépítésekor automatikusan elvégezhetőek.



20. ábra - Android Studio 1.5.1

6.4.3. TESZTELÉS

A mobil applikáció tesztelését valós eszközökön valósítjuk meg. Az Android Studio jelenleg beépített emulátora nem biztosít lehetőséget a Bluetooth és Wifi kapcsolat teszteléséhez, így ezeknek a moduloknak a kipróbálását és leellenőrzését mindenféleképpen valódi készülékeken kell letesztelni. Ezen felül a beépített emulátor túl lassan működik, így ettől függetlenül is javasolt alkalmazásaink valós környezetben való tesztelése. A tesztelésnél célszerű minél több eszközön megvizsgálni a szoftverünk funkcionális működését és grafikai megjelenítését, hiszen az eltérő rendszer verziókból és képernyő felbontásokból adódó hibákat így nagyobb valószínűséggel leszünk képesek detektálni. A teszteléshez mi egy Samsung Galaxy S4 LTE mobiltelefont és egy Sony Xperia Z3 Compact okos telefont szeretnék felhasználni.

*21. ábra - Samsung Galaxy S4 LTE[13] - Android 5.0.1, 1080*1920, 5”*

*22. ábra - Sony Xperia Z3 Compact [14]- Android 5.1.1, 720*1280, 4,6”*

6.5. Modul szükségletek

Rendszerünk megvalósításához elengedhetetlenül szükségesek bizonyos érzékelők, illetve a rendszert mozgató szervók és motorok. Ezeket mi előre elkészített modulok felhasználásával szeretnénk megvalósítani. Így a prototípus elkészítésénél megspórolunk némi időt és pénzt az elektronikai tervezés és a legyártatás folyamatának mellőzésével.

6.5.1. 3D NYOMTATOTT ELEMEK

- 1db spirál tár
- 2db kapu
- 2db henger
- 3 db golyó
- 1db kör alakú alappanel

6.5.2. FEGYVERZET ALRENDSZER

- 2db szervo (kapukhoz), típus: sg90
- 2db léptető motor (360° a hengerekhez), típus: jq24-125H670
- 1db léptető motor (az alappanel mozgatásához), típus: 28byj48-5v
- 1db akkumulátor, típus: Gardena 12V 3Ah NiMH
- 1db relay modul, típus: Tianbo 05vdc

[13] http://samsung-updates.com/wp-content/uploads/2013/09/samsung-galaxy-s4.jpg
[14] http://api.sonymobile.com/files/xperia-z3-compact-black-1240x840-2f1d546fc795ff2d1295547982a23cb4.jpg

6.5.3. ÉRZÉKELŐ ALRENDSZER

- 1db szervo, típus: sg90
- 1db ultrahang, típus: hc-sr04
- 1db mozgásérzékelő, típus: hc-sr501

6.5.4. KAMERA ALRENDSZER

- 1db léptető motor, típus: 28byj48-5v
- 1db kamera, típus: ov-5647

6.5.5. KOMMUNIKÁCIÓS ALRENDSZER

- 1db bluetooth 4.0 ble, típus: hm-10 (Arduino)
- 1db wifi adapter, típus: Realtek rtl8188cus (Raspberry)

6.5.6. SÉRÜLÉS DETEKTÁLÓ ALRENDSZER

- 1db gyroscope & gyorsulásmérő, típus: mpu-6050

6.5.7. ILLUSZTRÁCIÓK

23. ábra - sg90 szervo

24. ábra - jq24-125H670 12V DC motor

25. ábra - 28byj48 5V DC motor

26. ábra - hc-sr04 ultrahang szenzor

27. ábra - hc-sr501 mozgás érzékelő

28. ábra - ov5647 kamera modul

29. ábra - hm-10 bluetooth modul

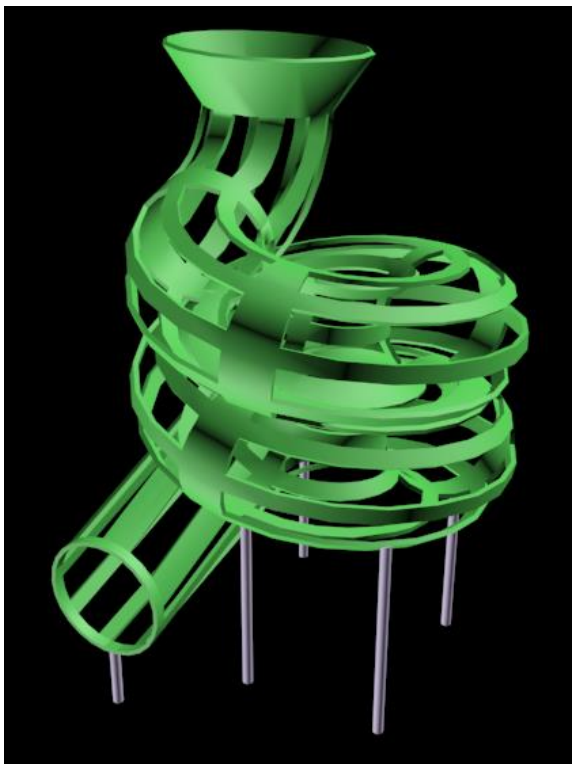
30. ábra - rtl8188cus wifi modul

31. ábra - mpu-6050 gyroscope és gyorsulásmérő

7. IMPLEMENTÁCIÓ

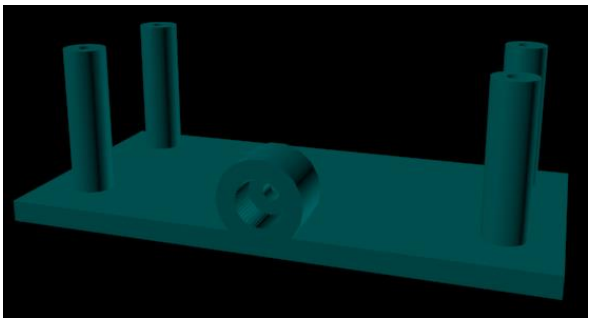
7.1. 3D nyomtatott alkatrészek

Projektünk implementációs szakaszát a 3D nyomtatott alkatrészek megvalósításával kezdtük. Terveinkkel ellentétben nem a MakerBot Mini 3D nyomtatót használtuk, mert túl költséges lett volna vele a megvalósítás. Több itthoni és kínai gyártótól kértünk árajánlatot és úgy vettük észre, hogy napjainkban mivel a 3D nyomtatás még egy viszonylag újszerű technológiának számít, nincs nagy különbség a két ország árazása között. A legdrágább alkatrész a tárnak bizonyult, mely a magyarországi árajánlatok szerint 26.250Ft körüli összegből lett volna megvalósítható. Ez természetesen csak egyetlen alkatrész költsége, a projekt megvalósítása körülbelül 200.000Ft körüli összegből lenne megvalósítható. Mivel árajánlatot nem az összes alkatrészre kaptunk, csak pár objektumra, ezért ez az összeg csak egy becsült érték, ami szerintünk jól megközelíti a valóságot.

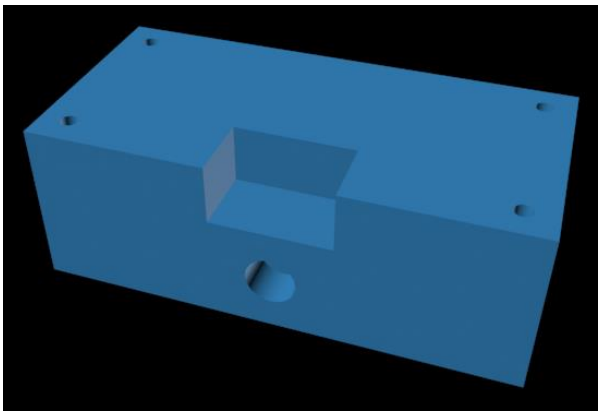


32. ábra - A legnagyobb költségű elem: 26.250Ft-ból megvalósítható

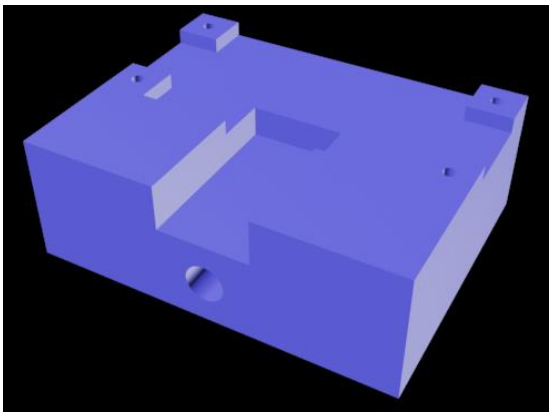
Ez a végösszeg túl nagy kiadás lett volna számunkra, így egy másik alternatívát kellett keresnünk. Ha valaki a környezetünkben rendelkezne saját 3D nyomtatóval, akkor a bérnyomtatás helyett egy jóval kedvezőbb megoldás lenne tisztán anyagköltségből megvalósítani a nyomtatást. Egyetemünk rendelkezik saját 3D printerrel és ingyenesen elérhetővé tette számunkra a nyomtató használatát. Mivel az alapanyagot is egyetemünk állta, ezért nekünk a nyomtatás egy nullszaldós költségvetéssel készülhetett el. Ám ez a választás minőségbeli és technika kompromisszumokkal járt, melynek során többször is át kellett terveznünk alkatrészeinket. Egyetemünk alsóbb kategóriás, gyengébb minőségű nyomtatókkal rendelkezik, mely oktatási célokra nagyon jól megfelel, ám az általunk készített tervek túl komplexek és megvalósíthatatlanok egy ilyen nyomtató számára. A legnagyobb problémát, a magas henger alakú kiállások és testek azon kiugró részei jelentették, melyek nem érintkeznek közvetlenül a nyomtatási felülettel.



33. ábra - Ultrahang szenzortartó szerkezet - 1. verzió

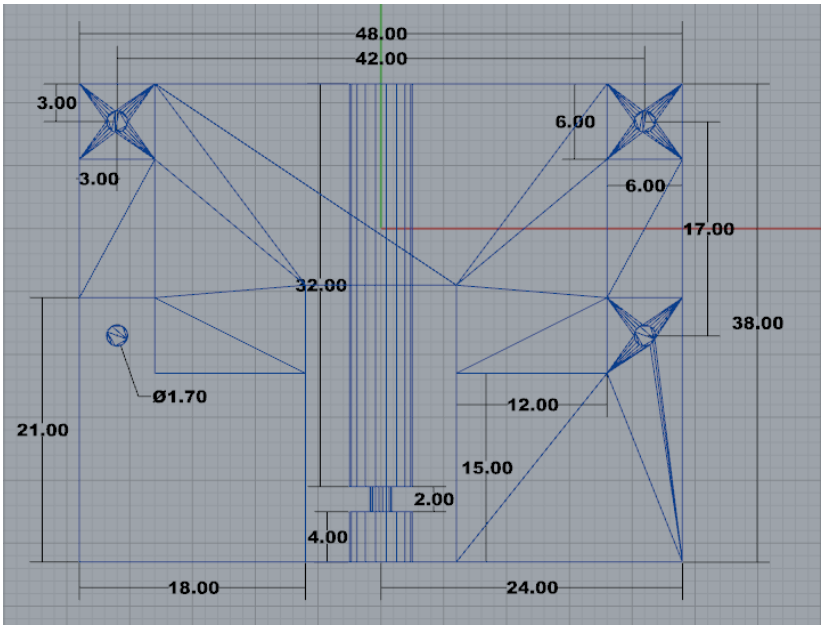


34. ábra - Ultrahangtartó szerkezet - 2. verzió



35. ábra - Ultrahangtartó szerkezet - 3. verzió

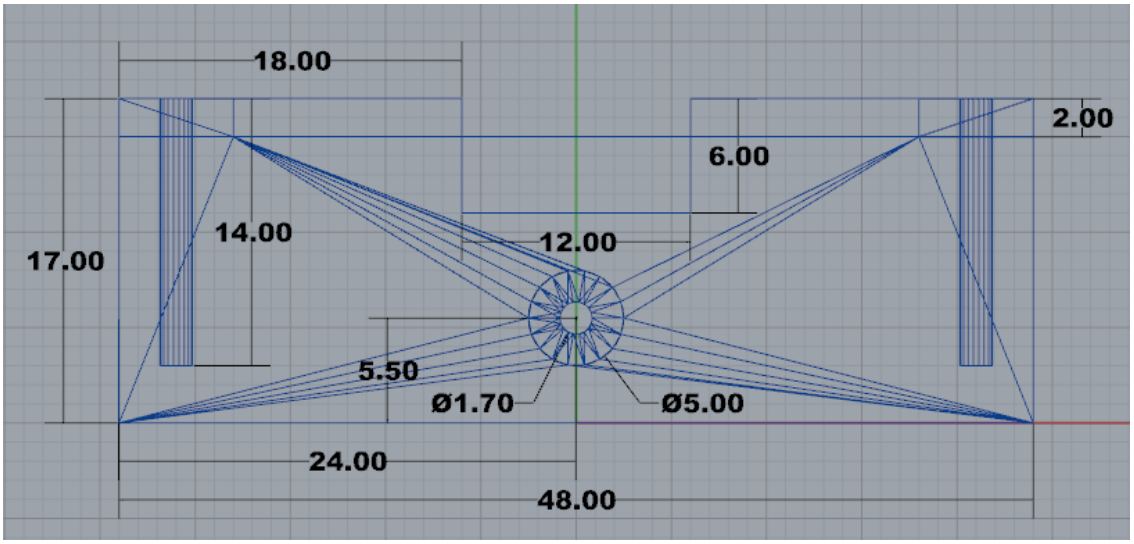
A 33. ábrától a 35. ábráig tartó 3D-s objektumokból renderel-t képek az ultrahang szenzortartó szerkezetének három lépcsős verziófejlődését szemléltetik. Az ábrákon jól érzékelhető, hogy a gömbölyded alakzatokat hasábosabb kialakítású, tömörebb tárgyakra kellett átalakítanunk, melynek során a nyomtatott elemek tömege kicsit megnőtt. Ennek az átalakításnak a segítségével a nyomtatás során, az alsóbb rétegeknek van ideje addig megszilárdulni, amíg a felsőbb rétegekhez eljutunk, így az alakzatok nem fognak szétfolyt végeredményt eredményezni.



36. ábra - Felül nézeti CAD terv az ultrahangtartó szerkezetről

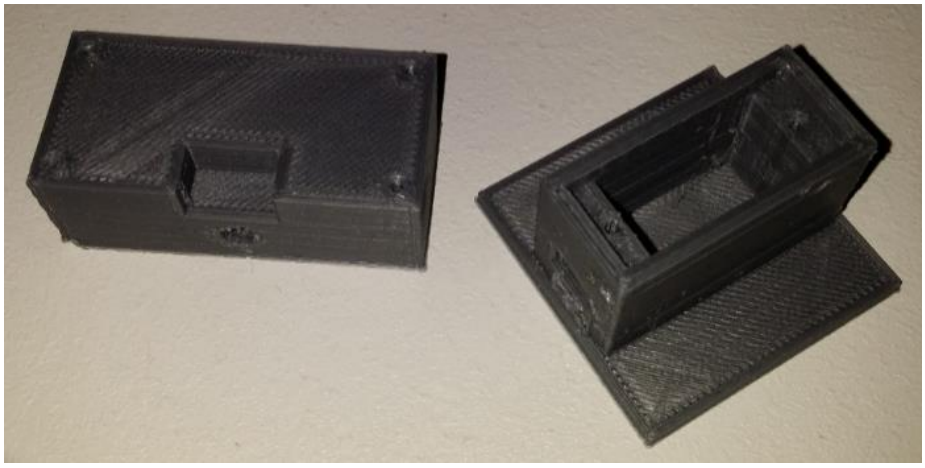
A 36. ábrán lévő CAD-es tervrajzon megfigyelhető, hogy a csavar furatok középpontját az objektum széleitől 3mm távolságra helyeztük beljebb, így a nyomtatás után biztosak lehettünk abban, hogy a csavarok becsavarása közben nem fog elpattanni, vagy letörni az enyhén merev műanyag a széleken. Továbbá megfigyelhető, hogy a mértékegységek tizedes pontosságúak, ugyanis az egyetemen található nyomtató gyengébb minőséget eredményez az eredetileg tervezettnél.

A 37. ábrán látható, hogy a testből legkisebb nyomtatandó kiállás 2 milliméter nagyságú. Az ettől kisebb önmagában álló nyomtatandó terület a nyomtatás során már deformációt eredményez a végeredményben. Végkövetkeztetésként levonhatjuk, hogy az általunk használt nyomtató, nem csak a túl nagy, hanem a 2-3mm-nél kisebb kiálló részletekkel is hibás végterméket bocsájt ki magából.

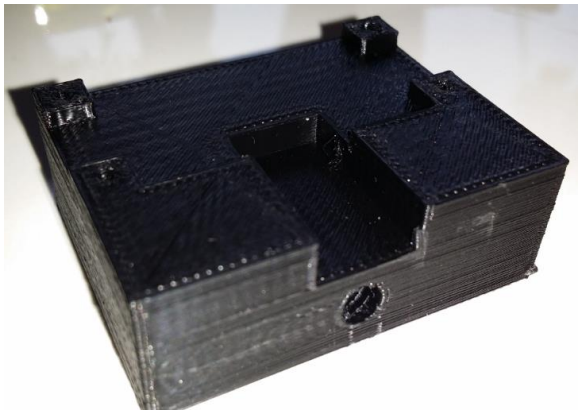


37. ábra - Elöl nézeti CAD terv az ultrahangtartó szerkezetről

Az elkészült végtermékek felülete bordázott, felszínükön a nyomtatás minőségéből fakadó, a nyomtatásból visszamaradt kiálló műanyagdarabkák találhatóak. Ezek a megfelelő célszerszámmal pillanatok alatt eltávolíthatóak és az elemek használhatóságát nem befolyásolják.



38. ábra - A 3D nyomtatással készített ultrahang- és szervo-tartó szerkezet



39. ábra - A kinyomtatott ultrahangtartó szerkezet - 3. verzió

7.2. Kommunikációs alrendszer

7.2.1. BLUETOOTH KOMMUNIKÁCIÓ

A fegyverrendszerben elhelyezett bluetooth kommunikáció, az Arduino-ra csatlakoztatott szenzorok által mért értékek továbbítására szolgál. A megvalósítás során egy bluetooth 4.0-át támogató modult használtunk fel, amely képes az úgynevezett bluetooth low energy technológia megvalósítására. A BLE technológia lehetővé teszi számunkra, hogy eszközünk a lehető legkisebb mértékben terhelje az akkumulátorokat. A modul adatátvitel közben csupán 8.5mA-t, készenléti állapotban pedig körülbelül 400uA és 1,5mA között veszi fel az áramot. Ez azt jelenti, hogy egy átlagos alkáli elem, amelynek 2700mAh-ás kapacitása van több mint 317 órán keresztül képes táplálni bluetooth modulunkat folyamatos adatátvitel mellett a legnagyobb 8,5mA-es áramfelvételt feltételezve.

$$\frac{2700mAh}{8{,}5mA} = 317{,}65h$$

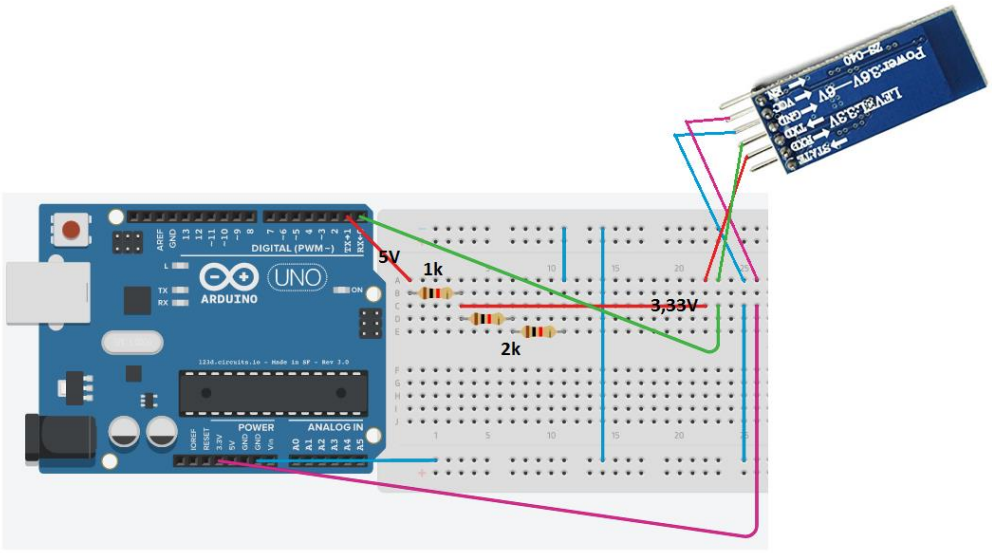
1. képlet – A bluetooth modul maximális futásideje átlagos alkáli elemet feltételezve

Az általunk használt HM-10-es bluetooth modul üzemfeszültsége 3,3V. Ezzel szemben az Arduino Mega 5V-on üzemel, így elsődleges problémánk a jelszintek átalakítása. Az Arduino a soros kommunikáció során 5V-os magas jelet küldhet, ami tönkreteheti a bluetooth-t. Azonban egy feszültségosztó ellenálláspár segítségével 3,3V-os jelszintre hozhatjuk a bemeneti 5V-os jelet. Az egyszerűség kedvéért mi egy 1 kilóohmos és egy 2 kilóohm-os ellenállással számoltunk. A megvalósítás során a 2kOhm-os ellenállást 2 darab 1kOhm-ossal helyettesítettük. A feszültségosztó képlet használatával leellenőrizhető, hogy ez az értékpár helyes feszültségszintet eredményez: 3,3V=5,5V*(2kOhm/1kOhm).

$$Uki = Ube * (\frac{R2}{R1 + R2})$$

2. képlet – A felhasznált feszültségosztó képlet

A visszafelé irányuló kommunikáció során a bluetooth modul csupán 3,3V-os jelszintet produkál, de az Arduino 5V-osat vár. Az Arduino által magas jelnek akkor tekinthető egy érték 5V-os jelszint esetén, ha az legalább 3V-os. A bluetooth által küldött 3,3V-os jelszint esetén pedig, akkor ha az legalább 2V-os feszültségű. A szintkülönbségből adódóan az Arduino minden olyan magas jelet, tévesen alacsonynak fog értékelni, ami a bluetooth jelszintjével mérve ugyan magas, de a 2 és 3V-os feszültség intervallum közé esik. Stabil jelszintet feltételezve ez nem okozhat gondot a két eszköz között. Éles tesztünk során sem ütköztünk adatvesztéses problémába. Azonban, hogy minden kétséget kizáróan biztos működést eredményezzen a rendszer, egy fet közbeiktatásával felhúzhatjuk az alacsony jelszintet 5V-os jelszintre.



40. ábra - A jelszintek biztosítása feszültségosztó ellenálláspár közbeiktatásával

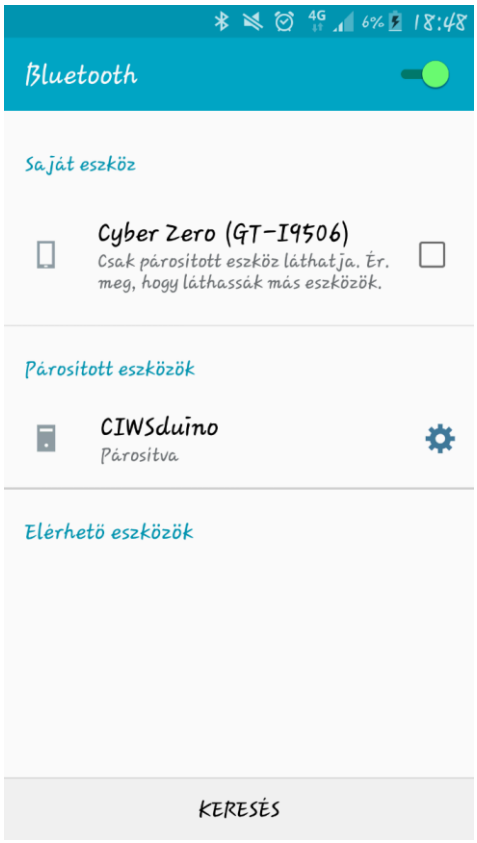
Első használatkor az eszköz felkonfigurálásához egy soros porton kommunikáló terminál szükséges. A beállítások során mi az Arduino IDE által biztosított Serial Monitort alkalmaztuk. A konfigurálás úgynevezett AT parancsok segítségével történik, melyhez egy részletes segítséget nyújt a HM-10-es bluetooth dokumentációja. A baud ráta és a kommunikáció alapjáraton 9600baud-ra és 8N1-re van beállítva. Nekünk ez a 8-bites adat, paritás bit nélkül, 1 lezáró bittel a teszteléshez megfelel, így ennek átállítására nincs szükség. Elsőként beállítjuk a bluetooth kapcsolódáshoz használatos nevet CIWSduino-ra. Ezt követően szükséges az alapértelmezett PIN kód átállítása egy egyéni kódra. Esetünkben ez az 198708-as számsorozat.

```
Terminalban kuldjuk: AT+NAMECIWSduino
Bluetooth valasza: +NAME=CIWSduino
OK

Terminalban kuldjuk: AT+PASS198709
Bluetooth valasza: +PASS=198709
OK
```

41. ábra- A bluetooth konfigurálása AT parancsokkal

A beállítás utolsó szakaszában át kell állítanunk a kapcsolódás típusát a párosítás és pin kód használata kombinációra, hogy a modul képes legyen az adatcsomagok fogadására a 2,4GHz-es frekvencián. Ennek a végső parancsnak az alkalmazása után, célszerű újraindítanunk a modult a későbbi zökkenőmentes működése eléréséhez. Újraindítás után az eszköz használatra készen áll, megkísérelhetjük a mobiltelefonról történő csatlakozást. Androidon válasszuk ki az elérhető bluetooth eszközök közül a CIWSduino nevű kapcsolódási pontot, és adjuk meg a korábban beállított 198709-es jelszót a párosításhoz. Sikeres párosítás esetén az Android jelzi számunkra a kapcsolódás tényét, melyek után megtörténhet a tényleges adatátvitel.



42. ábra - Androidos képernyőkép a bluetooth párosításáról

7.2.2. WIRELESS KOMMUNIKÁCIÓ

A Wi-Fi kapcsolat a Raspberry Pi és a kliensek között a Raspberry-re kötött videó streamelésére szolgál. A kapcsolat létrejöttéhez a szervernek (Raspberry) illetve a klienseknek azonos alhálózaton kell lenniük. Hogy a kapcsolódás egyszerű

legyen a felhasználó számára az implementáció során fontos szempont volt, hogy az eszközök zéró konfigurációval megtalálják egymást a hálózaton és megkezdjék a streamelést. Mivel azt szeretnénk, hogy a rendszer a lehető legkisebb mértékben függjön az adott alhálózat beállításaitól, ezért nem hagyatkozhatunk arra, hogy a csatlakozott eszközök statikus ip-címet kapnak. A kliensnek és a szervernek meg kell találniuk egymást a hálózaton annak ismerete nélkül. A feladatot az UDP protokoll segítségével oldjuk meg. A kliens eszközök a hálózatra való csatlakozás után bizonyos időközönként UDP csomagokat küld a hálózat broadcast ip címére egy előre meghatározott porton. Ezzel a módszerrel, a hálózat összes működő eszköze megkapja a csomagot. A szerveren egy Node.js-ben megírt alkalmazás figyeli a beérkező UDP csomagokat és amint talál egy csomagot, amely az egyik kliens küldött, válaszol egy kifejezetten a kliensnek célzott üzenettel. Miután a kliens megkapta a választ a szervertől nincs szükség tovább UDP csomagokat küldeni. Ezzel a módszerrel kerüljük el, hogy a szervernek kelljen broadcastolnia az ip címét, ami hálózati szempontból nem javasolt (UDP flooding, DoS), valamint a Wi-Fi multicast lock feloldását az Android rendszeren. Az Android rendszer alapértelmezetten blokkol minden csomagot, amit nem explicit az eszköznek címeztek ezért a broadcast üzenetek fogadásához szükséges a Wi-Fi multicast lock feloldása, ami nem javasolt.

Ha nem érhető el hálózat az eszköz telepítésének helyén, a Raspberry konfigurálható Wi-Fi hozzáférési pont létrehozására is a hostapd (Wi-Fi modul konfigurálása AC módba) és dnsmasq (DNS és DHCP szerver) eszközök használatával. A raspbian újabb verzióiban a hálózati interfészek konfigurálásáért a dhcpd.conf file a felelős. A dhcpd.conf és a wpa_supplicant.conf file megfelelő szerkesztésével engedélyezhetjük a Wi-Fi interfész hozzáférési pont üzemmódját és valamint konfigurálhatjuk a hálózatot.

Ezen módszerek használatával a kamera alrendszer egyszerre több különböző eszközről is elérhető akár zéró konfiguráció mellett, valamint bármilyen kliens rendszeren használható lesz, hiszen csak egy Wi-Fi és stream lejátszására képes eszközre van szükség (Okostelefon, Tablet, Személyi számítógép, TV, stb.)

7.3. Érzékelő alrendszer

7.3.1. ULTRAHANGOS RADAR

Az általunk megépített ultrahangos radar egy ultrahang szenzorból és egy szervóból áll, melyeket 3D nyomtatási technológiával készített alkatrészek kötnek össze. A szervo szerkezet a radar mozgatásáért felelős. Az SG90-es szervót 180°-os szögtartományban alkalmazhatjuk és pulzus modulált jelek segítségével állíthatjuk a kívánt szögbe. Programozáskor az Arduino beépített Servo library-ját használtuk feladatunk megkönnyítésére. Használata során létre kell hoznunk egy Servo típusú példányt, majd ehhez a példányhoz hozzá kell rendelnünk az Arduino-nak azon PWM képes lábát, melyhez a szervo vezérlőjele tartozik. A radarhoz tartozó Arduino sketch-ben a szervót úgy állítottuk be, hogy fokonként léptetődjön és minden egyes szögben addig várakozzon, amíg az ultrahangos szenzor mérései be nem fejeződtek. Az SG90-es szögsebessége 0,1sec/60°, ami azt jelenti, hogy 0,1 másodperc kell ahhoz, hogy 60°-ot forduljon a szervo. Ez azt jelenti, ha képes lenne 360°-ot fordulni, akkor, amíg ezt a fordulatot megteszi, addig 600ms tellene el.

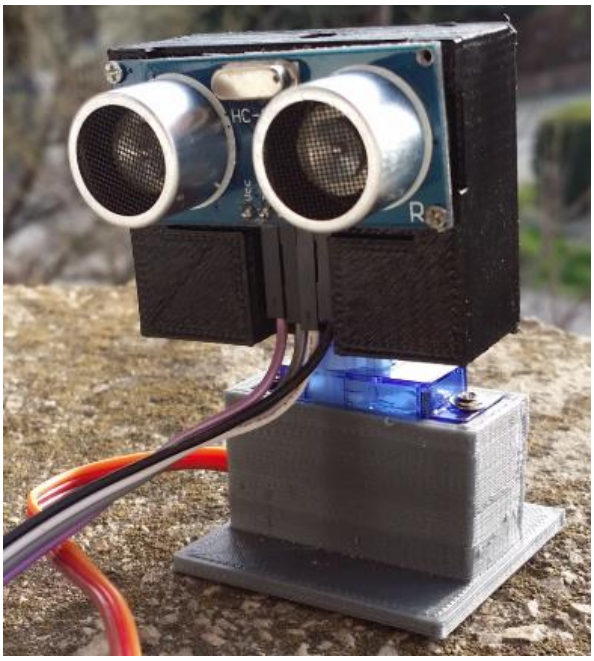
$$0{,}1s * \left(\frac{360}{60}\right) = 0{,}6s$$

3. képlet – A 360° megtételéhez szükséges idő kiszámításának képlete

Ebből kiszámítható, hogy ahhoz, hogy 1°-ot el tudjon mozdulni, körülbelül 2 milliszekundumra van szükség. Ez lesz a radar mozgatásából adódó késleltetése.

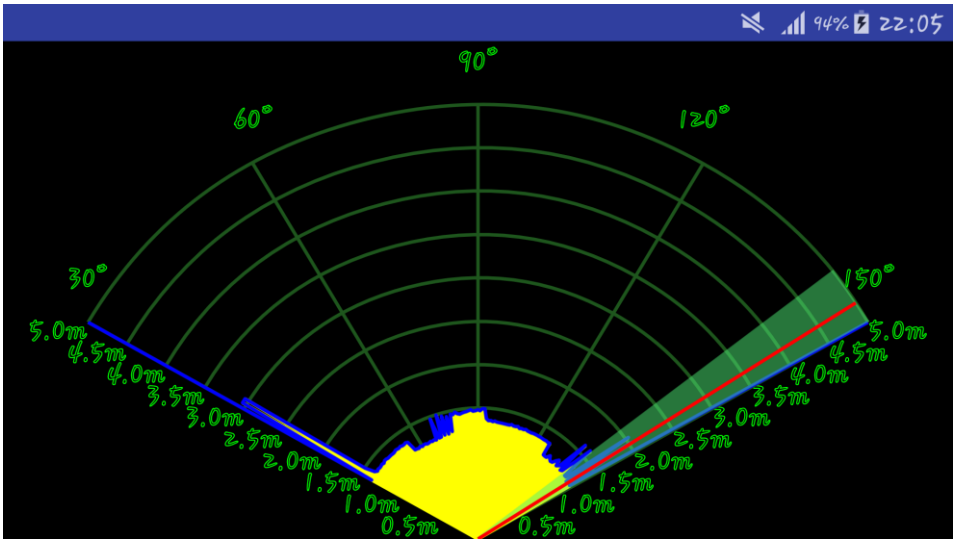
$$\frac{600ms}{360°} = 1{,}67ms$$

4. képlet – A mozgásból adódó késleltetés kiszámításának képlete



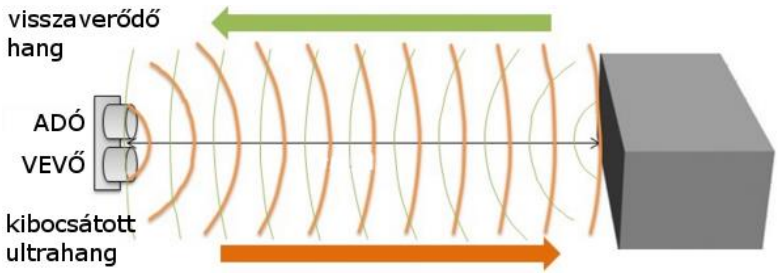
43. ábra – Fénykép az ultrahangos radarról

A HC-SR04 típusú ultrahang szenzorunk egyetlen ultrahang adóból és egy ultrahang vevőből áll. Adatlap szerint 2cm-től 500cm-ig tartó tartományban képes érzékelni a tárgyakat. Ám ezt erősen befolyásolja az adott környezet és a tárgyak elhelyezkedése. Az ultrahang jelek visszhangos zavarai ugyan csökkenthetőek az adóra és a vevőre szerelt kis papírtölcsérek alkalmazásával, de az ilyen olcsóbb szenzorok nagy erőfeszítések árán sem fognak tökéletesen működni. A mért értékek között megjelenhetnek olyan a többitől erőteljesen kiugró mérési adatok, melyek az ultrahang jelek magas fokú pontatlanságából adódnak. Ezeknek az értékeknek a számát a mérések során alkalmazott medián szűrő segítségével redukálhatjuk le. Medián szűrő alkalmazása mellett, egyidejűleg több mérést is elvégzünk, melyeket távolság szerint sorba rendezünk. Végezetül a rendezés után előállt elemsor középső értékét vesszük eredményül.



44. ábra - Képernyőkép az Android-os radarképről

Arduino alatt a Tim Eckel által 2012-ben kifejlesztett NewPing nevű könyvtárat használjuk fel az ultrahang szenzor lekezelésére. Ez a könyvtár lehetővé teszi számunkra egy olyan függvény használatát, amely alapértelmezetten 5 mérésből számított értéket ad számunkra eredményül a korábban említett medián szűrő alkalmazásával. Az ultrahang jelsorozatoknak oda-vissza meg kell tenniük ezt a maximum 5 méteres utat, ami egy viszonylag időigényes művelet.



45. ábra - Az ultrahang mérésének szerkezeti ábrája

Az medián módszer alkalmazása során a felhasznált könyvtár 5 ping jelsorozatot küld ki egymás után az adóból. Ezeknek a visszaérkezési idejéből számítható ki az ultrahang által megtett út a hangsebesség értékének felhasználásával. A hangsebesség a hőmérséklet és közvetítő közeg függvényében változó mennyiség. Számításainknál a levegőben és szobahőmérsékleten történő 343m/s-os sebességét vesszük alapul. A hang által megtett távolság egyszerűen kiszámítható, ha az eltelt időt beszorozzuk a hang sebességével. Természetesen az ultrahang kétszer járja be a teljes utat, egyszer az adótól a tárgyig, majd másodszor visszacsapódáskor a tárgytól a vevőig. Ebből az okból fakadóan az előbb kiszámított értéket, még kettővel le kell osztanunk.

$$távolság = \frac{(mért\ idő * hangsebesség)}{2}$$

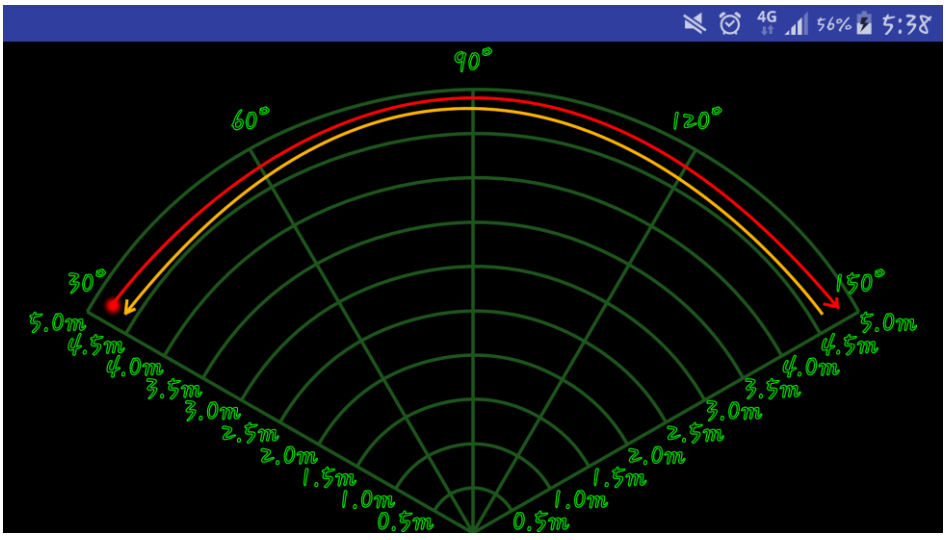
5. képlet – Az ultrahanggal mért távolság kiszámításának képlete

Ebből következik, hogyha a maximum 5 méteres utat teszi meg a kiküldött ultrahang, akkor ez a távolságot (2*5m)/(343m/s)=0,0292s, azaz körülbelül 29ms alatt tudja megtenni. Ez azt jelenti, hogyha egyetlen jelből álló mérésünk lenne, akkor 29 milliszekundumot kellene várjunk a szervo-val, hogy biztosak lehessünk abban, hogy vissza érkezett a kiküldött ping-re érkező echo válasz. De az általunk használt median metódus 5 ping jellel dolgozik, így a mérésekből adódó késleltetés 5*29, azaz 145ms. Amennyiben a mozgatásból és a mérésből származó késleltetéseket összegezzük, akkor a radarunk pozíciók közötti átállása összesen 147ms késletetéssel kell, hogy kalkuláljunk.

$$2 * (120° * 147ms) = 35280ms$$

6. képlet – A mozgásból adódó információ frissülés időszükséglete szélsőséges esetben

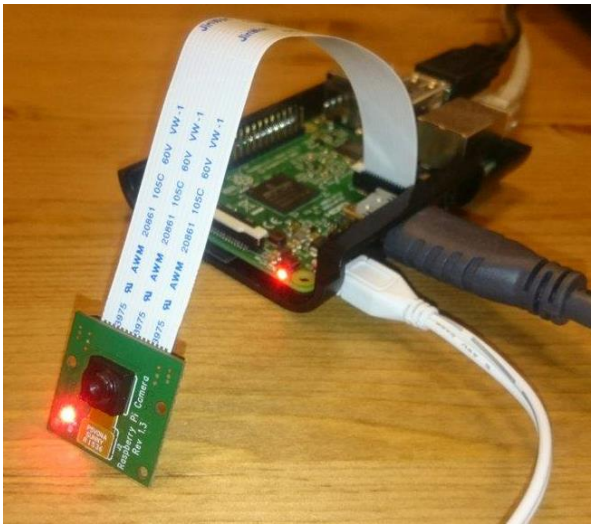
Számunkra ez azt a következtetést engedi levonni, hogy a legrosszabb esetben, amelyben az érzékelő az érzékelési tartomány széléről elindulva elér a másik széléhez, majd onnan visszafordulva ismét elérkezik a kiindulási ponthoz, több mint 35 másodperc telik el, amíg az adott ponthoz tartozó érték lefrissülhet. Átlagos esetben ez 17,64sec-os késleltetést jelent ezen a 120°-os tartományon, amely elfogadható érték egy ilyen kis teljesítményű szenzor esetén.



46. ábra - A radar 35 másodperc alatt járja be a 120°-ot

7.4. Kamera alrendszer

A video streamelése a Raspberry operációs rendszerének alap szoftvereinek illetve néhány harmadik féltől származó szoftver telepítésével van lehetőség. A Raspberry Pi hivatalos kameramoduljának kezelésére szolgáló alap programok a raspivid illetve a raspistill. Segítségükkel a ma rendelkezésre álló streamelési technológiák és protokollok egyaránt implementálhatók. A video streameléséhez a Real-Time Tranport Protocol-t (RTP) választottuk ami TCP kapcsolaton keresztül valósítja meg a streamelést. Így bármilyen videólejátszó szoftver, amely alkalmas streamelésre http-n vagy RTSP-n keresztül, képes lesz csatlakozni a Raspberry-hez és lejátszani a streamet.



47. ábra - Fotó a Raspberry PI-re csatlakoztatott HD kameráról

A kamera modul csatlakoztatása és engedélyezése után, a raspivid parancssori alkalmazással készíthetünk h.264 kódolású video streamet. A h.264 kódolás előnye, hogy – a streameléshez ideális – tömörítést használ. A kódolás célja minél jobb minőségű video elérése alacsonyabb bitráta mellett.

A video streameléséhez a nyílt forráskódú VLC media lejátszó cvlc parancssori alkalmazását használtuk. A VLC telepíthető a Raspbian alapértelmezett csomag telepítő alkalmazásán keresztül, valamint lefordítható az online elérhető forráskódból.

A h.264 kódolt videót a Linux pipe-ok segítségével juttatjuk el a cvlc alkalmazásnak, mely ezután elindítja a streamelést egy megadott porton keresztül. A szerver streamelés előtt kb. 300-1000ms hosszú videó anyagot gyorsítótáraz a hálózaton elérhető sávszélesség függvényében. A kliens oldalon elérhető média lejátszó alkalmazások ezen felül kliens oldali gyorsító tárazást használnak, ami 1 és 5 másodperc között jellemző érték, így elkerülhetetlen egy bizonyos nagyságú késleltetés a videó rögzítése és megjelenítése között.

48. ábra - Anroid-os képernyőkép a stream-elt kameraképről

8. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Projektünk megvalósítása közben, számtalan ötlet és lehetőség merült fel bennünk, mellyel továbbfejleszthetnénk CIWS rendszerünket, melyeket az alábbi felsorolásban tüntetünk fel. A felsorolás pontjainak egy része rendszerünk tökéletesítésével foglalkozik, mely pontok megvalósításával a fegyverzet méreteit leredukálhatnánk, bővíthetnénk az érzékelés terét és jobban lefedhetnénk azoknak a felhasználóknak a körét, melyeket kirekesztettünk egyes korlátozásainkkal. A lista másik része a fegyverrendszer, mai feltörekvő, modern technikákkal való egyesítésére tér ki, melyekkel rendszerünk jobban eleget tenne az innovációs elvárásoknak. Ilyenek például a hang alapú vezérlés, vagy a virtuális valóságban történő vezérlés lehetősége.

- az összes főbb mobil platform támogatása alkalmazásunkkal
- saját nyák tervezése, mely optimálisabb működést eredményez, és kevesebb helyet foglal
- a fegyverrendszer és az ultrahangos érzékelés átültetése térbe a sík működésről
- több fegyverrendszer összehangolt, autonóm működésének kiaknázása
- a fegyverrendszer éjjellátó kamerával való bővítése
- hang alapú érzékelés bevezetése
- a rendszer kerekekkel való bővítése, mozgásra való képességgel való felvértezése, terület védelmezéshez egy bejárandó útvonal segítségével
- a fegyverzet hanggal történő irányítása mobiltelefonról
- virtuális valóság szemüveggel történő vezérlés a virtuális térben
- a fegyverrendszer gondolatokkal való vezérlése

9. A PROJEKT ÉRTÉKELÉSE

Fegyverrendszerünk megtervezésekor azon cél lebegett a szemünk előtt, hogy egy olyan viszonylag kis méretű Close in Weapon rendszer megvalósítását tegyük lehetővé, melyen jól szemléltethetőek ezen rendszerek főbb jellegzetességei. A projekt megvalósítása során arra törekedtünk, hogy az elkészülő modell, a nagy katonai alkalmazású társaival azonos működésű, de alacsonyabb költségvetésű legyen. Amíg egy Phalanx CIWS előállítási költsége 3,8 millió $, addig az általunk elkészített demó változat teljes költsége 33.650Ft.

Megnevezés	Típus	Mennyiség	Egységár	Összesen
Arduino klón	Mega 2560	1 db	2000 Ft/db	2000 Ft
Raspberry klón	PI 2	1 db	11000 Ft/db	11000 Ft
Bluetoot modul	hm-10	1 db	900 Ft/db	900 Ft
Wifi adapter	rtl8188cus	1 db	700 Ft/db	700 Ft
Mozgásérzékelő	hc-sr501	1 db	300 Ft/db	300 Ft
Ultrahang szenzor	hc-sr04	1 db	300 Ft/db	300 Ft
Kamera modul	ov-5647	1 db	3700 Ft/db	3700 Ft
Gyorsulásmérő	mpu-6050	1 db	600 Ft/db	600 Ft
Szervó	sg90	3 db	500 Ft/db	1500 Ft
Léptető motor	jq24-125H670	2 db	700 Ft/db	1400 Ft
Léptető motor 2	28byj48-5v	2 db	450 Ft/db	900 Ft
Akkumulátor	12V 3Ah NiMH	1 db	10000 Ft/db	10000 Ft
Relay modul	Tianbo 05vdc	1 db	350 Ft/db	350 Ft
Összesen				**33650 Ft**

1. táblázat – A projekt megvalósításának kerekített költségei

Természetesen egy ilyen kicsinyített modell sohasem lesz olyan nagy kapacitású, érzékelési távolságú és pontosságú, mint egy nagy költségvetésű, több száz kutató és mérnök által kivitelezett katonai projekt, ám kitűnően reprezentálhatja azokat a működés során felmerülő eseteket, melyekkel szembe találkozhatunk működtetésük során.

Megnevezés	CIWSduino	AK-630	Phalanx	Goalkeeper
Súly	2 kg	9114 kg	6200 kg	9902 kg
Fegyverzet	Hengeres kilövő	GSh-6-30	M61 Vulcan	GAU-8
Tüzelési gyorsaság	X	5000 db/perc	4500 db/perc	4200 db/perc
Tüzelési tartomány	4,5 m	4 km	3,6 km	2 km
Lőszer tárolás	20 db	2000 db	1550 db	1190 db
Kezdősebesség	X	900 m/s	1100 m/s	1109 m/s
Magassági szögtartomány	0°	-12 és +88° között	- 25 és +85°között	- 25 és +85°között
Szélességi szögtartomány	+-90°	+-180°	+-180°	360°

2. táblázat – A CIWS rendszerek paramétereit összehasonlító táblázat

Sajnos a betervezett alrendszereket, a megvalósításra rendelkezésre álló rövid időkereten belül, nem sikerült hiánytalanul megvalósítanunk. Az elkövetkezendő időszakban a hiányzó modulok megvalósítására szeretnénk összpontosítani. Ettől eltekintve kijelenthetjük, hogy projektünk ezen szakasza sikeresen zárult. A rendszerünk eddig elkészült elemei a terveknek megfelelően működnek, a pótolandó részek megvalósítása jó irányba halad. A még kinyomtatásra váró alkatrészek elkészültével, illetve a még hiányzó, de már megrendelt modulok megérkeztével záros határidőn belül elkészülhet az oktatási célokat szolgáló CIWSduino fegyverrendszer.

10. IRODALOMJEGYZÉK

[1] Simon Monk: Programming Arduino : Getting started with sketches, *TheMcGraw-Hill Companies*, 2012

[2] Mike Riley: Programming your home: Automate with Arduino, Android, and your computer, *The Pragmatic Programmers LLC*, 2012

[3] Andrew K. Dennis: Raspberry Pi Home Automation with Arduino, *Packt Publishing Ltd.*, 2013

[4] CIWS (2016 január): https://en.wikipedia.org/wiki/Close-in_weapon_system

[5] SGR-A1 (2015 szeptember): https://en.wikipedia.org/wiki/Samsung_SGR-A1

[6] Phalanx (2016): http://www.raytheon.com/capabilities/products/phalanx/

[7] Phalanx (2014 június): http://www.naval-technology.com/features/featurelaser-quest-phalanx-laws-and-the-future-of-close-in-weapon-systems-4295413/

[8] Goalkeeper http://www.thales7seas.com/html5/products/313/GOALKEEPER.pdf

[9] Goalkeeper (2016 január): https://en.wikipedia.org/wiki/Goalkeeper_CIWS

[10] AK-630 (2016 február) https://en.wikipedia.org/wiki/AK-630

[11] MakerBot (2016 március) https://eu.makerbot.com/shop/en/3d-printer/replicator-mini/93/makerbot-replicator-mini

[12] Filament (2016 március) http://store.makerbot.com/filament/pla#mini-truered

[13] Arduino Mega (2016 március) https://www.arduino.cc/en/Main/arduinoBoardMega

[14] Arduino Mega (2016 március) https://www.arduino.cc/en/Main/ArduinoBoardMega2560

[15] Arduino Software (2016 március) https://www.arduino.cc/en/Main/Software

[16] Raspberry PI2 (2016 március) https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[17] Raspbian (2016 március) https://www.raspbian.org/RaspbianFAQ

[18] Android (2016 március) http://developer.android.com/about/dashboards/index.html

[19] Android Studio (2016 március) http://developer.android.com/sdk/index.html#top

[20] Git (2016 március) https://git-scm.com/

[21] Gradle (2016 március) http://gradle.org/whygradle-build-automation/

[22] Samsung Galaxy S4 (2016 március) http://www.gsmarena.com/samsung_i9506_galaxy_s4-5542.php

[23] Sony Xperia Z3 Compact (2016 március) http://www.gsmarena.com/sony_xperia_z3_compact-6538.php