

VARGA ROLAND
SZAKDOLGOZAT

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
GÉPÉSZMÉRNÖKI KAR
MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK



SZAKDOLGOZATOK



BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
GÉPÉSZMÉRNÖKI KAR

MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK

VARGA ROLAND

SZAKDOLGOZAT

**Ember és robot kooperációjának demonstrálása
Sakközö iixa robotkar segítségével**

Demonstrating human-robot collaboration

With chess-playing iiwa robotic arm

Konzulens:

Magyar László
tesztmérnök

Témavezető:

Dr. Czmerk András
egyetemi adjunktus

Budapest, 2018

Ide kell befűzni az eredeti feladatkiírási lapot!

NYILATKOZATOK

Beadhatósági nyilatkozat

A jelen szakdolgozat az üzem által elvárt szakmai színvonalnak mind tartalmilag, mind formailag megfelel, beadható.

Kelt,

Az üzem részéről:

üzemi konzulens

Elfogadási nyilatkozat

Ezen szakdolgozat a Budapesti Műszaki és Gazdaságtudományi Egyetem Gépészmeérnöki Kara által a Diplomatervezési és Szakdolgozat feladatokra előírt valamennyi tartalmi és formai követelménynek, továbbá a feladatkiírásban előírtaknak maradéktalanul eleget tesz. E szakdolgozatot a nyilvános bírálatra és nyilvános előadásra alkalmasnak tartom.

A beadás időpontja:

témavezető

Nyilatkozat önálló munkáról

Alulírott, Varga Roland (XZYX5L), a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója, büntetőjogi és fegyelmi felelősségem tudatában kijelentem és sajátkezű aláírásommal igazolom, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és dolgozatomban csak a megadott forrásokat használtam fel. minden olyan részt, melyet szó szerint vagy azonos értelemben, de átfogalmazva más forrásból átvettettem, egyértelműen, a hatályos előírásoknak megfelelően, a forrás megadásával megjelöltem.

Budapest, 2018

sziigorló hallgató

TARTALOMJEGYZÉK

1. Bevezetés	1
1.1. Célkitűzés	1
1.2. Áttekintés – még nem végleges	1
2. Irodalomkutatás	2
2.1. Ipar 4.0 eredete	2
2.2. Előnyei	2
2.3. Kialakítási alapelvek	3
2.3.1. Összekapcsolás	3
2.3.2. Információs átláthatóság	4
2.3.3. Decentralizált döntéshozatal	5
2.3.4. Technikai asszisztens	5
2.4. Ember-robot kollaboráció	6
2.4.1. Fogalmak tisztázása	6
2.4.2. Biztonsági szempontok ember-robot kollaboráció esetén . . .	7
2.4.3. Érzelmi megfontolások	10
2.5. KUKA-specifikus biztonsági funkciók	10
2.5.1. Használt szakkifejezések	11
2.5.2. Biztonságszempontrú funkciók	14
3. Alkalmazáshoz szükséges műszaki feltételek elemzése	16
3.1. Projekt részletes leírása	16
3.2. A megfogó vezérlése	17
3.2.1. A vezérlés hardveres kialakítása	17
3.2.2. A megfogó szoftveres konfigurációja	18
3.3. Sakkbabák kialakítása, felismerése a képeken	20
3.3.1. A sakkbabuk formatervezése	20
3.3.2. Bábuk pozíciójának azonosítása	23
3.4. QR-kód alapú képfeldolgozás - továbbfejlesztési irány	26
3.4.1. A forráskód build-je	27
3.4.2. ZXing könyvtár beimportálása és használata	27
3.4.3. QR kód generálás és olvasás	28
4. Robotkar kalibráció, referencia felvétel kidolgozása	30
4.1. Koordinátarendszerek	30
4.2. Szerszámkalibráció - XZY 4-pont metódus	32

TARTALOMJEGYZÉK

4.3. Báziskalibráció - 3 pont metódus	33
4.4. A szerszám tömegeloszlásának meghatározása	35
5. Robotkar mozgásának definiálása és programozása	37
6. Sakkalgoritmus beágyazása	38
6.1. A chess.algorithm csomag - az algoritmusok működtetője	39
6.2. A chess.core csomag - a játék magja	40
6.3. A chess.properties csomag - kiegészítő funkciók	43
6.4. A sakkprogram kiegészítése a projekthez	43
7. Eredmények értékelése	45

JELÖLÉSEK JEGYZÉKE

A táblázatban a többször előforduló jelölések magyar és angol nyelvű elnevezése, valamint a fizikai mennyiségek esetén annak mértékegysége található. Az egyes mennyiségek jelölése – ahol lehetséges – megegyezik hazai és a nemzetközi szakirodalomban elfogadott jelölésekkel. A ritkán alkalmazott jelölések magyarázata első előfordulási helyüknél található.

1. BEVEZETÉS

1.1. Célkitűzés

A szakdolgozat célja megismertetni az olvasóval az Ipar 4.0 (az ipari fejlődés egyik legújabb és legmeghatározóbb trendje) fontos elemét, az ember-robot kollaborációt. A fókuszban egy gyakorlati megvalósítás, egy demo áll, viszont általános irányelvek és módszerek is taglalásra kerülnek. A szakdolgozat feltételezi, hogy az olvasó rendelkezik alapszintű ismeretekkel az Ipar 4.0 főbb elemeit illetően. A dolgozat fő szerepe az Ipar 4.0 egy kisebb részletének praktikus, szemléletes bemutatása, de mégis érdemes rendszerszemléletűen hozzájondolni a többi elemét is, mivel így kaphatunk csak teljes képet ennek funkciójáról.

1.2. Áttekintés – még nem végleges

Első körben a 4. ipari forradalom főbb vonásai kerülnek bemutatásra, különös tekintettel az ember-robot együttműködésre (későbbiekben HRC - Human-Robot Collaboration). Ezen belül több balesetmegelőző lépcsőfokról is szó lesz azért, hogy szemléletesekkel legyenek az előnyök. Ezt követően a szakdolgozat magját jelentő demóhoz használt KUKA LBR iiwa 7 R800 robotkarhoz tartozó biztonságtechnikai fogalmakat és specifikációkat ismertetem, mivel ennek keretében beszélhetünk a HRC-vel kapcsolatos funkciókról.

2. IRODALOMKUTATÁS

2.1. Ipar 4.0 eredete

Az Ipar 4.0 koncepcióját Németországban dolgozták ki, ahol világszinten kiemelkedő a termelési iparág, illetve világvezető a gyártó eszközök területén. Az Ipar 4.0 a német kormány stratégiai kezdeményezése volt, mely nagy mértékben támogatta az ipari szektor fejlesztését. Ilyen értelemben az Ipar 4.0-ra tekinthetünk egy olyan mozgalomként, amelynek célja megőrizni Németország befolyását a gépiparban és az autógyártás területén.[1]

Az alap koncepciót először a Hanover Fair-en¹ prezentálták 2011-ben. A bemutató óta az Ipar 4.0 Németország vezető téma a kutatások területén, egyetemi és ipari környezetekben különféle eseményeken. A fő irányvonal az új technológiákban és koncepciókban rejő potenciál kihasználása felé mutat, ilyen területek:

- az IoT (Internet of Things²) elérhetősége és kihasználása,
- a technológiai és gazdasági folyamatok integrációja cégen belül,
- a valóság virtuális leképezése,
- ‘okos’ gyárak beleértve az ‘okos’ gyártást és termékeket.

2.2. Előnyei

Amellett hogy a digitalizáció és az új technológiák természetes következménye, az Ipar 4.0 megjelenése szintén kapcsolatban áll azzal a tényel, hogy a gyártásban a profit növelésére irányuló kezdeményezések, lehetőségek nagy része kiaknázásra került, új megoldásokat kellett keresni. A gyártási költségek csökkentek a Just-In-Time (röviden JIT) termelés bevezetésével, a lean elveinek alkalmazásával és a gyárak olyan helyre telepítésével, ahol a munkaerő lényegesen olcsóbb. Ha az előállítási költségek minimalizálása a célunk, az Ipar 4.0 egy igéretes megoldásnak tűnik. Számos forrás alapján az Ipar 4.0 alkalmazása csökkentheti[2]:

- a gyártás költségét 10-30%-kal,
- a logisztikával kapcsolatos kiadásokat 10-30%-kal,

¹A hannoveri a világ egyik legnagyobb kereskedelmi bemutatója. Körülbelül 6500 kiállító és 250.000 látogató vesz részt ezen a rendezvényen.

²A Dolgok internete fizikai eszközökből, járművekből, otthoni felszerelésekkel és további elektronikát, szoftvert, szenzorokat, aktuátorokat tartalmazó tételekből álló hálózat, amelyek képesek egymással kapcsolatba lépni, adatot fogadni és küldeni.

- a minőségmenedzsmenthez köthető költségeket 10-20%-kal.

Ezeken kívül a koncepció alkalmazásának számos egyéb előnyéről szólhatunk: (1) új termékek piacra kerülési ideje csökken, (2) érzékenyebb reagálás a megváltozott vásárlói igényekre, (3) lehetővé teszi a személyreszabott tömeggyártást az összgyártási költség jelentős növelése nélkül, (4) rugalmasabb és barátságosabb munkakörnyezetet teremt, (5) a természetes erőforrásokat hatékonyabban hasznosítja.

2.3. Kialakítási alapelvek

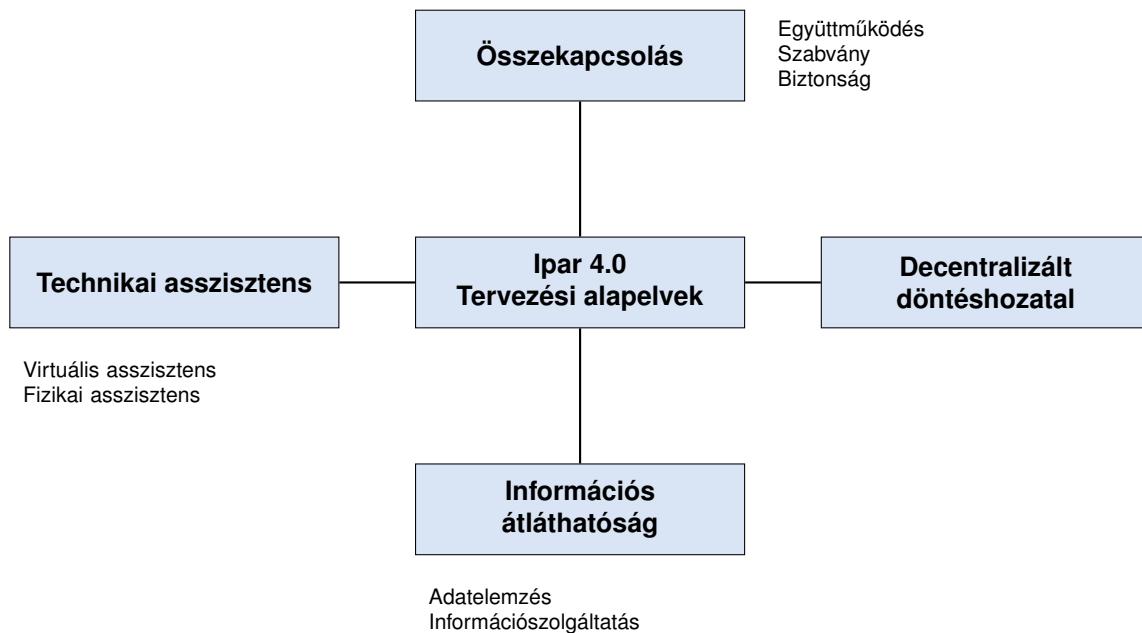
A számos szövegelemzés és átfogó irodalmi áttekintés négy fő dizájn elvet emelt ki, hogy irányvonalat mutasson a szakértőknek és tudósoknak az Ipar 4.0 környezet kialakításához: összekötés, információs átláthatóság, decentralizált döntéshozatal és technikai asszisztens (2.1. ábra). Ezek az alapelvek a következő alfejezetekben kerülnek részletes tárgyalásra az egyetemi és ipari publikációkban használt kifejezések (és következésképpen a kialakítási alapelvek) rövid elemzése után.

Összességeiben a két különböző típusú publikáció szövegelemzése nem mutat lényeges eltérést, mindenkorral a külön-külön elemzése ugyanazokat a kialakítási alapelveket eredményezi. Azonban szembe tűnő, hogy egyes dizájn elemeket gyakrabban tárgyalnak a gyakorlati publikációkban. Az ember-robot kollaboráció, adat- és információbiztonság és a decentralizált döntéshozatal gyakrabban fordul elő ipari kiadványokban. Az első kettővel kapcsolatos értekezések magas száma rávilágít az Ipar 4.0 eredményes implemetálásának legnagyobb kihívásaira amivel az iparban dolgozók szembesülnek. Mindeközben a decentralizált döntéshozatal tekintik az Ipar 4.0 legproblémásabb elemének, és ezért ez rendkívül részletes és átfogó tárgyalásra kerül.

2.3.1. Összekapcsolás

Gépek, eszközök, szenzorok és emberek kapcsolatba lépnek IoT-n (Internet of Things - Dolgok interne) és IoP-n (Internet of People - Emberek interne[3]) keresztül és így formálnak egy IoE-t (Internet of Everything - minden interne[4]). A vezetéknélküli technológiák kiemelkedő szerepet játszanak az interakciók során, mivel lehetővé teszik az internetes hozzáférést mindenfelé. Az IoE-n keresztül összekötött emberek és eszközök képesek egymással információt megosztani, ami a kollaboráció alapját jelenti a közös célok elérése érdekében. 3 különböző típust különböztethetünk meg az IoE kapcsán: ember-ember együttműködés, **ember-robot kollaboráció** és robot-robot kollaboráció.[5]

A különböző gépek, eszközök, érzékelők és emberek egymás közti interakciója során elengedhetetlen szerepe van a széles körben elfogadott kommunikáci-



2.1. ábra: Ipar 4.0 tervezési szempontok

ós szabványoknak. Ezek teszik lehetővé a különböző gyártóktól érkező moduláris eszközök rugalmas kombinálását. Ez a modularizáció az alapfeltétel, hogy az Ipar 4.0 'okos' gyárai alkalmazkodni tudjanak a folyamatosan változó piaci igényekhez vagy a személyreszabott rendelésekhez.

Ahogy nő az IoE-ben részt vevők száma, a monetáris¹ és politikai érdekek meg fogják növelni az ilyen létesítmények elleni káros támadások számát, így az igény is nőni fog a magasabb fokú informatikai biztonság iránt.

2.3.2. Információs átláthatóság

Az összekapcsolt objektumok és emberek növekvő számának köszönhetően, a fizikai és a virtuális világ egybeolvadása lehetővé tesz egy újfajta információs modellt[6]. Az érzékelők összekapcsolása révén képezhetünk egy digitális, virtuális leképezést a világunkról.

Az összefüggés-tudatos információ az IoE résztvevői számára elengedhetetlenek a megfelelő döntések meghozatalához. Az ilyen összefüggés-tudatos rendszerek a feladataikat virtuális és a fizikai világból érkező információk alapján látják el. A virtuális világból érkező információkra példák az elektronikus dokumentumok, rajzok, szimulációs modellek. A fizikai világ információi például a pozíció vagy a szerszám állapota. A fizikai világ elemzéséhez az érzékelők felől érkező nyers adatokat magasabb szintű értelmezési és egyéb információval kell kiegészíteni. Ahhoz,

¹pénzhez vagy valutához kötődő

hogy az átláthatóságot fenntartsuk, az adatelemzés eredményeit egy olyan kisegítő rendszerbe kell bevinni, ami minden IoE résztvevő számára elérhető. A folyamatkritikus információk esetén a valós idejű adatszolgáltatás elengedhetetlen.

2.3.3. Decentralizált döntéshozatal

A decentralizált döntések meghozatalának két alappilére az objektumok és emberek összekapcsolása, illetve a termelő létesítményen belülről és kívülről érkező információk átláthatósága. Az összekapcsolt és decentralizált döntéshozó egységek lehetővé teszik a lokális információk globálissal együtt felhasználását egyazon időben, így elősegítve az átgondoltabb döntéshozatalt és így növelve összességében a termelékenységet. Az egyes IoE elemek a feladataikat annyira önállóan látják el, amennyire csak lehet. A feladatok csak kivételek, zavarok vagy ellentmondásos célok esetén kerülnek továbbításra magasabb szintre.

Gyakorlati szempontból a decentralizált döntéshozatalt a kiber-fizikai rendszerek teszik lehetővé. Ezek beágyazott számítóegységeinek, szenzorainak és aktuátorainak felhasználásával történik fizikai világ autonóm nyomon követése és az irányítása.

2.3.4. Technikai asszisztens

Az Ipar 4.0 'okos' gyáraiban az ember szerepe alapvetően megváltozik, gépkezelő helyett inkább stratégiai döntéshozóvá és rugalmas problémamegoldóvá válik. A termelési folyamatok növekvő komplexitása miatt, ahol a kiber-fizikai rendszerek összetett hálózatot alkotnak és decentralizált döntéseket hoznak, az embereknek támogató rendszerekre van szükségük. Ezeknek a rendszereknek a szerepe az információk összegyűjtése és megjelenítése egyértelműen és érthetően annak érdekében, hogy az emberek jól megalapozott döntéseket tudjanak hozni, és magas prioritású problémákat tudjanak megoldani rövid időn belül. Jelenleg az embereket főként az okostelefonjaik és táblagépeik kötik össze az IoT-vel[7]. A hordozhatóság kiemelkedően fontossá fog válni a jövőben amint a jelenlegi kihívásokon (mint például az energiaellátás) sikerül felükkerekedni.

Az emberek robotok általi fizikai kisegítése (a robotika területen elért fejlesztésekkel) szintén a technikai asszisztens szerep részét képezi. A robotok számos feladatot képesek elvégzni, amelyek az ember számára kellemetlenek, túl fárasztóak vagy veszélyesek más munkásokra nézve[8]. Az emberek fizikai feladatokban hatékony, sikeres és biztonságos segítésének érdekében szükséges, hogy a robotok az ember társaikkal zökkenőmentesen és intuitívan működjenek együtt[8]. Ezen felül elengedhetetlen, hogy az emberek megfelelő képzésben részesüljenek az adott

ember-robot kollaborációhoz[9].

2.4. Ember-robot kollaboráció

2.4.1. Fogalmak tisztázása

Az ember és a robot közösen végzett feladataikat különböző interakciós szinteken valósíthatják meg, ezeket érdemes egymástól elhatárolni¹ (2.2. ábra):

1. Robot cella (Robotic cell): a robot önállóan végzi a feladatát az embertől kerítéssel elválasztva. Ez esetben nem beszélhetünk ember-robot együttműködésről.
2. Együttes jelenlét (Coexistence): a robot és az ember közel helyezkedik el egymáshoz védőkerítés nélkül, de nincs közös munkaterük. A robotnak van saját meghatározott tere.
3. Szinkronizált munkavégzés (Synchronized work): olyan elrendezés, melyben az ember és a robot osztozik egy közös munkateren, de egyszerre csak egyikük aktív. A munkamenet az ember és a robot jól definiált 'koreográfiája'.
4. Kooperáció (Cooperation): a két „partner” mindegyike a saját feladatával foglalkozik. A munkaterük lehet közös, de nem dolgozhatnak sem ugyanazon a terméken, sem ugyanazon a munkadarabon.
5. Kollaboráció (Collaboration): olyan elrendezés, amely esetén az ember és a robot közösen és szimultán dolgozik egyazon terméken vagy munkadarabon. Tipikusan a robot megfogja, átnyújtja és tartja a munkadarabot amíg a munkás dolgozik rajta.



2.2. ábra: Balról jobbra: együttes jelenlét, kooperáció, kollaboráció²

¹Forrás: <https://hohmannchris.wordpress.com/2017/02/08/cobots-more-cooperation-than-collaboration/>

2.4.2. Biztonsági szempontok ember-robot kollaboráció esetén

A biztonságos ember-robot együttműködés érdekében az elmúlt években különböző stratégiák lettek kifejlesztve. Ezek a módszerek különböző biztonságtípusra építenek, többek közt[10]:

- az ütközésbiztonság érdekében csak 'biztonságos'/kontrollált ütközésre kerülhet sor robotok, emberek és akadályok között. Az emberekre gyakorolt erő/-nyomaték határolása a fő szempont.
- aktív biztonsági rendszer az ember és a berendezés közötti közelgő ütközések időben történő észlelésére és a műveletek megszakítására ellenőrzött módon. Távolságérzékelők, látórendszerök és erő/érintkező érzékelők használhatóak erre a célra.
- a hardver elemek működtetése során adaptív biztonsági stratégia a beavatkozáshoz, illetve korrekciós eljárások implementálása, amelyek az ütközéseket a munkafolyamat megszakítása nélkül kerülik el.

Ilyen irányban nemzeti és nemzetközi szabványokat, irányelvezeket és törvényeket vezettek be annak érdekében, hogy a rendszerintegrátorok könnyen be tudják építeni a biztonsági methódusokat a saját rendszerükbe. Tekintettel arra, hogy egy kolaborációs munkatéren nem csak az ember és a robot tartózkodik, hanem egyéb segédberendezések is megjelennek (például: elektromos csavarbehajtók, elektromos szorítóberendezések stb.) minden egyes cella egyedülálló kockázatot jelent biztonságtechnikai szempontból. Ebből kifolyólag minden berendezés és folyamat esetén be kell tartani az ezekre vonatkozó hatályos törvényeket és szabványokat. A 2.1., 2.2. és 2.3. táblázatok indikatív példákat tartalmaznak az EU legismertebb törvényei és általános normából válogatva.

2.1. táblázat: EU Directives

Title	Description
2006/42/EC	Machinery Directive (MD)
2009/104/EC	Use of Work equipment Directive
89/654/EC	Workplace Directive
2001/95/EC	Product Safety Directive
2006/95/EC	Low Voltage Directive
2004/108/EC	Electromagnetic Compability Directive (EMC)

²Képek forrása: <https://www.safetysolutions.net.au/content/machine/article/safety-solutions-for-intelligent-human-robot-collaboration-990038334>

2.2. táblázat: Indicative general standards

Title	Description
EN ISO 12100	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN ISO 13849-1/2	Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design, Part 2: Validation
EN 60204-1	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
IEC 62061	Safety of machinery - Functional safety of safety related electrical, electronic and programmable electronic control systems

2.3. táblázat: Robot standards

Title	Description
EN ISO 10218-1	Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots
EN ISO 10218-2	Robots and robotic devices - Safety requirements for industrial robots - Part 2: Robot systems and integration
ISO/PDTS 15066	Robots and robotic Devices - Collaborative Robots

Jelenleg közel 30 aktív uniós irányelv és körülbelül 600 szabvány létezik a biztonságtechnikával kapcsolatosan. Különösen robot cellákkal kapcsolatban 2.3. táblázat előrevetít számos különböző stratégiát a megfelelő biztonság kialakítására. Pár a legfontosabbak közül[10]:

- **Biztonsággal kapcsolatos vezérlőrendszer teljesítménye:** biztonságkritikus vezérlőrendszer-egységeknek biztosítaniuk kell, hogy az egyes hibákra van oly mértékű tolerancia, hogy a biztonságot nem veszélyezteti.
- **Robotmegállító funkciók:** minden robotnak rendelkeznie kell védőleállító funkcióval, továbbá független vészleállító funkcióval. Külső biztonsági berendezésekkel való kapcsolatot is biztosítani kell.
- **Sebességkorlátozás:** a robot végén található beavatkozó és a szerszámközéppont (TCP - Tool Center Point) sebessége kontrollálható kell legyen. Kollaboratív munkaállomások esetén a TCP sebessége nem haladhatja meg a 250 mm/s-ot.
- **Kollaboratív műveletekre vonatkozó követelmények:** együttműködésre képes robotok kell hogy működtessenek valamilyen látható indikátorral amikor a robot kollaboratív műveletet hajt végre. A következő követelmények érvényesek:

- **Biztonsági rendszer által kezdeményezett felügyelt megállás:** a robotnak kötelező megállnia, amikor az ember a kollaboratív munkaterületen tartózkodik. Amint az ember elhagyja ezt a területet, a robot folytathatja az automatikus munkavégzési folyamatot.
- **Kézi vezetés (Hand guiding):** a kézi vezetéshez alkalmazott berendezések tartalmaznia kell egy vészleállítót és egy engedélyező szerkezetet. E művelet során biztonságkritikai szempontból tanúsított módon nyomon kell követni a robot sebességét. Számos technológia felhasználható az ember által kézzel történő irányításhoz, mint például az impedancia vagy a merevség szabályozása.
- **Sebesség és pozíció monitorozása¹:** a robot fent kell tartson egy adott távolságot az üzemeltetőtől. Ezt integrált jellemzők vagy külső inputok kombinációjának figyelésével tudja megvalósítani.
- **Teljesítmény- és erőhatárolás belső dizájnnal:** a teljesítmény-/erőhatároló funkcióknak be kell tartatniuk a szabvány által meghatározott határértékeket. Ha ezt túllépik, megállást kell eszközölniük.
- **Vezérlőrendszer által határolt erő és teljesítmény:** egy vezérlőfunkció biztosítja, hogy a maximális erő és teljesítmény értékeket ne lehessen túlélni.
- **Robot mozgásterének korlátozása:** feleslegesen nagy tér elhatárolása szükséges, ha a robotnak engedélyezett a teljes mozgástartomány kihasználása. A mozgástér leszűkítése lehetséges a robotba integrált rendszerek kihasználásával vagy külső korlátozó eszközök telepítésével, illetve e kettő kombinációjával. Dinamikus korlátozás érhető el vezérlőeszközök (kapcsolók, fényfüggönyök stb.) alkalmazásával, ha további limitációkra van szükség a robotprogram végrehajtása során.
- **Minimális távolság meghatározása:** feladattól függően kockázatelemzés után kerül meghatározásra az ember és a robot közötti minimális távolság. Az elemzés során érdemes figyelembe venni a) a végeffektorokhoz és az esetlegesen ezek által tartott munkadarabokhoz kapcsolódó kockázatokat, b) a munkaterület elrendezését, c) a munkás feladatát és d) a rendszer felhasználhatóságát.
- **Ütközésérzékelés:** a biztonsági funkcióknak meg kell tudniuk határozni mind az ember, mind pedig a robot pillanatnyi pozíciója és sebessége alapján, hogy a minimális távolság alá csökkenhet-e az ember és a robot távolsága (ütközés).

¹folyamatos megfigyelés

- **Esetleges ütközések elkerülése:** ez a funkció lehetővé teszi a robot számára, hogy megelőzze az ütközéseket a) lassítás vagy megállás, b) az meghatározott útja során eszközölt irányváltás és c) egy másik biztonságos útvonalon haladás által.
- **Technológiai és ergonómiai követelmények:** az ember és robot közötti esetleges ütközés esetén megfelelő óvintézkedéseknek kell biztosítaniuk, hogy éles, hegyes vagy durva felületek nem találhatóak az érintkezési zónában. Továbbá a környező munkaterületet (ahol az ember a kollaboratív robottal ütközhet) úgy kell kialakítani, hogy a felhasználó számára elegendő tér biztosítva legyen ahhoz, hogy kockázatos helyzeteket elkerülje.

Ezeket a funkciókat szabványosítással foglalkozó szervek dolgozták ki a különböző biztonsági szempontok átfedés nélküli ellenőrzésének érdekében. Átfedések előfordulhatnak az egyes projektek kivitelezésekor; a biztonsági követelményeknek teljes mértékben való megfelelést minden esetben felül kell vizsgálni.

2.4.3. Érzelmi megfontolások

Az ember-robot együttműködésnek és a kollaboratív celláknak a kialakítása, valamint az ember és a robot között hatékony feladatmegosztás szorosan kapcsolódik társadalom-fiziológiai kérdésekhez. A mélyebb megértés és az ember viselkedésének megjósolása, valamint a HRI (Human-Robot Interaction - Ember-Robot Interakció) fejlődése részben a kognitív mérnökséghez, pszichológiához és szociológiához kötődik. A bizalom, a terheltség és a kockázatosság a legfontosabb emberi tényezők, amik befolyásolják az automatizálási technológiák alkalmazását. Az emberközpontú robotcella kialakítása során az emberi tényezők, mint a terheltség, éberség, helyzetfelismerés, hibák stb. figyelembe vétele ugyanúgy fontos, mint a kognitív mérnöki szempontok és az ergonomia[11].

2.5. KUKA-specifikus biztonsági funkciók

A következő részben a KUKA Sunrise OS 1.16[12] biztonsági funkciói kerülnek taglalásra. Az alábbi fogalmak és szempontok egy része már a 2.4.2 fejezetben előkerült általánosságban, viszont gyakorlati szempontok miatt ezek specifikus tárgyalására is szükség van. Az itt leírtak tájékoztató jellegűek, céljuk a szakdolgozat értelmezhetőségének javítása, nem foghatóak fel általános iránymutatásként. Adott OS (Operating System) és egyéb kiegészítő berendezések esetén mindenkoruknak ezekhez mellékelt biztonsági utasítás, ezek betartása elengedhetetlen a hatékony és biztonságos működtetéshez.

A biztonsági funkciókat két csoportra lehet osztani:

- Biztonságkritikus funkciók az emberi biztonság fenntartása érdekében
Az ipari robot biztonságkritikus funkciói megfelelnek a felsorolt biztonsági követelményeknek:
 - EN ISO 13849-1 (2.2. táblázat) szabvány 3-as kategóriája (Category 3), illetve a d teljesítményszint besorolás (Performance Level d)
 - EN 62061 (2.2. táblázat) SIL 2 követelménye Ezek a követelmények azonban csak a következő feltételek fennállásakor teljesülnek:
 - Az ipari robot minden biztonsági szempontból releváns mechanikus, illetve elektromechanikus komponense a biztonságos működés szempontjából le lett tesztelve a beüzemeléskor és legalább egyszer minden évben, hacsak nem a kockázatelemzők másképp határoztak. Ilyen eszközök:
 - * Vészstop eszköz (Emergency stop device) a smartPAD-en
 - * Engedélyező kapcsoló a smartPAD-en
 - * Kulcsos kapcsoló a smartPAD-en
- Nem biztonságkritikus funkciók a gépek védelme érdekében
Az ipari robot nem biztonságkritikus funkciói nem tartoznak kitüntetett biztonsági követelményekhez.

2.5.1. Használt szakkifejezések

2.4. táblázat: A használt szakkifejezések

Kifejezés	Leírás
Tengely mozgástartománya (Axis range)	Az a tartomány, amelyben a tengely mozoghat. Ezt a tartományt minden tengelyre külön meg kell határozni.
Megállási út (Stopping distance)	Megállási út = reakcioidő alatt megtett út + fékút. A megállási út része a veszélyzónának.
Munkatér (Workspace)	A beavatkozó (manipulator) ezen tartományon belül mozoghat. A munkatér az egyes tengelyek mozgástartományából származtatott.
Automatikus (AUT) (Automatic)	A programvégrehajtáshoz tartozó üzemmód. A beavatkozó a programozott sebeséggel mozog.
A következő oldalon folytatódik	

2.4. táblázat – a 11. oldalon kezdődik

Kifejezés	Leírás
Gépkezelő/Felhasználó (Operator/User)	Az ipari robot üzemeltetője lehet a menedzsment, alkalmazott vagy megbízott ember, aki felelős a robot működtetéséért.
Veszélyzóna (Dangerzone)	A veszélyzóna a munkaterületből és a megállási útból áll.
Élettartam (Service life)	A biztonságkritikus komponensek élettartama az ügyfélnek kiszállítástól számítandó. Az élettartam nem függ attól, hogy a komponens a robotvezérlőbe került beépítésre vagy másba, vagy akár egyáltalán nem is volt használva, mivel a biztonságkritikus alkatrészek is ki vannak téve előregedésnek a tárolás alatt.
CRR	Irányított Robot Visszatérés (Controlled Robot Retraction) A CRR egy olyan üzemmód, amelybe akkor lehet lépni, ha az ipari robotot megállította a biztonsági vezérlő a következő okok valamelyike miatt: <ul style="list-style-type: none"> • A robot megsértett egy tengelyspecifikus vagy egy descartesi koordinátára vonatkozó szabályt. • A biztonság-orientált szerszám elhelyezkedése a megfigyelt tartományon kívülre esik. • A robot megsérti az erőre vagy a nyomatékra vonatkozó korlátozásokat. • A pozíószenzor nem ‘mastered’ vagy ‘referenced’.¹ • A csuklóban lévő nyomásszenzor nem ‘referenced’. CRR üzemmódba lépés után a robotot újra lehet mozgatni.
KUKA smartPAD	Lásd „smartPAD”
Beavatkozó (Manipulator)	A robotkar és a hozzá telepített elektronika.
A következő oldalon folytatódik	

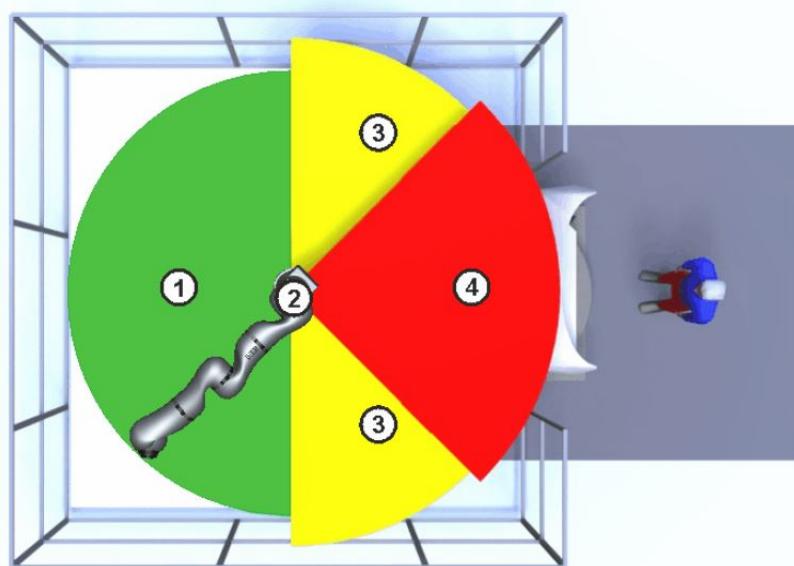
¹ A két kifejezést nem célszerű lefordítani magyarra.

2.4. táblázat – a 11. oldalon kezdődik

Kifejezés	Leírás
Biztonsági sáv (Safety zone)	A beavatkozó nem léphet a biztonsági sávba. Ez a tartomány kívül esik a veszélyzónán.
Biztonsági megállás	A biztonsági megállást a biztonsági vezérlő idézheti elő, így megszakítva a munkamenetet és megállítva a robot mozgását. Biztonsági megállás esetén a programadatok nem vesznek el, így a program folytatódhat a megszakítás pontjától kezdve. A biztonsági megállás lehet 'Stop category 0', 'Stop category 1' vagy 'Stop category 1 (path-maintaining)' (útvonal követés).
smartPAD	A smartPAD egy kézbevehető irányító panel a robotcellához (-állomáshoz). A robotállomás működtetéséhez szükséges összes üzemeltetői és megjelenítési funkciót tartalmazza.
Stop category 0	A motorok kikapcsolnak és a fékek összezárnak.
Stop category 1	A robotkar lefékeződik és közben nem marad a programozott útvonalon. A beavatkozó a motorok segítségével jut álló pozícióba. Amint megállnak a csuklók a motorok kikapcsolnak és a fékek behúzódnak.
Stop category 1 (path-maintaining)	A beavatkozó lefékeződik és közben a programozott útvonalon marad. Amint megállnak a csuklók a motorok kikapcsolnak és a fékek behúzódnak. Ha 'Safety category 1 (path-maintaining)'-et idéz elő a biztonsági vezérlő, akkor ez felügyeli is a megállítás folyamatát. A fékek behúzódnak és a motorok legkésőbb 1 másodperc elteltével kikapcsolnak. Hiba esetén 'Stop category 1' kerül végrehajtásra.
Rendszerintegrátor (System integrator / plant integrator)	A rendszerintegrátorok feladata az ipari robot integrálása egy rendszerbe és ennek hitelesítése.
T1	Teszt üzemmód, manuálisan csökkentett sebesség (Manual Reduces Velocity) ($\leq 250 \text{ mm/s}$) Megjegyzés: kézi vezetés T1-ben esetén a sebesség nem kimondottan csökkentett, inkább limitált biztonságszempontú sebességmonitorozás által.
A következő oldalon folytatódik	

2.4. táblázat – a 11. oldalon kezdődik

Kifejezés	Leírás
	Megjegyzés: a 250 mm/s maximális sebesség a mobil platformokra nem vonatkozik.
T2	Teszt mode, manuálisan állított nagy sebesség (> 250 mm/s megengedett)



2.3. ábra: Példa a mozgástartományokra: 1. munkaterület, 2. beavatkozó, 3. megálísi út, 4. biztonsági sáv

2.5.2. Biztonságszempontú funkciók

Az alábbi funkciók permanenten jelen vannak az ipari robot esetén:

- Vészleállító eszköz (EMERGENCY STOP device)
- Engedélyező eszköz (Enabling device)
- Az üzemmód lezárása (a kulcsos kapcsoló által)

A következő biztonságszempontú funkciók előre konfiguráltak és a robotvezérlő biztonsági interfészén keresztül bármikor a rendszerbe integrálhatóak:

- Üzemeltető biztonsága (= a fizikaileg jelen lévő biztonsági kerítések megfigye-lésére alkalmas csatlakozó)
- Külső vészleállító eszköz

- Külső, útvonalkövető megállást előidéző eszköz (Stop category 1 (path- maintaining)) (2.4. táblázat)

Egyéb biztonsági funkciók konfigurálása esetleges, pl.¹:

- Külső engedélyező eszköz
- Tengelyspecifikus munkatér monitorozás
- Descartes-i munkatér megfigyelés
- **Descartes-i védett tér monitorozás**
- Sebesség nyomon követés
- **Tengelyekben ébredő nyomatékok megfigyelése**
- **Üktözés detektálása**

Az egyes pontokról további információ a technikai specifikációban található[12].

¹A vastag betűvel szedett pontok a szakdolgozat fontos elemei

3. ALKALMAZÁSHOZ SZÜKSÉGES MŰSZAKI FELTÉTEK ELEMZÉSE

3.1. Projekt részletes leírása

A projekt célja egy olyan robot demo hardveres és szoftveres kidolgozása, amely képes egy emberrel (továbbfejlesztés után akár egy másik robottal) lejátszani egy sakkjátszmát. A demo az ember-robot kollaboráció bemutatására szolgál, fontos szempont az interakció biztonságos megvalósítása mind az emberre, mind a környező tárgyakra tekintettel.

A megvalósításhoz a következő problémák megoldására van szükség:

1. szükséges biztonsági funkciók beüzemelése,
2. a bábuk helyzetének felismerése az egyes lépések előtt és után,
3. a bábuk megfogása és mozgatása (ide tartozik a kalibráció és a referenciafelvétel),
4. sakkalgoritmus beágyazása a programba,
5. a sakkbábúk és a tábla megtervezése és megvalósítása,
6. jelzés a robotkar számára, ha lépés történt.

A felsorolt pontok a projekt során következőképpen kerültek kidolgozásra:

- A bábuk helyzetének detektálása a projekt során QR-kód kereső és olvasó képfeldogozó eljárásokon alapul (a bábuk tetején található a kód). A kamera a roboton kerül rögzítésre.
- A bábuk mozgatása egy elektromosan vezérelt, párhuzamos megfogó (gripper) segítségével történik.
- Ahhoz hogy a bábuk megfogása egyszerű legyen, azonos magasságú és azonos módszerrel megfogható bábuk készülnek.
- A biztonsági funkciók főként az tengelyekben ébredő plusz nyomatékok monitorozására és biztonsági zónák definiálására épül.
- A tábla és a robotkar, illetve a megfogó (egy jól definiált pontja) és a robotkar relatív helyzetének kalibrálására a robotvezérlő szoftverben elérhető alapfunkciók kerültek felhasználásra.

- A sakklépés megtörténtét a robotvezérlőhöz kötött külső gombbal tudja a felhasználó jelezni.
- A képek fogadása, feldolgozása és a sakkalgoritmus futtatása mind a robotvezérlőn történik.
- Mivel a robotvezérlőn Java alapú környezet fut magas szinten, így a képfeldolgozó és a sakkozó programok is ebben lettek implementálva.

3.2. A megfogó vezérlése

3.2.1. A vezérlés hardveres kialakítása

Ahhoz hogy a megfogót (grippert) a robotvezérlőn futó programból lehessen irányítani több kiegészítő eszközre is szükség van a gripperen és a vezérlőegységén kívül. Többféle konstrukcióval is el lehet érni ezt a célt; a projekt során használt összeállítás elemei (3.1):

- **Megfogó (gripper):** párhuzamosan mozgó, két pofajú, elektromos megfogó; pontos típusa: SCHUNK MEG 50 EC[13].
- **Grippervezérlő:** a megfogóhoz tervezett vezérlő; pontos típusa: SCHUNK Controller MEG EC[13]
- **Analóg és digitális I/O modulok¹:** a robotvezérlő ezen modulok segítségével tudja irányítani a grippervezérlőt. A felhasznált eszközök Beckhoff gyártmányúak. Pontos tipusaik:
 - EL1809: 16 csatornás, digitális bemeneti modul²
 - EL2809: 16 csatornás, digitális kimeneti modul³
 - EL3002: 2 csatornás, analóg bemeneti modul⁴
 - EL4032: 2 csatornás, analóg kimeneti modul⁵
- **EtherCAT⁶ Coupler:** az I/O modulok az ú.n. E-bus-on keresztül kommunikálnak. Ahhoz hogy a robotvezérlőhöz lehessen kötni az E-bus-t, EtherCAT Coupler-re van szükség. Pontos tipusa: Beckhoff EK1100⁷.

¹I/O ((Input/Output): bemeneti és kimeneti modulok

²<https://www.beckhoff.com/english.asp?ethercat/el1809.htm>

³<https://www.beckhoff.com/english.asp?ethercat/el2809.htm>

⁴<https://www.beckhoff.hu/english.asp?ethercat/el3002.htm>

⁵<https://www.beckhoff.com/english.asp?ethercat/el4032.htm>

⁶Általános ismertető az EtherCAT-ről: <https://en.wikipedia.org/wiki/EtherCAT>

⁷<https://www.beckhoff.com/english.asp?ethercat/ek1100.htm>

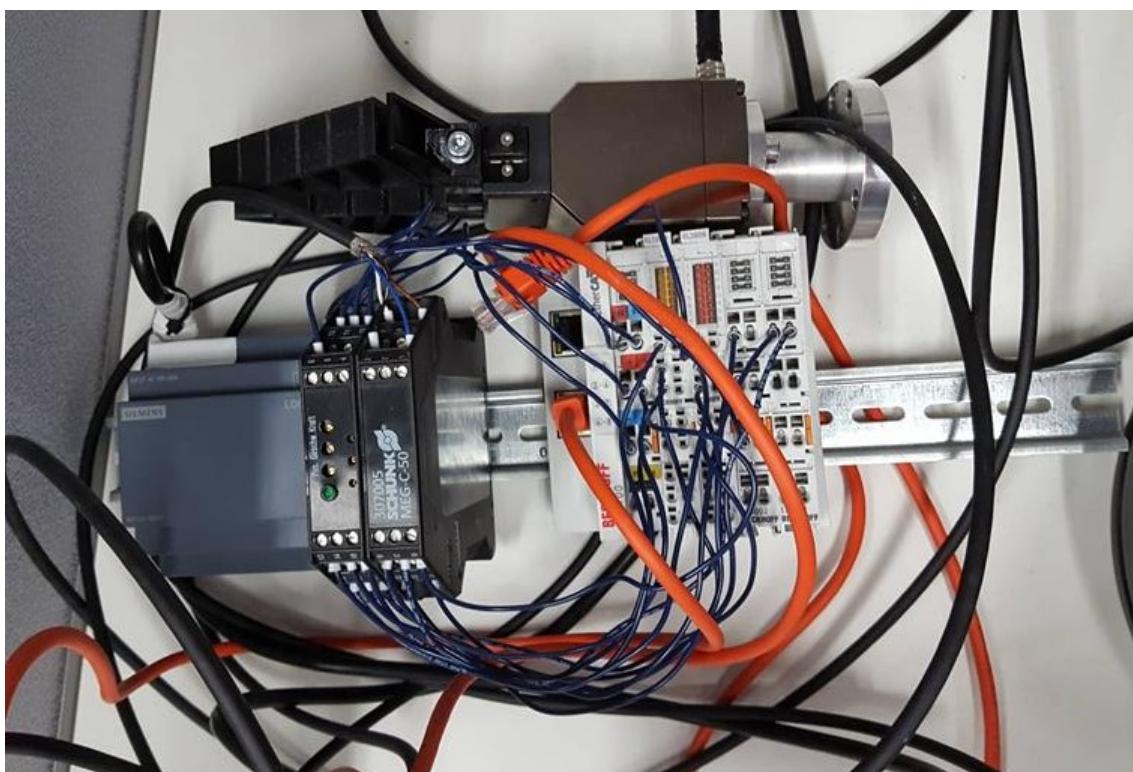
- **Tápegység:** a fentebb felsorolt eszközök mindegyike 24 V megtáplálást igényel, ezt a projekt során egy Siemens tápegység szolgáltatja.

3.2.2. A megfogó szoftveres konfigurációja

KUKA SunriseOS esetén (és általánosságban a KUKA robotok esetén) a különböző bemeneti, kimeneti és kommunikációs csatornák kezelésére I/O konfigurációs fájl generálására van szükség. Ezeket a fájlokat a WorkVisual nevű program segítségével lehet létrehozni és szerkeszteni. Adott SunriseOS-hez meghatározott a kompatibilis Sunrise Workbench verzió (ebben a programban a legegyszerűbb a robotvezérlőn futó program megírása, feltöltése). Adott Sunrise Workbench verzóval kompatibilis WorkVisual verzóra vonatkozó információkat a Sunrise Workbench-et megnyitva a Help->Sunrise.OS Release Notes menüpont alatt találunk. A szakdolgozathoz felhasznált szoftverek és környezet:

- SunriseOS 16
- Work Visual 5.0.5 build600
- Windows 10 a PC-n

A megfelelő IOConfig elkészítéséhez az alábbi lépések szükségesek:



3.1. ábra: Kép a megfogóhoz tartozó konstrukcióról

1. Ahhoz hogy a WorkVisual verziót összekössük a Sunrise Workbench-csel importálni kell a Workbench-hez tartozó Sunrise.kop fájl a WorkVisual-ba. Ez a fájl a telepített Workbench verzióhoz tartozó mappán belül a 'WorkVisual AddOn' nevű almappában található. Az importálás menete: WorkVisual->Extras->Option package management->Install... gomb. A felugró ablakban lehet ki-választani a megfelelő Sunrise.kop fájlt és telepíteni. Ahhoz hogy az egyes Beckhoff modulokkal lehessen kommunikálni szükség van a Device description file'-ok importálására. Ezek a fájlok a gyártó oldaláról letölthetők¹. A (XML) fájlok importálásához a WorkVisual->Import / Export->Import device description file lehetőséget kell választani (ezt a műveletet adott eszközön csak egyszer kell elvégezni, új projekt esetén ezt a lépést már ki lehet hagyni). A megfelelő fájlok kiválasztása és importálása után szükség van a DTM Catalog frissítésére (WV->Extras->DTM Catalog Management->Search for installed DTMs). A fájlok használatához a 'Known DTMs' részről át kell emelni az elemeket a Current DTM Catalog' részre (3.2).
2. A Sunrise Workbench-ben új projekt létrehozása után a projektre jobb egérgomb->New->I/O Configuration. A WorkVisual automatikusan elindul. A projektre jobb egérgombbal kattintva (WV-ban) a 'Set as active controller' lehetőséget kell választani. A busz struktúrákhoz hozzá kell adni a 'KUKA Extension Bus (SYS-X44)' elemet ahhoz, hogy az EtherCAT kommunikációt inicializáljuk a robotvezérlőben. Ehhez adhatjuk hozzá az EK1100 EtherCAT Coupler-t, ami az összeköttetést biztosítja a robotvezérlőben található EtherCAT hálózat és a modulok között (E-bus). Az egyes modulokat ehhez adhatjuk hozzá a programban. **Fontos:** az egyes fájlokat olyan sorrendben kell hozzáadni, ahogy azok fizikailag kapcsolódnak egymáshoz (3.3 ábra).
3. Ahhoz hogy ezeket a be- és kimeneteket a Sunrise Workbench-ben használni tudjuk szükség van Sunrise I/O Group létrehozására (VW->IO Mapping->Sunrise I/Os->Creates signals at the provider). Az I/O Group-nak tetszőleges nevet adhatunk. Az egyes be- és kimenetek kezeléséhez létrehozhatunk változókat az I/O Group-on belül. A gripper nyitásához és zárásához alapvetően 3 változó deklarálása elegendő²), pl.:
 - OpenGripper: bool, output, digital
 - CloseGripper: bool, output, digital

¹Device description files: <https://www.beckhoff.com/english.asp?download/elconfig.htm>

²A gripper nyitásához és zárásához elegendő csak a digitális ki- és bementeket használni, de például a pozíció követéséhez használni kell analóg bemeneti modult, a fogóerő programból történő beállításához pedig analóg kimeneti modult.

- Status: bool, input, digital

Az egyes változókat a kívánt bemenetekkel és kimenetekkel Drag and drop módszerrel lehet összerendezni (3.4). Megfelelő összekötés esetén a változók melllett szürke nyilak zölddé változnak.

4. Az elkészült I/O konfigurációt exportálni kell ahhoz, hogy a Sunrise Workbenchben használni lehessen (WV->File->Import / Export->Export I/O Configuration to Sunrise Workbench project). A Workbench-ben ezek után megjelenik az src' mappában egy ioAccess nevű csomag. Ezen belül található az IO Group-hoz tartozó osztály (pl.: GripperContolIOGroup.java). Ez tartalmazza a szükséges metodusokat a gripper vezérléséhez, ezeket lehet meghívni a programból.

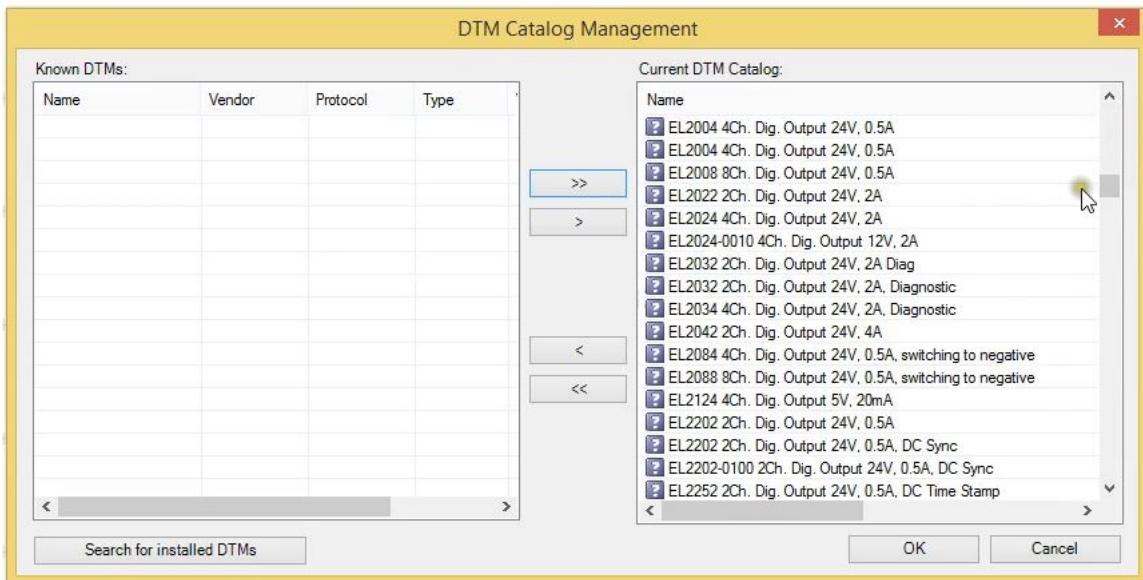
3.3. Sakkbabuk kialakítása, felismerése a képeken

A sakkprogram működtetéséhez elegendő a bábuk új pozíójának felismerése azután, miután az ember lépett. Elég csak az ember bábuit figyelni (a robot vezeti magának a bábuk pillanatnyi helyzetét). A sakkbabuk felismerére többfajta módszer is megfelelő lehet. A képfeldolgozást lehet saját vagy szabványos mintára alapozni (pl.: QR-kód alapú felismerés - 3.4 fejezet), illetve színes kamerakép esetén adott színek keresésére. A szakdolozat során a második módszer került beágyazásra, viszont a sakkbabuk formai kialakításakor a QR-kód alapú felismerhetőség is fontos szempont volt, így biztosítva az ilyen irányú továbbfejlesztés lehetőségét.

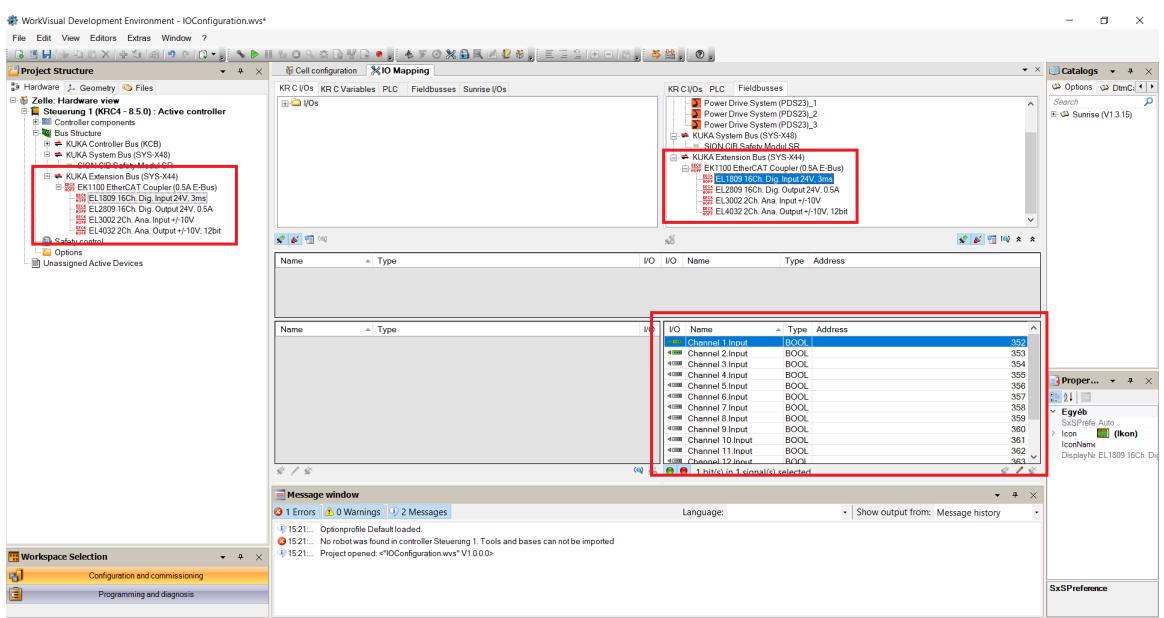
3.3.1. A sakkbabuk formatervezése

Ahhoz hogy a megfogó fel tudja emelni a bábukat könnyebség, ha a bábuk egyforma magasak, vagy ha a megfogó mindegyik bábút egy kitültetett, egy magasságban lévő részénél fogja meg (akár a talprészénél vagy alá is nyúlhatna). A talprésznel megfogás jó megoldás lehet, ha maguk a bábuk színesek, és így biztosított a képeken való felismerhetőségük. A QR-kódos továbbfejlesztéshez viszont elkerülhetetlen, hogy a QR-kód a bábuk tetején legyen elhelyezve. Ezen megfontolások alapján a bábuk végleges konstrukciója (3.5 ábra):

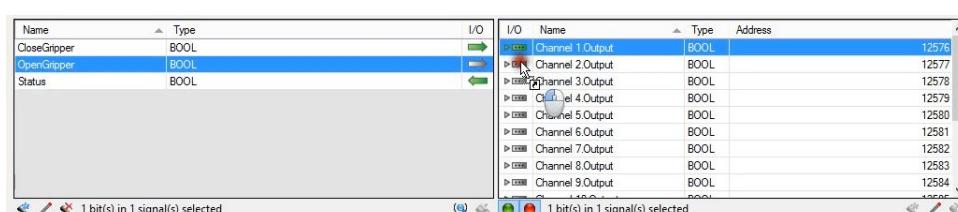
- Mindegyik bábu tetején található egy négyzet alapú, szélességéhez viszonyítva alacsony hasáb, amire az adott színt/színeket/QR-kódot rá lehet ragasztni.
- Ennél a hasábnál fogva emeli meg a megfogó a bűbut, így kevésbé kell lenyúlnia a robotkarnak, kisebb az esély arra, hogy egy másik bábút felborít.



3.2. ábra: Az xml fájlok sikeres beimportálása után láthatjuk a Device description fájlokat



3.3. ábra: Az EtherCAT konfiguráció



3.4. ábra: Az I/O változók hozzárendelése a be- és kimenetekhez

- A hasáb 6 mm vastag, hogy a megfogó viszonylag nagy terhelését (min. 50 N) elviselje.
- Mindegyik bábu egyforma magas (40 mm), így a robotkarral felemelés programja és a képfeldolgozás is nagy mértékben leegyszerűsödik.
- A bábuk viszonylag alacsonyak és a talpréssük vastagított, tömör, hogy kevésbé legyenek borulékonyak.
- Annak érdekében, hogy a bábukat kevesebb támaszanyag felhasználásával lehessen nyomtatni a bábuk tetején elhelyezkedő hasábok külön lettek kinyomatva és csak utólag lettek a bábukra felragasztva.

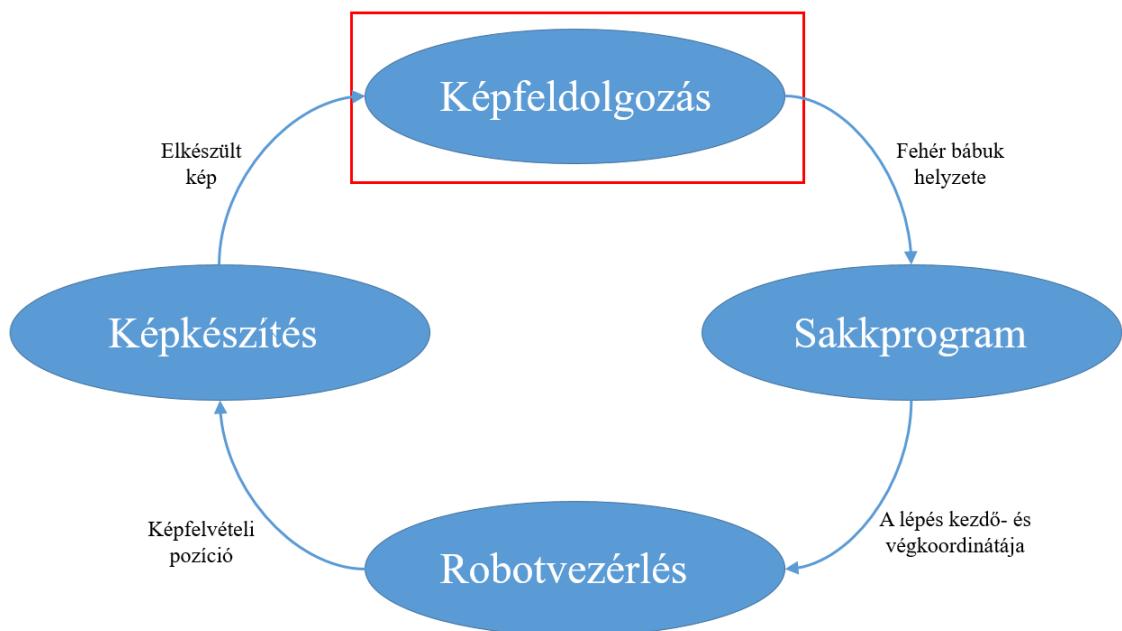


3.5. ábra: A sakkbábuk 3D-s terve



3.6. ábra: A kinyomtatott sakkbábuk

3.3.2. Bábuk pozíciójának azonosítása



3.7. ábra: A képfeldolgozás helye a folyamatban

A bábuk pozíciójának azonosításához szükség van valamilyen kalibrációs eljárásra, hogy a képek egyes részeihez a sakktábla mezőit tudjuk rendelni. A mezők definiálásához kötődő módszer alapjait OpenCV-s függvények jelentik. A kidolgozott eljárás lényege, hogy a kalibrációs képen egy sakktáblaminta belső sarokpont-

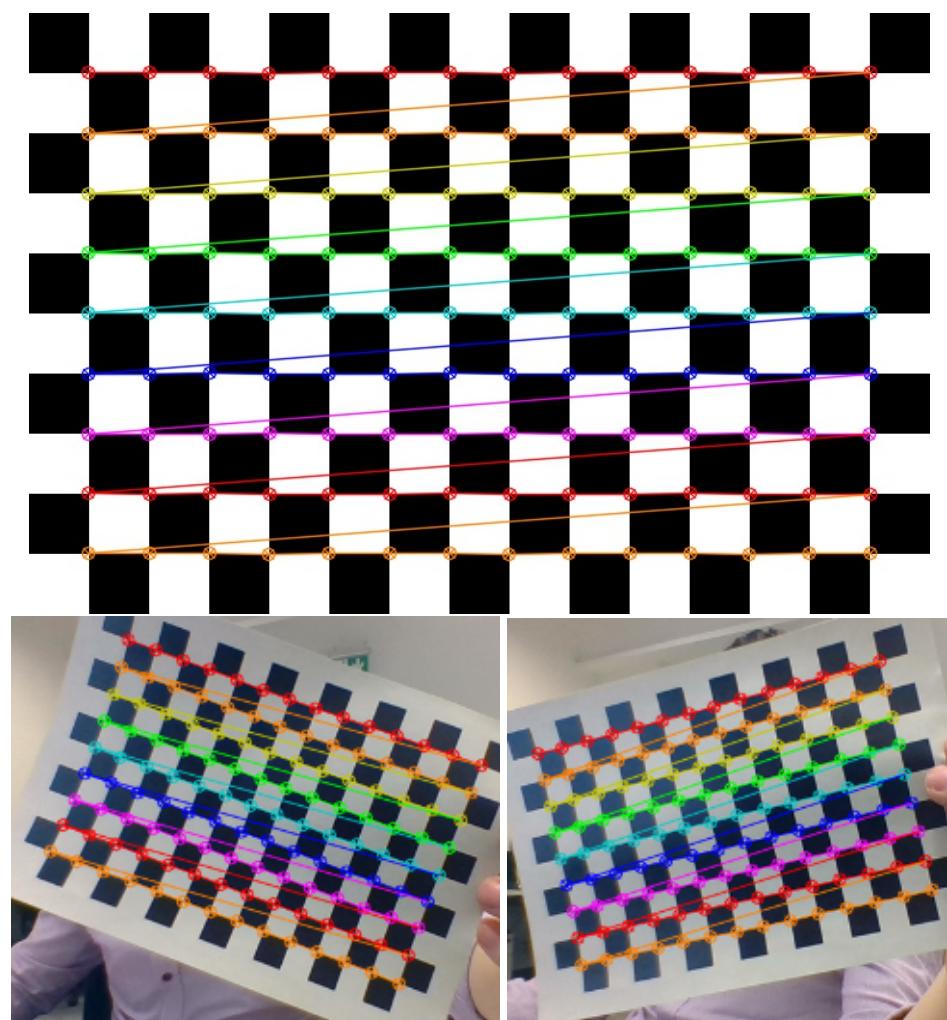
jait (azon sarokpontok, amelyek 4 mezőnek együttes sarokpontjai) határozza meg a képen. A képkoordináták alapján történik az egyes mezők felosztása az élek mentén szomszédos pontok képzeletbeli összekötésével. Mivel a perspektivikus hatások elhanyagolhatók a tábla mezőre nézve, így 2 szemközti pontra illesztett téglalap jó közelítés és az egyes képek kivágásához. Fontos elem, hogy a kalibrációs sakkmin-ta mezőszámai vízszintes és függőleges irányban különbözzenek, mert így fogja a sarokpontkereső függvény minden ugyanabban a sorrendben visszaadni a pontok koordinátáját. A sorszámozás a minta bal felső részénél lévő, fekete mező sarkától indul és soronként halad (a hosszabb élek mentén) a 3.8 ábrán látható módon¹. Az algoritmus az elmosódásokra, nagyobb szögekre ($50\text{-}70^\circ$) és a papír gyűrűdéseire nem érzékeny.

Fontos elem ezeken túl, hogy a kalibrációs lap abban a síkban helyezkedjen el, ahol a bábukon a zöld minta, így biztos megfelelő helyen keressük a bábukat. Ha az alsó tábla szintjére helyeznénk a kalibrációs lapot, akkor az egyes mezőkre helyezett bábukon lévő minták átlóghatnának a szomszédos mezőkre. Ezt mutatja be a 3.9 ábra. Ezek alapján a kalibrációs eljárás lépései:

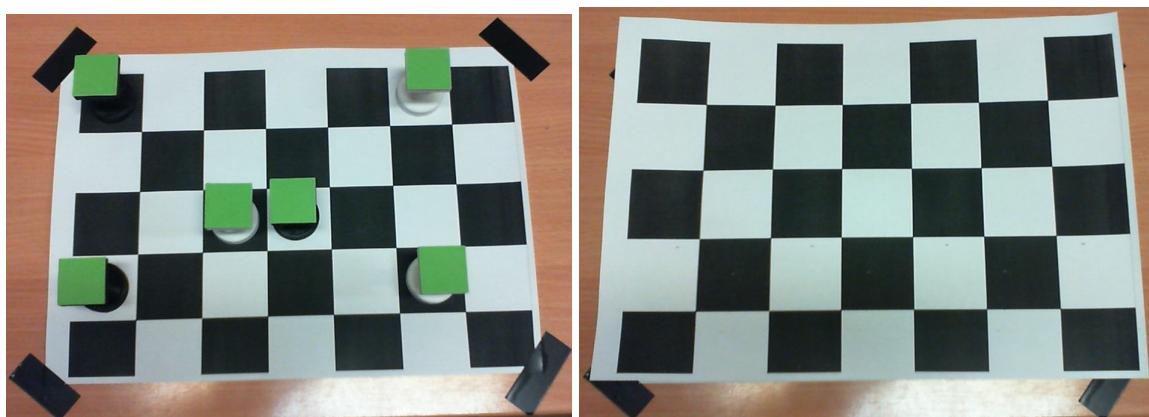
1. Néhány sakkbábú táblára helyezése.
2. A bábuk teteje által kifeszített síkra egy sakktáblamintás papír ráhelyezése úgy, hogy az egyes mezők az alsó sakktábla mezői felett legyenek (mindegyik alsó sakktáblamező felett legyen egy belső sarokpontokkal határolt mező).
3. Kép készítése abból a pozícióból és szögből, ahova a robotkat a kamerával minden képkészítés előtt vissza fog térti.
4. A képen a sarokpontok keresése és a mezők meghatározása automatikusan történik a programban, de a sarokpontok számát vízszintes és függőleges irányban előre definiálni kell.

A képfeldolgozáshoz szükséges osztályokat az ‘imgprocess’ csomag tartalmazza (az OpenCV-s és a QR-kódhoz tartozó .jar fájlok a Build path-hoz hozzá kell még adni). A sakkbábuk pozíciójának felismerését végző metódusok a ScanPieces-Position osztályban vannak implementálva. Az osztályt példányosítani a (BufferedImage) kalibrációs képpel lehet, megadva még a horizontális és vertikális sarokpontok számát függvényparaméterként. A példányosítás hatására lefut a sarokpontkereső függvény. A ScanPosition függvényt erre a példányra meghívva visszakapjuk a fehér bábuk helyét egy boolean mátrix formájában, ahol a mátrix egyes elemei a tábla egyes mezői. A kép mezőkre darabolását a CropImages privát függvény végzi,

¹<http://stadatum.blogspot.com/2016/05/corner-identification-in-computer.html>



3.8. ábra: A sakktáblaminta feltérképezett pontjai



3.9. ábra: Kalibrációs kép felvétele

amely a kalibráció során meghatározott sarokpontok alapján darabolja fel a képet: a kapott képek a mezők bal felső és jobb alsó pontjai által meghatározott téglalapon belüli részek.

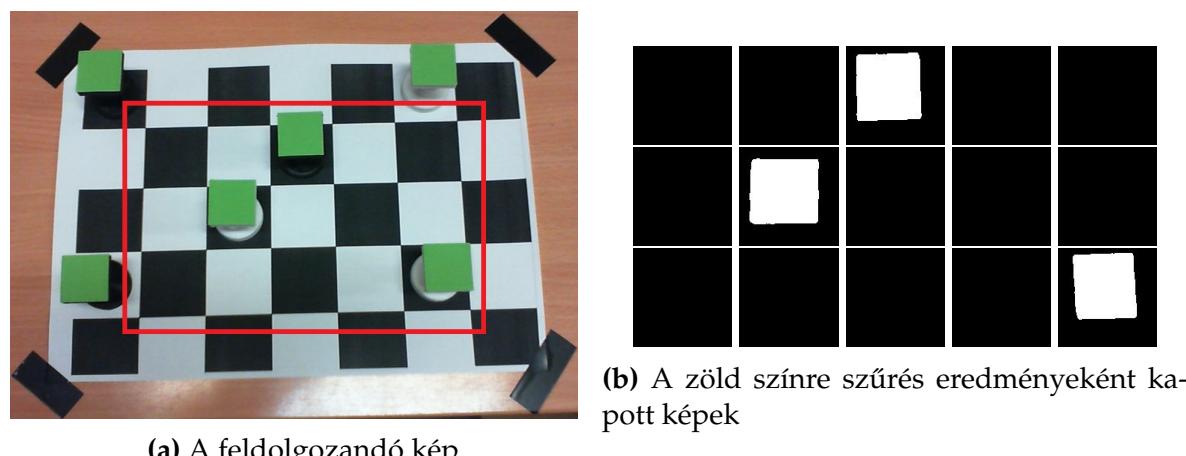
A kapott képekről a FindGreen függvény dönti el, hogy található-e rajtuk jelenősebb kiterjedésű zöld szín. A zöld színre szűrést RGB színtartományban vizsgálja a program (HSV színskálába transzformálással a szűrés még finomítható és kevésbé lesz érzékeny a különböző fényviszonyokra is). Az egyes képek minden pixele esetén kiszámol a program egy értéket a következő képlet alapján:

$$value = greenComponent - \frac{redComponent}{2} - \frac{blueComponent}{2}$$

Ha ez a ‘value’ érték egy megadott határértéknél nagyobb (jelenleg 40 egy 0-tól 255-ig terjedő skálán), akkor a pixel színét a szűrő algoritmus fehérre állítja, ellenkező esetben feketére. Utolsó lépésként a szűrő algoritmus minden mezőre külön-külön kiszámítja a fehér és a fekete pixelek számának az arányát. Ha ez az érték 10%-nál nagyobb egy mező esetén, akkor ott található bábu, azaz a hozzá tartozó boolean mátrix elem true értékű lesz. A visszaadott mátrixot a sakkprogram kapja majd meg, ami ez alapján és az előző állás alapján dönti el, hogy mi volt a lépés.

3.4. QR-kód alapú képfeldolgozás - továbbfejlesztési irány

Ahhoz, hogy sakkjátékot tetszőleges állapotból lehessen kezdeni, illetve folytatni szükség van a bábuk helyének felismerésén túl azok típusának egyértelmű felismerésére. Erre alkamas megoldás lehet a bábuk tetejére helyezett QR-kód, amelyben kódolva van a bábu típusa. A QR-kódok kezelésére jó választás a nyílt forráskódú



3.10. ábra: Zöld színre szűrés (a bal oldali képen a pirossal bekeretezett rész lett kalibrálva, azokat a mezőket vizsgálja)

ZXing („Zebra Crossing”) program¹. Ez a Java könyvtár (library) alkamas különböző formátumú, egy- és kétdimenziós vonalkódokkal kapcsolatos képfeldolgozásra, amelynek csak egyik eleme a QR-kód olvasás és generálás.

3.4.1. A forráskód build-je

A könyvtár egyszerű használatához célszerű .jar fájlokat generálni a forráskódból. Az első lépés a Github-on elérhető forráskód letöltése vagy klónozása a saját számítógépre. A programkód számos mappába és almappába van rendszerezve az egyes moduloknak megfelelően (pl.: core/ és javase/). **Fontos:** a mapparendszert olyan helyre tegyük a számítógépen, melynek elérési útjában nem található szóköz karakter (ékezetes karakter is probléma lehet)! minden Java alapú modul esetén található egy pom.xml fájl, amit Apache Maven² segítségével lehet használni.

Szükségünk van megfelelő java verzió telepítésére. A JRE (Java Runtime Environment) helyett a JDK (Java Developement Kit) valamelyik verzióját (a projekt jdk 10.0.2 verziót használ) érdemes telepíteni, ha fejlesztői funkciókat is igénybe szeretnénk venni (a JRE csomagot ez már tartalmazza). Az Apache Maven telepítése után szükségünk van az ehhez és a JDK-hoz tartozó környezeti változók beállítására. Ezt a Vezérlőpult->Rendszer->Speciális rendszerbeállítások->Környezeti változó...->Rendszerváltozók címszó alatt tehetjük meg. Szükségünk van egy ‘JAVA_HOME’ és egy ‘M2_HOME’ változóra (3.11. ábra).

Parancssorban navigálunk a ZXing projekt gyökeréhez és futtassuk a ‘mvn install’ parancsot a fordításhoz, a tesztekhez és az összes modul felépítéséhez. A „-DskipTests” paraméter hozzáadásával a unit teszteket kihagyhatjuk. Szükség lehet a -Drat.ignoreErrors=true’ paramétere a licensz tesztekkel kapcsolatos problémák ignorálásához. A build folyamat akkor mondható sikeresnek, ha minden modul felépítése sikeres (‘ANDROID_HOME’ környezeti változó beállítása nélkül az Androidhoz kapcsolódó modulokat nem build-eli) (3.12. ábra).

A lefordított .jar fájlokat ezt követően az egyes modulokon belül találjuk. Például a lefordított core/ kód helye a core/target/core-x.y.z.jar. Ezeket lehet beimportálni a képfeldolgozást megvalósító projektbe.

3.4.2. ZXing könyvtár beimportálása és használata

Az iiwa robotkart programozni Sunrise Workbench használatával a legegyszerűbb, ami egy JAVA Eclipse platformú szoftver. Emiatt praktikus okokból a szakdolgozat képfeldolgozási és sakkalgoritmus beágyazási része túlnyomó részt Eclipse-ben

¹További információk és forráskód: <https://github.com/zxing/zxing>

²Az Apache Maven program ingyenesen letölthető: <https://maven.apache.org/>

Rendszerváltozók	
Változó	Érték
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\PROGRA~1\Java\jdk-10.0.2
JAVA_TOOL_OPTIONS	-Dfile.encoding=UTF8
LSTC_LICENSE	ANSYS
M2_HOME	C:\PROGRA~1\Apache\maven
NUMBER_OF_PROCESSORS	4
OMP_NUM_THREADS	4
OS	Windows NT

Új... Szerkesztés... Törlés

3.11. ábra: Szükséges környezeti változók beállítása

```
[INFO] Reactor Summary:
[INFO]
[INFO] ZXing 3.3.4-SNAPSHOT ..... SUCCESS
[INFO] ZXing Core ..... SUCCESS
[INFO] ZXing Java SE extensions ..... SUCCESS
[INFO] ZXing zxing.org web app 3.3.4-SNAPSHOT ..... SUCCESS
[INFO] -----
[INFO] BUILD SUCCESS
```

3.12. ábra: Sikeres build végeredménye

történt (verzió: 4.9.0)¹.

A könyvtár beimportálásához létrehoztam egy java projektet. Erre jobb egér-gombbal kattintva elnavigáltam a ‘Configure Build Path...’ menüponthoz (3.13. ábra). A ‘Classpath’-t kiválasztva jobb oldalt aktivizálódik az ‘Add External Jars...’ gomb. Kiválasztottam a *core/* és a *javase/* modulok .jar fájljait. Ezekben belül lehetőség van forráskód (Source attachment) és dokumentáció (Javadoc location) csatolására (3.13. ábra). ‘Apply and Close’ után megjelennek a csomagok a hivatkozott könyvtárak (Referenced libraries) pont alatt.

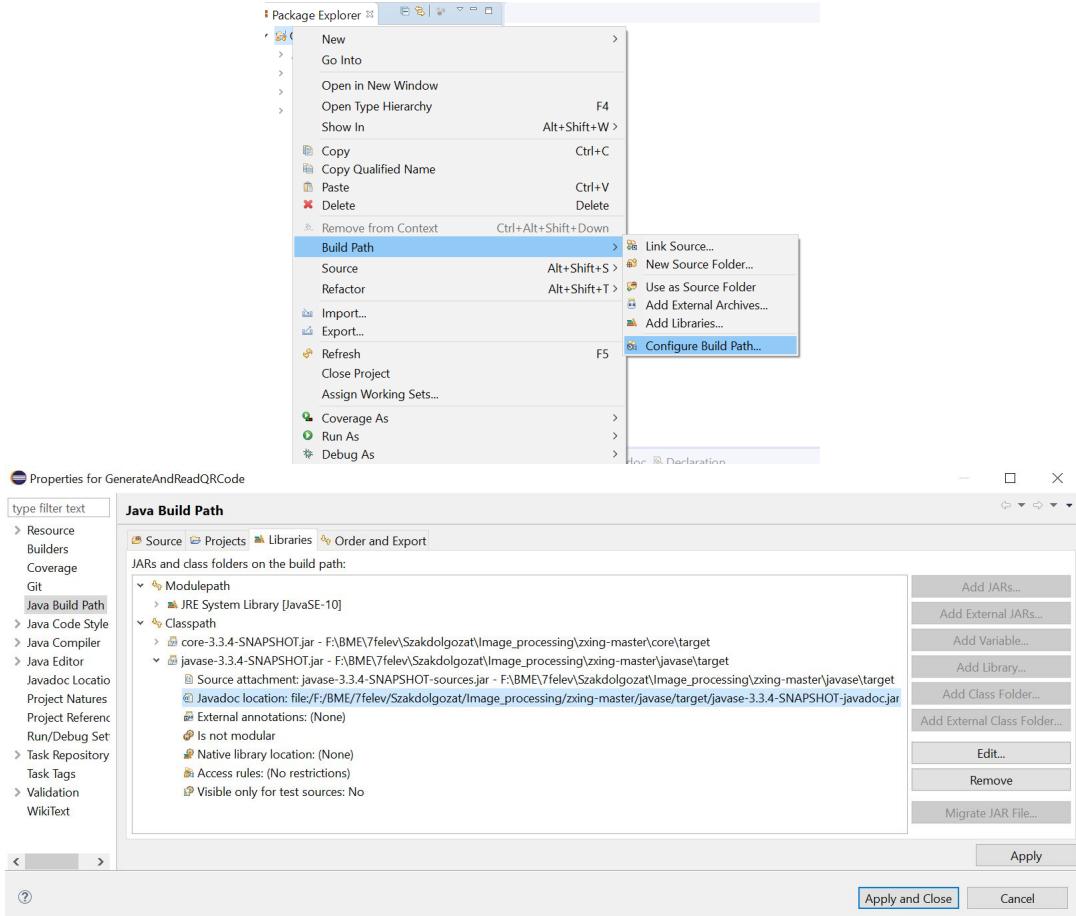
3.4.3. QR kód generálás és olvasás

A ZXing könyvtárt felhasználó, ehhez a projekthez szükséges elemek az ‘imgprocess’ csomagban találhatóak a ‘QRCodeMethods.java’ fájlban. A függvények leírása a 3.1 táblázatban találhatóak².

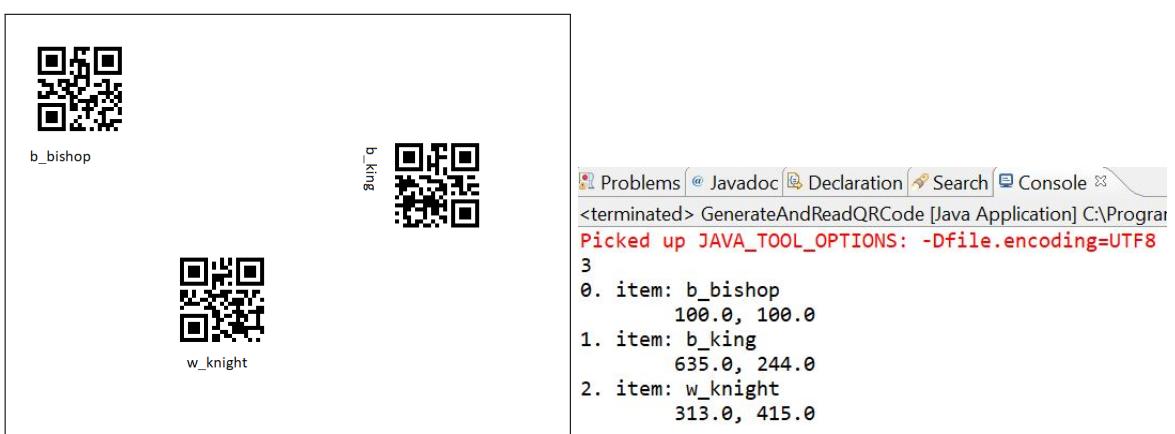
¹Link: <https://www.eclipse.org/>

²Forráskód: <https://github.com/rolandvarga601/Szakdolgozat>

3.4 QR-kód alapú képfeldolgozás - továbbfejlesztési irány



3.13. ábra: ZXing hozzáadása a hivatkozott könyvtárakhoz



3.14. ábra: Példa a több QR-kódot tartalmazó kép kiolvasására

Függvény neve	Paraméterek	Leírás
createQRCode	<ul style="list-style-type: none">String qrCodeData - a kódolt szövegString filePath - a képek mentési helyeString charset - karakterkódolásint qrCodeheight - a kép magasságaint qrCodewidth - a kép szélessége	Nincs visszatérési értéke, a QR-kódot tartalmazó képet a megadott elérési helyre menti.
readQRCode	BufferedImage image - a kép amelyen a QR-kód található	Visszaadja a képen található QR-kódba kódolt szöveget.
readMultiQRCode	String FileName - a kép elérési útja, amelyen a QR-kódok találhatóak	Képes egyazon képen különböző szögekben elhelyezkedő QR-kódok felismerésére és dekódolására (3.14 ábra). Visszatérési érték: Result[], amely tömb elemei tartalmazzák többek között az egyes QR-kódok helyét a képeken és a kódolt szöveget.

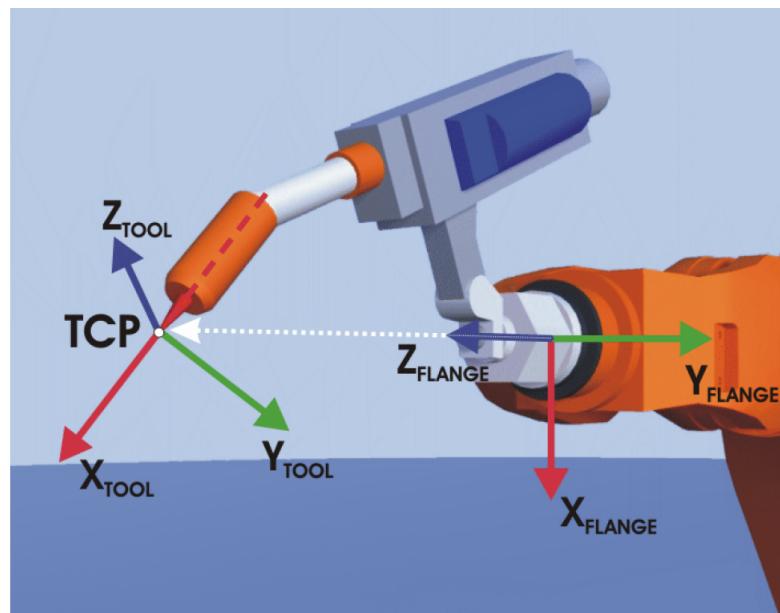
3.1. táblázat: QRCodeMethods.java függvényei

4. ROBOTKAR KALIBRÁCIÓ, REFERENCIA FELVÉTEL KIDOLGOZÁSA

4.1. Koordinátarendszerek

Ha a robotkar pozíciójáról beszélünk, akkor elsősorban a végpontjának vagy az arra szerelt szerszámnak a pozíójára és orientációjára vagyunk kiváncsiak. Az ilyen elhelyezkedést a térben különböző koordinátarendszerekkel és a közöttük felírható transzformációkkal tudjuk jellemzni. A főbb koordinátarendszereket foglalja össze a 4.1 táblázat.

Ahhoz hogy egy test pozícióját és orientációját definiálni tudjuk a térben legalább 6 lineárisan független koordinátára van szükség. A viszonyítási koordinátarendszerben felírt 3 transzlációs és 3 rotációs koordináta megfelel erre a célra.



4.1. ábra: A TCP (Tool Center Point) elhelyezkedése[12]

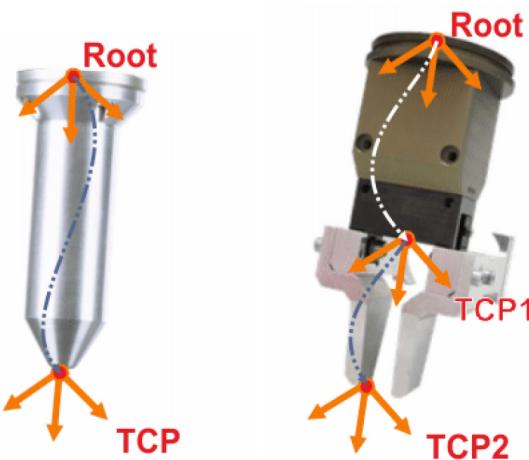
- Transzlációs vektorok:
 - X távolság: a referencia KR X tengelye mentén vett transzláció
 - Y távolság: a referencia KR Y tengelye mentén vett transzláció
 - Z távolság: a referencia KR Z tengelye mentén vett transzláció
- Rotációs vektorok:
 - A szög: a referencia KR Z tengelye körül vett forgatás
 - B szög: a referencia KR Y tengelye körül vett forgatás
 - C szög: a referencia KR X tengelye körül vett forgatás

Koordinátarendszer (KR)	Leírás
Világ KR	A világ KR egy állandó, Descartes-i (Cartesian) koordinátarendszer. Ez minden más KR gyökere, mint például a bázis KR-é vagy a robot bázis KR-é. Alapértelmezetten a világ KR a robot bázispontjában található.
Robot bázis KR	A robot bázis KR-e olyan Descartes-i KR, amely a robotkar bázispontjában található. Ez definiálja a robotkar relatív helyzetét a világ KR-hez képest. Alapértelmezetten a robot bázis KR megegyezik a világ KR-rel. Meg lehet adni egy elforgatási vektort a Sunrise.Workbench-ben, amely definiálja a robot relatív forgatását a világ KR-hez képest. Alapbeállítésként a padlóra rögzített robot felszerelési orientációja: ($A=0^\circ$, $B=0^\circ$, $C=0^\circ$).
Bázis KR	Ahhoz hogy a Descartes-i térben mozgásokat definiálhassunk szükség van referencia KR (bázis) felvételére. Sztenderd módon a világ KR a mozgás bázis KR-e. További bázis KR-eket lehet definiálni a világ KR-hez képest. Ezt mutatja be a FEJEZET HIVATKOZÁS!!!!.
Flange KR	A flange KR írja le a robot flange középpontjának a pozícióját és orientációját. Ennek az elhelyezkedése nem fix, a robottal együtt mozog. A flange KR használható a rá fogatott szerszámokhoz kötődő KR-ek origójaként. Például a szakdolgozat keretében a gripper egy jól definiált pontjára illesztett KR a robot flange KR-éhez képest relatív lett megadva (4.1).
Szerszám KR	A szerszám KR az a Descartes-i KR, amely a felszerelt szerszám munkapontjára illeszkedik. Ezt hívják szerszám középpontnak (Tool Center Point - TCP). Bármennyi frame-et lehet definiálni egy szerszámhoz és ezek mindegyike kiválasztható TCP-nek. A szerszám KR rendszerint a flange KR-ból származtatott.

4.1. táblázat: A főbb koordinátarendszerek

4.2. Szerszámkalibráció - XZY 4-pont metódus

A sakkprojekt kivitelezése során szükséges a robotkarra szerelt megfogó kalibrálása ahhoz, hogy a pozicionálás minél pontosabb legyen. A robotkar szoftverében ez egy alapfunkció, kiegészítő csomag nem szükséges hozzá. A kalibrációs eljárás alapvetően 2 lépésből áll:



4.2. ábra: Szerszám 1 (balra) és 2 (jobbra) TCP-vel[12]

1. a TCP origójának meghatározásból,
2. az origóra illesztett koordinátarendszer orientálásából.

A második lépés jelen esetben elhagyható, megfelelő ha a TCP az orientációját a referencia KR-től örökli.

A TCP origójának meghatározására használt módszer: XYZ 4-pont metódus. Ehhez ki kell választanunk a szerszám egy adott pontját adott helyzetben (pl.: gripper egyik csúcspontja nyitott állásban); ez lesz a TCP. Az eljárás lényege, hogy a kalibrálni kívánt szerszám ezen pontját egy referenciaponthoz vezéreljük 4 különböző irányból. A referencia pont szabadon választható. A robot kontroller a különböző flange pozíciókból számolja ki a TCP helyzetét.

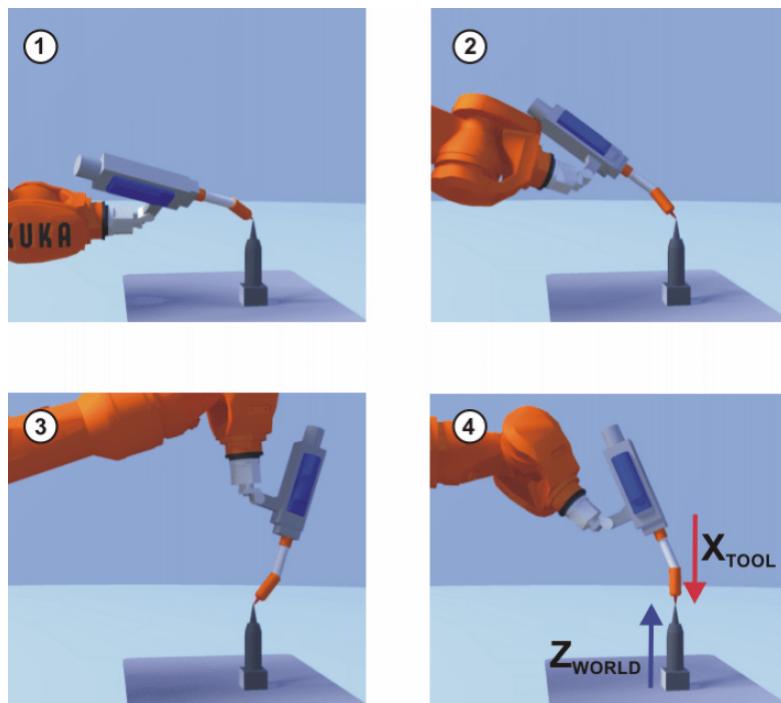
A 4 flange pozíció egymás közötti távolságainak meg kell haladniuk egy előre definiált minimumot. Ha a pontok túl közel vannak egymáshoz, akkor a pozíció adatokat nem lehet elmenteni. Erre hibaüzenet figyelmeztet.

A kalibráció minősége a transzlációs vektor hibájával mérhető, amit a program kalibráció közben számít ki. Ha ez a hiba meghalad egy definiált határértéket, akkor célszerű a TCP-t újból kalibrálni.

A minimális távolságok és a maximális számítási hiba a Sunrise Workbench-ben konfigurálható. Részletesebb leírás az eljárásról a megfelelő Sunrise.OS verzió manuáljában található[12].

4.3. Báziskalibráció - 3 pont metódus

A báziskalibráció során a felhasználó egy Descartes-i koordinátarendszert (bázis koordinátarendszert) rendel a bázisnak választott frame-hez. A bázis KR-nek a felhasználó által választott, tetszőleges helyen lehet az origója.



4.3. ábra: XYZ 4-pont metódus 4 lépése[12]

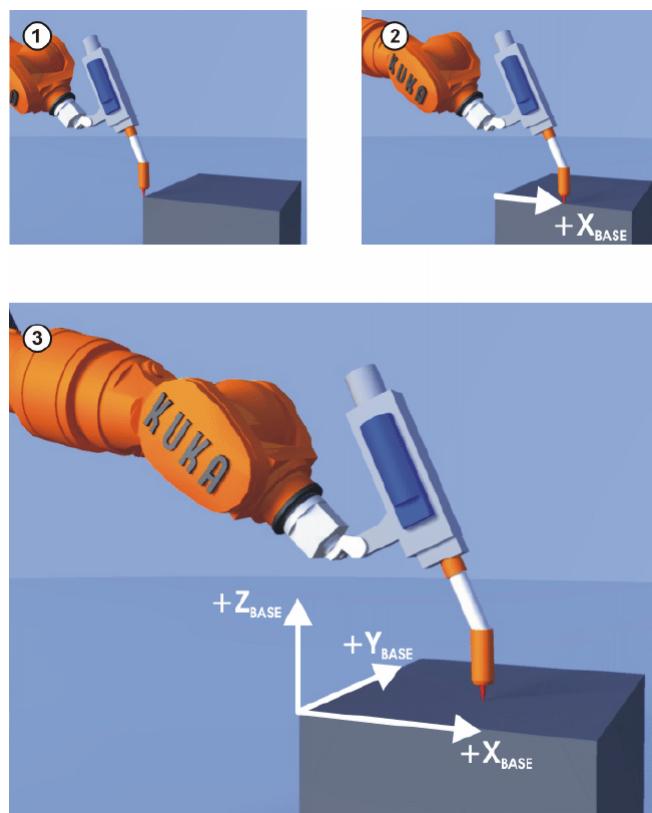
A bázis kalibráció előnyei:

1. A TCP végigvezérelhető a munkafelület szélein vagy egy munkadarabon.
2. A bázishoz viszonyítva lehet felvenni a szükséges pontokat. Ez a szakdolgozat során azért fontos, mert így nem kell a sakktábla egyes mezőihez tartozó pontokat külön felvenni, a pozíciókat meg lehet adni a bázisponthoz képest, ami lehet például a sakktábla egyik sarokpontja.

A 3-pont metódus során az origó és a bázis 2 további pontja kerül rögzítésre. Az origó felvétele után a kívánt X tengely pozitív felén egy tetszőleges pont rögzítése következik. Végezetül az XY sík első térnegyedébe eső, szintén tetszőlegesen választott pontot kell felvenni. Ez a 3 pont egyértelműen meghatározza a bázist.

A rögzített pontok origótól vett távolságának meg kell haladnia egy minimumot, illetve az egyenesek között is meg kell lennie egy minimum szögnek (origó - X tengely és origó - XY síkon felvett pont). Ha a pontok túl közel vannak vagy az említett szögek túl kicsik, akkor a pozíció adatokat nem lehet elmenteni. Erre hibaüzenet figyelmeztet.

A minimális távolságok és szögek a Sunrise Workbench-ben módosíthatók. Részletesebb leírás az eljárásról a megfelelő Sunrise.OS verzió manuáljában található[12].



4.4. ábra: Báziskalibráció 3-pont metódussal[12]

4.4. A szerszám tömegeloszlásának meghatározása

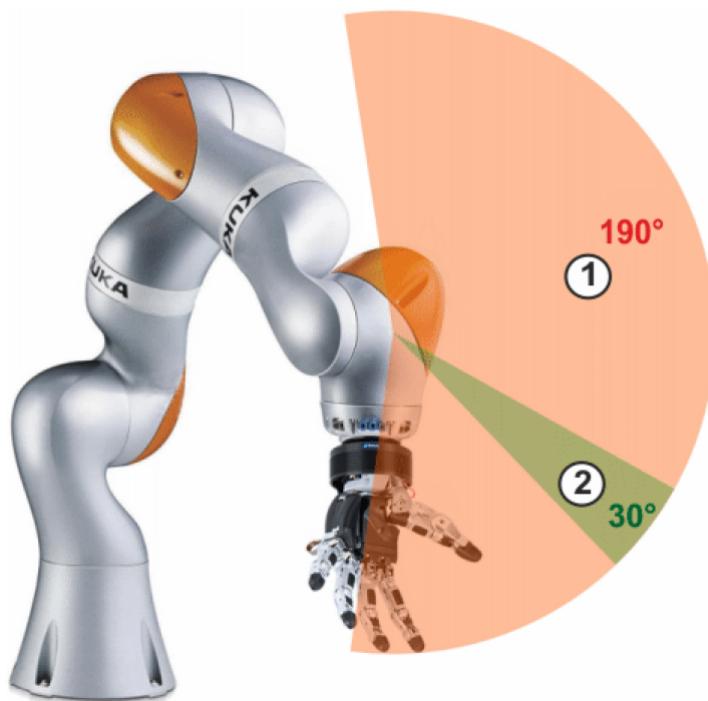
A robotkaron tengelyeiben ébredő többlet nyomatékok meg lehet határozni egy maximális értéket, így ha a munkaterében lévő objektummal ütközik, detektálja és megáll. A robot saját tömegének dinamikus mozgatásához szükséges nyomatéka nem adódik hozzá ehhez a többlet nyomatékhöz. Adott szerszám robotkarra erősítésével viszont előfordulhat, hogy a maximális járulékos nyomatékérték átlépi valamelyik tengelyen a megengedett nyomatékértéket. Ezen hatás korrigálásához szükséges a szerszám tömegeloszlásának kalibrálása (a sakkbábuk, mint munkadarabok kalibrálása nem szükséges, azok tömege elhanyagolható).

A tömegeloszlás meghatározásához a robot először különböző méréseket futtat automatikusan a csuklótengelyek (utolsó három tengely: A5, A6, A7) mozgatásával. Ez alapján a robotkarra szerelt szerszám tömege és a tömegközéppontjának helye meghatározható. Másik lehetőség a szerszám tömegének megadása (például katalogus alapján), majd ez alapján a tömegközéppont helyének kalibrálása.

Az eljárás futtatásához nincs szükség plusz csomagra, a funkció a robotszoftver beépített eleme. A lépések a következők (automatikusan történik, a folyamat 1-2 percert vesz igénybe):

1. A folyamat kezdetén az A7 tengelyt a vezérlő a 0 pozícióba mozgatja. Ezen kívül az A5 tengelyt úgy pozicionálja, hogy az A6-os tengelyre többletnyomatékot a súly ne fejtsen ki. Ekkor a szerszám tömegközéppontja abba a síkba esik, amit az A6 tengely és a gravitációs térerősségvektor kifeszít.
2. A mérés futása során az A6 és az A7 tengelyek egy bizonyos tartományban vesznek fel helyzeteket:
 - Alapbeállításként az A6 tengely -95° és $+95^\circ$ közötti részét vesz fel pozíciókat. Ha a munkatér nem elég nagy ehhez a mozgástartományhoz, akkor ez a tartomány csökkenthető.
 - Az A7 tengely 0° és -90° között mozog.

A mérést befolyásoló tényezőkről információ a Sunrise kézikönyvében[12] található.



4.5. ábra: Az A6 tengely mozgástartománya: 1-es esetben teljes, 2-es esetben csökkentett tartomány[12]

5. ROBOTKAR MOZGÁSÁNAK DEFINIÁLÁSA ÉS PROGRAMOZÁSA

6. SAKKALGORITMUS BEÁGYAZÁSA

A robotprogramhoz felhasznált sakkalgoritmusok és főbb Java osztályok alapja egy versenyre készített, nyílt forráskódú sakk alkalmazás [14]. 2005-ben Arwid (Arvydas) Bancewicz első helyezést ért el ezzel az alkalmazásával az OBEA Számítógépes Programozó Versenyen (OBEA Computer Programming Contest) 17 évesen. Az általa készített program rendelkezik grafikus felhasználói felülettel (továbbiakban GUI - Graphical User Interface), a forráskód nagy része ennek megfelelő működtetéséhez lett megírva.

A szakdolgozat keretében megvalósított projektem során külön a sakk alkalmazáshoz felhasználói felületet létrehozni nem volt szükségszerű. A robotprogram továbbfejleszthető olyan formában, hogy folyamatosan kijelezze a jelenlegi sakkállást, így még inkább nyomon követhető a játék menete, így megkönnyítve a továbbfejlesztést. Ezen a felületen keresztül akár tetszőleges felállást lehetne konfigurálni a játék kezdetéhez.

A sakkprogram beágyazásának első fő kihívása a GUI elhagyásával egy konzolalkalmazás létrehozása. A program forráskódja viszonylag hosszú, sok osztály lett implementálva különböző csomagokba rendezve (12 csomag és ezeken belül 93 osztály). A program működésének megértéséhez jó kiindulási pont a mellékelt README fájl és a „program documentation” mappában található szotver leírás (Software Documentation.doc)[14]. Habár ezek főként a GUI működését taglalják, az ehhez tartozó főprogram kódjából a meghívott függvények alapján ki lehet következtetni a működés struktúráját. Ezen felül a Java oszányoknak és az egyes csomagoknak beszédes nevük van, például a következő sakklépést az algorithm (algoritmus) csomagban található különböző algoritmusokra alapozott osztályok metódusai határozzák meg.

A szakdolgozathoz felhasznált csomagok az alábbiak (ezeken belül is bizonyos funkciók ki lettek kapcsolva, de ezek jelentik a program magját):

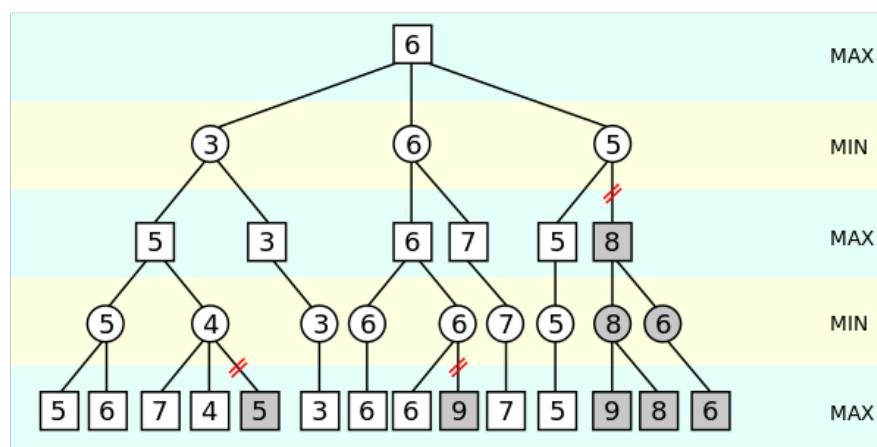
- chess.algorithm: Ebben a csomagban kerültek implementálásra a különböző sakkalgoritmusokat meghívó és végrehajtó utasítások.
- chess.core: Ez a csomag tartalmazza a sakkhoz szorosan kötődő osztályokat, metódusokat (pl.: bábuk szabályos lépései, sakkóra...).
- chess.properties: Itt találhatóak a programozott sakkjáték szempontjából praktikus funkciók (pl.: játék jelenlegi állása, játékosok neve...).

6.1. A chess.algorithm csomag - az algoritmusok működtetője

A programban az alábbi sakkalgoritmusok lettek implementálva:

- Alfa-béta vágás (AlphaBeta.java)
- Minimax elv (MiniMax.java)
- NegaScout algoritmus (NegaScout.java)
- Principal variation search (PrincipalVariation.java)
- Kvázi véletlenszerűen választott szabályos lépés (RandomGen.java)

A sakkprogram alapbeállításként az Alfa-béta vágást használja. Az Alfa-béta vágás egy olyan kereső algoritmus, amely igyekszik csökkenteni a minimax algoritmus keresési fájában lévő kiértékelt elemek számát. Ezt a módszert gyakran alkalmazzák kétszemélyes játékok (pl.: amőba, sakk, go, stb.) esetén gépi játékos programozására. Egy adott lépés kiértékelését akkor szakítja meg teljesen, ha legalább egy válaszlépés bebizonyítja, hogy a lépés rosszabb, mint a korábban vizsgált lépés. Az ilyen lépések további vizsgálata felesleges. Ha egy hagyományos minimax fára alkalmazzák, akkor ugyanazt a lépést adja majd vissza, mint amit a minimax algoritmus adna, de kimetszi azokat az ágakat a fában, amik a kimenetet nem befolyásolják.¹



6.1. ábra: Az alfa-béta vágás szemléltetése. A kiszürkített részfák vizsgálata felesleges (a lépések jobbról balra történő kiértékelésekor). A max és min szintek reprezentálják a játékos, illetve az ellenfelének a lépéseiit.²

¹Szöveg forrása angol nyelven: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

²A kép forrása: https://upload.wikimedia.org/wikipedia/commons/thumb/9/91/AB_pruning.svg/600px-AB_pruning.svg.png

Az algoritmus két változót értékel ki minden lépésben, alfát és bétát. Az alfa érték reprezentálja azt a minimum pontszámot, amely az u.n. ‘maximalizáló’ játékos számára már biztosítva van, illetve a béta érték az a maximum pontszám, ami az u.n. ‘minimalizáló’ játékos számára biztosított. Kezdetben alfa értéke negatív végiglen, béta pozitív végtelen, azaz mindenkét játékos a saját legrosszabb lépéssel indít. A fa rekurzív vizsgálata során folyamatosan változik az értékük a játékosok által garantáltan elérhető értékekre. Amint a béta érték az alfa alá csökken az azt jelenti, hogy ez az állás (ha minden játékos részéről a legjobb lépéseket tételezzük fel) nem állhat elő, így további vizsgálatuk felesleges.

Az algoritmusokat a sakk szabályaihoz és sajátosságaihoz a MoveAlgorithm illeszti. Ez az osztály az alábbi funkciókat tartalmazza (egyéb kiegészítő, teszteléshez megírt függvényeken felül):

- Definiálja az egyes bábuk lehetséges lépései mezőkre lebontva, tehát az erre vonatkozó metódusok pontosan megadják, hogy az egyes bábuk melyik mezőkről melyikekre léphetnek. Példának okáért a gyalog esetén külön a fehér bábukra és külön a feketékre is meg van határozva (getPawnMoves(Coord c) függvény), hogy a kiindulási pontjukról az ellenfél irányába egyet vagy ket-töt is léphetnek. Bármely más esetben 1-et léphetnek előre (az ütések külön függvények definiálják, ezek elkülönítése a robotprogramozás esetén is kifejezetten előnyös).
- Az egyes bábuk lehetséges ütéseit külön függvények határozzák meg. A gyalog kivételével a többi bábunál ez megegyezik az előző pontban leírt lépésekkel.
- A bábuk játékhelyzettől függő összes lehetséges lépését, ütését a getRealMoves’, a getRealAttacks’ és a getRealAll’ függvények adják vissza. Ezek kizártján azon lépéseket, melynél az adott játékos királya a lépés előtt és utána is sakkban áll. Ezt úgy vizsgálja meg a program, hogy adott játékállásban „meglépi” a kívánt lépést, majd ellenőrzi, hogy a király sakkban maradt-e.
- Itt találhatóak még az egyes játékállások „költségét” meghatározó függvények, amelyek által visszaadott értékek az algoritmusok működtetésének alapjait jelentik.

6.2. A chess.core csomag - a játék magja

A chess.core csomag definiálja az algoritmusok működtetéséhez nem szorosan kapcsolódó osztályokat. A fő fájl a ChessGame.java, ennek segítségével lehet egy játékot elkezdeni. A példányosításakor az alábbi inicializáló lépések futnak le:

1. A játék állapota inicializáltra változik (a játékállapotokról a chess.properties csomag kapcsán lesz bővebben szó).
2. Alapértelmezetten az Alfa-béta kereső algoritmus kerül beállításra. Ezt a játék során bármikor lehet módosítani a setAlgorithm' függvény segítségével.
3. A játékosok neve alapértelmezetten „Black” és „White”, ezt is bármikor lehet módosítani a játék közben. Kiindulásként a fehér játékos az ember, a fekete a robotkar, de a program viszonylag könnyedén átalakítható ha ezt meg szeretnék cserélni.
4. A hagyományos sakkjátszmák során a játékosok egy sakkórát (6.2 ábra) használnak arra, hogy maximalizálják a játék idejét. Mindkét játékosnak előre meghatározott és beállított ideje van a gondolkozásra. Ha az adott játékos lépett, lenyom az órán egy kapcsolót, melynek hatására a másik játékos ideje telik addig, amíg ő is vissza nem kapcsolja az órát. A program két virtuális órát használ erre a célra, melyek indítása és megállítása automatikusan történik minden lépésnél.

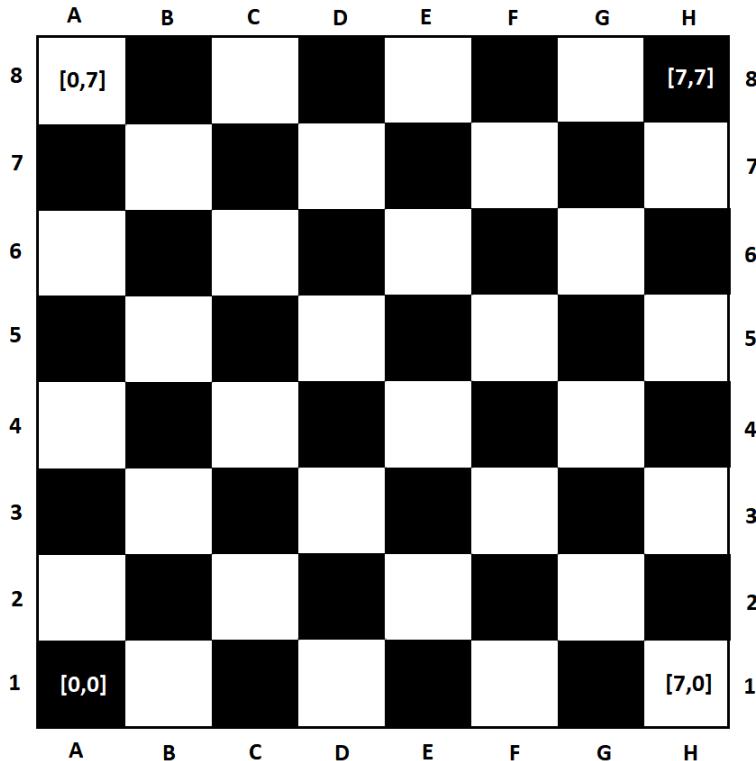


6.2. ábra: Hagyományos játékokhoz használt sakkóra - a piros zászló leesése jelzi, ha a játékos ideje elfogyott¹

5. A tábla és az alap felállás inicializálása a Board osztály példányosításával történik. Ez az osztály tartja számon, hogy melyik mezőn milyen bábu található, emellett praktikus okokból a két király jelenlegi helyzetét külön változókban tárolja.

¹A kép forrása: <https://www.polishchess.com/garde-analog-chess-clock-classic-p-161.html>

A bábuk pozíciója kapcsán külön érdemes kiemelni, hogy a pozíciót a bábuk koordinátája határozza meg. A lehetséges koordinátákat egy 8x8-as tömb jelképezi. A tömb [0,0] indexű eleme a tábla A1 mezője, a [7,7] pedig a H8 (6.3 ábra).



6.3. ábra: A sakktábla mezőihez koordináták rendelése

Létrehozni lépést a Move osztály példányosításával lehet. A konstruktur egy kezdő és egy végponti x és y koordinátát vár, amelyek a bábu kiindulási- és végkoordinátái. Azt, hogy egy lépés szabályos volt-e úgy lehet ellenőrizni, hogy meghívjuk a sakkjáték példányunk checkIfLegalMove' függvényét paraméterként átadva a lépést. Fontos, hogy a jelenlegi sakkjáték példányunkra hívjuk meg ezt a metódust, így biztos a pillanatnyi állás alapján dönt.

A lépések végrehajtása a szintén a ChessGame példányon belül implementált 'movePiece' függvénnyel lehet. Ez első körben értelemszerűen megváltoztatja a bábuk helyzetét. Második lépésként az adott lépést hozzáadja a lépésekkel vezetett listához (a ChessTableModel osztály segítségével). Ezek után a program átállítja, hogy ki a soron következő játékos, elindítja az óráját és ellenőrzi, hogy véget ért-e a játék, azaz mattot adott-e valamelyik fél.

6.3. A chess.properties csomag - kiegészítő funkciók

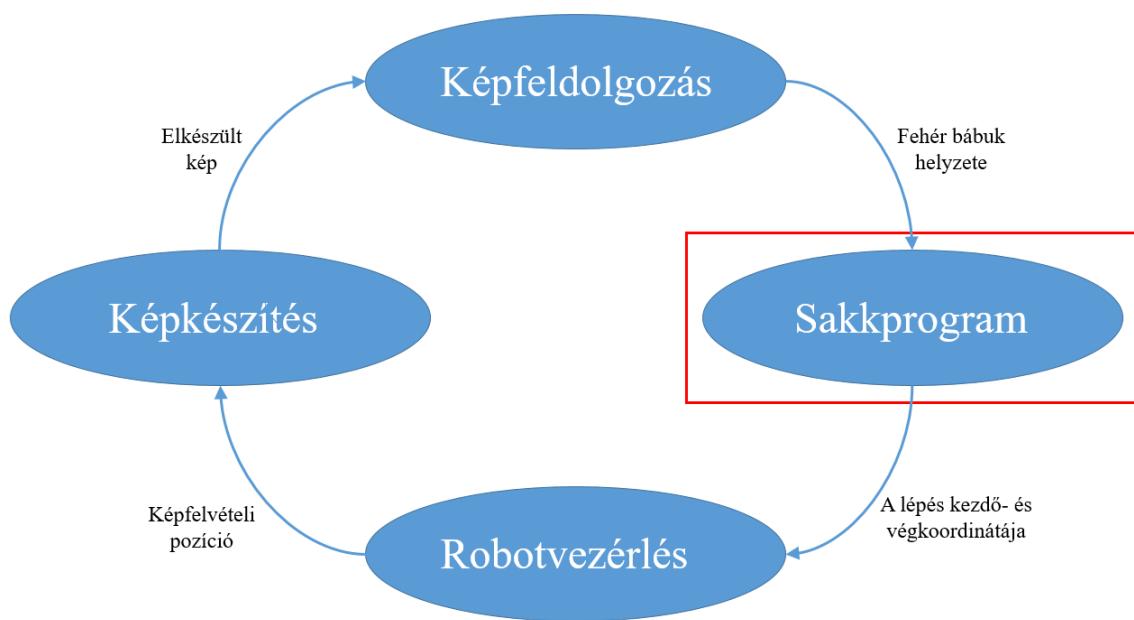
Ezen a csomagon tartalmaz számos a hagyományos sakkjáték végigjátszásához elengedhetetlen és néhány extra funkciót (a nélkülözhetetlen elemek ki lettek emelve a felsorolásban):

- **BoardParameters:** továbbfejlesztés részeként a smartPAD-en meg lehetne jeleníteni a jelenlegi állást. Ehhez hasznos függvényeket és beállításokat tartalmaz ez az osztály.
- **ChessColors:** az előző pontban említett megjelenítéshez különböző színdefiníciókat bocsát rendelkezésre.
- **ChessPreferences:** segítségével el lehet menteni a jelenlegi játékhelyzetet. A program képes tetszőleges helyzetből kezdeni egy játékot, tehát ennek segítségével lehetséges egy játék későbbi folytatása.
- **GameParameters:** itt lehet beállítani az adott játékhoz használt sakkalgoritmus paramétereit (az algoritmus kiválasztása, kereső fa szintjeinek száma). Ezen felül itt lehet a játéknak címet adni, ez tárolja a játékosok nevét és azt, hogy melyik játékos helyett játszik a gép.
- **State:** meghatározza a játék jelenlegi állását (Vége van-e? Meg lett-e állítva? Inicializálva lett-e már? stb.).

6.4. A sakkprogram kiegészítése a projekthez

Ahogy a 6.4 ábrán is látszik, a projekt megvalósításához a sakkprogramot össze kellett kötni a képfeldolgozást végző résszel és a robotvezérlővel. A képfeldolgozó rész meghatározza, hogy melyik mezőkön helyezkednek el a fehér bábuk, ez a sakkprogram bemenete. A kimenete pedig a szükséges lépés kezdő- és végpontja (pl.: [0,0] mezőről a [0,7] mezőre). Ezek fizikai koordinátkra transzformálása már a robotvezérlőn futó főprogramban történik.

A képfeldolgozás kimenete egy 8x8-as tömb (mátrix), amely indexelése meggyezik a sakkprogramnál bemutatott mezőindexeléssel (6.3 ábra). Ez a tömb az ember lépése utáni állapotban mutatja a bábuk elhelyezkedését. Ahhoz, hogy a megtett lépést meg lehessen határozni, ezt az állapotot össze kell vetni a lépés előttivel. A programban vezetett állást bármikor le lehet kérdezni (sakkjáték példány -> board -> b). A kapott elem egy 8x8-as tömb, amely egyes elemei vagy null értékűek vagy az ott található bábut írják le. minden bábu esetén le lehet kérdezni, hogy fekete-e vagy fehér (white: bábu boolean tulajdonsága - igaz ha fehér, hamis ha fekete). Ezek



6.4. ábra: A sakkprogram helye a folyamatban

alapján létre lehet hozni egy olyan logikai értékeket tartalmazó tömböt, amely formalag megegyezik a képfeldogozó rész által szolgáltatott tömbbel.

A lépés előtti és utáni tömböket a 'FindMove' függvény értékeli ki. Elemenként hasonlítja össze a tömböket a program. Ha két elem megegyezik, az azt jelenti, hogy az ott lévő bábu nem mozdult el. Ha különbözik a két érték, akkor két eshetőség állhat fenn:

- a bábu ellépett onnan (lépés után 'false' a mező értéke) - ekkor ez a mezőkoordináta lesz a lépés kiindulópontja, vagy
- a bábu oda lépett (lépés után true' a mező értéke) - ez a mező a lépés végpontja.

Két fehér bábu egy kör alatt csak sánszoláskor mozdulhat el. Ezt az eshetőséget a lépés meghatározásakor külön meg kell vizsgálni. Mivel a fekete bábul pozícióját a sakkprogram követi, így nem igényel külön eljárás kialakítását az, ha a fehér játékos leüti a fekete egy bábuját. Ilyen esetben a leütött bábu pozíciójához a lépés előtt 'false' érték tartozott, ezt a sakkprogram adja meg. A képfeldolgozó rész csak a fehér bábulat keresi (azokon van csak zöld jelölő), így az ütés után a mező értéke true' lesz.

A lépés validálására érdemes meghívni a sakkjáték checkIfLegalMove' függvényét mielőtt a lépést a programban végrehajttnánk, mivel szabálytalan lépést is be lehetne vinni.

7. EREDMÉNYEK ÉRTÉKELÉSE

HIVATKOZÁSOK

- [1] Andreja Rojko. Industry 4.0 concept: Background and overview. 2017.
- [2] Thomas Bauernhansl, Jörg Krüger, Gunther Reinhart, and Günther Schuh. Wgp-standpunkt industrie 4.0. Technical report, Wissenschaftliche Gesellschaft für Produktionstechnik, 2016.
- [3] T. Vilarinho, B. A. Farshchian, J. Floch, and B. M. Mathisen. *A Communication Framework for the Internet of People and Things Based on the Concept of Activity Feeds in Social Computing*. July 2013.
- [4] F. J. N. d. Santos and S. G. Villalonga. Exploiting local clouds in the internet of everything environment. In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, March 2015.
- [5] C. Wesch-Potente G. Schuh, T. Potente and Annika Hauptvogel. Sustainable increase of overhead productivity due to cyber-physicalsystems. In *Proceedings of the 11th Global Conference on Sustainable Manufacturing*, March 2013.
- [6] Henning Kagermann. Change through digitization : value creation in the age of industry 4.0. 2015.
- [7] J. Miranda, N. Mäkitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, and J. M. Murillo. From the internet of things to the internet of people. *IEEE Internet Computing*, 19(2):40–47, Mar 2015.
- [8] M. Awais and D. Henrich. Human-robot interaction in an unknown human intention scenario. In *2013 11th International Conference on Frontiers of Information Technology*, pages 89–94, Dec 2013.
- [9] S. Jaschke. Mobile learning applications for technical vocational and engineering education: The use of competence snippets in laboratory courses and industry 4.0. In *2014 International Conference on Interactive Collaborative Learning (ICL)*, pages 605–608, Dec 2014.
- [10] George Michalos, S Makris, Panagiota Tsarouchi, Toni Guasch, Dimitris Kontovrakis, and George Chryssolouris. Design considerations for safe human-robot collaborative workplaces. *Procedia CIRP*, 37:248–253, 12 2015.
- [11] Olessia Ogorodnikova. Human weaknesses and strengths in collaboration with robots. *Periodica Polytechnica Mechanical Engineering*, 52(1):25–33.

- [12] KUKA Deutschland GmbH. *System Software - KUKA Sunrise.OS 1.16 - Operating Instructions for End Users*, 2018.
- [13] SCHUNK GmbH & Co. KG. *Electrical parallel gripper MEG 50 EC Assembly and Operational Manual*, 3 2016. Rev. 02.01 EN.
- [14] Arwid (Arvydas) Bancewicz. Arwid-chess, 2005.
<https://github.com/Arwid/chess>.

HIVATKOZÁSOK

HRC Human Robot Collaboration

IoT Internet of Things

IoE Internet of Everything

TCP Tool Center Point

HRI Human-Robot Interaction