ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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

SZAKDOLGOZAT

Rizmajer Bence
T010848/FI12904/K

OE-KVK
2025.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Rizmajer Bence
Szakdolgozat száma: SZD2506301042197245Y2MVWY
Törzskönyvi száma: T010848/FI12904/K **Neptun kódja:** [redacted]

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

A dolgozat címe: Kamerarendszer-alapú célpont lokalizáció és metrikus pontszámítás OpenCV segítségével
A dolgozat címe angolul: Camera system-based target localization and metric point calculation using OpenCV
A feladat részletezése: Készítsen darts táblához automatikus kiértékelőt, amely rögzített kamerarendszerről szól, mely élő képfeldolgozás segítségével képes detektálni a dobást és annak értékét megjeleníti.

Intézményi konzulens neve: Sándor Tamás

A kiadott téma elévülési határideje: 2027. 06. 30.
Beadási határidő: 2025. 12. 15.

A szakdolgozat: Nem titkos.

Kiadva: Budapest, 2025. 10. 14.

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

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

HALLGATÓI NYILATKOZAT

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

Budapest, 2025.11.22.

.....

hallgató aláírása

1 Tartalomjegyzék

2 Bevezetés

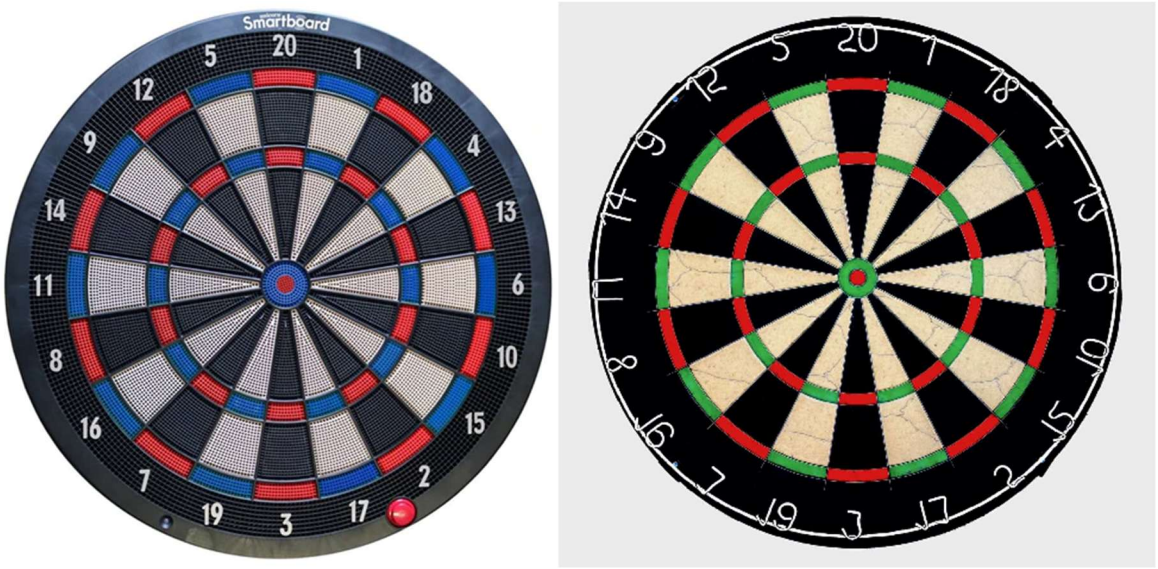
Az utóbbi évtizedekben a számítógépes látás rohamos fejlődésen ment keresztül, és napjainkra mind a tudományos kutatásban, mind az ipari alkalmazásokban meghatározó szerepet játszik. Az önvezető járművek, az ipari minőségellenőrző rendszerek, a robotika vagy akár a biztonságtechnika mind erősen támaszkodnak a képfeldolgozás és a mesterséges intelligencia legújabb innovációira. A kamerarendszerek és az azokhoz kapcsolódó algoritmusok lehetővé teszik, hogy a számítógépek egyre pontosabban érzékeljék és értelmezzék a környezetüket.

A képfeldolgozás egyik fontos felhasználási területe a célpontok detektálása és a pontos lokalizáció. Ez a feladat nemcsak a katonai, vagy kutatási, hanem a szórakoztató és sporteszközök világában is releváns. Ez egy olyan rendszer, amely képes kameraképek alapján meghatározni egy célpont helyzetét, illetve a dobások összeget egy külön szoftver komponens dolgozza fel és tárolja adatbázisban. A megvalósítás során az OpenCV (Computer Vision) könyvtár nyújtja az alapvető metódusokat a képfeldolgozási feladatok elvégzésére, például az objektumok detektálására, geometriai transzformációk alkalmazására, valamint a képi zajok kiszűrésére.

A téma gyakorlati jelentősége abban rejlik, hogy a megvalósított rendszer valós időben képes támogatni egy darts játék mérését és kiértékelését, ezáltal modern és automatizált alternatívát kínál a hagyományos manuális pontszámítás helyett.

3 Irodalomkutatás

A darts automatikus pontszámlálására irányuló megoldások a gyakorlatban és a kutatásban két alapvető irányra oszthatók: a beépített érzékelőket használó elektronikus táblákra, illetve a meglévő bristle steel-tip táblákhoz kamerákon és számítógépes látáson alapuló megoldásokra. A kereskedelmi elektronikus soft-tip táblák tömegesen elérhetők, egyszerű telepítésükkel és beépített számlálóegységükkel azonnali visszajelzést adnak, ezért népszerűek otthoni és szórakoztató környezetben. Ezek a rendszerek megbízhatóan működnek soft-tip nyilakkal, és számos játékmódot, hálózati vagy statisztikai funkciót kínálnak; ugyanakkor alapvető korlátuk, hogy nem kompatibilisek professzionális steel-tip környezettel, ahol a bristle-tábla és a nyilak mechanikája eltérő. [1],[4]



1. ÁBRA SOFT-TIP ELEKTROMOS ÉS BRISTLE STEEL-TIP DARTS TÁBLA [31] [30]

A profi és versenycélú alkalmazások, illetve a már meglévő bristle táblák megtartását célzó fejlesztések a számítógépes látás (computer vision) és a mélytanulás eszközeire támaszkodnak. Egy jelentős kutatási eredmény a „DeepDarts” rendszer, amely a kulcspont-detekció feladatát úgy modellezi, hogy a nyílhegyeket, mint objektumokat detektálja, és egyetlen kameraképből képes a dobások helyét és ezáltal a pontértéket megbecsülni. A szerzők által összeállított, több tízezres képadatbázison végzett vizsgálatokban a DeepDarts a tesztképek többségében helyes összpontszámot adott, és bemutatja, hogy single-image alapú megközelítéssel költséghatékony, széles körben telepíthető megoldás érhető el, bár a pontosság függ a nézőponttól és a képzési adathalmaz

sokszínűségétől. [2], [3] A DeepDarts-hoz hasonló kutatások rávilágítanak arra, hogy a single-camera megoldásoknál a fő problémák: (1) a kamera nézőpontjából eredő torzítások és kalibráció szükségessége, (2) az occlusion és több nyíl közeli elhelyezkedése miatti különböző detekciós nehézségek, valamint (3) a megvilágítás- és tábla-variancia kezelése. Ezek a kihívások a publikus irodalomban többször felbukkanó kutatási kérdésekké váltak, és gyakori javaslat a többkamerás fúzió, task-specifikus adataugmentációk és robusztus keypoint-modellek alkalmazása. [2], [5]

A kutatási eredmények mellett a piacon és a hobbi közösségben is megjelentek gyakorlati, kamerára épülő megoldások. Az Autodarts Vision Kit egy kereskedelmi jellegű, kamerát/kamerákat és szoftvert tartalmazó rendszer, amely a gyártó állítása szerint magas (a marketinganyagban említett 99,5%-os) pontossággal detektálja a találatokat és automatikusan számolja a pontokat, így lehetővé teszi hagyományos bristle táblákhoz való utólagos automata számlálást és online játékokat is.[3], [6] Az ilyen kereskedelmi „vision kit” megoldások előnye, hogy viszonylag egyszerű telepítéssel és felhasználói felülettel járulnak hozzá az otthoni vagy klub-szintű automatizáláshoz, ugyanakkor függenek a gyártó kalibrációs ajánlásaitól, és jellemzően nem oldanak meg minden, kutatásban feltárt nehézséget, mint például a szélsőséges megvilágítás vagy az extrém nézőpontok.[3], [6]

Egy másik fontos szereplő a rangsorban a szoftveres/hibrid platformok köre. A DartConnect például elsősorban egy statisztikai, ligamenedzsment és kézi/asszisztált számlálást támogató szolgáltatás, amely VideoConnect funkcióján keresztül lehetőséget ad kamerás megfigyeléshez és online meccsek lebonyolításához. Az ilyen platformok nem feltétlenül végeznek automatikus, teljesen autonóm CV-alapú detekciót, de integráló szerepük fontos a felhasználói élmény és a versenyszervezés terén.[7], [12]

A kutatói és hobbi közösségben számos open-source és egyetemi projekt is található, amelyek OpenCV-re és egyedi deep-learning megoldásokra építve demonstrálnak automata pontszámlálást. Egyes egyetemi projektek és GitHub-repository-k egyszerű, kalibrált webkamera-alapú rendszereket mutatnak be, amelyek jó kiinduló pontot nyújtanak prototípusokhoz és összehasonlító vizsgálatokhoz, ugyanakkor gyakran hiányzanak belőlük a nagyobb, kipróbált adatbázisok és az ipari szintű felhasználói tesztek.[4], [8]

Összességben a jelen szakirodalom és piaci állapot kettős pályát mutat. Az egyszerű, beépített érzékelőkkel működő elektronikus táblák stabil, olcsó és széles körben elterjedt

megoldást kínálnak, míg a steel-tip bristle táblákhoz történő automata számlálás felé a számítógépes látás és a mélytanulás adja a legígéretesebb irányt. A DeepDarts és a hasonló kutatások bebizonyították, hogy single-image, illetve többkamera-alapú rendszerekkel megbízható pontszámítás érhető el, de a gyakorlatban a teljes megbízhatóság eléréséhez továbbra is szükséges többnézetes fúzió, kiterjedt adataugmentációk és robusztus kalibrációs eljárások kidolgozása. A piaci „vision kit" megoldások és a szoftveres platformok közötti együttműködés (például automata detektálás + kézi felülvizsgálat) jelenleg a legpraktikusabb út a széles körű elterjedés felé.

4 Specifikáció

A rendszer tervezésekor olyan hardver és szoftverkörnyezetet választottam, amely a valós idejű képfeldolgozás, a megbízható adatkommunikáció és a stabil backend működés követelményeit a lehető legjobban teljesíti. A választott komponensek minden esetben műszaki, teljesítménybeli és fejlesztési szempontok alapján kerültek kiválasztásra, figyelembe véve a fejlesztési időt, rendelkezésre álló erőforrásokat és a rendszer hosszú távú bővíthetőségét.

A képfeldolgozó algoritmusok implementálásához Python nyelvet használok, amely magas szintű absztrakciójának és gazdag könyvtárkészletének köszönhetően ideális választás gyors kutatás-fejlesztési ciklusokhoz. A Python ökoszisztéma kulcsfontosságú eleme az OpenCV, amely ipari szabvány a képfeldolgozás területén, és natív hardvergyorsítási lehetőségekkel is rendelkezik. Míg C++ implementációval elméletileg magasabb futási sebesség érhető el, a fejlesztési idő jelentősen megnövekedne, valamint a dinamikus tesztelés és algoritmus-iterációk nehezebbé válnának. Java és más magas szintű nyelvek szintén alternatívát jelentenének, azonban könyvtártámogatottságuk és rugalmasságuk a Python–OpenCV párossal nem vethető össze, különösen képfeldolgozás-orientált projektekben.

A backend komponens megvalósításához Java nyelvet és a Spring Boot keretrendszert alkalmazom. A Java erőssége a típusbiztonság, a hosszú távú stabilitás, valamint a kiemelkedő teljesítmény JVM-alapú alkalmazások esetén. A Spring Boot moduláris felépítése, széleskörű integrációja és REST API-k fejlesztésére optimalizált infrastruktúrája ipari szinten bizonyított. A Spring Security, JPA és WebFlux modulok elérhetősége megkönnyíti a rendszer bővítését és skálázhatóságát. A fejlesztés során IntelliJ IDEA-t használok, amely fejlett kódanalízis eszközöket, automatikus refaktorálási lehetőségeket és kiváló Spring-integrációt biztosít. Ezek a funkciók jelentősen csökkentik a fejlesztési időt és növelik a kódminőséget, így hatékonyabb alternatívát jelentenek más IDE-khez képest, például az Eclipse-hez vagy a NetBeans-hez.

A képi adatok gyűjtése három egységből álló kamerarendszerrel történik, ahol minden Raspberry Pi Camera Module 2 külön Raspberry Pi 4 mikroszámítógéphez kapcsolódik. Ezek az egységek lokálisan, a helyi erőforrásokra támaszkodva végzik el az előfeldolgozást (pl. zajszűrés, perspektíva-korrekció, kontúrdetektálás), így a backend felé már csak a releváns, kinyert adat kerül továbbításra, ami jelentősen csökkenti a hálózati terhelést és a késleltetést. A Raspberry Pi Camera Module 2 választásának oka a Raspberry Pi

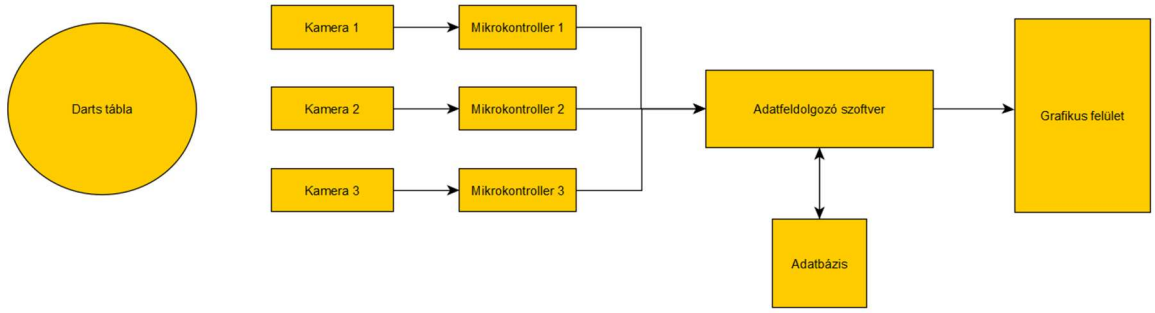
platformmal való natív kompatibilitás, az alacsony késleltetésű adatátvitel, valamint a jó minőségű, fix fókuszos szenzor, amely elegendő felbontást biztosít a darts tábla pontos követéséhez. Más USB-s kamerák gyakran nagyobb késleltetéssel, magasabb CPU-terheléssel vagy gyengébb dokumentációval rendelkeznek. A Raspberry Pi 4 alkalmazása indokolt a szükséges számítási teljesítmény miatt. A négymagos ARM Cortex-A72 architektúra, a 4–8 GB RAM elérhetősége és a hardveres videoprocesszorok lehetővé teszik a valós idejű képfeldolgozást. Más mikrokontrollerek, mint például az ESP32 vagy különféle STM32 sorozatok bár energiatakarékosak, de nem képesek nagy felbontású képek kezelésére, nem futtatnak teljes operációs rendszert, és nem tudnak Python–OpenCV környezetet támogatni.

A rendszer adatainak tárolására PostgreSQL adatbázist alkalmazok. Ez az adatbázis kiválóan alkalmas strukturált adatok, történeti rekordok és összetett lekérdezések kezelésére. Teljes mértékben ACID-kompatibilis és támogatja a többfelhasználós környezetekben szükséges tranzakciókezelést, amely elengedhetetlen a darts-eredmények és a kamerák által küldött időbélyeges adatok konzisztens tárolásához. A PostgreSQL előnye a MySQL-lel szemben a szabványosabb SQL-megvalósítás és a fejlettebb indexelési lehetőségek. Az SQLite-hoz képest pedig a hálózati működés, párhuzamos hozzáférés és skálázhatóság hozható fel előnyként.

5 Rendszerterv

5.1 Logikai rendszerterv

A logikus rendszerterv alkotja a teljes projekt strukturális alapját, amely három különálló, mégis szorosan integrált funkcionális egységet foglal magában: a kamerákkal és mikrovezérlőkkel felszerelt feldolgozó modulokat, a központi adatfeldolgozó egységet és a grafikus kijelző egységet. Ezen komponensek mindegyike meghatározott szerepet tölt be az általános architektúrán belül, miközben összehangolt interakciójuk biztosítja a rendszer hatékony és megbízható működését egészében. Ez a moduláris megközelítés rugalmasságot és skálázhatóságot ad, lehetővé téve az egyes alrendszerek jövőbeni módosítását vagy frissítését a teljes rendszer funkcionalitásának veszélyeztetése nélkül.



2. ÁBRA LOGIKAI RENDSZERTERV

Összességében ez a logikus rendszerfelépítés biztosítja, hogy a több kameramodulból származó adatok feldolgozása összehangolt és szinkronizált módon történjen, lehetővé téve az átlátható, pontos és valós idejű visszajelzést a felhasználó számára. Az alrendszerek közötti strukturált interakció zökkenőmentes információáramlást hoz létre, ami egy koherens és hatékony darts pontozási rendszert eredményez.

5.1.1 Darts tábla

A steel tip darts tábla egy nagy sűrűségű, kompresszált sisalrostból készült céltárgy, amelyet kifejezetten acélhegyű darts nyilak befogadására és tartós használatra terveznek. A sisalrost, amik az Agave sisalana növény hosszú, 100–300 mikrométer átmérőjű szálai, nagy rugalmas deformációs képességgel rendelkezik, így a tábla felületén keletkező perforációk a terhelés megszűnése után részben vagy teljesen visszazáródnak.

3. ÁBRA SISALROST [32]

Ez a tulajdonság, valamint a tábla 1,1–1,3 g/cm³ körüli átlagos tömörsége nagymértékben hozzájárul a táblák hosszú élettartamához és mechanikai stabilitásához. A táblát körülvevő hátlap jellemzően közepes sűrűségű farostlemezből vagy rétegelt lemezből készül 10–15 milliméter vastagsággal, amely a táblának további merevséget biztosít, illetve csökkenti az ütésből eredő rezgések amplitúdóját.

A szektorok elkülönítését biztosító drótrendszer, úgynevezett "spider", a tábla fontos funkcionális eleme. A modern konstrukciókban egyre elterjedtebb a háromszög keresztmetszetű elválasztódrót, amely a lepattanások csökkentését szolgálja azáltal, hogy a nyíl hegye a drótról könnyebben a sisalrostok közé csúszik, nem pedig visszaverődik róla. A drót vastagsága általában 0,6 és 1,2 milliméter között változik, rögzítése pedig tűzdróttal vagy ragasztással történik, amely minimalizálja a nem kívánt rezgésátadást. A drótrendszer pontos geometriáját a WDF (World Darts Federation) szabványok határozzák meg, és alapvető szerepet játszik a tábla pontosságában és játékélményében.

4. ÁBRA FÉLBEVÁGOTT HUZAL

A standard méretezés szerint a darts tábla átmérője 451 milliméter körüli (±3 mm toleranciával). A pálya 20 egyenlő, 18 fokos szögszektorra oszlik, amelyek logaritmikusan növekvő pontértékei a játék képesség és pontosságigényét erősítik. A dupla és tripla szektorokat határoló gyűrűk szélessége 8 milliméter, míg a köztes területek ennél nagyobb felülettel rendelkeznek. A bull terület két koncentrikus körből áll, a külső bull (single bull) átmérője 31–32 milliméter, míg a belső bull (double bull) 12–13 milliméter.

A steel tip darts tábla működésének fizikai sajátosságait a sisalrostok anyagtani jellemzői, a drótrendszer kialakítása és a tábla teljes szerkezeti integritása határozzák meg. A nyíl becsapódásakor a tábla jelentős lokális terhelést kap, a kinetikus energia pedig a rostok közötti súrlódáson és a komponensek belső csillapításán keresztül disszipálódik. A rugalmas tulajdonságok biztosítják, hogy a tábla torzulás nélkül képes hatékony energiaelnyelésre. Az optimális működéshez szükséges a homogén rosteloszlás, az egyenletes kompresszió, valamint a precízen megmunkált felosztás, amelyek együtt teszik

lehetővé a pontos és ismételhető mérési környezetet mind hobbi, mind versenyszintű használat esetén.

5.1.2 Kamerák és mikrokontrollerek

A kamera és a mikrovezérlő párost ugyanabból a termékcsaládból választottam ki, biztosítva a teljes hardverkompatibilitást és a megbízható kommunikációt. Összeköttetésük egy előre tervezett 15 tűs AWM szalagkábellel valósul meg, amely stabil és kompakt interfészt biztosít a két eszköz között, minimalizálva a jelveszteség vagy a zajinterferencia kockázatát. Ez a szabványosított csatlakozás leegyszerűsíti a fizikai beállítást és csökkenti a lehetséges kábelezési hibákat, így a teljes rendszer robusztusabbá és könnyebben reprodukálhatóvá válik.

A projekt ezen szakaszában az elsődleges cél a kamerák által rögzített élő videó valós idejű feldolgozása. Ezt a folyamatot egy három kamerából álló konfiguráció támogatja, amelyek mindegyike saját, dedikált mikrovezérlőhöz csatlakozik. A rendszer a következőképpen működik: minden mikrovezérlőre fel van töltve egy Python programozási nyelven fejlesztett képfeldolgozó szoftver. Ez a szoftver az OpenCV függvénykészletet használja a bejövő képfolyam elemzésére, a dobott nyíl helyének azonosítására, és a darts táblán észlelt becsapódási terület alapján a megfelelő pontszám meghatározására. Számítógépes látási algoritmusok alkalmazásával a szoftver hatékonyan képes érzékelni a mozgást, felismerni az egyes régiókat, és értékeket rendelni az észlelt találatokhoz.

A lehető legnagyobb mérési pontosság és rendszermegbízhatóság elérése érdekében minden egyes kamerát külön kell kalibrálni és kiértékelni. Ez a megközelítés lehetővé teszi a fejlesztő számára, hogy könnyen meghatározza, melyik kamera kódját, konfigurációját vagy fizikai elhelyezkedését kell módosítani a pontosság növelése érdekében. A kamera-visszacsatolás elkülönítése biztosítja, hogy a hibakeresés és az optimalizálás szisztematikusan elvégezhető legyen. Ezenkívül ez a beállítás lehetővé teszi az egy, két vagy három kamera azonos körülmények között történő konfigurációinak teljesítményének közvetlen összehasonlítását, ezzel segítve a pontos nyíl detektálás leghatékonyabb beállításának meghatározását.

A rendszerblokk adatkimenetét illetően, amikor egyetlen nyilat dobnak a táblára, minden mikrokontroller függetlenül feldolgozza a kamera képét, és továbbít egy adatkészletet a központi adatfeldolgozó egységnek. Következésképpen három különálló

adatpontot fogad, amelyek megfelelnek a három kamera által végzett egyedi észleléseknek. Ezeket az adatokat ezután egy Java szoftver elemzi, hogy meghatározza a végeredményt, biztosítva, hogy a rendszer pontosan azonosítani tudja a nyíl becsapódási pozícióját és kiszámítsa az eredményét.

5.1.3 Központi adatfeldolgozás

Az adatfeldolgozó egység központi elemként szolgál, ahol a képfeldolgozási fázis után az összes logikai művelet és számítási feladat történik. A teljes rendszer magját képező intelligenciát biztosítja, amely a kameráktól és a mikrokontrollerektől kapott információkat egységes keretbe integrálja. Ezen az egységen belül a feldolgozott képadatokat elemzi, értelmezi és értelmes információkká alakítja, amelyek felhasználhatók az egyes darts dobások értékeléséhez. Ez a szakasz biztosítja, hogy a kamerák nyers detektálási eredményeit pontos pontozási adatokká alakítsa, amelyeket később a teljesítmény nyomon követésére és vizualizációjára lehet felhasználni. A logika részletes működését, beleértve az adatértelmezést és a döntéshozatali mechanizmusokat, a dolgozat későbbi szakaszaiban ismertetem.

Miután egy dobást kiértékeltek és meghatározta a hozzá tartozó pontszámot, a szoftver automatikusan rögzíti az eredményt a megfelelő mérkőzésszám alatt a rendszer adatbázisában. Ez az adatbázis-struktúra több játékos és folyamatban lévő játék egyidejű kezelésére szolgál, lehetővé téve a hatékony adattárolást és -lekérdezést. A strukturált és szervezett adatrekordok fenntartásával a rendszer biztosítja, hogy minden dobás a megfelelő játékoshoz és játékmenethez legyen társítva. Az adatbázis részletes felépítését a későbbiekben ismertetem.

Továbbá az adatfeldolgozó egység különféle statisztikai elemzéseket végez a rögzített adatokon, például átlagokat, sikerességi arányokat vagy teljesítménytrendeket számít ki az időbeli lefolyásból. Ezek a statisztikai számítások lehetővé teszik értékes információk gyűjtését a játékosok teljesítményéről, segítve a haladás nyomon követését és a fejlesztendő területek azonosítását. Ezenkívül a szoftver kezeli az összes releváns mutató szinkronizálását és biztonságos tárolását, biztosítva, hogy a játékosok statisztikái következetesen frissüljenek és elérhetőek legyenek a rendszer egészében. Ezáltal az adatfeldolgozó egység nemcsak számítási központtá, hanem a hosszú távú adatkezelés és a teljesítményértékelés kritikus elemévé is válik.

5.1.4 Grafikus megjelenítés

A grafikus kijelzőmodul a teljes rendszer felhasználói felületeként szolgál, világos és interaktív módon biztosítva a játékosok számára a folyamatos eredmények és az általános játékteljesítmény vizualizálását. Elsődleges funkciója a mérkőzés során szerzett pontok valós idejű megjelenítése, biztosítva, hogy a játékosok minden dobás után folyamatosan nyomon követhessék a haladásukat. Ez a vizuális visszajelzés nemcsak a játékélményt fokozza, hanem hozzájárul egy vonzóbb és átláthatóbb mérkőzéskörnyezethez is. A felület intuitív és könnyen olvasható, és olyan lényeges információkat jelenít meg, mint a játékosok nevei, az aktuális pontszámok és a fennmaradó pontok strukturált és vizuálisan vonzó elrendezésben.

A közvetlen eredmények egyszerű megjelenítésén túl a grafikus kijelző fontos szerepet játszik a gyűjtött adatok összefoglalásában és értelmezésében is, miután a mérkőzés véget ért. Minden mérkőzés végén a kijelző átfogó statisztikai elemzést mutat a játékosok teljesítményéről, beleértve olyan mutatókat, mint az átlagos pontszámok, a találatok pontossága, a konzisztencia és a körönkénti összehasonlítások. Ezeket a statisztikai összefoglalásokat a rendszer adatbázisában tárolt feldolgozott adatok felhasználásával generálja, amelyeket az adatfeldolgozó egység számít ki.

A valós idejű vizualizáció és a részletes játék utáni elemzés kombinálásával a grafikus kijelző áthidalja a szakadékot a technikai adatok és a felhasználói élmény között. A numerikus információkat világos, könnyen értelmezhető vizuális megjelenítéssé alakítja, amelyek segítenek a játékosoknak megérteni teljesítményük trendjeit és időbeli fejlődését. Lényegében ez a komponens nemcsak informatív kimeneti felületként szolgál, hanem motiváló és elemző eszközként is, amely mind a laza játékmenetet, mind a komoly teljesítménykövetést támogatja a darts pontozási rendszeren belül.

5.2 Fizikai rendszerterv

A fizikai rendszertervben a hangsúly a rendszer elméleti és funkcionális leírásáról a kézzelfogható megvalósításra helyeződik át, részletezve, hogy a logikai rendszerterv komponensei hogyan valósulnak meg fizikailag és hogyan kapcsolódnak egymáshoz. Ez a szakasz a korábban meghatározott architektúra konkrét megvalósítását kívánja bemutatni, elmagyarázva, hogyan épülnek fel, vannak elrendezve és integrálva egy teljesen működőképes rendszerbe. Míg a logikai rendszerterv meghatározza az egységek közötti kapcsolatokat és adatáramlást, a fizikai terv világosan ábrázolja, hogyan jelennek meg ezek az egységek a valós hardverben, beleértve térbeli konfigurációjukat, kábelezésüket és támogató infrastruktúrájukat.

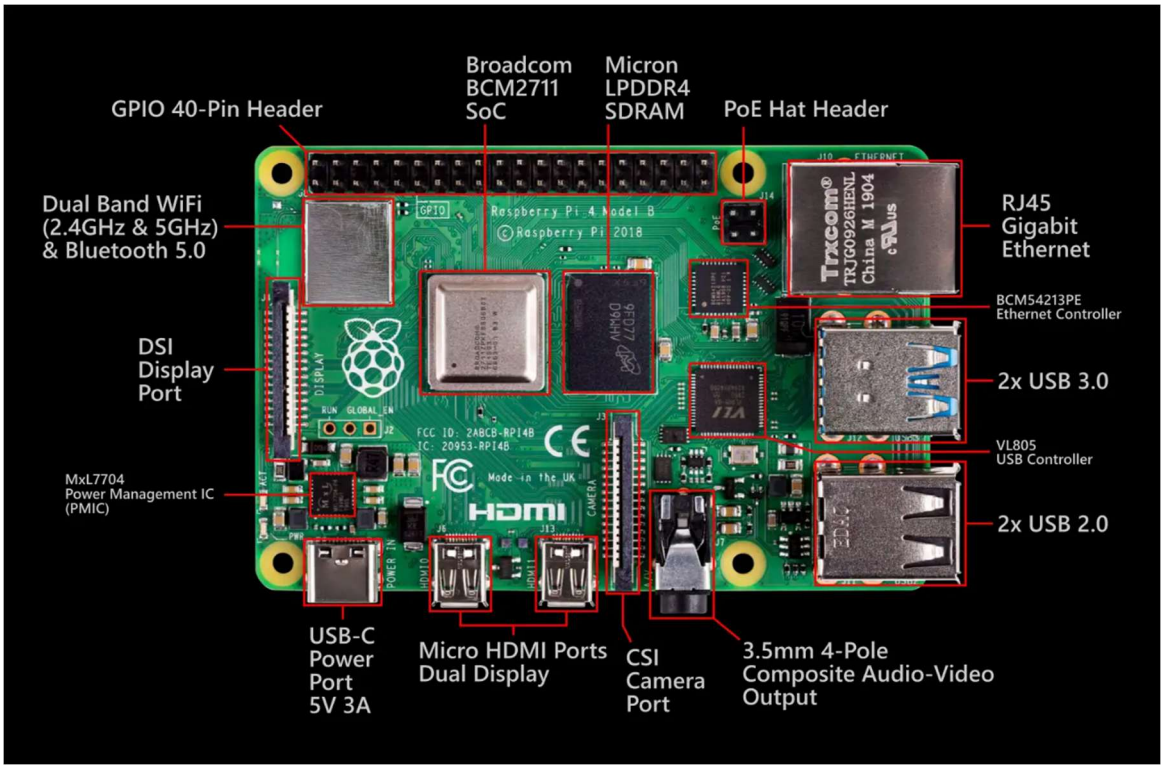
A fizikai elrendezés leírása mellett ez a terv az alrendszerek közötti elektromos csatlakozásokat is tárgyalja. Ez magában foglalja a tápegység elosztásának, az adatátviteli kábelek, a kommunikációs interfészek és a biztonsági szempontok részletes ismertetését a stabil és interferenciamentes működés biztosítása érdekében. A szabványosított csatlakozók, például a kamerák és a mikrovezérlők közötti 15 tűs AWM szalagkábel használata hozzájárul a letisztult és moduláris kialakításhoz, amely könnyen összeszerelhető, karbantartható vagy bővíthető.

5.2.1 Komponensek

A fizikai rendszer négy alapvető elemre bontható, ezek közül több szorosan együttműködve alkotja meg a műszaki logikát, és van amelyik a rendszer vizuális megjelenését, vagy éppenséggel a mechanikai megoldását szolgálja ki.

5.2.1.1 Mikrokontroller

A képfeldolgozó szoftver megfelelő üzemeltetésére egy Raspberry Pi 4 Model B mikrokontrollert választottam, ugyanis az élő képfeldolgozás szignifikáns mennyiségű számítási kapacitást vesz igénybe. Ez az eszköz, mondhatni bőven a minimális elvárások felett teljesít, így költséghatékonyság szempontjából nem feltétlenül a legjobb megoldás, azonban ezt a minőség nagyban kompenzálja, ugyanis sok tapasztalatom van az e féle termékek/szoftverek használatában és rettentően kizökkentő az, amikor a játék dinamikáját megakadályozza a szoftver lassú működése.



1. FIGURE RASPBERRY PI 4 MODEL B BOARD KIOSZTÁS [33]

A kontroller nem minden része van felhasználva fejlesztés, illetve működtetés közben, így a mikrokontroller azon részeit szeretném bemutatni, amik projekt relevanciával bírnak.

- **CPU**

Paraméter	Leírás
ISA	ARMv8-A (64-bit)
Mikroarchitektúra	Cortex-A72
Kiadás dátuma	2019 Q2
Magok száma	4
Szálak száma	4
Mag architektúra	Normál
Frekvencia	1.5GHz
Gyártástechnológia	28nm
L1 gyorsítótár	32KB adat + 48KB utasítás L1 gyorsítótár magonként
L2 gyorsítótár	1MB megosztva az összes mag között
Maximális memória sávszélesség	4.4GB/s
Maximális PCIe sávok	4
TDP	4W

Minden számítógép több szintű memória-gyorsítótárral rendelkezik. A leggyorsabb adatátvitel a gyorsítótárakba és a gyorsítótárakból a CPU-ba ágyazott gyorsítótárak. Ezeket L1, L2, L3 stb. gyorsítótáraknak nevezik, ami 1., 2., 3. stb. szintet jelent. Minél magasabb a szint száma, annál nagyobb lehet a gyorsítótár, de lassabb lesz. Az ARM Cortex-A72 processzor memória-architektúrája kétszintű gyorsítótár-struktúrával rendelkezik: egy 1. szintű (L1) adat-gyorsítótárral és egy egységes 2. szintű gyorsítótárral. Magonként egy L1 gyorsítótár található, és mind a négy mag osztozik az L2 gyorsítótáron. Az elsődleges memória a harmadik és egyben utolsó szint az L2 gyorsítótár után.

Az RPi4B Cortex A72 egyik előnye az elődeihez képest a 15 utasításos futószalag-mélység, amely sorrenden kívüli végrehajtást biztosít, így nem kell megvárnia, amíg az egyik folyamat kimenete elindul egy másikon. Ez sokkal gyorsabbá teszi, mint az RPi 3B+.

- **GPU**

Paraméter	Leírás
GPU Integrált grafika	Broadcom VideoCore VI (32-bit)
GPU végrehajtó egységek	4
GPU árnyaló egységek	64
GPU órajel	500MHz
GPU FP32 lebegőpontos teljesítmény	13.5GFLOPS – 32GFLOPS (becslések és számítások alapján változhat)
Maximális kijelző felbontás	4Kp60
Videó dekódolás	H.265 4Kp60, H.264 1080p60
Videó kódolás	H.264 1080p30

Mint minden számítógépnél, a beépített grafikus processzorok is RAM-ot használnak grafikus memóriának, amelyet megosztanak a CPU-val. Az RPi konfigurációs dokumentációja szerint a konfigurációs fájlban (config.txt) található GPU memóriabeállítások határozzák meg a CPU és a GPU közötti memóriamegosztást. Bármilyen értékre is van beállítva a GPU memória, a CPU kapja a fennmaradó memóriát. A minimális érték 16 MB, az alapértelmezett érték 64 MB, a technikai maximális érték pedig 944 MB, bár az 512 MB feletti értékek nem biztosítanak nagyobb teljesítményt, és nem szabad használni őket.

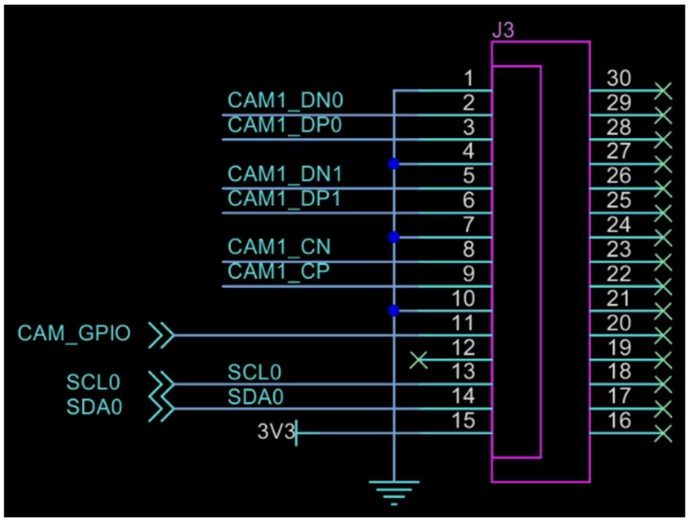
- **WIFI**

A CYW43455 20 MHz, 40 MHz és 80 MHz csatornaszélességet támogat. A WiFi adatátviteli sebesség a választott kapcsolattól és sávszélességtől függ. A MagPi által végzett benchmark tesztek körülbelül 58,3 Mbps sebességet mutattak 2,4 GHz-en és 114 Mbps-ot 5 GHz-en. Az átviteli sebességet végső soron az RPi4B hardver korlátozza, amely egyetlen adatfolyamot támogat MSC7-tel, így az adatsebesség 72 Mbps-ra korlátozódik egy 20 MHz-es csatornán, 150 Mbps-ra egy 40 MHz-es csatornán és 325 Mbps-ra egy 80 MHz-es csatornán. A CYW43455 nem képes egyszerre WiFi-t futtatni kétsávos (2,4 GHz és 5 GHz), más néven valódi egyidejű kétsávos üzemmódban.

- **Camera Serial Interface (CSI)**

Ez a port nagy sebességű kapcsolatot biztosít egy kifejezetten erre a célra tervezett RPi kameramodullal, amely képek és videók rögzítésére képes. A CSI porthoz csatlakoztatott kameramodulok kevesebb energiát fogyasztanak, mint az USB webkamerák, így nem terhelik a tápegységet és nem merítik le az akkumulátorokat.

Az alább látható CSI csatlakozó kiosztása az RPi4B kapcsolási rajzából származik.



5. ÁBRA RASPBERRY PI 4 CSI KONNEKTOR PINOUT [33]

A CSI egy soros buszt határoz meg egy nagysebességű órajelsávval (CAM1_CN/CAM1CP) és két párhuzamosan használt adatsávval az adatok továbbítására (a 0. sáv CAM1_DN0 és CAM1_DP0, az 1. sáv pedig CAM_DN1 és CAM1_DP1). Mindegyik sávot két vezeték vezeti az alacsony feszültségű differenciális jelzéshez. A 11-es és 12-es lábakon található I2C soros órajel (SCL0) és adatvezetékek (SDA0) a kamera

funkcióinak konfigurálására/vezérlésére szolgálnak, például a képfelbontás és a képkockasebesség kiválasztására. A CAM_GPIO a kamera meghajtására és be-/kikapcsolására szolgál, amikor erre szükség van.

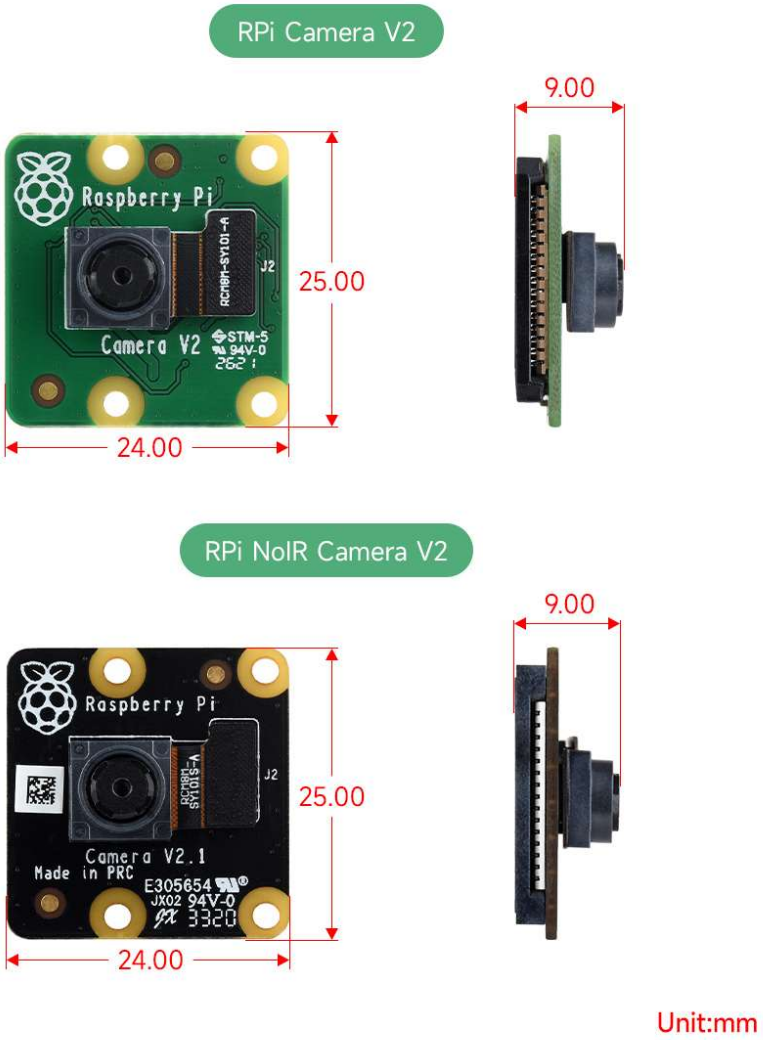
Ezen a csatlakozón keresztül kommunikál a kontroller a Raspberry Camera Module 2-vel.

5.2.1.2 Kamera

A Raspberry Pi Camera Module 2 egy kompakt, nagy teljesítményű kameramodul, amelyet kifejezetten a Raspberry Pi számítógépekhez terveztek. A modul lelke a Sony IMX219PQ típusú CMOS képérzékelő, amely 8 megapixeles felbontást biztosít, 3280×2464 aktív képponttal. A szenzor 1/4”-os optikai méretű, a pixelek mérete pedig 1,12×1,12 mikrométer. A Sony Exmor R hátulról megvilágított szenzortechnológiájának köszönhetően a kamera jó fényérzékenységgel és alacsony zajszinttel működik. A modul egy fix fókuszú optikai rendszert használ, amelynek fókusztávolsága megközelítőleg 3,04 milliméter, fényereje pedig F2.0. A látószög vízszintes irányban körülbelül 62,2°, függőleges irányban pedig 48,8°. A fix fókusz miatt a mélységélesség nagy tartományt fed le, jellemzően 10 centimétertől egészen a végtelenig.

Videós képességeit tekintve a kamera többféle felbontást és képkockasebességet támogat: 1080p felbontást 30 képkocka/másodperccel, 720p felbontást 60 fps-sel, illetve 640×480-as VGA felbontást akár 90 fps sebességgel. A modul képes akár 11,76 másodperc hosszúságú expozíciós idő alkalmazására is.

A modul fizikailag rendkívül kisméretű, mindössze körülbelül 25×24×9 milliméter nagyságú, tömege pedig nagyjából 3 gramm. A Raspberry Pi számítógéphez egy 15 tűs szalagkábellel csatlakozik a CSI-2 interfészen keresztül. Üzemi hőmérséklet-tartománya -20 °C és +60 °C között van,



6. ÁBRA RASPBERRY CAMERA MODULE 2 [33]

5.2.1.3 Feldolgozó egység

Ez a komponens tekinthető a teljes rendszer legmodulárisabb és platformfüggetlenebb elemének, mivel Java szoftverrel és a hozzá tartozó grafikus felhasználói felülettel valósult meg. A Java inherens platformfüggetlen képességeinek köszönhetően az alkalmazás zökkenőmentesen működhet gyakorlatilag bármilyen modern eszközön anélkül, hogy jelentős módosításokra vagy speciális hardverre lenne szükség. Ez a sokoldalúság lehetővé teszi, hogy a szoftver ugyanolyan jól fusson okostelefonokon, laptopokon, asztali számítógépeken, vagy akár megfelelő feldolgozási teljesítménnyel rendelkező beágyazott rendszereken is. Továbbá ez a modularitás jelentősen leegyszerűsíti a telepítést és a karbantartást, mivel a program frissítései vagy fejlesztései minimális erőfeszítéssel terjeszthetők az összes platformon. Ugyanaz a futtatható kódbázis több

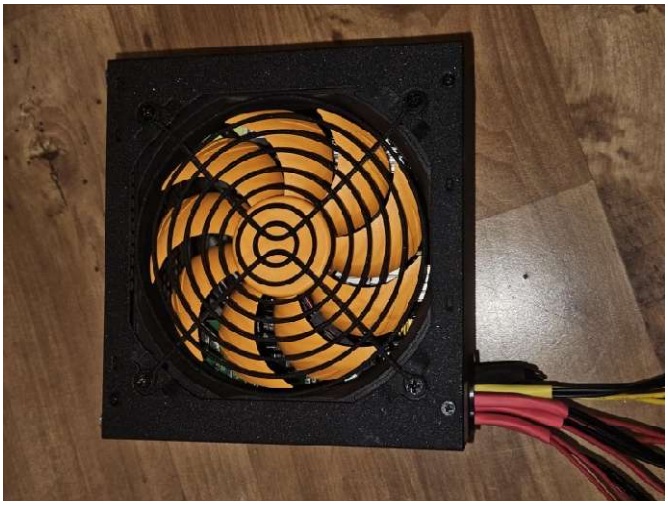
eszközön is használható, így a játékosok és az üzemeltetők kiválaszthatják a környezetükhöz legmegfelelőbb beállítást. Összefoglalva, ez az elem a darts pontozórendszer legrugalmasabb és legalkalmazkodóbb részét képviseli. A Java platformfüggetlen jellegének és grafikus felhasználói felületének integrációjának kihasználásával biztosítja, hogy az alkalmazás univerzálisan elérhető, felhasználóbarát és könnyen skálázható maradjon, függetlenül attól, hogy milyen eszközön vagy operációs rendszeren fut.

5.2.1.4 Tápellátás

A rendszer energiaellátásának megvalósításához egy 450 W-os, 80 Plus minősítésű PC-tápegységet választottam. Az eszköz a számítógépes alaplapok és perifériák számára készült, több különböző feszültségszintű kimenettel rendelkezett: +3.3 V, +5 V, +12 V1, +12 V2, –12 V és +5 Vsb. Mivel a projektben három Raspberry Pi 4 egység és egy 12 V-os LED-szalag tápellátását kellett biztosítani, az eredeti kialakítás módosítása elengedhetetlenné vált.

Az átalakítás célja az volt, hogy a tápegység biztonságosan és stabilan képes legyen egyidejűleg kiszolgálni a teljes rendszert. A három Raspberry Pi 4 egyenként 5 V / 3 A áramfelvételt igényelt, míg a LED-szalag 12 V feszültségről, 24 W teljesítménnyel működött. Így összesen közel 9 A terhelés jelentkezett az 5 V-os ágon, és körülbelül 2 A a 12 V-os ágon.

Az átalakítás során a tápegység +5 V-os ágát használtam a Raspberry Pi egységek táplálására, mivel ez az ág 13 A leadására volt képes, ami elegendő tartalékot biztosított. Kialakítottam három külön kimeneti csatlakozást, amelyek mindegyike egy-egy Raspberry Pi-hez tartozott. A külön ágak használatával elkerülhetővé vált a feszültségesés és az elektromos zavarok átvitele a modulok között.
A +12 V-os ágat a LED-szalag táplálására használtam, mivel annak áramigénye mindössze 2 A körüli volt, amit a tápegység ezen ága biztonsággal el tudott látni. A LED-ág kimenetét biztosítékkal és polaritásvédelmi diódával egészítettem ki, ezzel növelve a rendszer megbízhatóságát.



7. ÁBRA TÁPEGYSÉG ÉS KIMENETEK

A PC-tápegységek alapvetően az ATX szabvány szerint működnek, így alaplap nélkül nem indulnak el automatikusan. Ennek megoldására egy indító áramkört építettem be, amely a PS_ON (zöld) vezetéket egy földponthoz kapcsolta. Ez a módosítás lehetővé tette a tápegység önálló indítását.
A kimeneti ágakat sorkapcsokra vezettem ki, így a csatlakozások könnyen hozzáférhetők, mérhetők és szerelhetők lettek. A vezetékek megfelelő keresztmetszetű rézvezetékből készültek, így a terhelés alatti feszültségesés minimális maradt.

Az átalakítás során külön figyelmet fordítottam az érintésvédelemre és a földelésre. A tápegység fémháza megfelelően földelve maradt, a hálózati bemenetnél pedig megtartottam az eredeti zavarszűrő és biztosítékolási elemeket.
A kimeneti ágak elé egyenként biztosítékok kerültek, így egy esetleges rövidzár vagy meghibásodás esetén csak az érintett ág válik le a rendszerről, az egész táp nem kapcsolt le. Ezzel sikerült jelentősen növelni a rendszer üzembiztonságát.

Az átalakított tápegység stabil, megbízható és jól terhelhető energiaforrást biztosított a teljes rendszer számára. A módosításnak köszönhetően a három Raspberry Pi 4 és a LED-szalag egyetlen, központi tápról üzemelt. Ez a megoldás egyszerűsítette a kábelezést, csökkentette a meghibásodás esélyét, valamint biztosította a folyamatos és zavartalan működést a teljes mérőrendszerben.

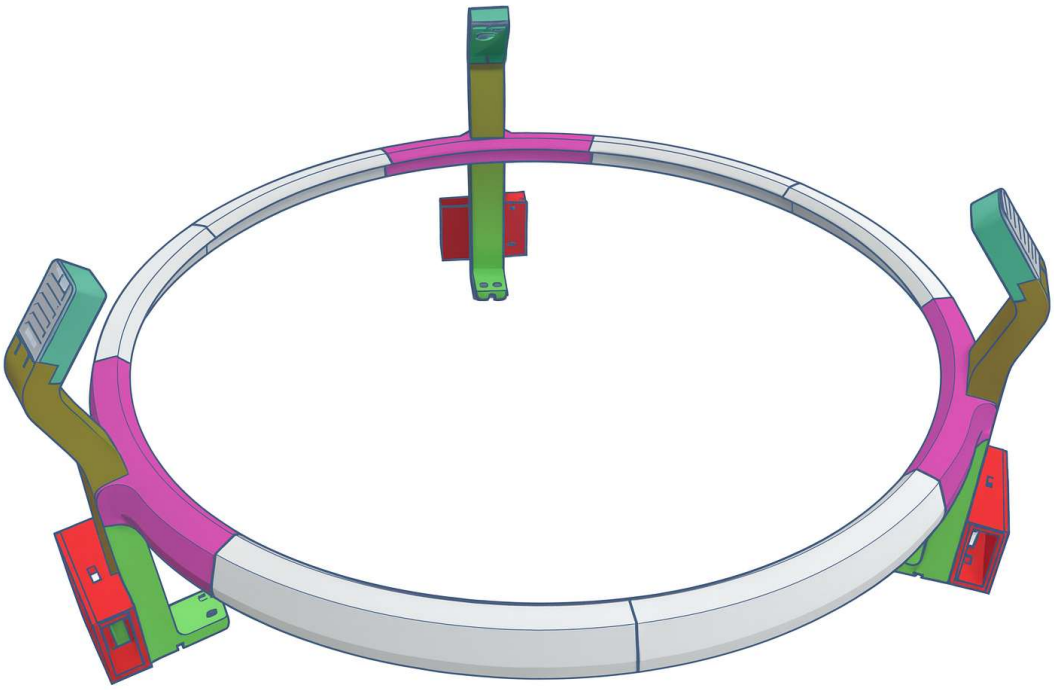
5.2.1.5 Mechanikai megoldás

Ahhoz, hogy a rendszer teljes mértékben funkcionális és megbízható egységként működjön, elengedhetetlen egy gondosan megtervezett tartószerkezet. Ez a szerkezet nemcsak a szükséges mechanikai stabilitást biztosítja az összes alkatrész számára, hanem jelentősen hozzájárul a termék általános esztétikai megjelenéséhez is. Más szóval, kettős célt szolgál: a rendszer hardverének pontos illesztését és biztonságos rögzítését biztosítja, miközben vizuálisan vonzó és professzionális külsőt is biztosít, amely fokozza a felhasználói élményt.

A tartósság és a kifinomult vizuális kialakítás elérése érdekében a tartószerkezetet OSB (Oriented Strand Board) lap és egyedi 3D nyomtatott szerkezeti elemek kombinációjával jött létre. Az OSB lap erős és stabil alapként szolgál, amely ellenáll a fizikai igénybevételnek, a rezgésnek és az ismételt használatnak, így ideális az alkatrészek rögzített pozícióinak fenntartásához működés közben. Felülete szilárd alapot biztosít a 3D nyomtatott alkatrészek és elektronikus modulok rögzítéséhez, miközben a teljes szerkezet könnyű és könnyen kezelhető marad.

A 3D nyomtatott elemeket ezzel szemben úgy terveztem, hogy pontosan pozicionálják a kamerákat, mikrovezérlőket és a LED szalagot. Geometriájukat úgy alakítottam ki, hogy biztosítsák az optimális kameraorientációt és látómezőt, lehetővé téve a képfeldolgozó rendszer maximális pontosságú működését. Ezenkívül a 3D nyomtatás modularitása megkönnyíti az egyes alkatrészek módosítását vagy cseréjét, lehetővé téve a rugalmas alkalmazkodást a különböző rendszerkonfigurációkhoz vagy a jövőbeli frissítésekhez.

Funkcionális céljukon túl a 3D nyomtatott alkatrészek a rendszer vizuális vonzerejét is fokozzák. Különböző színekben, formákban vagy felületi textúrákban nyomtathatók, így a végterméknek letisztult és modern megjelenést kölcsönöznek. Ez a tervezési odafigyelés nemcsak a használhatóságot javítja, hanem segít a rendszert egy letisztult, professzionális eszközként bemutatni, amely alkalmas demonstrációs vagy kereskedelmi alkalmazásokhoz.



8. ÁBRA 3D MODELL

A csatolt képen látható 3D-s modellt tökéletes forgásszimmetriával lett tervezve, hogy biztosítsák mind a mechanikai egyensúlyt, mind a funkcionális egységességet. Ennek fizikai megvalósítása a következő ábrán látható.

9. ÁBRA NYOMTATOTT 3D MODELL RÖGZÍTETT ÁLLAPOTBAN

A szerkezet három függőleges „oszlopból” áll, amelyek egyenletesen, 120°-os intervallumokban helyezkednek el a kör alakú keret középpontja körül. Ez a szimmetrikus elrendezés nemcsak kiváló szerkezeti stabilitást biztosít, hanem kulcsfontosságú szerepet játszik az összes kamera konzisztens képalkotási feltételeinek elérésében is. Ennek a precíz elrendezésnek köszönhetően minden kamera azonos térbeli perspektívából rögzít képet a darts táblához képest, ami jelentősen leegyszerűsíti a képfeldolgozási folyamatot. Ennek eredményeként a képfelismerő algoritmust csak egyszer kell kifejleszteni és finomhangolni egyetlen kamerához, majd egyszerű átparaméterezéssel a többire is alkalmazható, ami nagymértékben javítja a fejlesztés hatékonyságát és biztosítja az állandó pontosságot.

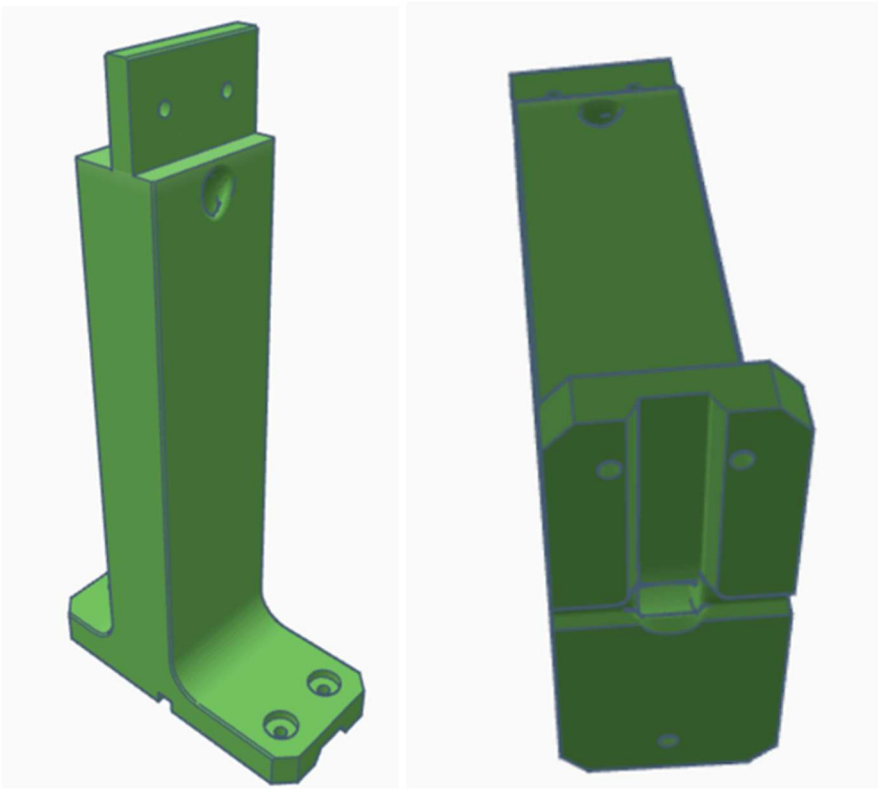
Maguk a kamerák a függőleges oszlopok felső szakaszaiban, a fejszerű szerkezetek belsejében vannak felszerelve. Két kis csavarral vannak biztonságosan rögzítve, amelyek megakadályozzák a nem kívánt rezgést vagy elmozdulást, amely befolyásolhatná a képstabilitást. Minden fej rácsszerű kialakítású, amely fontos hőfunkciót lát el, ugyanis lehetővé teszi a levegő keringését az elektronikus alkatrészek körül, megakadályozva a kameramodulok túlmelegedését hosszabb működés közben.

Minden oszlop alján piros házak találhatók, amelyek a Raspberry Pi 4 mikrovezérlők befogadására szolgálnak. Ezek a házak egyik oldalukon nyitottak a megfelelő légáramlás és hőelvezetés biztosítása érdekében, biztosítva a mikrovezérlők stabil teljesítményét folyamatos működés közben is. Az elrendezés a portokhoz és csatlakozókhoz való könnyű hozzáférést is megkönnyíti, lehetővé téve a kényelmes karbantartást és újraprogramozást.

A körgyűrű belső kerületét úgy terveztem, hogy befogadjon egy LED-szalagot, amely egyenletes megvilágítást biztosít a darts tábla felületén. Ez az integrált világítás biztosítja, hogy mindhárom kamera konzisztens fényerő- és kontrasztviszonyok mellett rögzítsen képeket, ezáltal növelve a képérzékelés megbízhatóságát és csökkentve az árnyékok vagy a környezeti fényváltozások hatását. Ezenkívül az egyik oszlop egy dedikált kábelcsatornával rendelkezik, amely biztonságosan elvezeti a táp kábelt a LED-rendszerhez.

Tekintettel a működési környezetre, ahol a dartsok időnként célt tévesztettek és a környező területet találták el, az elektromos vezetékek védelme rendkívül fontos. Ezért minden kábelt a szerkezet megerősített szakaszain belül kell elvezetni, hogy

megakadályozzuk az éles nyílak okozta véletlen sérüléseket. Ez nemcsak a biztonságot növeli, hanem a teljes rendszer élettartamát és megbízhatóságát is meghosszabbítja.



10. ÁBRA KÁBELCSATORNA

6 Szoftver komponensek

6.1 Képfeldolgozó szoftver

Ez a szoftver komponens Python nyelven íródott és feladata volt a kamera inicializálása, a referencia-képek kezelése, az új dobások felismerése, a nyíl hegyének meghatározása, valamint az ebből számított pontérték meghatározása és mentése.

Python verzió: 3.9.13

Az alábbiakban bemutatom a main() függvény működését és logikáját.

INICIALIZÁLÁS ÉS MODELL BETÖLTÉSE:

A program indításakor a rendszer először naplózási (logging) beállításokat hajt végre, majd megkísérli betölteni a tanuló modellt (*tip_correction_model_v4.pkl*), amely a nyíl hegypontjának finomkorrekcióját végezi el.
Ha a modell nem volt elérhető, a program nyers, gépi tanulás nélküli detektálási módba kapcsol.

A kód ezen része biztosította, hogy a program képes legyen mind offline (modell nélkül), mind online (modell alapú) feldolgozásra, ami növelte a rendszer rugalmasságát.

MODELL:

A hegypont korrekciójához egy úgynevezett felügyelt tanuláson alapuló gépi tanulási modellt hoztam létre, amelynek célja az volt, hogy a nyíl detektált hegypontját a valós fizikai pozícióhoz igazítsa.
A modell betanítása offline történt, korábban rögzített képi adatok alapján.

A tanításhoz minden dobás után a rendszer a *trainer.trainer_mode()* függvény segítségével mentette el a detektált hegypont koordinátáit és a hozzájuk tartozó valós, kézzel korrigált pozíciókat.
A képekből a *neural.extract_patch_around_tip()* függvény vágott ki kis méretű képrészleteket (patch-eket), amelyek középpontja a detektált hegypont volt.

A tanulómodellt a scikit-learn könyvtár segítségével építettem fel, és a kísérletek során több architektúrát is kipróbáltam.
A legjobb eredményt egy Random Forest Regressor adta, mivel jól kezelte a nemlineáris összefüggéseket és zajos adatokat, miközben gyorsan tanult.

A modell minden mintára két valós értéket jósolt:

$$[\widehat{\Delta X}, \widehat{\Delta Y}] = f_{model}(I_{norm})$$

A cél a predikciós hiba minimalizálása volt, amit négyzetes hibafüggvénnyel mértem:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\Delta X_i - \widehat{\Delta X_i}\right)^2 + \left(\Delta Y_i - \widehat{\Delta Y_i}\right)^2\right]$$

A tanulás során az adatokat 80–20%-os arányban osztottam fel tanító és teszt halmazra. A modell teljesítményét az átlagos euklideszi hibával (E_{avg}) értékeltem:

$$E_{avg} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\left(\Delta X_i - \widehat{\Delta X_i}\right)^2 + \left(\Delta Y_i - \widehat{\Delta Y_i}\right)^2}$$

A tréning végén a modellt joblib formátumban mentettem el (*tip_correction_model_v4.pkl*), amelyet a főprogram futáskor automatikusan betöltött.

KAMERA ÉS REFERENCIA KÉPEK INICIALIZÁLÁSA

Az *image_process.camera_init()* függvény indította el a csatlakoztatott Raspberry Pi kamera modul inicializálását.
Ezt követően a program létrehozta a referenciaképeket, amelyek a tábla háttérképét jelentették dobás nélkül.
A későbbi képkockák ehhez a referenciához kerültek összehasonlításra, így a mozgás vagy pontosabban a dobott nyíl könnyen detektálható volt.

Matematikailag a mozgásdetektálás az alábbi képlettel írható le:

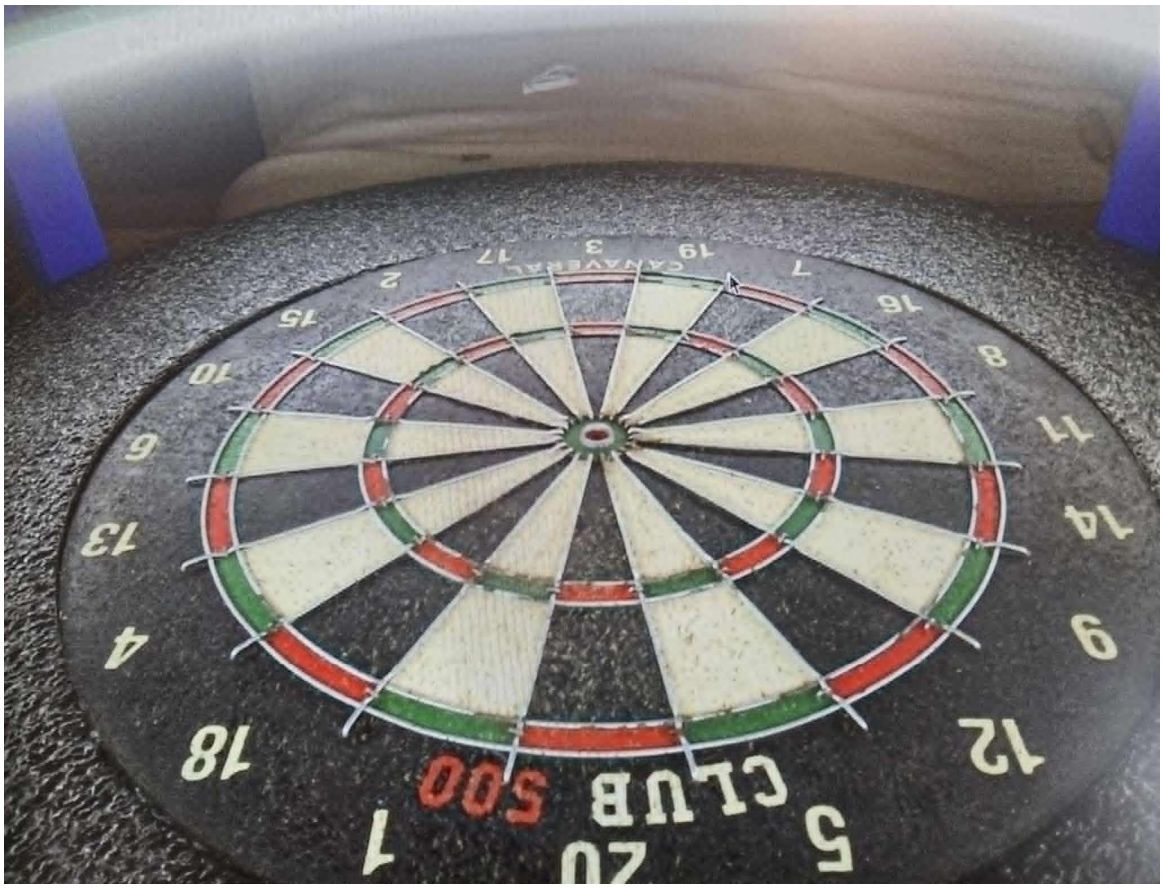
$$D(x, y) = \left|I_t(x, y) - I_{ref}(x, y)\right|$$

ahol

- $I_t(x, y)$a pillanatnyi képkocka intenzitása,
- $I_{ref}(x, y)$a referencia-kép intenzitása,
- $D(x, y)$pedig a különbségkép, amely a mozgást mutatja.

A *detect_new_dart()* függvény ezt a különbségképet binárissá alakította, majd zajszűrést és terület-számítást végzett.

Ha a különbség meghaladta a küszöbértéket ($D(x, y) > T$), és a detektált objektum területe (A) nagyobb volt a minimális értéknél ($A > A_{min}$), akkor a rendszer új nyílként azonosította a mozgást. A küszöbértékek pontos meghatározás segít abban, hogy a szoftver tudja mikor megy oda és veszi ki a nyilat a táblából a játékos és mikor történt dobás.



11. ÁBRA KAMERA ALAPKÉPE

SZEMBÖLI NÉZET LÉTREHOZÁSA

Miután a program új nyilat észlelt, az aktuális képkockát előbb perspektívatranszformáción keresztül alakította át elölnézeti képpé a *create_front_view()* függvénnyel. Ez az átalakítás egy projektív transzformáció, amelyet a következő mátrixegyenlet ír le:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \qquad ahol\ H \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

A transzformáció eredményeképp a kamera ferde nézetéből származó kép elülnézeti vetületbe került, ami a pontosság szempontjából kulcsfontosságú volt.

A NYÍL HEGYÉNEK DETEKTÁLÁSA

A *detect_dart_tip_fast()* függvény ezután a nyíl hegyét detektálta.
A hegy keresésénél az algoritmus a kontraszt-különbségekre és a kontúrformákra támaszkodott.
Az éldetektálást Sobel-operátorral végezte:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I$$

A gradiens nagysága:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

A nyíl hegye azon pontként lett azonosítva, ahol a kontúr záródott, és a lokális fényességmaximum illeszkedett a gradiensirányhoz.

MODELL ALAPÚ HEGYPONT KORREKCIÓ

Ha a betöltött modell elérhető volt, a *neural.extract_patch_around_tip()* függvény egy kisebb képrészletet vágott ki a hegypont körül.
Ez a részlet normalizálva lett ($I_{norm} = I/255.0$), majd a betanított neurális háló a hegypont elmozdulását becsülte meg:

$$[\Delta x, \Delta y] = f_{model}(I_{norm})$$

A korrigált hegypont ezután a következőképpen számítódott:

$$(x_c, y_c) = (x_t + \Delta x, y_t + \Delta y)$$

Ezzel sikerült csökkenteni a torzításokat, amelyeket a kamera dőlésszöge és a lencsetorzítás okozott.

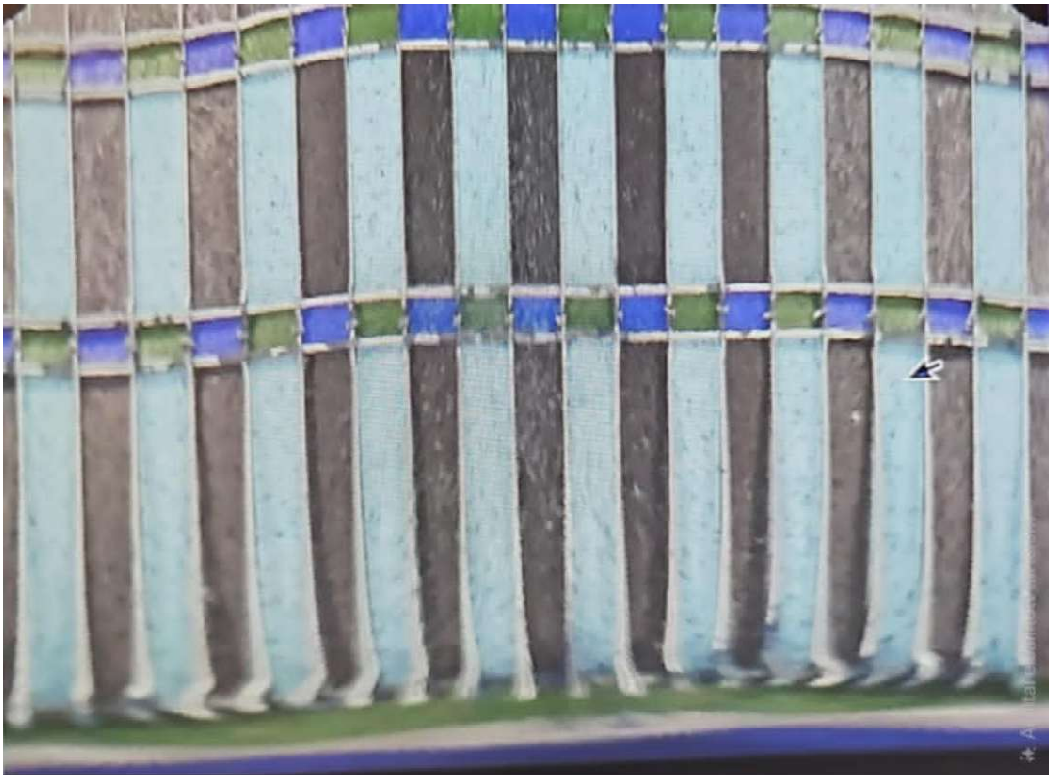
KOORDINÁTA ÁTALAKÍTÁSA

A korrigált hegypont a *front_to_polar_xy()* függvényen keresztül került polárkoordinátás rendszerbe.
Ez a lépés a tábla geometriai szimmetriáját használta ki, hiszen a pontszámokat a szögtartomány és a sugár függvényében lehet meghatározni.

Az átalakítás alapegyenletei:

$$r = \sqrt{(x_c - x_0)^2 + (y_c - y_0)^2}, \qquad \theta = arctan\left(\frac{y_c - y_0}{x_c - x_0}\right)$$

ahol (x_0, y_0)a tábla közepe.



12. ÁBRA POLÁR KOORDINÁTARENDSZERBE TRANSZFORMÁLT KÉP

PONTSZÁMÍTÁS

A *calculate_score.score_dart()* függvény ezután a (r, θ)értékek alapján meghatározta, melyik gyűrűbe és szektorba esett a találat.
A szektorokat a *sectors.json*, a gyűrűhatárokat pedig a *rings.json* fájl tartalmazta.

A pontszám tehát függvényként írható le:

$$S = f(r, \theta)$$

ahol S a dobás pontértéke.

REFERENCIA KÉPEK FRISSÍTÉSE

Minden sikeres dobás után a program frissítette a reference_raw és reference_front képeket. Ez azért volt szükséges, hogy a következő detektálásnál a korábbi nyíl ne befolyásolja az új mozgás-azonosítást.

A referencia-kép frissítését az alábbi rekurzív módon lehet értelmezni:

$$I_{ref}^{(t+1)} = \alpha I_t + (1 - \alpha) I_{ref}^t$$

ahol α kis érték (pl. 0,1), ami simítja a háttér frissülését.

6.2 Java szoftver

A rendszer második fő komponense egy Spring Boot alapú Java backend alkalmazás, amely a képfeldolgozó modul által szolgáltatott adatokat fogadja, feldolgozza, tárolja és továbbítja a felhasználói felület, illetve a statisztikai modul felé. A komponens egyaránt felel a mérkőzések, játékosok és dobások nyilvántartásáért, valamint a rendszer egészének üzleti logikájáért.

A képfeldolgozó szoftver összegyűjti az aktuális adatokat és egy JSON objektumba kerülnek, majd Wi-Fi kapcsolaton keresztül, HTTP POST kéréssel továbbítódnak a Spring Boot alkalmazás felé. A kommunikáció REST alapú, a képfeldolgozó kliens a backend megfelelő végpontjára küldi a feldolgozott dobási adatokat.

A beérkező adatokat a backend első lépésben sémaszinten ellenőrzi. A JSON objektum kötelező mezői közé tartozik a mérkőzés azonosítója (matchId), a játékos azonosítója (playerId), a kör (roundNumber), a nyíl sorszáma (dartNumber) és a dobott érték (hitValue). Az adatcsomag tartalmazhat egy totalScoreAfterThrow mezőt is, amelyet a képfeldolgozó rendszer számít ki, így a backend képes összevetni a két fél által kalkulált pontszámot és az esetleges eltéréseket azonosítani. Az üzenetben található idempotencyKey mező egyedi kulcsot biztosít arra az esetre, ha a képfeldolgozó modul hálózati hiba miatt újraküldené ugyanazt a dobást: a backend ez alapján felismeri, ha egy dobás már korábban rögzítésre került, és nem hoz létre duplikált rekordot.

A backend alkalmazás az adatokat egy tranzakciós feldolgozási folyamaton keresztül kezeli. Először ellenőrzi, hogy a megadott mérkőzés (match_id) és játékos (player_id)

létezik-e, valamint hogy a játékos valóban részt vesz-e az adott mérkőzésben a match_players tábla alapján. Ezután megnézi a mérkőzés aktuális állapotát a matches táblában, csak „in_progress” állapotú meccs esetén enged további dobásokat. Ha minden előfeltétel teljesül, a rendszer új rekordot hoz létre a throws táblában, amely tartalmazza a dobás minden releváns adatát (kör, nyíl sorszáma, érték, eredő pontszám és időbélyeg). Ezzel párhuzamosan frissül a match_players tábla is, ahol az adott játékos hátralévő pontszámát (score_left) a dobás eredményének megfelelően csökkenti.

A tranzakció sikeres lefutása után a Spring Boot komponens eldönti, hogy a dobás véget vetett-e a mérkőzésnek. Amennyiben a játékos elérte a nullát vagy más, a szabályokban meghatározott feltétel teljesül, a rendszer a matches tábla status mezőjét „finished” értékre állítja, és rögzíti a befejezési időpontot (end_time). A meccs lezárását követően frissül a leaderboard tábla is, amely a játékos hosszú távú statisztikáit tartalmazza: növekszik a matches_played számláló, győzelem esetén pedig a matches_won mező. Ezen felül a rendszer számítja és eltárolja a legmagasabb elért pontszámot (highest_score) és az átlagos dobási teljesítményt (average_score), így biztosítva a játékosok objektív összehasonlíthatóságát.

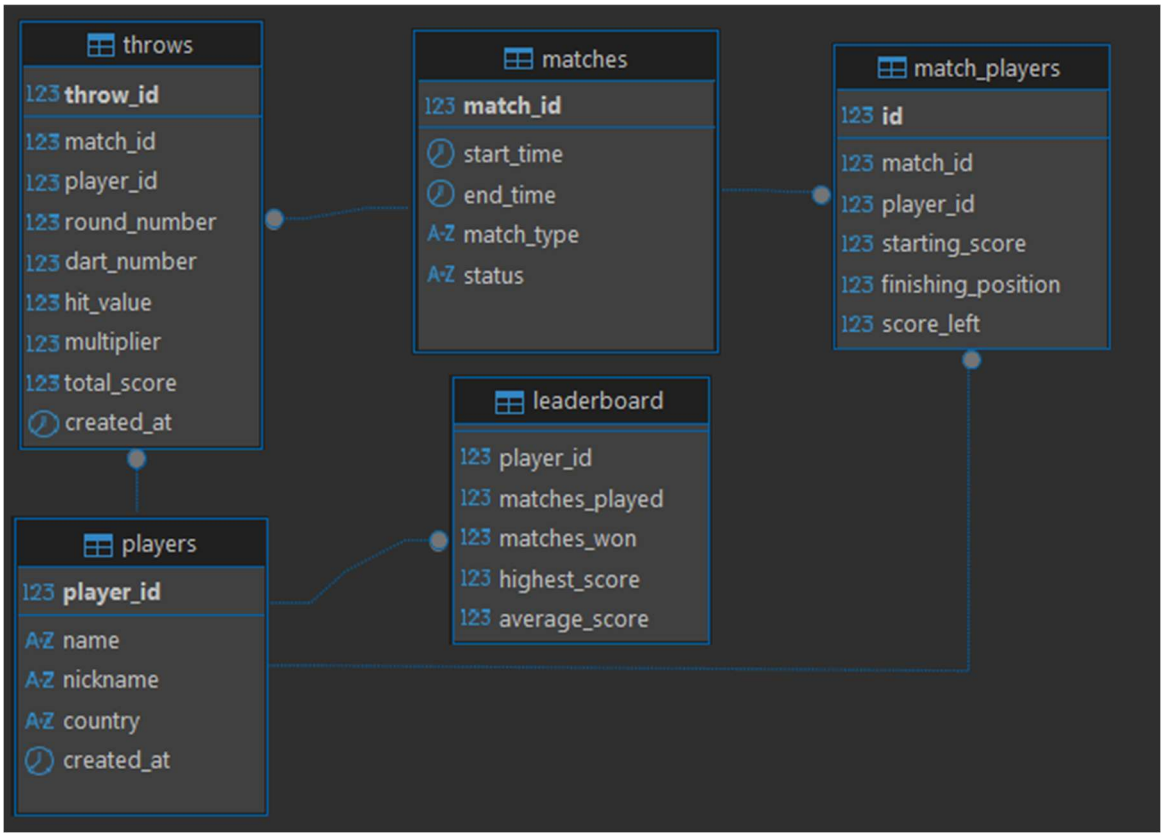
A Spring Boot alkalmazás nemcsak a dobások rögzítését végzi, hanem valós idejű metrikákat is szolgáltat. A rendszer folyamatosan gyűjti és aggregálja az adatokat, például a percenkénti dobások számát, az aktív mérkőzések mennyiségét, a feldolgozási késleltetéseket és a dobások átlagos feldolgozási idejét. Ezeket az adatokat a Micrometer könyvtár segítségével a Spring Boot beépített „actuator” interfészén keresztül lehet lekérdezni, ami lehetővé teszi a rendszer teljesítményének és megbízhatóságának monitorozását.

A modulok között a kommunikáció REST-alapú és JSON formátumot használ, így a rendszer egyszerűen integrálható külső alkalmazásokkal vagy más szoftverkomponensekkel. Az architektúra rétegei világosan elkülönülnek, a Controller réteg felel az adatok fogadásáért és a válaszok küldéséért, a Service réteg tartalmazza az logikát és a tranzakciókezelést, míg a Repository réteg a PostgreSQL adatbázissal történő kommunikációt valósítja meg a Spring JDBC és Liquibase segítségével.

Az így kialakított rendszer biztosítja, hogy a Python képfeldolgozó modul által érzékelt események biztonságosan, konzisztensen és valós időben kerüljenek be az

adatbázisba. A backend alkalmazás egyszerre látja el a valós idejű játékirányítást, a hosszú távú statisztikai elemzést és a rendszermonitorozást, miközben megbízható, skálázható és könnyen továbbfejleszthető alapot biztosít a darts mérkőzések digitális feldolgozásához.

6.2.1 Adatbázis



13. ÁBRA ADATBÁZIS STRUKTÚRA

A rendszer adatbázisa a darts mérkőzések lebonyolítását és statisztikai feldolgozását támogatja. A struktúra relációs modellre épül, amelyben az entitások között egyértelmű idegen kulcsos kapcsolatok biztosítják az adatintegritást és a kaszkádolt törlést. Az adatbázis elsődleges célja a játékosok, mérkőzések, dobások és eredmények nyilvántartása, valamint a hosszú távú teljesítményadatok rögzítése.

A ’players’ tábla tartalmazza a játékosok alapadatait. Minden játékos egyedi azonosítóval (player_id) rendelkezik, amelyet a rendszer automatikusan generál. A tábla tárolja továbbá a játékos nevét, becenevét, országát, valamint a rekord létrehozásának időpontját (created_at). A name mező egyedi azonosításra is alkalmas, mivel a backend logika ennek alapján dönti el, hogy egy játékos már létezik-e az adatbázisban, vagy új bejegyzést kell létrehozni számára.

A ’matches’ tábla a rendszer központi eleme, hiszen itt kerülnek rögzítésre az egyes mérkőzések. Minden mérkőzés egyedi match_id értékkel rendelkezik, továbbá tartalmazza a kezdési (start_time) és befejezési (end_time) időpontokat, a játéktípust (match_type), valamint a mérkőzés aktuális állapotát (status). A match_type mezőben az adott játékmód (például 301 vagy 501) kerül tárolásra, míg a status mező az életciklust írja le, amely tipikusan „in_progress” (folyamatban) vagy „finished” (befejezett) értéket vehet fel.

A ’match_players’ tábla kapcsolótábla szerepet tölt be a játékosok és a mérkőzések között, lehetővé téve, hogy több játékos vegyen részt ugyanazon mérkőzésen. Az id az elsődleges kulcs, míg a match_id és player_id mezők idegen kulcsok a matches és a players táblákra. Ezen kívül a tábla a mérkőzéshez kapcsolódó játékos-specifikus adatokat is tárolja, úgymint a kezdő pontszámot (starting_score), a játékos aktuális hátralévő pontját (score_left), valamint a befejezési helyezést (finishing_position). A két idegen kulcs közötti kapcsolat biztosítja, hogy egy játékos egy mérkőzésen belül csak egyszer szerepelhessen, így megakadályozva az adatredundanciát. A kapcsolatok „on delete cascade” beállítása garantálja, hogy ha egy mérkőzés vagy játékos törlésre kerül, az ehhez tartozó rekordok a match_players táblában automatikusan eltávolításra kerülnek, így megőrizve az adatintegritást.

A ’throws’ tábla a rendszer legdinamikusabb eleme, amely minden egyes dobást külön sorban tárol. Minden dobás egyedi throw_id azonosítót kap, és hivatkozást tartalmaz arra, hogy melyik mérkőzéshez (match_id) és melyik játékoshoz (player_id) tartozik. A tábla mezői részletesen rögzítik a dobás körülményeit és eredményét: a kör sorszámát (round_number), az adott nyíl számát (dart_number), a találat értékét (hit_value), a szorzót (multiplier), valamint az adott dobás után elért összesített pontszámot (total_score). A created_at mező időbélyeget tartalmaz, amely lehetővé teszi az események időrendi visszakövetését. A throws tábla segítségével a rendszer pontosan rekonstruálni tudja egy mérkőzés lefolyását és a játékosok teljesítményét dobásról dobásra.

A teljesítményadatokat a ’leaderboard’ tábla foglalja össze. Ez a tábla szintén a player_id-re épül, amely idegen kulcsként hivatkozik a players táblára. Minden játékoshoz statisztikai mutatók tartoznak, például a lejátszott mérkőzések száma (matches_played), a megnyert mérkőzések száma (matches_won), a legmagasabb elért pontszám (highest_score), valamint az átlagos pontszám (average_score). Ez a tábla alkalmas a játékosok teljesítményének hosszú távú elemzésére és rangsorok generálására.

Ez a struktúra ideális alapot biztosít egy többjátékos darts mérkőzés automatizált eredményrögzítéséhez és statisztikai feldolgozásához. A tábla-kapcsolatok jól tükrözik az alkalmazás logikai rétegében is használt modellhierarchiát, ahol a Player, Match, Throw és Leaderboard entitások között egyértelmű egy-több relációk valósulnak meg.

6.3 Grafikus megjelenítés

A rendszer felhasználói felületét (frontend) egy statikus, HTML-, CSS- és JavaScript-alapú webes alkalmazás biztosítja, amely a Spring Boot projekt erőforrásai közé integrálva fut. Ennek köszönhetően nincs szükség külön webszerverre vagy kliensoldali build környezetre, mivel a Spring Boot képes közvetlenül kiszolgálni a könyvtárban található HTML fájlokat. Az alkalmazás elindításakor a Spring Boot beépített Tomcat szervere gondoskodik a kérések kezeléséről, így a felhasználó a böngészőben egyszerűen elérheti a főoldalt a http://localhost:8080 címen.

A frontend egyszerű, de funkcionálisan teljes körű felületet nyújt a felhasználónak: az „Új játék” gomb megnyomásával elindítható egy új mérkőzés, ahol a felhasználó két játékos nevét adhatja meg, valamint kiválaszthatja a játék típusát (például 301 vagy 501). A JavaScript logika ezután egy JSON-objektumot állít össze, amely tartalmazza a felhasználói űrlapban megadott adatokat, majd ezt HTTP POST kéréssel elküldi a Spring Boot backend felé. A kapcsolat REST API-n keresztül valósul meg, a kliens a /api/game/start végpontot hívja meg, amelyet a GameController osztály kezel.

A kommunikáció az HTTP protokoll szabványos elemeit használja, a kliensoldali JavaScript a fetch() függvénnyel küld aszinkron kérést, amelynek törzse JSON formátumban tartalmazza az adatokat. A kérés fejléce „application/json” értéket kap, így a backend azonnal tudja, hogy a bejövő tartalmat JSON-ként kell deszerializálnia. A Spring Boot a Jackson könyvtár segítségével automatikusan átalakítja a JSON-objektumot egy Java objektummá, amelyet a @RequestBody annotációval ellátott paraméter vesz át. Ennek típusa a korábban definiált MatchStartRequest osztály, amelyben a player1, player2 és mode mezők reprezentálják a kliens által küldött adatokat. Miután a kérés beérkezik, a backend a MatchService komponens segítségével elvégzi az új mérkőzés létrehozásához szükséges lépéseket. A szolgáltatás először ellenőrzi, hogy a megadott játékosok szerepelnek-e az adatbázisban. Amennyiben nem, új rekordokat hoz létre számukra a

players táblában. Ezután a rendszer új mérkőzést hoz létre a matches táblában, beállítja a kezdési időpontot, a játéktípust és az alapértelmezett státuszt („in_progress”). Végül a két játékost hozzákapcsolja a mérkőzéshez a match_players táblán keresztül, ahol rögzíti az induló pontszámot (301 vagy 501) és a hátralévő pontokat.

A tranzakció sikeres lefutása után a Spring Boot válaszként egy HTTP 200 státuszkódot küld vissza, valamint egy egyszerű szöveges üzenetet vagy JSON-objektumot, amely tartalmazza az új mérkőzés azonosítóját (match_id). A frontend ezt az értéket a fetch() hívás befejezése után megkapja, és a JavaScript segítségével kiírja a képernyőre a „Meccs elindult, azonosító: 12” formában. A felhasználó számára ez azonnali vizuális visszajelzést ad arról, hogy a rendszer regisztrálta a kérést, és a mérkőzés ténylegesen létrejött az adatbázisban.

A továbbiakban a frontend feladata a mérkőzés során történő események megjelenítése és a dobások követése. Amikor a Python képfeldolgozó szoftver új dobásról küld adatot a backendnek, az alkalmazás a throws táblában rögzíti az eredményt, majd a frontend képes lekérdezni a legfrissebb állapotot. A kliens a /api/match/{id}/state végpont segítségével lekérheti az adott mérkőzés aktuális állapotát, a játékosok nevét, a fennmaradó pontokat és az eddigi dobások listáját.

A rendszer ezen szakaszában különösen fontos a valós idejű adatfrissítés kezelése. A jelenlegi implementációban a frontend bizonyos időközönként lekéri az aktuális mérkőzés állapotát, ami úgynevezett polling technikával valósul meg. A backend ezeket a kéréseket a GameController egyik GET végpontja fogadja, amely az adatbázisból összegyűjti az aktuális állapotot, és egy JSON struktúrában küldi vissza a kliensnek. Későbbi továbbfejlesztésként lehetőség van a WebSocket kommunikáció bevezetésére, amely lehetővé teszi, hogy a backend azonnal értesítse a frontendet, ha új dobás érkezik vagy megváltozik a mérkőzés állapota, így valós idejű frissítés jönne létre a böngészőben.

A frontend és a backend kapcsolatának másik kulcseleme a hibakezelés**.** A JavaScript oldalon a fetch() függvény try-catch blokkban fut, így hálózati hiba vagy érvénytelen adat esetén a rendszer felhasználóbarát hibaüzenetet tud megjeleníteni. A backend oldalán minden beérkező kérés ellenőrzésen megy keresztül, ha bármely mező hiányzik, hibás típusú, vagy a megadott játékos nem létezik az adatbázisban, a Spring Boot automatikusan 400-as hibát küld vissza, amelyet a frontend kezel. Az ilyen szigorú validálás

kulcsfontosságú a rendszer stabilitása és adatkonzisztenciája szempontjából, hiszen megakadályozza, hogy hibás vagy hiányos adatok kerüljenek az adatbázisba.

A felhasználói felület vizuális megjelenését letisztult, modern dizájn jellemzi, amely CSS segítségével valósul meg. A gombok, beviteli mezők és visszajelző elemek egységes színvilágot és tipográfiát használnak, biztosítva a professzionális megjelenést és az egyszerű használatot. A frontend így nemcsak a rendszer működtetését szolgálja, hanem a felhasználói élményt is támogatja.

A két réteg közötti kapcsolat szigorúan adatvezérelt, a frontend nem tartalmaz üzleti logikát, hanem kizárólag az adatok megjelenítéséért felel, míg a backend végzi az adatfeldolgozást és az üzleti szabályok érvényesítését. Ez a rétegzett architektúra lehetővé teszi a kód újrahasznosítását és karbantartását, valamint biztosítja, hogy a későbbiekben akár más típusú kliensalkalmazások (például mobilapp vagy asztali program) is könnyedén csatlakozhassanak ugyanahhoz az API-hoz.

7 Továbbfejlesztési lehetőségek

A kialakított rendszer számos ponton tovább fejleszthető a megbízhatóság, a felhasználói élmény és a skálázhatóság növelése érdekében. A mechanikai felépítés tekintetében az egyik legfontosabb jövőbeli fejlesztési irány a 3D-nyomtatott tartószerkezet módosítása. A jelenlegi verzióban a kamera és a mikrokontroller közötti szalagkábel kívül helyezkedik el, így sérülékenyebb és kevésbé esztétikus. A következő fejlesztés célja egy olyan továbbfejlesztett konstrukció létrehozása, amely a kábeleket és magát a mikrokontrollert is teljes egészében a tartó belsejében rejti el. Ez egyrészt növeli a mechanikai védelmet, másrészt letisztultabb, professzionálisabb külsőt ad a rendszernek.

A Python képfeldolgozó algoritmusok pontosságának további növelése. A jelenleg alkalmazott korrekciós modell korlátozott mennyiségű tanítóadat alapján készült, így a rendszer pontossága tovább javítható nagyobb adathalmaz gyűjtésével és feldolgozásával. A hosszabb távú cél egy új, komplexebb gépi tanulási modell kialakítása, amely a jelenleginél robusztusabban kezeli a különböző fényviszonyokat, kameraállásbeli eltéréseket és képzajokat. Egy ilyen fejlettebb modell a nyíl hegypontjának becslését jelentősen pontosíthatja, csökkentve a hibaarányt és növelve a rendszer megbízhatóságát.

A backend komponens esetében szintén több irányban van lehetőség bővítésre. A jelenlegi rendszer elsősorban lokális, egy adott helyszínen futó darts mérkőzések kezelésére készült. A jövőben cél egy olyan online platform létrehozása, amely lehetővé teszi a felhasználók számára, hogy távolról, interneten keresztül mérkőzzenek meg egymással. Ehhez szükség van felhasználói fiókok, autentikáció és profilkezelés bevezetésére, valamint egy olyan infrastruktúrára, amely valós időben képes a különböző eszközökről érkező dobásadatok szinkronizálására. Ez jelentősen növelné a rendszer interaktivitását és a felhasználói közösség bevonását.

Végül egy hosszabb távú, koncepcionálisan új fejlesztési irány egy olyan megoldás kialakítása, amelyben a képfeldolgozást nem külön kamerarendszer végzi, hanem maga a felhasználó eszköze, például egy okostelefon. Ebben az esetben az alkalmazás előfizetéses modellben lenne elérhető, a felhasználó telefonjának kamerája rögzítené a dobásokat, és a képfeldolgozó algoritmus közvetlenül a mobil eszközön futna le. A felismerési eredmények ezt követően a backend szerver felé továbbítódnának. Ez a megoldás drasztikusan csökkentené a rendszer hardveres költségeit, könnyebben hozzáférhetővé tenné a technológiát szélesebb felhasználói kör számára, és új üzleti modell megvalósítására is lehetőséget nyitna.

8 Összefoglalás

A szakdolgozat célja egy olyan automatikus darts pontszámító rendszer megtervezése és megvalósítása volt, amely valós időben képes a darts játék során keletkező találatok felismerésére, értelmezésére és rögzítésére. A rendszer alapját három Raspberry Pi 4 mikroszámítógép és a hozzájuk kapcsolt Raspberry Pi Camera Module 2 kamerák adják, amelyek több nézőpontból rögzítik a darts tábla felületét. A képfeldolgozás Python nyelven, OpenCV könyvtár segítségével történik, míg a mért értékek feldolgozását, tárolását és a grafikus megjelenítéshez szükséges adatközlést egy különálló, Spring Boot alapú Java backend végzi.

A dolgozat részletesen bemutatja a steel-tip darts táblák fizikai felépítését és működését, ami alapvető a hiteles képfeldolgozás megtervezéséhez. A hardverkomponensek kiválasztása során elsődleges szempont volt a megbízhatóság, a valós idejű működéshez szükséges teljesítmény, valamint a rendszer hosszú távú bővíthetősége. A kamerák elhelyezéséhez és a Raspberry Pi egységek rögzítéséhez egy egyedi, 3D-nyomtatott és OSB alapú mechanikai szerkezet készült, amely biztosítja a fix perspektívát, az optimális megvilágítást és a kábelek védelmét. A homogén fényviszonyokat egy körgyűrűbe épített LED szalag garantálja, jelentősen javítva a detekció stabilitását.

A képfeldolgozás több lépésben valósul meg. A rendszer referenciaképet készít dobás nélkül, majd mozgásdetektálással határozza meg az új nyíl érkezését. A kamera dőlésszögeiből adódó torzítást perspektívatranszformáció kompenzálja, amely elölnézeti képet hoz létre a tábla síkjáról. A nyíl hegypontjának meghatározása kontrasztalapú detekcióval történik, amelyet opcionálisan egy gépi tanulási modell finomít. A hegypontot a rendszer polárkoordinátákba transzformálja, majd kiszámítja a pontértéket a tábla geometriai paraméterei alapján. Minden dobás után frissülnek a referenciaképek, biztosítva, hogy a táblán maradt nyilak ne zavarják a következő detektálást.

A Spring Boot alapú backend fogadja a kameráktól érkező adatokat, elvégzi a szükséges érvényességi ellenőrzéseket, kezeli a mérkőzéseket, játékosokat, dobásokat, és a rögzített adatokat PostgreSQL adatbázisban tárolja. A rendszer gondoskodik a valós idejű pontszámításról, a mérkőzés állapotainak követéséről és a hosszú távú statisztikák frissítéséről. A grafikus felület a játék közben élő eredményeket jeleníti meg, a mérkőzés végén pedig részletes statisztikai összefoglalót biztosít.

A fejlesztett rendszer bizonyította, hogy többkamerás, Python alapú képfeldolgozással és egy modern, Java alapú backend architektúrával hatékonyan automatizálható a darts pontszámítás folyamata. A dolgozat bemutatja, hogy megfelelő hardver és szoftvertervezéssel megbízható, bővíthető és valós időben működő megoldás hozható létre, amely alkalmas lehet további kutatások, fejlesztések vagy akár későbbi kereskedelmi rendszerek alapjául is.

9 Summary

The aim of the thesis was to design and implement an automatic darts scoring system that is capable of recognizing, interpreting and recording hits in real time during a darts game. The system is based on three Raspberry Pi 4 microcomputers and the Raspberry Pi Camera Module 2 cameras connected to them, which record the darts board surface from multiple perspectives. Image processing is done in Python using the OpenCV library, while the processing, storage and data communication required for graphical display of the measured values are performed by a separate, Spring Boot-based Java backend.

The thesis presents in detail the physical structure and operation of steel-tip dartboards, which is essential for the design of authentic image processing. The primary considerations when selecting hardware components were reliability, the performance required for real-time operation, and the long-term expandability of the system. A unique, 3D-printed and OSB-based mechanical structure was created to place the cameras and mount the Raspberry Pi units, ensuring a fixed perspective, optimal illumination, and cable protection. Homogeneous lighting conditions are guaranteed by an LED strip built into a circular ring, significantly improving the stability of the detection.

Image processing is performed in several steps. The system takes a reference image without a throw, then uses motion detection to determine the arrival of a new arrow. Distortion due to camera tilt angles is compensated for by perspective transformation, which creates a front view image of the board plane. The arrow tip is determined by contrast-based detection, optionally refined by a machine learning model. The system transforms the tip into polar coordinates and then calculates the point value based on the geometric parameters of the board. The reference images are updated after each throw, ensuring that arrows remaining on the board do not interfere with the next detection.

The Spring Boot-based backend receives data from cameras, performs the necessary validity checks, manages matches, players, throws, and stores the recorded data in a PostgreSQL database. The system provides real-time scoring, tracking of match states, and updating long-term statistics. The graphical interface displays live results during the game and provides a detailed statistical summary at the end of the match.

The developed system has proven that the darts scoring process can be effectively automated with multi-camera, Python-based image processing and a modern Java-based backend architecture. The thesis demonstrates that with appropriate hardware and software

design, a reliable, scalable, and real-time solution can be created, which can be suitable for further research, development, or even as a basis for later commercial systems.

10 Irodalomjegyzék

[1] M. Health, „The 7 Best Electronic Dartboards, According to Editors,” 2. október 2024.. [Online]. Available: https://www.menshealth.com/technology-gear/g62449830/best-electronic-dart-board/. [Hozzáférés dátuma: 22. november 2025.].

[2] P. W. K. V. A. W. J. M. William McNally, „DeepDarts: Modeling Keypoints as Objects for Automatic Scorekeeping in Darts using a Single Camera,” in *IEEE*, Nashville, TN, USA, 2021.

[3] W. J. M. e. al., „DeepDarts: Modeling Keypoints as Objects for Automatic Scorekeeping in Darts using a Single Camera,” arXiv:2105.09880, május 2021.. [Online]. Available: https://arxiv.org/abs/2105.09880. [Hozzáférés dátuma: 5. november 2025.].

[4] Autodarts, „Autodarts Vision,” Autodarts, [Online]. Available: https://autodarts.io/vision. [Hozzáférés dátuma: 13. október 2025.].

[5] L. G. (LarsG21), „Darts_Project,” GitHub, [Online]. Available: https://github.com/LarsG21/Darts_Project. [Hozzáférés dátuma: 19. október 2025.].

[6] A. webshop, „Autodarts Vision Kit — termékoldal,” buy.autodarts.io, 2025. [Online]. Available: https://buy.autodarts.io/en-eu/products/autodarts-vision-kit. [Hozzáférés dátuma: 6. november 2025.].

[7] DartConnect, „VideoConnect,” DartConnect, [Online]. Available: https://www.dartconnect.com/videoconnect/. [Hozzáférés dátuma: 11. november 2025.].

[8] G. McGarry, „Applying Artificial Intelligence to Automatically Score Darts,” LinkedIn post / közösségi cikk; illetve különböző egyetemi projektek és GitHub felhasználók, [Online]. Available: https://github.com/LarsG21/Darts_Project. [Hozzáférés dátuma: 12. november 2025.].

[9] IEEE, *IEEE 802.11 Wireless Local Area Networks (Wi-Fi) Standard, Institute of Electrical and Electronics Engineers,* USA: IEEE.

[10] C. Systems, „Channel State Information (CSI) in Wireless Networks – Technical Overview, Cisco Documentation,” USA.

[11] R. P. Ltd, „Raspberry Pi 4 Model B – Product Specifications,” Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Hozzáférés dátuma: 13. november 2025.].

[12] R. P. Ltd, „Raspberry Pi Camera Module v2 – Product Brief,” Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/. [Hozzáférés dátuma: 12. november 2025.].

[13] Python, „The official home of the Python Programming Language,” Python Software Foundation, [Online]. Available: https://www.python.org/. [Hozzáférés dátuma: 11. november 2025.].

[14] IntelliJ IDEA, „the IDE for pro Java and Kotlin Development,” JetBrains, [Online]. Available: https://www.jetbrains.com/idea/. [Hozzáférés dátuma: 4. október 2025.].

[15] Java, „Oracle Java is the #1 programming language and development platform.,” Oracle Corporation, [Online]. Available: https://www.java.com/en/. [Hozzáférés dátuma: 5. október 2025.].

[16] PostgreSQL, „The world’s most advanced open source database,” PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/. [Hozzáférés dátuma: 7. október 2025.].

[17] OpenCV, „Open Source Computer Vision Library,” Open Source Vision Foundation, [Online]. Available: https://opencv.org/. [Hozzáférés dátuma: 1. szeptember 2025.].

[18] F. A. a. A. B. Mansoor, „ Computer vision based,” INMIC, 2008.

[19] P. R. Aryan, „Vision based automatic target scoring,” ICACSIS, 2012.

[20] Z. C. a. N. Vasconcelos, „Delving into high quality object detection,” CVPR, 2018..

[21] Z. W. Y. P. Z. Yilun Chen, „Cascaded pyramid network,” CVPR, 2018.

[22] A. D. N. U. H. R. Anthony Cioppa, „Multimodal and multiview distillation for real-time player," CVSports, 2020.

[23] X. Z. X. F. a. Q. Penghua Ding, „Design of automatic target-scoring system of shooting game based on computer vision," ICAL, 2009.

[24] J. D. T. D. a. J. Ross Girshick, „Rich feature hierarchies for accurate object detection," CVPR, 2014..

[25] D. A. D. E. C. Wei Liu, „Single shot multibox detector," ECCV, 2016..

[26] K. Y. a. J. D. Alejandro Newell, „Stacked hourglass networks for human pose estimation," ECCV, 2016..

[27] R. M. Parag, *Sequential recognition and scoring of archery,* Master's thesis, 2017..

[28] S. D. R. G. a. A. Joseph Redmon, „You only look once: Unified, real-time object detection," CVPR, 2016..

[29] J. R. a. A. Farhadi, „Yolo9000: better, faster,," CVPR, 2017..

[30] „myDartspfei," 2025. [Online]. Available: https://mydartpfeil.com/en-eu/products/dartscheibe-mydartpfeil-starter-steeldartscheibe-dartboard-fuer-steeldart?variant=47022703051094&country=CZ&currency=EUR&trc_gcmp_id=19475207756&gad_campaignid=23321596014.

[31] „DoubletopDartsShop," 2025. [Online]. Available: https://www.doubletopdartshop.com/blogs/news/steel-tip-or-soft-tip-darts.

[32] „CreativeCompany," 2025. [Online]. Available: https://www.cchobby.de/sisal-rost-8-g-1-pck.

[33] DataCaptureControl. [Online]. Available: https://datacapturecontrol.com/articles/io-devices/single-board-computers/raspberry-pi-4b.

11 Ábrajegyzék