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 Sakkozó iiwa 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 Gépészmérnöki 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 átvettem, egyértelműen, a hatályos előírásoknak megfelelően, a forrás megadásával megjelöltem.

Budapest, 2018 .....

szigorló hallgató

TARTALOMJEGYZÉK

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. ALKALMAZÁSHOZ SZÜKSÉGES MŰSZAKI FELTÉTELEK ELEMZÉSE

1.1. 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 megtörténtétét 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.

1.2. A megfogó vezérlése

1.2.1. 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 (1.1):

- **Megfogó (gripper):** párhuzamosan mozgó, két pofájú, elektromos megfogó; pontos típusa: SCHUNK MEG 50 EC[**?**].
- **Grippervezérlő:** a megfogóhoz tervezett vezérlő; pontos típusa: SCHUNK Controller MEG EC[**?**]
- **Analóg és digitális I/O modulok**[1]**:** a robotvezérlő ezen modulok segítségével tudja irányítani a grippervezérlőt. A felhasznált eszközök Beckhoff gyártmányúak. Pontos tipusaik:
  - EL1809: 16 csatornás, digitális bemeneti modul[2]
  - EL2809: 16 csatornás, digitális kimeneti modul[3]
  - EL3002: 2 csatornás, analóg bemeneti modul[4]
  - EL4032: 2 csatornás, analóg kimeneti modul[5]
- **EtherCAT**[6] **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 EK1100[7].

[1] I/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
[6] Általános ismertető az EtherCAT-ről: https://en.wikipedia.org/wiki/EtherCAT
[7] https://www.beckhoff.com/english.asp?ethercat/ek1100.htm

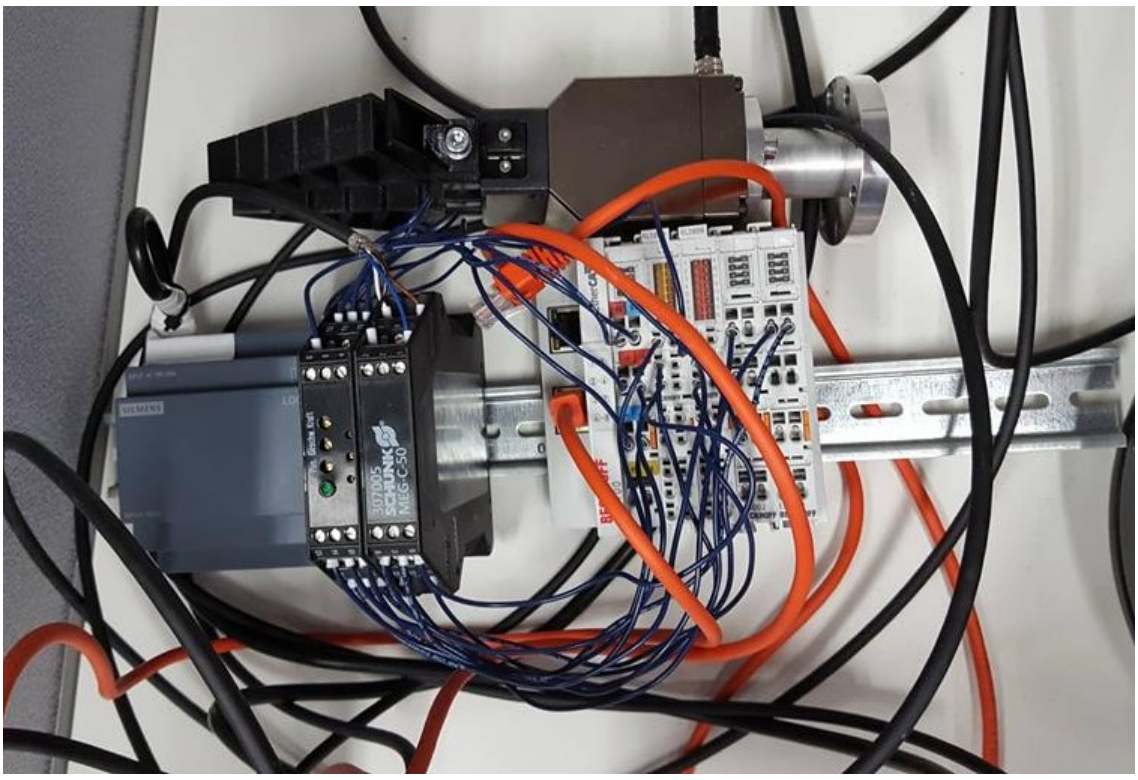
- **Tápegység:** a fentebb felsorolt eszközök mindegyike 24 V megtáplálást igényel, ezt a projekt során egy Siemens tápegység szolgáltatja.

1.2.2. 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:



1.1. 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 kiválasztani 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 letölthetőek[1]. 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 (1.2).

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 (1.3 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 elegendő[2]), pl.:

    - OpenGripper: bool, output, digital
    - CloseGripper: bool, output, digital

[1] Device description files: https://www.beckhoff.com/english.asp?download/elconfg.htm

[2] A 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 összerendelni (1.4). Megfelelő összekötés esetén a változók mellletti 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 Grouphoz tartozó osztály (pl.: GripperContolIOGroup.java). Ez tartalmazza a szükséges methódusokat a gripper vezérléséhez, ezeket lehet meghívni a programból.

1.3. Sakkbábúk kialakítása, felismerése a képeken

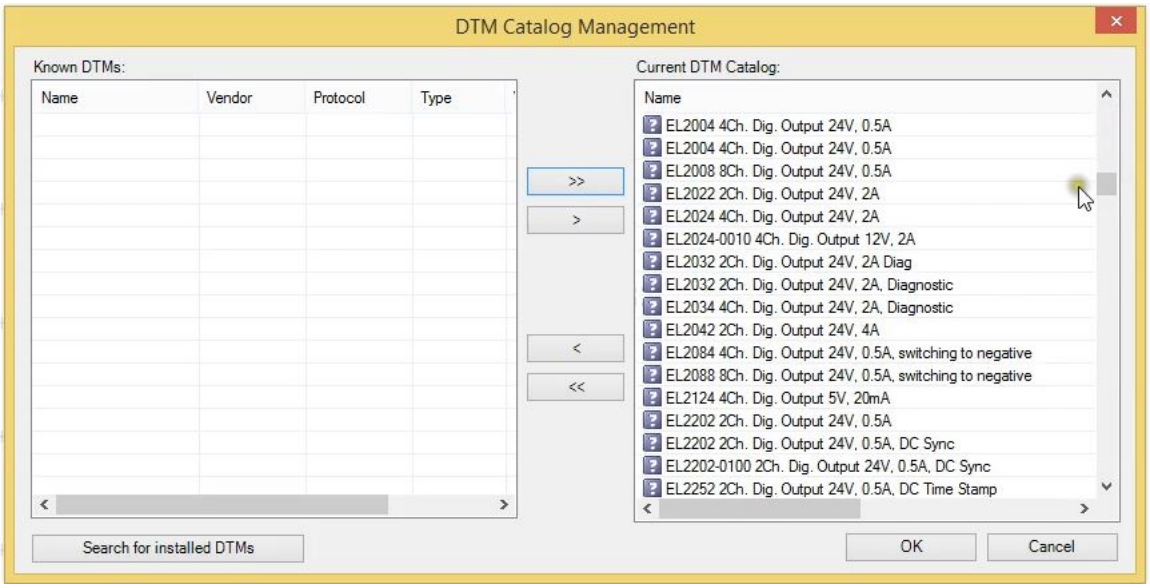
A sakkprogram működtetéséhez elegendő a bábúk új pozíciójának felismerése azután, miután az ember lépett. Elég csak az ember bábuit figyelni (a robot vezeti magának a bábuk pillanatnyi helyzetét). A sakkbábúk felismerére többfajta módszer is megfelelő lehet. A képfeldolgozást lehet saját vagy szabványos mintára alapozni (pl.: QR-kód alapú felismerés - 1.4 fejezet), illetve színes kamerakép esetén adott színek keresésére. A szakdolozat során a második módszer került beágyazásra, viszont a sakkbábuk formai kialakításakor a QR-kód alapú felismerhetőség is fontos szempont volt, így biztosítva az ilyen irányú továbbfejlesztés lehetőségét.

1.3.1. A sakkbábuk formatervezése

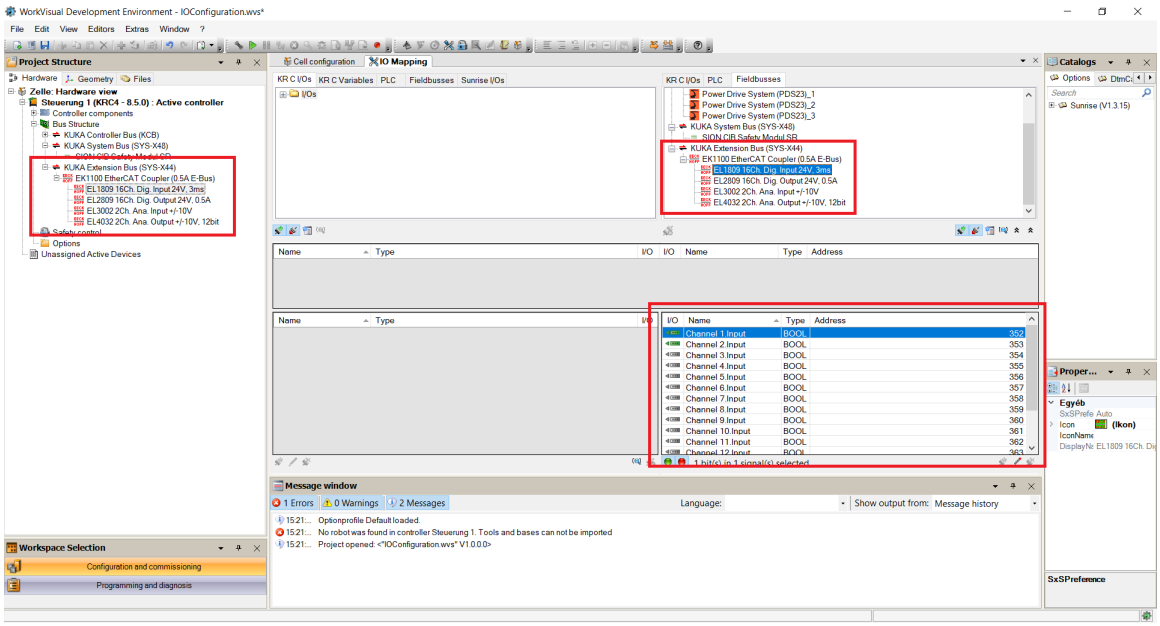
Ahhoz hogy a megfogó fel tudja emelni a bábukat könnyebség, ha a bábuk egyforma magasak, vagy ha a megfogó mindegyik bábut egy kitültetett, egy magasságban lévő részénél fogja meg (akár a talprészénél vagy alá is nyúlhatna). A talprésznél megfogás jó megoldás lehet, ha maguk a bábuk színesek, és így 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 (1.5 ábra):

- Mindegyik bábu tetején található egy négyzet alapú, szélességéhez viszonyítva alacsony hasáb, amire az adott színt/színeket/QR-kódot rá lehet ragasztani.

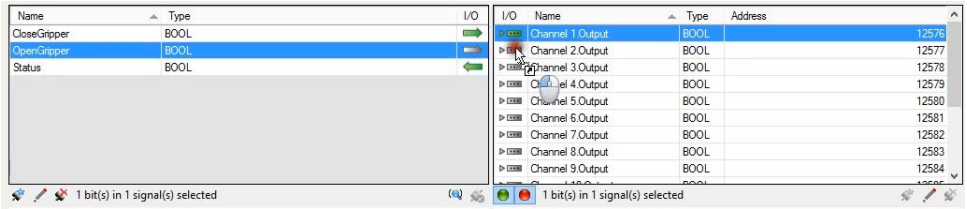
- Ennél a hasábnál fogva emeli meg a megfogó a bűbut, így kevésbé kell lenyúlnia a robotkarnak, kisebb az esély arra, hogy egy másik bábut felborít.



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

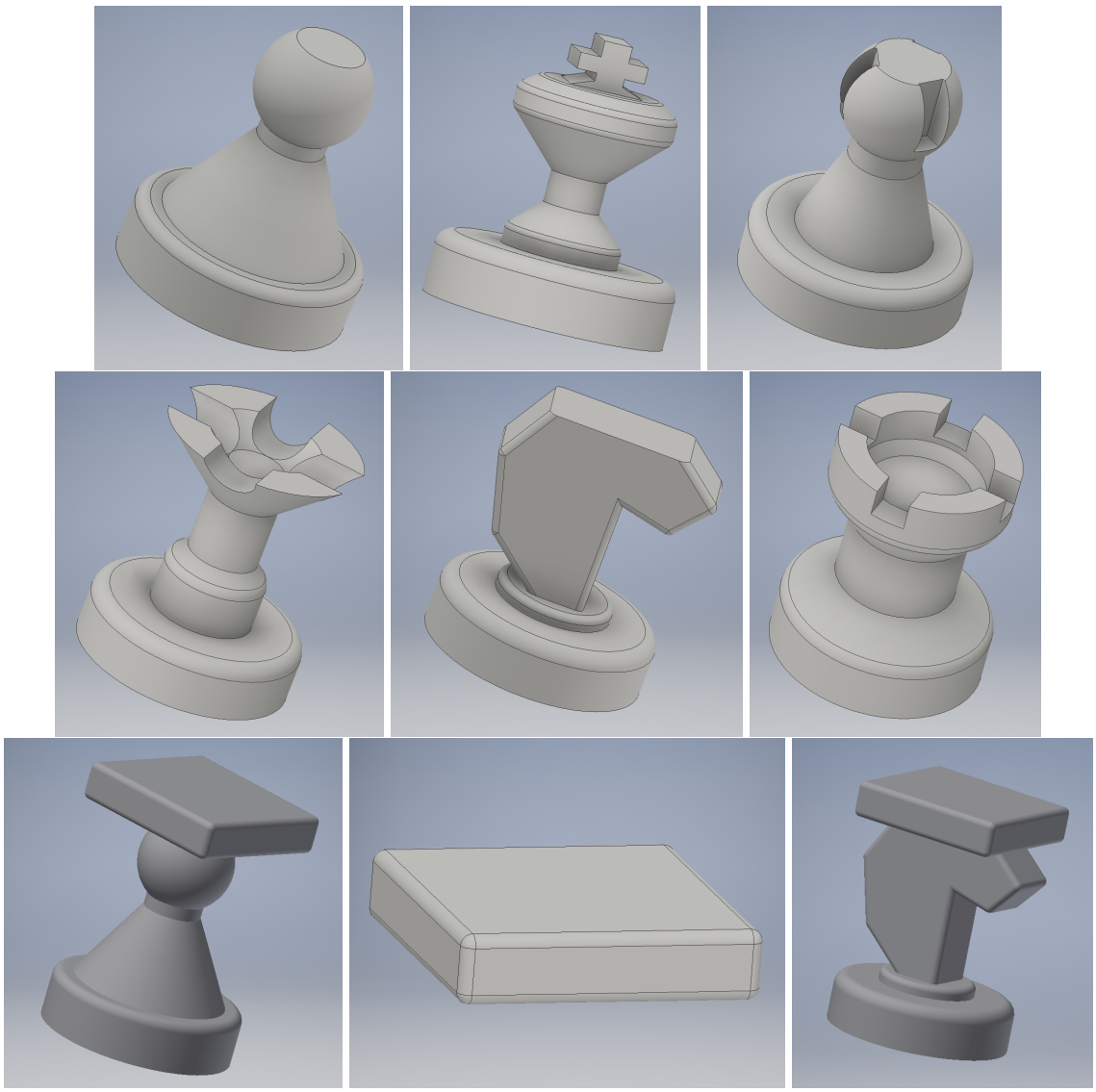


1.3. ábra: Az EtherCAT konfiguráció



1.4. 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 talprészük vastagított, tömör, hogy kevésbé legyenek borulékonyak.
- Annak érdekében, hogy a bábukat kevesebb támaszanyag felhasználásával lehessen nyomtatni a bábuk tetején elhelyezkedő hasábok külön lettek kinyomtatva és csak utólag lettek a bábukra felragasztva.



1.5. ábra: A sakkbábúk 3D-s terve

1.6. ábra: A kinyomtatott sakkbábuk

1.3.2. Bábuk pozíciójának azonosítása – TODO: referencia másik fejezetre

A bábuk pozíciójának azonosításához szükség van referenciafelvételre, hogy a kép egyes részeihez a sakktábla mezőit tudjuk rendelni. A

1.4. QR-kód alapú képfeldolgozás - továbbfejlesztés

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 helyzett QR-kód, amelyben kódolva van a bábu típusa. A QR-kódok kezelésére jó választás a nyílt forráskódú ZXing („Zebra Crossing") program[1]. Ez a Java könyvtár (library) alkamas különböző formátumú, egy- és kétdimenziós vonalkódokkal kapcsolatos képfeldolgozásra, amelynek csak egyik eleme a QR-kód olvasás és generálás.

1.4.1. A forráskód build-je

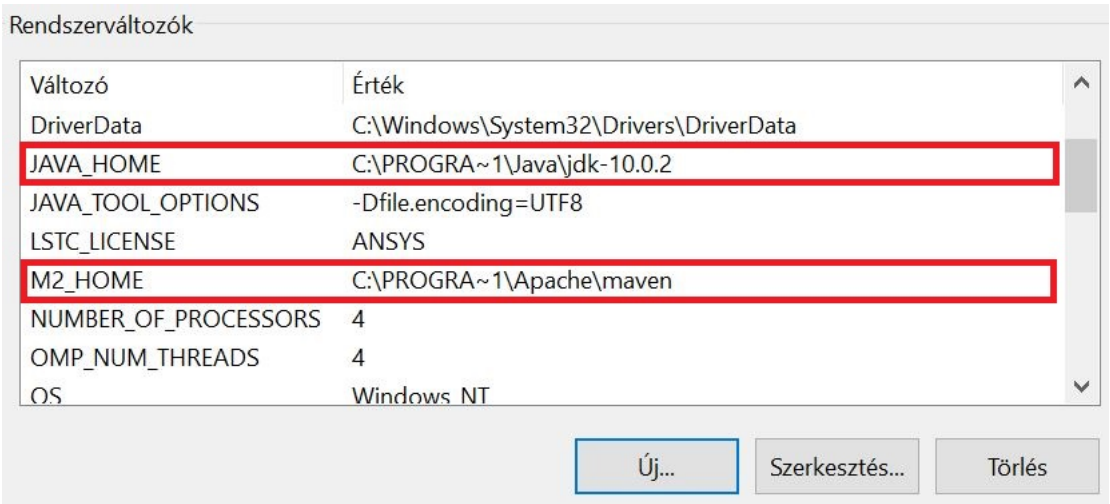
A könyvtár egyszerű használatához célszerű .jar fájlokat generálni a forráskódból. Az első lépés a Github-on elérhető forráskód letöltése vagy klónozása a saját számítógépre. A programkód számos mappába és almappába van rendszerezve az egyes moduloknak megfelelően (pl.: core/ és javase/). **Fontos:** a mapparendszert olyan helyre tegyük a számítógépen, melynek elérési útjában nem található szóköz karak-

[1]További információk és forráskód: https://github.com/zxing/zxing

ter (ékezetes karakter is probléma lehet)! Minden Java alapú modul esetén található egy pom.xml fájl, amit Apache Maven[1] 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 változók...->Rendszerváltozók címszó alatt tehetjük meg. Szükségünk van egy 'JAVA_HOME' és egy 'M2_HOME' változóra (1.7. ábra).

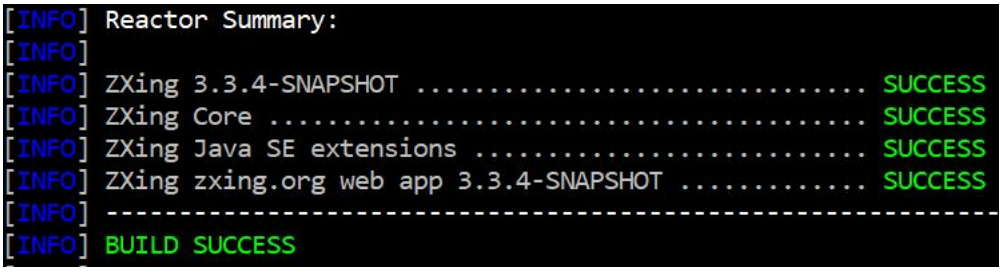
Parancssorban navigáljunk a ZXing projekt gyökeréhez és futtassuk a 'mvn install' parancsot a fordításhoz, a tesztekhez és az összes modul felépítéséhez. A „-DskipTests" paraméter hozzáadásával a unit teszteket kihagyhatjuk. Szükség lehet a -Drat.ignoreErrors=true' paraméterre a licensz tesztekkel kapcsolatos problémák ignorálásához. A build folyamat akkor mondható sikeresnek, ha mindegyik modul felépítése sikeres ('ANDROID_HOME' környezeti változó beállítása nélkül az Androidhoz kapcsolódó modulokat nem build-eli) (1.8. ábra).



1.7. ábra: Szükséges környezeti változók beállítása

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.

[1] Az Apache Maven program ingyenesen letölthető: https://maven.apache.org/



1.8. ábra: Sikeres build végeredménye

1.4.2. ZXing könyvtár beimportálása és használata

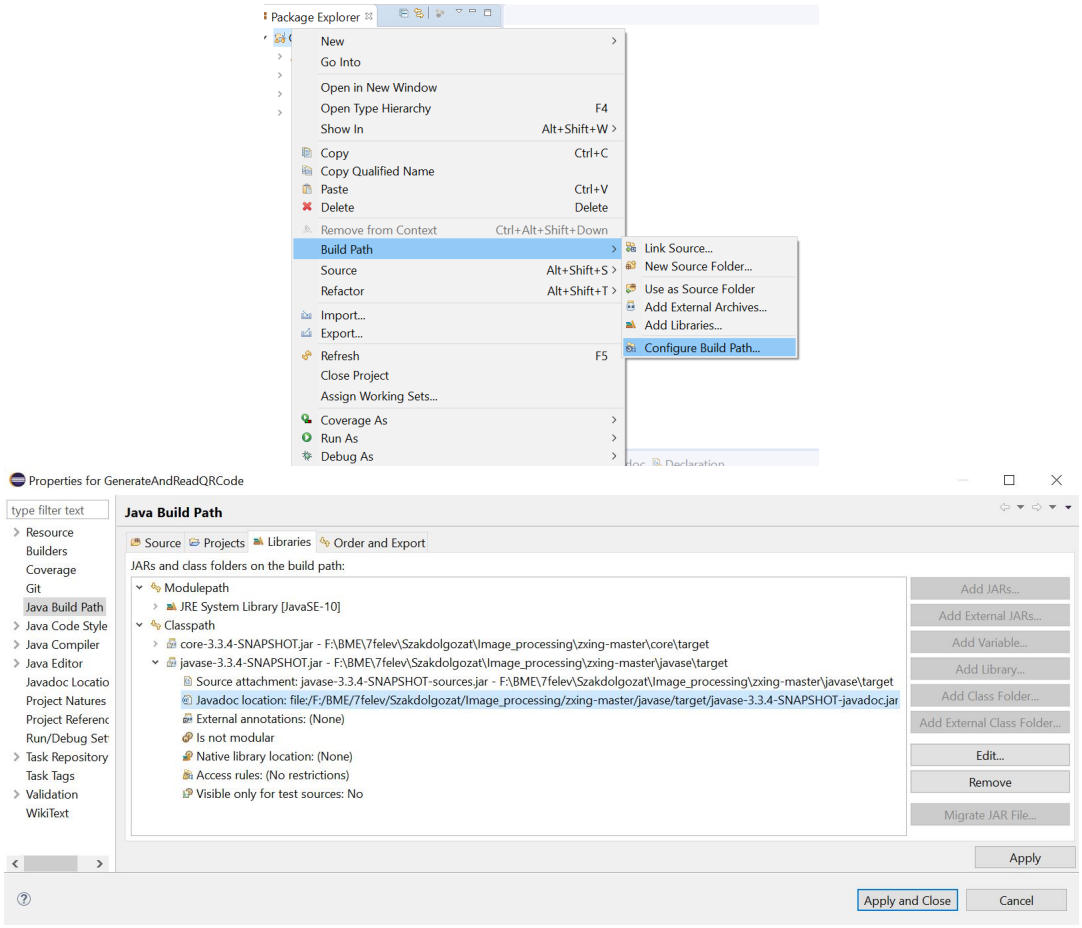
Az iiwa robotkart programozni Sunrise Workbench használatával a legegyszerűbb, ami egy JAVA Eclipse platformú szoftver. Emiatt praktikus okokból a szakdolgozat képfeldolgozási és sakkalgoritmus beágyazási része túlnyomó részt Eclipse-ben történt (verzió: 4.9.0)[1].

A könyvtár beimportálásához létrehoztam egy java projektet. Erre jobb egérgombbal kattintva elnavigáltam a 'Configure Build Path...' menüponthoz (1.9. ábra). A 'Classpath'-t kiválasztva jobb oldalt aktivizálódik az 'Add External Jars...' gomb. Kiválasztottam a *core/* és a *javase/* modulok .jar fájljait. Ezeken belül lehetőség van forráskód (Source attachment) és dokumentáció (Javadoc location) csatolására (1.9. ábra). 'Apply and Close' után megjelennek a csomagok a hivatkozott könyvtárak (Referenced libraries) pont alatt.

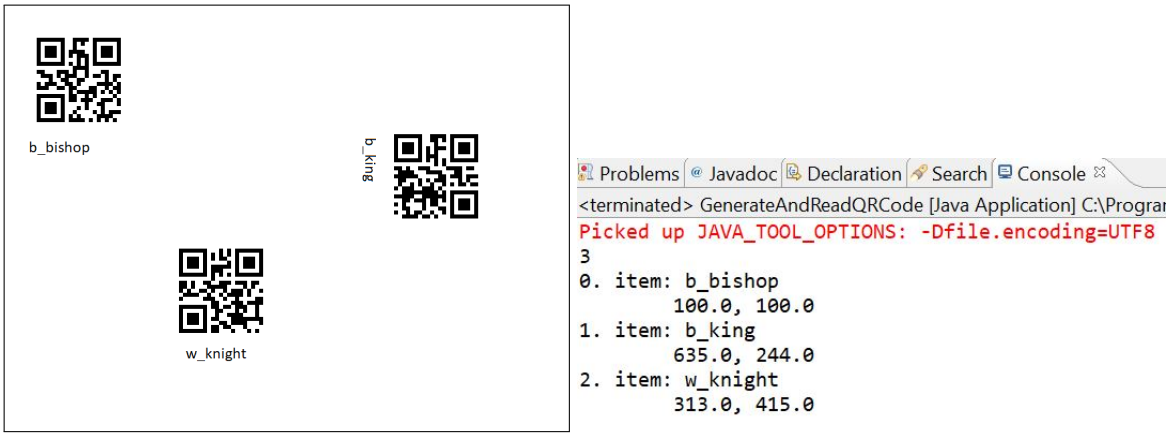
1.4.3. QR kód generálás és olvasás

A ZXing könyvtárt felhasználó, ehhez a projekthez szükséges elemek az 'imgprocess' csomagban találhatóak a 'QRCodeMethods.java' fájlban. A függvények leírása az 1.1 táblázatban találhatóak[2].

[1] Link: https://www.eclipse.org/
[2] Forráskód: https://github.com/rolandvarga601/Szakdolgozat



1.9. ábra: ZXing hozzáadása a hivatkozott könyvtárakhoz



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

2. ROBOTKAR KALIBRÁCIÓ, REFERENCIA FELVÉTEL KIDOLGOZÁSA

Függvény neve	Paraméterek	Leírás
createQRCode	• String qrCodeData - a kódolt szöveg • String filePath - a képek mentési helye • String charset - karakterkódolás • int qrCodeheight - a kép magassága • int 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 (1.10 ábra). Visszatérési érték: Result[], amely tömb elemei tartalmazzák többek között az egyes QR-kódok helyét a képeken és a kódolt szöveget.

1.1. táblázat: QRCodeMethods.java függvényei

3. ROBOTKAR MOZGÁSÁNAK DEFINIÁLÁSA ÉS PROGRAMOZÁSA

4. SAKKALGORITMUS BEÁGYAZÁSA

5. EREDMÉNYEK ÉRTÉKELÉSE

HRC . . . . . Human Robot Collaboration

IoT . . . . . Internet of Things

IoE . . . . . Internet of Everything

TCP . . . . . Tool Center Point

HRI . . . . . Human-Robot Interaction