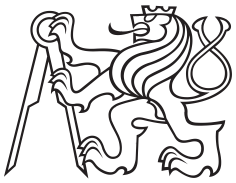
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Model pro detekci přítomných dílů v zásobníku robotické buňky

Radim Průdek

Školitel: Ing. Martin Hlinovský, Ph.D.
Obor: Kybernetika a robotika
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Průdek** Jméno: **Radim** Osobní číslo: **507380**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Model pro detekci přítomných dílů v zásobníku robotické buňky

Název bakalářské práce anglicky:

A model for detecting the parts present in the robotic cell stack

Pokyny pro vypracování:

1. Napište PLC program pro automatickou stavbu stavebnic.
2. Napište potřebne robotické programy.
3. Za použití kamery umístěné v robotické buňce, vytvořte model pro detekci přítomných dílů v zásobníku. Z detekovaných dílů v zásobníku zvolte nejvhodnější stavbu a tu navrhnete operátorovi pomoci HMI.
4 Integrujte výsledek do plánovače, který zvolí vhodné programy pro stavbu.

Seznam doporučené literatury:

[1] TIA Portal - SCL programování PLC, part 1 - 9: https://www.blaja.cz/software/tia-portal-scl-programovani-plc-part-1.html
[2] Process Simulate Tutorial: https://www.youtube.com/watch?v=2mHXlLdmkKk
[3] Process Simulate 1 - 18: https://www.youtube.com/watch?v=SaD0efG8AF4&list=PLnjh283r6h7WfukhtgsRwQjHadVawxk_m&index=1
[4] TIA Portal V18 full tutorial step by step in 11 hours: https://www.youtube.com/watch?v=Vvfk-H4AxFs

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Hlinovský, Ph.D. katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **19.01.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Martin Hlinovský, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Michael Šebek, DrSc. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

Rád bych poděkoval Ing. Martinu Hlinovskému Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 25. května 2024

. . . . . .

Abstrakt

Cílem této práce je realizace staveb z dřevěných stavebních dílů umístěných v zásobníku a jejich následné rozebrání za pomoci průmyslového robotického manipulátoru KUKA.

Pro identifikaci přítomnosti stavebních dílů v zásobníku a překážek v pracovním prostoru je použita 2D RGB kamera. Obraz z této kamery je následně zpracován v Python programu běžícím na průmyslovém PC. Úkolem tohoto programu je provedení segmentace obrazu a zpracování dat. Pro barevnou segmentaci je použita metoda prahování v HSV spektru. Výstupem programu je pak vyhodnocení přítomnosti jednotlivých dílů v zásobníku, určení druhů staveb, které lze z těchto dílů postavit, a identifikační čísla dílů pro jednotlivé uskutečnitelné stavby.

Centrálním řídicím prvkem této úlohy je PLC automat SIMATIC S7-1500, který je propojen s průmyslovým PC a robotem přes komunikační rozhraní PROFINET.

Všechny informace o běhu manipulace a vyhodnocených datech z kamery jsou zobrazeny na uživatelských obrazovkách. Na nich si uživatel může vybrat některou z realizovatelných staveb, spustit tuto stavbu, ovlivňovat průběh stavby a po jejím dokončení zvolit její rozebrání.

Tato práce může sloužit pro demonstrativní účely spolupráce manipulační robotické buňky se strojovým viděním.

Klíčová slova: PLC, robot KUKA, průmyslové PC, 2D RGB kamera, barevná segmentace, PROFINET, OPC UA

Školitel: Ing. Martin Hlinovský, Ph.D.

Abstract

The aim of this work is the realization of buildings from wooden building parts placed in a stack and their subsequent disassembly with the help of an industrial robotic manipulator KUKA.

A 2D RGB camera is used to identify the presence of building parts in the stack and obstacles in the workspace. The image from this camera is then processed in a Python program running on an industrial PC. The task of this program is to perform image segmentation and data processing. For color segmentation, HSV spectrum thresholding method is used. The output of the program is then the evaluation of the presence of the individual parts in the stack, the determination of the types of buildings that can be built from these parts and the identification numbers of the parts for each realizable building.

The central control unit of this task is the SIMATIC S7-1500 PLC automaton, which is connected to the industrial PC and robot via the PROFINET communication interface.

All information about the manipulation run and the evaluated camera data is displayed on the user screens. On these screens, the user can select one of the realizable builds, start to build it, influence the progress of the construction and choose to disassemble it when it is finished.

This work can be used to demonstrate the collaboration of a robotic manipulation cell with machine vision.

Keywords: PLC, robot KUKA, industrial PC, 2D RGB camera, color segmentation, PROFINET, OPC UA

Title translation: A model for detecting the parts present in the robotic cell stack

Obsah

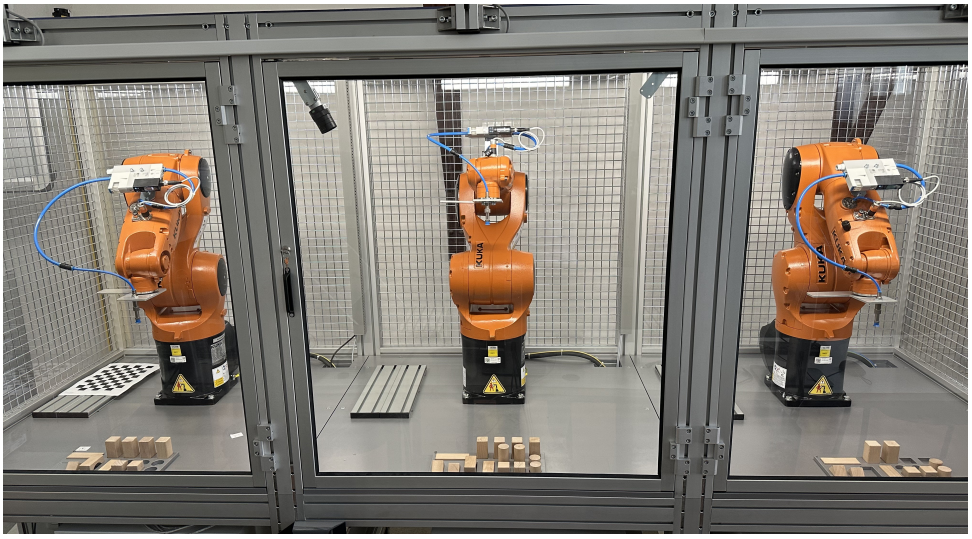
Obrázky

Tabulky

Kapitola 1
Úvod

Cílem této bakalářské práce je vytvoření pracovního prostředí pro konstrukci stavby z dřevěných kostek, které jsou systematicky uskladněny v zásobníku. Pro samotný stavební proces je využíván průmyslový robot KUKA [1], jehož pracovní prostředí je vybaveno statickou kamerou [2] pevně připevněnou k rámu robotické buňky. Kamera sleduje prostor zásobníku stavebních dílů včetně prostoru staveb. Pro analýzu obrazu z kamery a detekci stavebních dílů je implementován průmyslový počítač [3], který je propojen přes průmyslový síťový protokol PROFINET [4] s programovatelným logickým automatem (PLC) [5]. Celý proces je tak řízen a koordinován tímto PLC, jež zajišťuje přesné a spolehlivé řízení robota na základě vyhodnocených informací z kamery.

1.1 Popis pracovního prostoru



Obrázek 1.1: Pracovní prostor.

1.1.1 PLC

Pro tuto práci je použit PLC automat SIMATIC S7-1500, CPU 1516F-3 PN/DP [5] od společnosti Siemens, viz obrázek 1.2.

Obrázek 1.2: PLC automat SIMATIC S7-1500, CPU 1516F-3 PN/DP.

Jedná se o výkonný PLC automat umožňující rychlé zpracování řídicích algoritmů a operací v průmyslovém prostředí. Centrální jednotka podporuje komunikaci přes průmyslový komunikační protokol PROFIBUS (DP) a PROFINET (PN) [4]. Řídicí systém rovněž nabízí metody OPC UA [6] prostřednictvím jeho integrovaného serveru. Veškerá konfigurace a následně i PLC program jsou vytvořeny v prostředí TIA Portal.

Pro vytváření PLC programu je k dispozici několik možností [7].

1. Nejstarší a nejjednodušší způsob programování PLC je **Ladder Diagram (LD)** - "Žebříčkový Diagram". LD je grafický programovací jazyk založený na schématech obvodů reléové logiky.

2. Dalším oblíbeným grafickým programovacím jazykem PLC je **Function Block Diagram (FBD)** - "Funkční Blokový Diagram". Skládá se z různých druhů bloků, které mají vstupy a výstupy. Bloky se navzájem spojují.

3. Další grafický programovací jazyk je **Sequential Function Charts (SFC)** - "Sekvenční Funkční Diagram". Jazyk se skládá z kroků a přechodů. Kroky jsou akce, které mají být provedeny a přechody jsou logické podmínky, které je třeba splnit před přechodem k dalšímu kroku.

4. **Instruction List (IL)** - "Seznam Instrukcí" je nízkoúrovňový textový programovací jazyk. Jazyk IL se skládá z jednoduché řady instrukcí a je podobný strojovému jazyku assembler. Programování v jazyku IL není v dnešní době považováno za produktivní.

5. **Structured Text (ST)** - "Strukturovaný Text" je považován za jazyk vyšší třídy. Je to textový jazyk se syntaxí srovnatelnou s programovacími jazyky C nebo C++.

Všechny PLC programy v této úloze byly napsány za pomocí programovacího jazyka **ST - Structured Text**.

1.1.2 Robot KUKA

Průmyslový robot použitý v této práci je KUKA KR 6 R700 sixx [1], viz obrázek 1.3.

Obrázek 1.3: KUKA KR 6 R700 sixx.

Tento průmyslový robotický manipulátor s šesti stupni volnosti je navržen pro manipulaci s materiálem o nízké hmotnosti. Optimální dynamické schopnosti robota jsou zajištěny při nominální hmotnosti zátěže 3 kg. Hlavní výhodou tohoto typu robota je jeho přesnost opakování polohy, kterou výrobce uvádí jako $\pm 0,03$ mm.

Robot KUKA je ovládán pomocí KUKA smartPADu, viz obrázek 1.4.

Obrázek 1.4: KUKA smartPAD.

Robot je programován pomocí jazyka **Kuka Robot Language (KRL)**. Jedná se o jazyk vyšší třídy, který umožňuje ovládat pohyby robota, jeho logiku a komunikaci s externími zařízeními. KRL má podobnou syntaxi jako programovací jazyk Pascal. Veškerá konfigurace robotické buňky, propojení signálů z PLC a následně i všechny robotické programy byly vytvořeny v prostředí Workvisual [8].

1.1.3 Průmyslové PC

Obrázek 1.5: Průmyslové PC NET-III 2I640DW.

NET-III 2I640DW [3] je kompaktní počítač bez ventilátoru s montáží na DIN lištu. Je navržen pro aplikace s omezeným prostorem a pro práci v průmyslovém prostředí.

Specifikace

- Procesor: Intel Elkhart Lake ATOM x6413E Quad Core 3.0 GHz CPU
- Paměť: DDR4 SODIMM, 32GB
- SSD disk: 256 GB SATA
- USB porty: 2x USB 3.0
- Ethernet porty: 3 x Intel GbE
- Sériové porty: 2 x RS232 / 422 / 485
- Podporované operační systémy: Windows 10 IOT, Windows 11 IOT, Ubuntu 22.04 a výše, Fedora 36 a výše
- Napájení: 24V DC

1.2 Návrh řešení

Samotný problém lze rozdělit na tři části. První část zahrnuje implementaci programů pro ovládání kamery a následné zpracování obrazu pomocí průmyslového PC 1.1.3. Druhá část spočívá ve vytvoření konfigurace a programu pro PLC 1.1.1, který bude řídit celý proces. Třetí část se týká vytvoření programu pro robota 1.1.2. Následně je nutno zajistit propojení těchto zařízení a bezchybnou komunikaci mezi nimi. Propojení a komunikace jsou podrobně popsány v kapitole 2.

1.2.1 Programy v průmyslovém PC

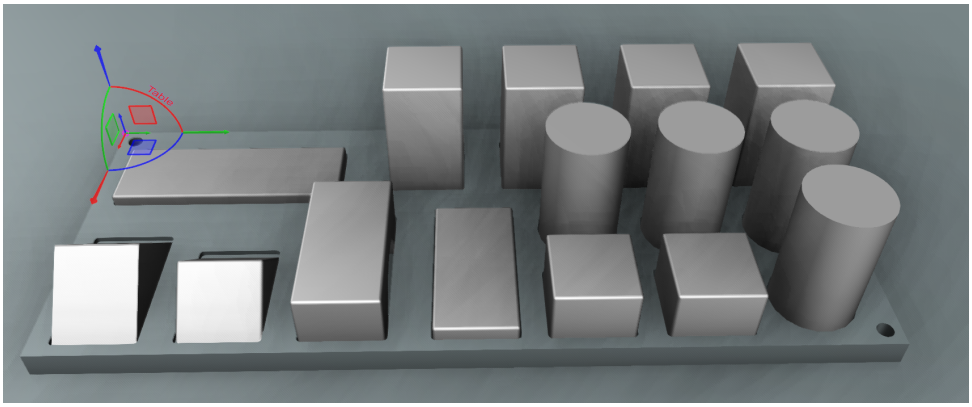
Hlavní úlohou průmyslového PC je ovládat kameru a následně zpracovávat data z ní. Ze zpracovaných dat identifikuje přítomné díly v zásobníku a detekuje překážky v pracovním prostoru. Vyhodnocovací program dále poskytuje dodatečné informace o tom, které stavby lze z přítomných dílů postavit, včetně identifikačních čísel dílů, ze kterých se stavba skládá. Implementace programů pro ovládání kamery a zpracování obrazu a dalších dat byla provedena v jazyce Python. Pro ovládání kamery je použito Open-source rozhraní pro Python **pypylon** [9]. Ke zpracování obrazu byly využity knihovny **OpenCV** [10] a **NumPy** [11]. OpenCV je světově nejpoužívanější open-source softwarová knihovna pro počítačové vidění a strojové učení. NumPy je univerzální standard pro práci s numerickými daty v jazyce Python. Podrobný popis zpracování obrazu je uveden v kapitole 3.

1.2.2 Programy v PLC

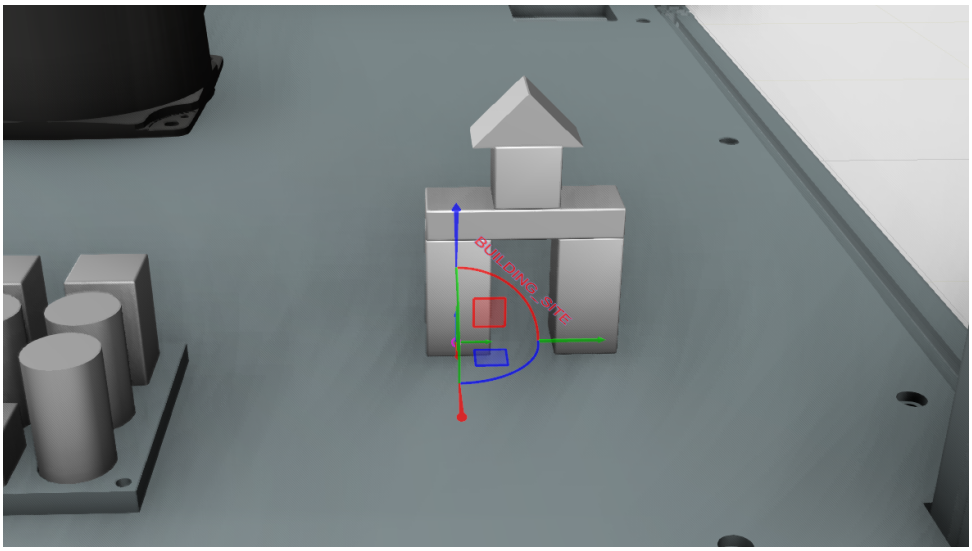
Hlavním úkolem PLC je řídit celý proces. PLC komunikuje s průmyslovým PC a přijímá data o přítomnosti dílů v zásobníku. Dále přijímá informace, které stavby lze postavit z přítomných dílů a také identifikační čísla dílů použitých pro každou možnou stavbu. V PLC jsou následně vypočítány polohy všech dílů v každé stavbě a jsou postupně posílány do robota jako souřadnice cílových poloh, kde má díl nabrat či kam odložit.

1.2.3 Program pro robota

Robot získává od PLC identifikační čísla dílů a data o jejich poloze. Získává též informaci, zda je spuštěn proces stavění nebo rozebírání stavby. Na základě těchto informací rozhoduje robot, v jakém pořadí bude vykonávat jednotlivé pohyby a v jaké bázi data o požadované poloze bude reprezentovat. Je tedy potřeba nastavit dvě nové báze. První určuje počátek v zásobníku dílů, viz obrázek 1.6. Druhá báze určuje počátek ve stavebním prostoru, viz obrázek 1.7.



Obrázek 1.6: Báze robota v zásobníku dílů.

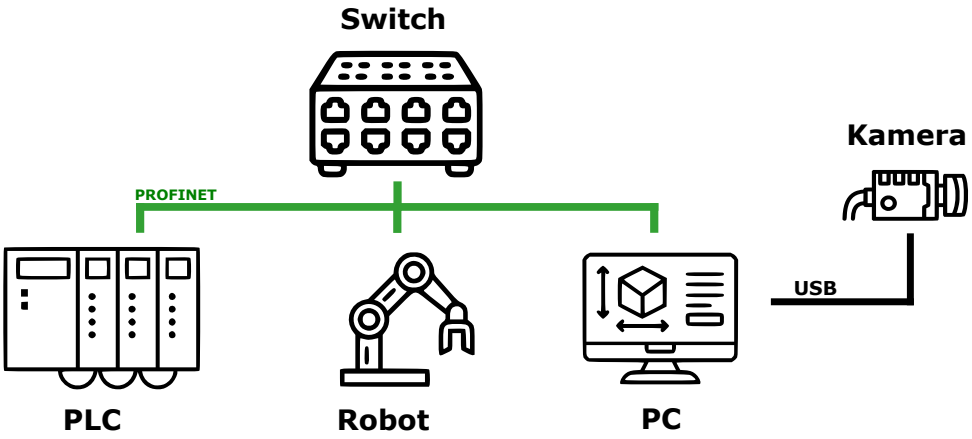


Obrázek 1.7: Báze robota ve stavebním prostoru.

Kapitola 2

Propojení a komunikace

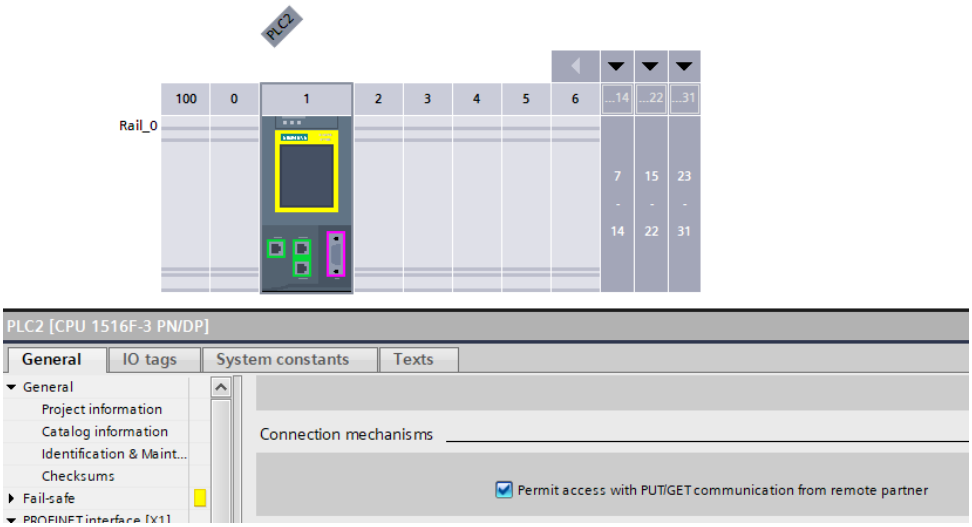
Vzájemná komunikace mezi zařízeními (PLC, robot a průmyslové PC) probíhá na bázi průmyslového síťového protokolu PROFINET [4]. Zařízení jsou navzájem propojena za pomocí LAN switche, viz obrázek 2.1. Kamera je k průmyslovému PC připojena přes USB rozhraní.



Obrázek 2.1: Schéma propojení.

2.1 Komunikace PLC s průmyslovým PC

PLC komunikuje s průmyslovým PC pomocí protokolu S7-TCP/IP známého jako ”RFC1006” nebo ”ISO on top of TCP” [12]. Použitá centrála S7-1500 má integrované Ethernet rozhraní. Pro zajištění této komunikace je nutno v Tia Portálu nastavit ve vlastnostech centrály povolení přístupu přes PUT/GET komunikace ke vzdáleným partnerům, viz obrázek 2.2.



Obrázek 2.2: Povolení komunikace pomocí protokolu S7-TCP/IP.

Komunikační část Python scriptu, běžící na průmyslovém PC, používá pro čtení a zápis dat z/do PLC Open-source knihovnu **Snap7** [13]. Tato knihovna je 32/64 bitová multiplatformní ethernetová komunikační sada pro nativní propojení s PLC automaty Siemens S7.

2.1.1 Podporované datové oblasti a datové typy

Pro komunikaci za pomoci knihovny **Snap7** [13] lze použít více možností datových oblastí. Pro protokol S7-TCP/IP jsou podporovány dané oblasti dat:

- I = Input: Fyzické vstupy. Bitové hodnoty, které lze číst ale nelze do nich zapisovat.
- Q = Output: Fyzické výstupy. Bitové hodnoty, které lze číst i zapisovat.
- M = Memory: Vnitřní paměť, kterou lze číst i zapisovat. Na rozdíl od datových bloků nebývá tato paměť perzistentní, tzn. po vypnutí PLC se paměť resetuje.
- DB = Data Block: Perzistentní paměť, kterou lze číst i zapisovat.
- T = Timer: Časovače (jen pro čtení).
- C = Counter: Čítače (jen pro čtení).

Následně je možno pro výměnu dat využít tyto podporované datové typy pro S7 komunikaci v knihovně **Snap7** [13].

- X: Bit, Boolean - Bit daný adresou bytu a číslem bitu.
- B: Unsigned Byte - 1-bytové celé neznaménkové číslo (od 0 do 255).

- W: Unsigned Word - 2-bytové celé neznaménkové číslo (od 0 do +65 535).
- DW: Unsigned Double Word - 4-bytové celé neznaménkové číslo (od 0 do +4 294 967 295).
- INT: Integer - 2-bytové celé číslo (od -32 768 do 32 767).
- UINT: Unsigned Integer - 2-bytové celé neznaménkové číslo (od 0 do +65 535)
- DINT: Double Integer - 4-bytové celé číslo (od -2 147 483 648 do +2 147 483 647).
- UDINT: Unsigned Double Integer - 4-bytové celé neznaménkové číslo (od 0 do +4 294 967 295).
- LINT: Long Integer - 8-bytové celé číslo.
- ULINT: Usigned Long Integer - 8-bytové celé neznaménkové číslo.
- SINT: Small Integer - 1-bytové celé číslo (od -128 do 127).
- USINT: Unsigned Small Integer - 1-bytové celé neznaménkové číslo (od 0 do 255).
- REAL: Real number, 4B IEEE - 4-bytové reálné číslo ve formátu IEEE-754 (32-bit).
- LREAL: Long Real number, 8B IEEE - 8-bytové reálné číslo ve formátu IEEE-754 (64-bit).
- CHAR: Signed Byte, Raw string - Posloupnost znaků se zadaným počtem znaků (1 znak = 1 byte). V PLC automatu je tento typ uložen bez informace o délce textu.
- STRING: S7 string - Text se zadaným počtem znaků (1 znak = 1 byte). V PLC automatu je tento typ uložen navíc s hlavičkou, ve které je informace o aktuální délce textu.
- DT: Date and Time, Datum a čas reprezentovaný jako ISO 8601 formátovaný datový řetězec.
- DTO: Date and Time, Datum a čas reprezentovaný jako Python objekt.
- Array: Není podporováno čtení/zápis pole přes jednu proměnnou (např. typu Array). Avšak proměnné typu pole se v PLC automatu chovají z hlediska průmyslového PC tak, jako by se jednalo o oblast jednotlivých proměnných. To znamená, že tyto proměnné lze číst/zapisovat jako jednotlivé proměnné, kde každá má svou adresu v databloku.

2.1.2 Komunikační datový blok

Pro komunikaci s průmyslovým PC je v PLC vyhrazen datový blok "DB_PC_com [DB100]", který obsahuje veškeré přenášené proměnné. Blok obsahuje boolovskou proměnnou, která udává povolení k zápisu dat z kamery do PLC a boolovskou proměnnou, která signalizuje, že v pracovním prostoru nejsou žádné překážky, viz obrázek 2.3.

1	Static			
2	Read_data	Bool	0.0	false
3	Clear_workspace_cam..	Bool	0.1	false

Obrázek 2.3: Proměnné povolení zápisu dat a detekce překážek v pracovním prostoru typu bool.

Dalším prvkem bloku je proměnná typu pole, která obsahuje boolovské proměnné určující přítomnost dílů v zásobníku, viz obrázek 2.4

3	Block_in_Base	Array[0..14] of Bool	2.0	
4	Block_in_Base[0]	Bool	2.0	false
5	Block_in_Base[1]	Bool	2.1	false
6	Block_in_Base[2]	Bool	2.2	false
7	Block_in_Base[3]	Bool	2.3	false
8	Block_in_Base[4]	Bool	2.4	false
9	Block_in_Base[5]	Bool	2.5	false
10	Block_in_Base[6]	Bool	2.6	false
11	Block_in_Base[7]	Bool	2.7	false
12	Block_in_Base[8]	Bool	3.0	false
13	Block_in_Base[9]	Bool	3.1	false
14	Block_in_Base[10]	Bool	3.2	false
15	Block_in_Base[11]	Bool	3.3	false
16	Block_in_Base[12]	Bool	3.4	false
17	Block_in_Base[13]	Bool	3.5	false
18	Block_in_Base[14]	Bool	3.6	false

Obrázek 2.4: Proměnná přítomnosti dílů v zásobníku typu pole boolů.

Následuje proměnná typu pole, která obsahuje boolovské proměnné, jež určují, zda daná stavba lze z přítomných dílů postavit, viz obrázek 2.5.

19	Building_Possibility	Array[0..15] of Bool	4.0	
20	Building_Possibility[0]	Bool	4.0	false
21	Building_Possibility[1]	Bool	4.1	false
22	Building_Possibility[2]	Bool	4.2	false
23	Building_Possibility[3]	Bool	4.3	false
24	Building_Possibility[4]	Bool	4.4	false
25	Building_Possibility[5]	Bool	4.5	false
26	Building_Possibility[6]	Bool	4.6	false
27	Building_Possibility[7]	Bool	4.7	false
28	Building_Possibility[8]	Bool	5.0	false
29	Building_Possibility[9]	Bool	5.1	false
30	Building_Possibility[10]	Bool	5.2	false
31	Building_Possibility[11]	Bool	5.3	false
32	Building_Possibility[12]	Bool	5.4	false
33	Building_Possibility[13]	Bool	5.5	false
34	Building_Possibility[14]	Bool	5.6	false
35	Building_Possibility[15]	Bool	5.7	false

Obrázek 2.5: Proměnná možnosti postavení stavby typu pole boolů.

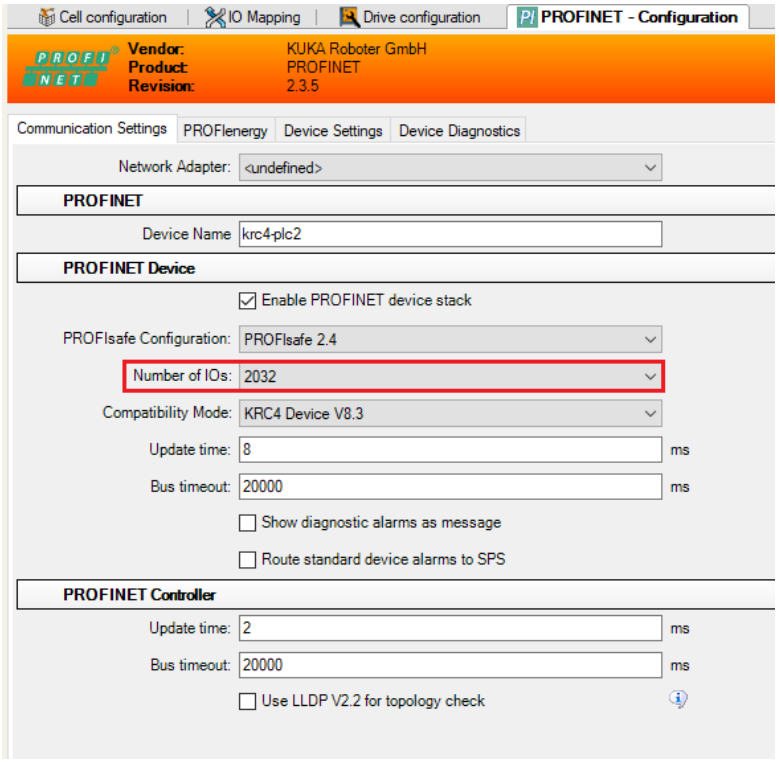
Poslední proměnná v datovém bloku je typu pole, obsahující struktury pro každou stavbu. Ve strukturách je uloženo pole bytů udávající posloupnost identifikačních čísel dílů stavby, viz obrázek 2.6.

36	Building	Array[0..15] of Struct	6.0		☐	☑	☑	☑	☐
37	Building[0]	Struct	6.0		☐	☑	☑	☑	☐
38	Block_id	Array[0..14] of Byte	6.0		☐	☑	☑	☑	☐
39	Block_id[0]	Byte	6.0	16#0	☐	☑	☑	☑	☐
40	Block_id[1]	Byte	7.0	16#0	☐	☑	☑	☑	☐
41	Block_id[2]	Byte	8.0	16#0	☐	☑	☑	☑	☐
42	Block_id[3]	Byte	9.0	16#0	☐	☑	☑	☑	☐
43	Block_id[4]	Byte	10.0	16#0	☐	☑	☑	☑	☐
44	Block_id[5]	Byte	11.0	16#0	☐	☑	☑	☑	☐
45	Block_id[6]	Byte	12.0	16#0	☐	☑	☑	☑	☐
46	Block_id[7]	Byte	13.0	16#0	☐	☑	☑	☑	☐
47	Block_id[8]	Byte	14.0	16#0	☐	☑	☑	☑	☐
48	Block_id[9]	Byte	15.0	16#0	☐	☑	☑	☑	☐
49	Block_id[10]	Byte	16.0	16#0	☐	☑	☑	☑	☐
50	Block_id[11]	Byte	17.0	16#0	☐	☑	☑	☑	☐
51	Block_id[12]	Byte	18.0	16#0	☐	☑	☑	☑	☐
52	Block_id[13]	Byte	19.0	16#0	☐	☑	☑	☑	☐
53	Block_id[14]	Byte	20.0	16#0	☐	☑	☑	☑	☐
54	Building[1]	Struct	22.0		☐	☑	☑	☑	☐
55	Block_id	Array[0..14] of Byte	22.0		☐	☑	☑	☑	☐
56	Block_id[0]	Byte	22.0	16#0	☐	☑	☑	☑	☐
57	Block_id[1]	Byte	23.0	16#0	☐	☑	☑	☑	☐
58	Block_id[2]	Byte	24.0	16#0	☐	☑	☑	☑	☐
59	Block_id[3]	Byte	25.0	16#0	☐	☑	☑	☑	☐
60	Block_id[4]	Byte	26.0	16#0	☐	☑	☑	☑	☐
61	Block_id[5]	Byte	27.0	16#0	☐	☑	☑	☑	☐
62	Block_id[6]	Byte	28.0	16#0	☐	☑	☑	☑	☐
63	Block_id[7]	Byte	29.0	16#0	☐	☑	☑	☑	☐
64	Block_id[8]	Byte	30.0	16#0	☐	☑	☑	☑	☐
65	Block_id[9]	Byte	31.0	16#0	☐	☑	☑	☑	☐
66	Block_id[10]	Byte	32.0	16#0	☐	☑	☑	☑	☐
67	Block_id[11]	Byte	33.0	16#0	☐	☑	☑	☑	☐
68	Block_id[12]	Byte	34.0	16#0	☐	☑	☑	☑	☐
69	Block_id[13]	Byte	35.0	16#0	☐	☑	☑	☑	☐
70	Block_id[14]	Byte	36.0	16#0	☐	☑	☑	☑	☐
71	Building[2]	Struct	38.0		☐	☑	☑	☑	☐
72	Building[3]	Struct	54.0		☐	☑	☑	☑	☐
73	Building[4]	Struct	70.0		☐	☑	☑	☑	☐
74	Building[5]	Struct	86.0		☐	☑	☑	☑	☐
75	Building[6]	Struct	102.0		☐	☑	☑	☑	☐
76	Building[7]	Struct	118.0		☐	☑	☑	☑	☐
77	Building[8]	Struct	134.0		☐	☑	☑	☑	☐
78	Building[9]	Struct	150.0		☐	☑	☑	☑	☐
79	Building[10]	Struct	166.0		☐	☑	☑	☑	☐
80	Building[11]	Struct	182.0		☐	☑	☑	☑	☐
81	Building[12]	Struct	198.0		☐	☑	☑	☑	☐
82	Building[13]	Struct	214.0		☐	☑	☑	☑	☐
83	Building[14]	Struct	230.0		☐	☑	☑	☑	☐
84	Building[15]	Struct	246.0		☐	☑	☑	☑	☐

Obrázek 2.6: Proměnná identifikačních čísel dílů pro každou stavbu typu pole struktur obsahující pole bytů.

2.2 Komunikace PLC s robotem KUKA

Komunikace PLC s robotem funguje na bázi přenášení jednotlivých binárních signálů přes PROFINET [4] rozhraní. Pro povolení této funkce je potřeba na straně robota mít přidán a nakonfigurován balíček volitelného příslušenství KUKA PROFINET M/S na řídicí jednotce robota. Instalace a konfigurace PROFINET rozhraní se provádí v programu Workvisual [8]. Pro výměnu dat je nastaveno množství vstupů a výstupů, které lze pro komunikaci využít. V naší konfiguraci je povoleno 2032 možných vstupů a výstupů, viz obrázek 2.7.



Obrázek 2.7: Konfigurace rozhraní PROFINET v programu Workvisual.

2.2.1 Mapování vstupů a výstupů

Dále je potřeba namapovat vstupy a výstupy v rozhraní robota s příslušnými vstupy a výstupy z PLC.

Binární výstupy na straně PLC jsou namapovány přímo na binární vstupy robota, viz obrázky 2.8 a 2.9

34		O_Prg_end	Bool	%Q2154.0	☐	☑	☑	☑
35		O_Prg_pause	Bool	%Q2154.1	☐	☑	☑	☑
36		O_Next_move	Bool	%Q2154.2	☐	☑	☑	☑
37		O_Building	Bool	%Q2154.3	☐	☑	☑	☑
38		O_Cleaning	Bool	%Q2154.4	☐	☑	☑	☑
39		O_Go_home	Bool	%Q2154.5	☐	☑	☑	☑

Obrázek 2.8: Binární výstupy z PLC.

$IN[193]	BOOL	PRG_END	02:01:0193 Input	BOOL		40.0
$IN[194]	BOOL	PROGRAM_PAUSE	02:01:0194 Input	BOOL		40.1
$IN[195]	BOOL	NEXT_MOVE	02:01:0195 Input	BOOL		40.2
$IN[196]	BOOL	BUILDING	02:01:0196 Input	BOOL		40.3
$IN[197]	BOOL	CLEANING	02:01:0197 Input	BOOL		40.4
$IN[198]	BOOL	GO_HOME	02:01:0198 Input	BOOL		40.5

Obrázek 2.9: Binární vstupy do robota.

Binární vstupy na straně PLC jsou namapovány na binární výstupy robota, viz obrázky 2.10 a 2.11

10		I_move_done	Bool	%I2131.1	☐	☑	☑	☑
11		I_robot_home	Bool	%I2131.2	☐	☑	☑	☑

Obrázek 2.10: Binární vstupy do PLC.

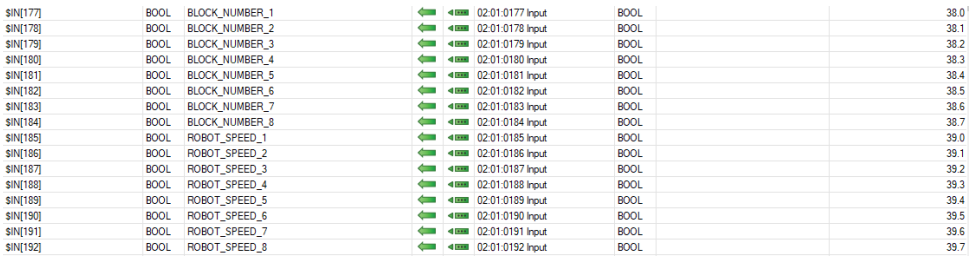
$OUT[10]	BOOL	MOVE_DONE	02:01:0010 Output	BOOL	16.1
$OUT[11]	BOOL	ROBOT_HOME	02:01:0011 Output	BOOL	16.2

Obrázek 2.11: Binární výstupy z robota.

Číselné výstupy na straně PLC typu byte jsou namapovány na skupinu osmi binárních vstupů robota, viz obrázky 2.12 a 2.13

28		O_Coord_X	DInt	%QD2136	☐	☑	☑	☑
29		O_Coord_Y	DInt	%QD2140	☐	☑	☑	☑
30		O_Coord_Z	DInt	%QD2144	☐	☑	☑	☑
31		O_Coord_B	DInt	%QD2148	☐	☑	☑	☑
32		O_Block_id	Byte	%QB2152	☐	☑	☑	☑
33		O_OV_speed	Byte	%QB2153	☐	☑	☑	☑

Obrázek 2.12: Číselné výstupy z PLC.

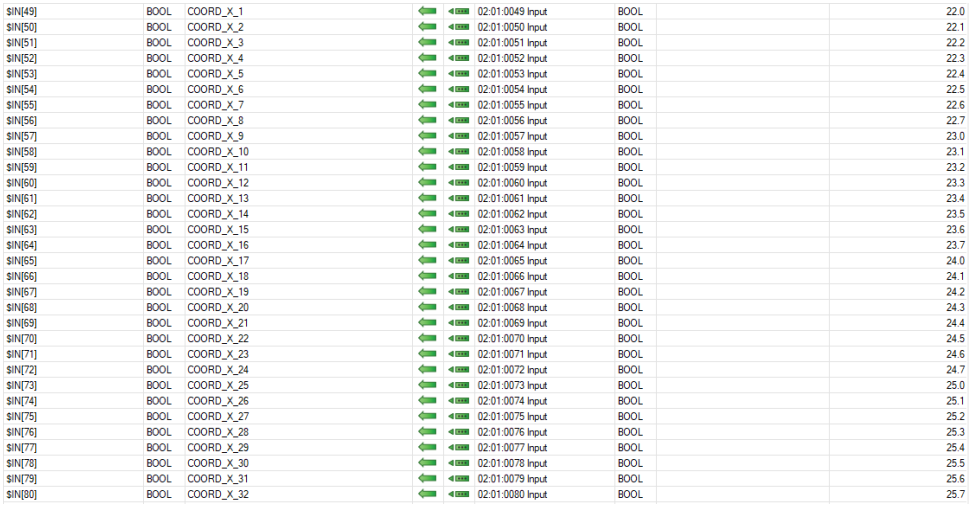


$IN[177]	BOOL	BLOCK_NUMBER_1		02:01:0177 Input	BOOL		38.0
$IN[178]	BOOL	BLOCK_NUMBER_2		02:01:0178 Input	BOOL		38.1
$IN[179]	BOOL	BLOCK_NUMBER_3		02:01:0179 Input	BOOL		38.2
$IN[180]	BOOL	BLOCK_NUMBER_4		02:01:0180 Input	BOOL		38.3
$IN[181]	BOOL	BLOCK_NUMBER_5		02:01:0181 Input	BOOL		38.4
$IN[182]	BOOL	BLOCK_NUMBER_6		02:01:0182 Input	BOOL		38.5
$IN[183]	BOOL	BLOCK_NUMBER_7		02:01:0183 Input	BOOL		38.6
$IN[184]	BOOL	BLOCK_NUMBER_8		02:01:0184 Input	BOOL		38.7
$IN[185]	BOOL	ROBOT_SPEED_1		02:01:0185 Input	BOOL		39.0
$IN[186]	BOOL	ROBOT_SPEED_2		02:01:0186 Input	BOOL		39.1
$IN[187]	BOOL	ROBOT_SPEED_3		02:01:0187 Input	BOOL		39.2
$IN[188]	BOOL	ROBOT_SPEED_4		02:01:0188 Input	BOOL		39.3
$IN[189]	BOOL	ROBOT_SPEED_5		02:01:0189 Input	BOOL		39.4
$IN[190]	BOOL	ROBOT_SPEED_6		02:01:0190 Input	BOOL		39.5
$IN[191]	BOOL	ROBOT_SPEED_7		02:01:0191 Input	BOOL		39.6
$IN[192]	BOOL	ROBOT_SPEED_8		02:01:0192 Input	BOOL		39.7

Obrázek 2.13: Číselné vstupy typu byte do robota.

Dále jsou přenášeny požadované cílové souřadnice pro pohyb robota. Ty jsou v PLC definovány jako typ REAL. Tento datový typ ovšem přes rozhraní PROFINET [4] nelze přenášet. Nejprve je nutná jejich konverze do formátu DINT, viz kapitola Konverze souřadnic typu REAL.

Číselné výstupy na straně PLC typu DINT jsou namapovány na skupinu třiceti dvou binárních vstupů robota, viz obrázky 2.12 a 2.14.



$IN[49]	BOOL	COORD_X_1		02:01:0049 Input	BOOL		22.0
$IN[50]	BOOL	COORD_X_2		02:01:0050 Input	BOOL		22.1
$IN[51]	BOOL	COORD_X_3		02:01:0051 Input	BOOL		22.2
$IN[52]	BOOL	COORD_X_4		02:01:0052 Input	BOOL		22.3
$IN[53]	BOOL	COORD_X_5		02:01:0053 Input	BOOL		22.4
$IN[54]	BOOL	COORD_X_6		02:01:0054 Input	BOOL		22.5
$IN[55]	BOOL	COORD_X_7		02:01:0055 Input	BOOL		22.6
$IN[56]	BOOL	COORD_X_8		02:01:0056 Input	BOOL		22.7
$IN[57]	BOOL	COORD_X_9		02:01:0057 Input	BOOL		23.0
$IN[58]	BOOL	COORD_X_10		02:01:0058 Input	BOOL		23.1
$IN[59]	BOOL	COORD_X_11		02:01:0059 Input	BOOL		23.2
$IN[60]	BOOL	COORD_X_12		02:01:0060 Input	BOOL		23.3
$IN[61]	BOOL	COORD_X_13		02:01:0061 Input	BOOL		23.4
$IN[62]	BOOL	COORD_X_14		02:01:0062 Input	BOOL		23.5
$IN[63]	BOOL	COORD_X_15		02:01:0063 Input	BOOL		23.6
$IN[64]	BOOL	COORD_X_16		02:01:0064 Input	BOOL		23.7
$IN[65]	BOOL	COORD_X_17		02:01:0065 Input	BOOL		24.0
$IN[66]	BOOL	COORD_X_18		02:01:0066 Input	BOOL		24.1
$IN[67]	BOOL	COORD_X_19		02:01:0067 Input	BOOL		24.2
$IN[68]	BOOL	COORD_X_20		02:01:0068 Input	BOOL		24.3
$IN[69]	BOOL	COORD_X_21		02:01:0069 Input	BOOL		24.4
$IN[70]	BOOL	COORD_X_22		02:01:0070 Input	BOOL		24.5
$IN[71]	BOOL	COORD_X_23		02:01:0071 Input	BOOL		24.6
$IN[72]	BOOL	COORD_X_24		02:01:0072 Input	BOOL		24.7
$IN[73]	BOOL	COORD_X_25		02:01:0073 Input	BOOL		25.0
$IN[74]	BOOL	COORD_X_26		02:01:0074 Input	BOOL		25.1
$IN[75]	BOOL	COORD_X_27		02:01:0075 Input	BOOL		25.2
$IN[76]	BOOL	COORD_X_28		02:01:0076 Input	BOOL		25.3
$IN[77]	BOOL	COORD_X_29		02:01:0077 Input	BOOL		25.4
$IN[78]	BOOL	COORD_X_30		02:01:0078 Input	BOOL		25.5
$IN[79]	BOOL	COORD_X_31		02:01:0079 Input	BOOL		25.6
$IN[80]	BOOL	COORD_X_32		02:01:0080 Input	BOOL		25.7

Obrázek 2.14: Číselné vstupy typu DINT do robota.

Konverze souřadnic typu REAL

Jelikož datový typ REAL nelze pomocí rozhraní PROFINET [4] přenášet, je nutná konverze do datového typu, který lze pomocí tohoto rozhraní přenést. Datový typ REAL je 32-bitové reálné číslo. Pro přenos tento typ můžeme převést na typ DINT, což je 32-bitové celé číslo. Pouhým převodem se ale reálné číslo zaokrouhlí na celé číslo a ztratí se tak informace o desetinných řádech. Pro zachování této informace lze nejprve před konverzí reálné číslo vynásobit konstantou, a pak následně po přenosu v programu robota zpětně číslo vydělit stejnou konstantou, viz obrázky 2.15 a 2.16. Zvýšením řádu konstanty se zvyšuje přesnost zaokrouhlení reálného čísla, ale snižuje se maximální rozsah hodnot, které lze použít. V našem případě byla použita konstanta 1000, díky níž dokážeme zachovat přesnost 10^{-3} mm.

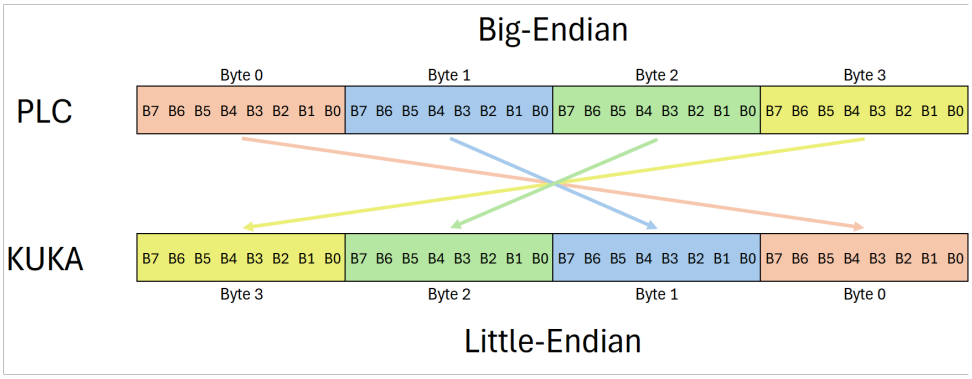
```
2 #Out_X_int := SWAP(TRUNC_DINT(#In_X_real * 1000));
3 #Out_Y_int := SWAP(TRUNC_DINT(#In_Y_real * 1000));
4 #Out_Z_int := SWAP(TRUNC_DINT(#In_Z_real * 1000));
5 #Out_B_int := SWAP(TRUNC_DINT(#In_B_real * 1000));
```

Obrázek 2.15: Konverze reálného čísla na celé číslo a vynásobení konstantou v PLC.

```
3   COORD_X = COORD_X_INT / 1000.0
4   COORD_Y = COORD_Y_INT / 1000.0
5   COORD_Z = COORD_Z_INT / 1000.0
6   COORD_B = COORD_B_INT / 1000.0
```

Obrázek 2.16: Konverze celého čísla na reálné číslo vydělením konstantou v robotovi.

Protože PLC a robot KUKA používají různé pořadí bytů, je navíc také potřeba udělat jejich prohození. PLC používá pro řazení bytů tzv. Big-Endian a KUKA používá pro řazení tzv. Little-Endian [14]. Řazení bytů je zobrazeno na obrázku 2.17.



Obrázek 2.17: Řazení bytů Big-Endian a Little-Endian.

Prohození řazení bytů je provedeno na straně PLC, viz obrázek 2.18.

```
2 #Out_X_int := SWAP(TRUNC_DINT(#In_X_real * 1000));
3 #Out_Y_int := SWAP(TRUNC_DINT(#In_Y_real * 1000));
4 #Out_Z_int := SWAP(TRUNC_DINT(#In_Z_real * 1000));
5 #Out_B_int := SWAP(TRUNC_DINT(#In_B_real * 1000));
```

Obrázek 2.18: Prohození řazení bytů v PLC.

Kapitola 3

Vidění robota

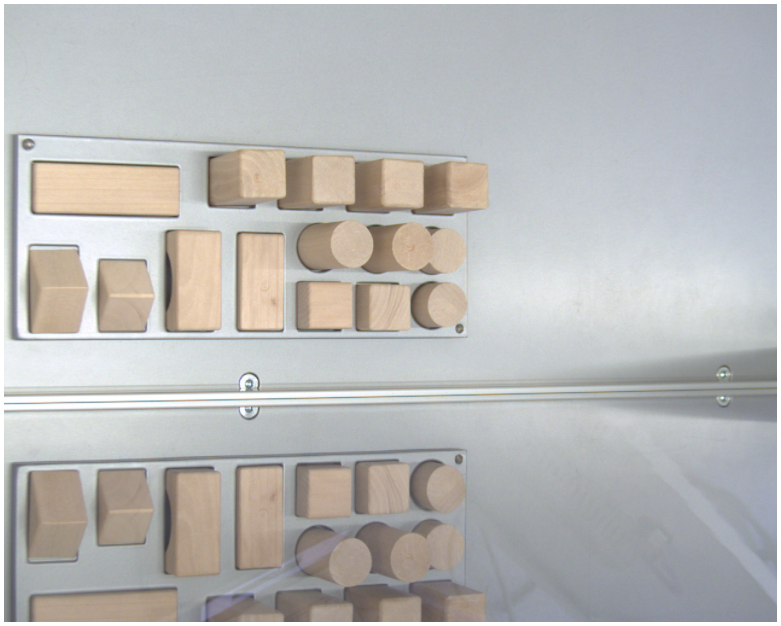
Robotické prostředí je vybaveno průmyslovou kamerou Basler acA1300-200uc [2] s danou specifikací uvedenou v tabulce 3.1, viz obrázek 3.1.

Obrázek 3.1: Kamera Basler acA1300-200uc.

Rozlišení (H x V pixely)	1280 x 1024
Senzor	PYTHON NOIP1SE1300A
Typ senzoru	CMOS
Snímkovací frekvence	203 fps
Rozhraní	USB 3.0

Tabulka 3.1: Specifikace kamery.

Kamera je ovládána přes Open-source rozhraní **pypylon** [9] pro Python, jak již bylo zmíněno v kapitole 1.2. Pomocí této knihovny lze provádět různé operace, jako je inicializace, nastavování parametrů a zachycení obrazu. Získaný snímek je ve formátu NumPy pole [11], což zjednodušuje následné zpracování obrazu. Na obrázku 3.2 lze vidět zachycený snímek kamerou.



Obrázek 3.2: Snímek pracovního prostoru zachycený kamerou Basler acA1300-200uc.

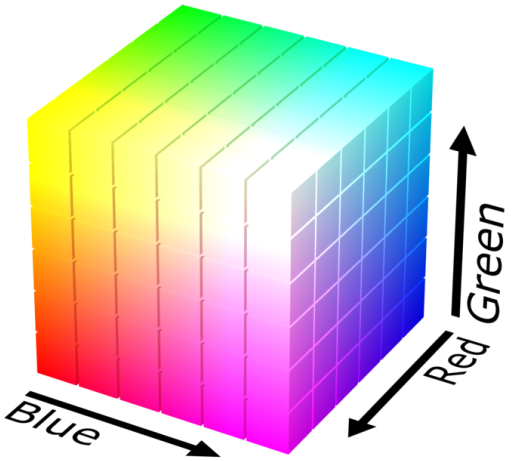
Jelikož cílem vidění robota je identifikovat, zda se díl nachází v zásobníku, je potřeba využít některé z metod segmentace. Pro charakter této úlohy je nejefektivnější využít metody barevné segmentace, neboť všechny díly jsou stejné barvy (hnědá) a pod stálým osvětlením.

3.1 Barevná segmentace

Barevná segmentace je metoda počítačového vidění, která se zaměřuje na identifikaci a oddělení objektů nebo regionů z obrazu na základě jejich barevných vlastností. Pro segmentaci je využito prahování, kde se nastaví horní a spodní hranice pro hodnoty v jednotlivých kanálech obrazu a porovnáním se vytvoří binární maska [15].

3.1.1 Výběr prostoru barev

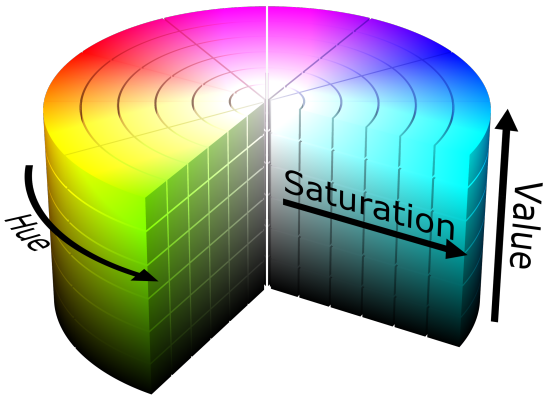
Nejčastější barevný model, používaný pro reprezentaci obrazu, je RGB (red - červená, green - zelená, blue - modrá). Je to aditivní barevný model, který mixuje tři barevné složky (červená, zelená, modrá) a tím vytváří všechny barvy spektra, viz obrázek 3.3.



Obrázek 3.3: RGB barevný model.

Tento model ale není vhodný pro barevnou segmentaci, protože špatně odděluje jasovou složku. Změní-li se osvětlení, pak se mohou změnit všechny tři kanály obrazu. Toto není příliš vhodné pro segmentaci pomocí prahování.

Vhodná volba je převedení barevného RGB spektra na HSV spektrum (hue - odstín, saturation - sytost barvy, value - hodnota jasu), viz obrázek 3.4. Jedná se o alternativní reprezentaci barevného modelu RGB a více odpovídá způsobu, jakým lidské vidění vnímá atributy vytvářející barvy. Je to vhodnější barevný model pro segmetaci pomocí prahování, protože není tolik ovlivněn změnou vnějšího osvětlení [16].



Obrázek 3.4: HSV barevný model.

Pro nalezení hodnot spodní a horní hranice jednotlivých kanálů HSV spektra byl použit Python script `segmentation_calibration.py` pro vytvoření GUI (Grafické uživatelské rozhraní), ve kterém lze zjistit optimální hodnoty hranic HSV spektra pro dokonalou segmentaci scény, viz obrázek 3.5. Je tedy potřeba zachytit dataset obrázků při různých úrovních osvětlení a najít optimální minimální a maximální hodnoty HSV segmentovaných objektů.

Obrázek 3.5: GUI pro nalezení hranic HSV spektra pro segmentaci.

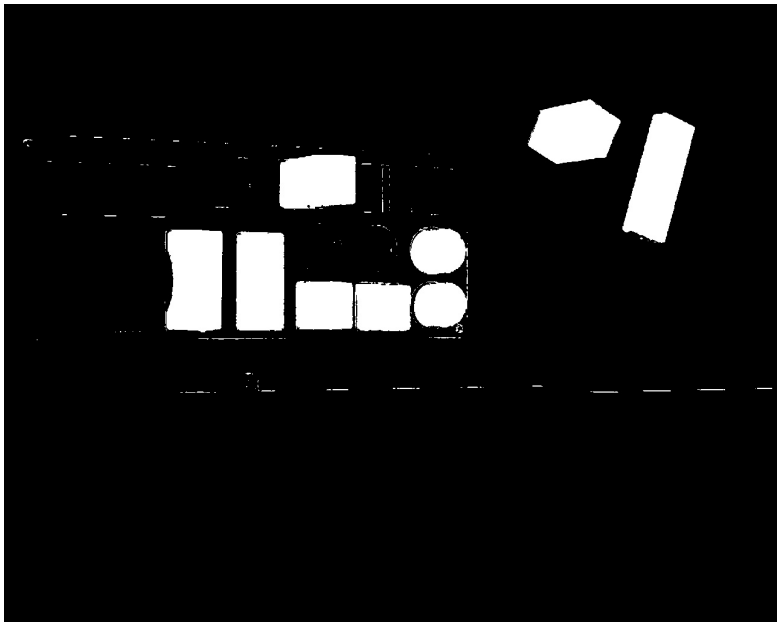
Barevné prahy byly určeny podle tabulky 3.2:

	H	**S**	**V**
min	0	35	33
max	30	255	255

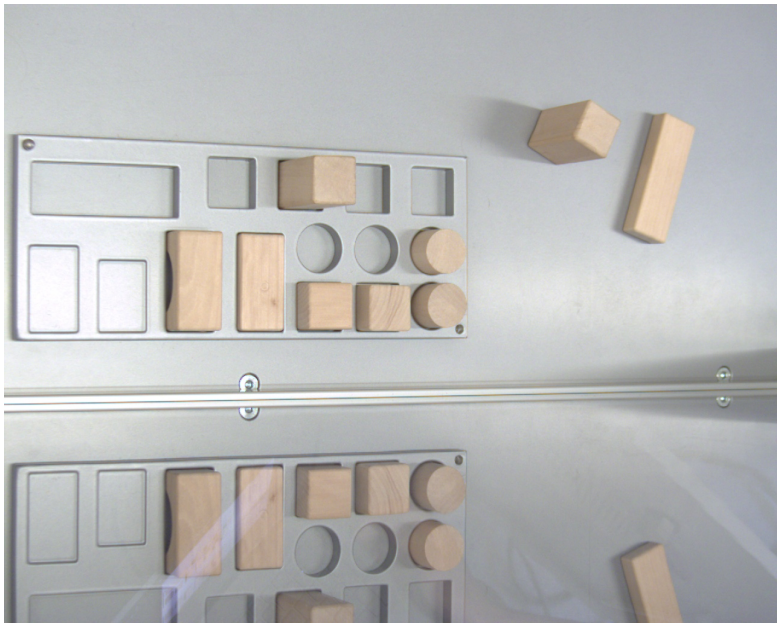
Tabulka 3.2: Hodnoty prahů HSV pro segmentaci.

3.1.2 Maskování scény

Se znalostí hodnot prahů v HSV spektru lze vytvořit binární masku pro každý zachycený obrázek. Na obrázku 3.6 je binární maska pro použitou scénu segmentace z obrázku 3.7.

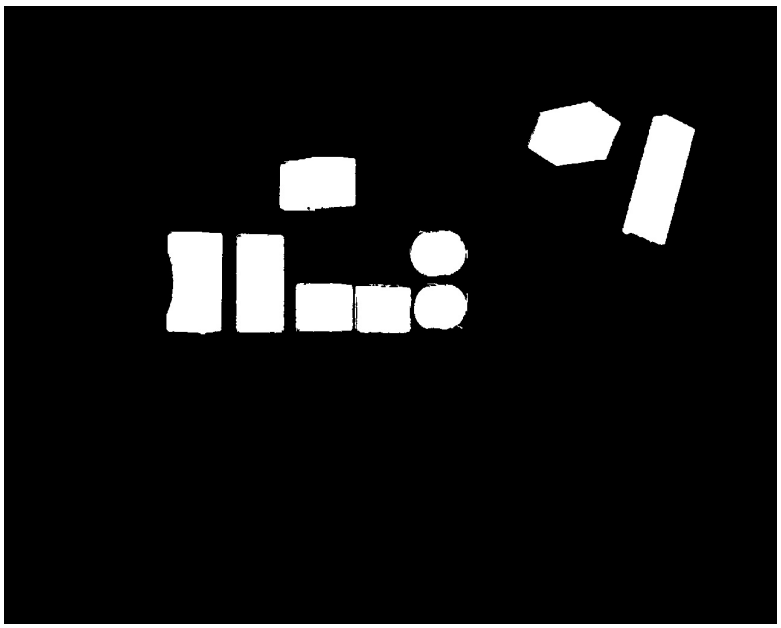


Obrázek 3.6: Binární maska.



Obrázek 3.7: Scéna použitá pro segmentaci.

Pro zlepšení segmentace můžeme ve vysegmentovaném obrázku najít všechny spojité komponenty a odstranit ty segmenty, které mají menší plochu než daný práh. V tomto případě je použit práh plochy 3000 pixelů. Vylepšená segmentační maska je zobrazena na obrázku 3.8



Obrázek 3.8: Binární maska s odstraněnými malými nebo deformovanými segmenty.

3.1.3 Identifikace přítomnosti dílů v zásobníku

Z vysegmentovaného obrázku následně můžeme jednoduše zjistit, zda je daný díl přítomen v zásobníku. Jelikož poloha dílů v zásobníku vůči kameře je neměnná, můžeme každému dílu přiřadit výřez z celého obrázku, ve kterém se nachází. Následně stačí vypočítat poměr nenulových pixelů z vysegmentovaného výřezu obrázku. V našem případě rozhodujeme, že díl je přítomen v zásobníku, pokud jeho výřez z celkového obrázku obsahuje alespoň 80% nenulových pixelů. Tímto způsobem zaručíme robustnost identifikace objektu i při méně přesné segmentaci.

Kapitola 4

Simulace a virtuální zprovoznění

Virtuální zprovoznění digitálního dvojčete znamená počítačově simulovat chování jeho 3D modelu. Snažíme se tedy zprovoznit model fyzického pracoviště, který se chová a komunikuje stejně jako ten reálný. Díky tomuto virtuálnímu nástroji jsme schopni urychlit vývoj, testovat funkčnost řešení a zaručit bezpečnost při vývoji a provozu [17].

Pro simulaci robota byl použit program KUKA SIM 4.3 [18] a spolu s reálnou testovací PLC centrálou SIMATIC S7-1500, CPU 1511-1 PN jsme schopni zprovoznit digitální dvojče našeho pracovního prostředí. Pokud by nebylo k dispozici reálné testovací PLC, je možno využít program PLC SIM Advanced [19] pro vytvoření virtuálního PLC.

4.1 S7-PLCSIM Advanced

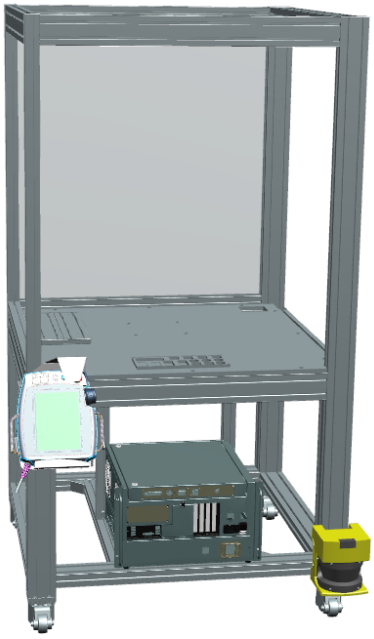
S7-PLCSIM Advanced je software od společnosti Siemens, který umožňuje vytvářet virtuální PLC pro testování PLC programů [19].

4.2 KUKA SIM

KUKA SIM je simulační prostředí od společnosti KUKA, které umožňuje detailní simulaci a virtuální zprovoznění průmyslových robotů. Simulátor umožňuje vytvářet detailní modely pracovního prostředí, včetně umístění a geometrie dílů, pracovních stanic a dalších zařízení. Toto umožňuje realistickou simulaci manipulace s objekty v reálném průmyslovém prostředí. Dále dovoluje psát robotické programy, které se mohou exportovat do prostředí Workvisual a následně nahrát do reálného robota [18].

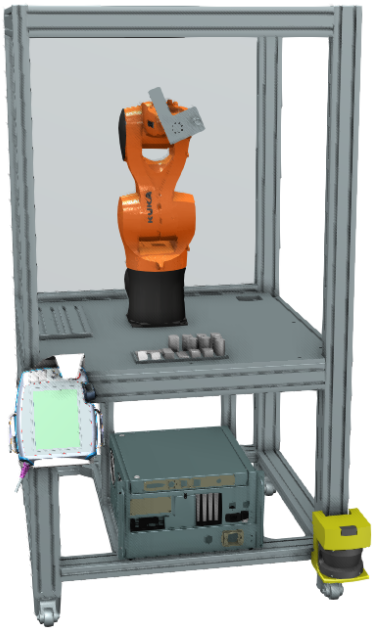
4.2.1 Vytvoření modelu v programu KUKA SIM

3D modely se do prostředí mohou vkládat jako CAD data ve formátu .jt, viz obrázek 4.1



Obrázek 4.1: 3D CAD model pracovního prostředí v programu JT2Go.

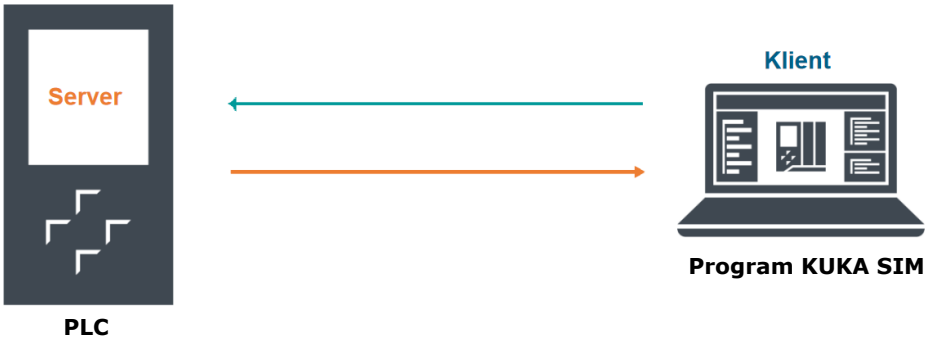
Následným výběrem průmyslového robota (v našem případě **KUKA KR 6 R700 sixx**) z katalogu dostupných modelů dostaneme virtuální dvojče fyzického prostředí, viz obrázek 4.2.



Obrázek 4.2: Virtuální dvojče v programu KUKA SIM.

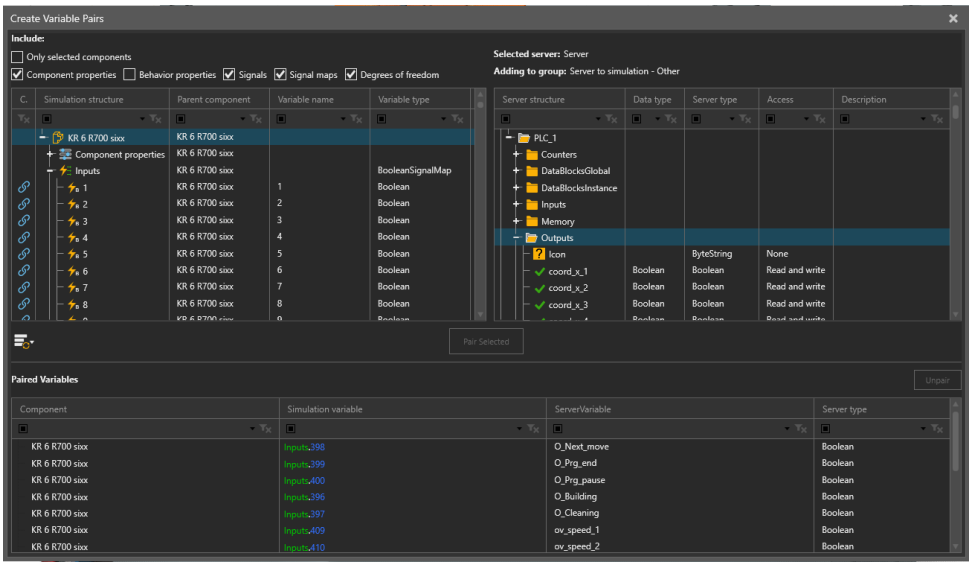
4.2.2 Propojení a komunikace programu KUKA SIM a PLC

Komunikace mezi programem KUKA SIM a reálným nebo virtuálním PLC je realizována prostřednictvím protokolu OPC UA, viz obrázek 4.3. OPC Unified Architecture (OPC UA) je průmyslový M2M (machine-to-machine) komunikační standard. Specifikace OPC UA je založena na předávání dat mezi OPC UA klientem a UPC UA serverem. Komunikace typu klient/server je založena na protokolu TCP/IP. Roli serveru zprostředkovává PLC automat, který má integrované metody OPC UA. Program KUKA SIM poté zastává roli klienta a při využití doplňku KUKA.Sim Connectivity dovoluje propojení s OPC UA serverem [6].



Obrázek 4.3: Schéma přenosu dat pomocí OPC UA komunikace.

V programu KUKA SIM [18] se následně propojí výstupní a vstupní proměnné, zasílané z PLC se vstupními a výstupními proměnnými robota, viz obrázek 4.4. Mapování proměnných mezi PLC a KUKA SIM je provedeno identicky jako mapování se skutečným robotem, které je zmíněno v kapitole 2.2.1.



Obrázek 4.4: Konfigurace mapování proměnných mezi PLC a KUKA SIM.

Kapitola 5

Uživatelské obrazovky

K řízení a monitorování chodu programu je k dispozici uživatelské rozhraní programovatelného logického automatu (PLC) se třemi hlavními obrazovkami. Každá obrazovka má svůj vlastní účel.

5.1 Hlavní obrazovka

Hlavní obrazovka je centrálním místem, kde uživatel ovládá celý program. Na této obrazovce jsou zobrazeny hlavní ovládací prvky pro řízení a monitorování stavu programu, viz obrázek 5.1.

Obrázek 5.1: Hlavní obrazovka.

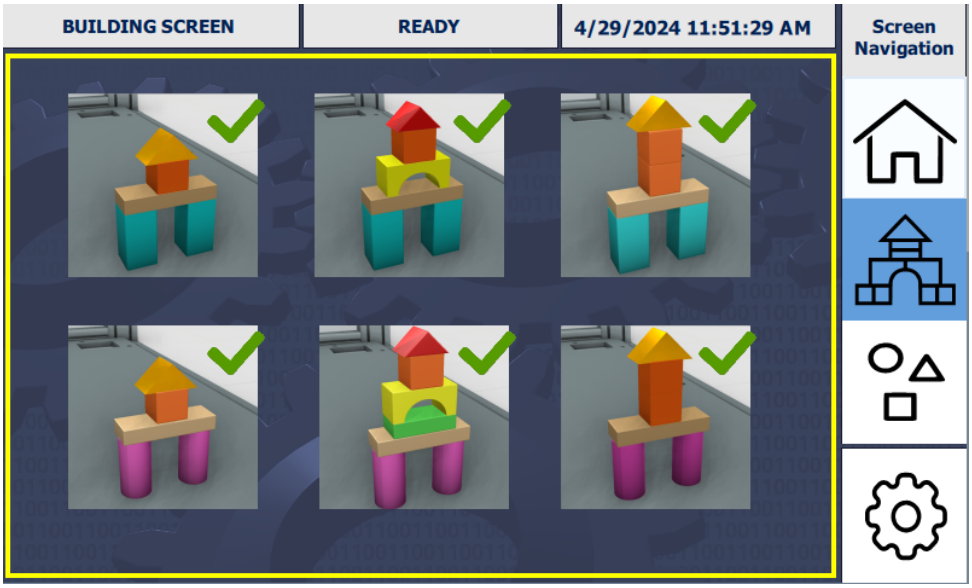
5.1.1 Popis tlačítek

- **Auto** - Navolit automatický režim. Kroky manipulace se vykonávají plynule.

- **Manual** - Navolit manuální režim. Další krok manipulace se vykoná po stisknutí tlačítka Next Move.
- **Next Move** - Další krok manipulace.
- **Start Building** - Zahájit proces navolené stavby.
- **Start Disassembling** - Zahájit proces rozebrání dokončené stavby.
- **Program End** - Ukončit manipulační program.
- **Program Pause** - Pozastavit pohyb robota.
- **Robot Speed** - Snížit nebo navýšit rychlost pohybu robotického manipulátoru.

5.2 Obrazovka staveb

Obrazovka staveb umožňuje uživateli vybrat konkrétní stavbu, kterou má robot postavit. Dále graficky podává informaci, zda je provedení dané stavby možné, viz obrázek 5.2.



Obrázek 5.2: Obrazovka se stavbami.

5.3 Obrazovka se zásobníkem dílů

Obrazovka dílů poskytuje uživateli přehled o dostupných stavebních dílech pro robota, viz obrázek 5.3.

Obrázek 5.3: Obrazovka se zásobníkem dílů.

Kapitola 6

Závěr

Pro tuto práci byly nejprve navrženy vhodné hardwarové prostředky potřebné pro realizaci strojového vidění. Vhodným výběrem pro tuto úlohu bylo průmyslové PC NET-III 2I640DW, které svou velikostí, specifikací a výkonem nejlépe splnilo požadavky na zabudování do stávající hardwarové konfigurace robotické buňky. Zároveň dovolilo připojení 2D RGB kamery pomocí USB rozhraní a síťové propojení s řídicím PLC.

Python program, běžící na tomto PC, zajišťuje ovládání kamery pomocí knihovny pypylon. Následně zpracovává obraz s použitím knihoven OpenCV a NumPy a komunikuje s nadřízeným PLC přes protokol S7-TCP/IP pomocí knihovny Snap7.

Pro indentifikaci dílů v zásobníku byla použita metoda barevné segmentace za pomoci prahování v HSV spektru. Převedení z RGB do HSV spektra bylo provedeno za účelem stabilní segmentace obrazu i při změně vnějšího osvětlení.

V PLC projektu byly navrženy vhodné datové komunikační bloky pro výměnu signálů a dat mezi PLC a průmyslovým PC a mezi PLC a robotem KUKA.

Pro jednoduchou a intuitivní obsluhu manipulační úlohy byly vytvořeny HMI uživatelské obrazovky.

Vhodně zvolená forma parametrizace cílových pozic robotického programu umožnila variabilitu procesu stavby, jejího rozebírání a jednoduché přidávání dalších variant staveb.

Literatura

[1] KR 6 R700 sixx: Small robots KR AGILUS sixx. In: KUKA.com [online]. 2022 [cit. 2024-05-10]. Dostupné z: https://www.kuka.com/-/media/kuka-downloads/imported/8350ff3ca11642998dbdc81dcc2ed44c/0000210361_en.pdf

[2] AcA1300-200uc. Basler Product Documentation [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://docs.baslerweb.com/aca1300-200uc

[3] NET-III 2I640DW: Compact Fanless Embedded Computer with Elkhart Lake ATOM® X6413E / J6412 Processor. In: LEX SYSTEM [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://www.lexsystem.com.cn/images/cgcustom/img020230621141619.pdf

[4] PROFINET System Description [online]. PROFIBUS Nutzerorganisation e. V. (PNO), 2014 [cit. 2024-05-10]. Dostupné z: https://us.profinet.com/wp-content/uploads/2012/11/PROFINET_SystemDescription_ENG_2014_web.pdf

[5] 6ES7516-3AN02-0AB0 [katalog výrobků]. Siemens [online]. 2022 [cit. 2024-05-10]. Dostupné z: https://mall.industry.siemens.com/mall/en/cz/Catalog/Product/6ES7516-3AN02-0AB0

[6] OPC 10000-1 UA Part 1: Overview and Concepts. OPC Foundation [online]. 2022 [cit. 2024-05-10]. Dostupné z: https://opcfoundation.org/developer-tools/documents/view/158

[7] Programmable controllers - Part 3: Programming languages [online]. 3rd ed. International Electrotechnical Commision, 2013 [cit. 2024-05-10]. ISBN 978-2-83220-661-4. Dostupné z: https://webstore.iec.ch/preview/info_iec61131-3%7Bed3.0%7Db.pdf

[8] KUKA.WorkVisual. KUKA [online]. c2024 [cit. 2024-05-19]. Dostupné z: https://www.kuka.com/cs-cz/produkty,-slu%C5%BEby/robotick%C3%A9-syst%C3%A9my/software/syst%C3%A9mov%C3%BD-software/kuka_systemsoftware/kuka_work-visual

[9] Pypylon: Basler's pylon Software Suite Interface for Python. Basler [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.baslerweb.com/en/software/pylon/pypylon/

[10] OpenCV modules. OpenCV [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://docs.opencv.org/4.9.0/index.html

[11] NumPy user guide. NumPy [online]. c2008-2022 [cit. 2024-05-10]. Dostupné z: https://numpy.org/doc/stable/user/index.html#user

[12] Open TCP/IP Communication via Industrial Ethernet [online]. Siemens, 2005 [cit. 2024-05-10]. A5E00711636-01. Dostupné z: https://cache.industry.siemens.com/dl/files/612/22146612/att_113921/v1/t-bausteine_e.pdf

[13] Python-snap7 documentation. Read the Docs [online]. c2013 [cit. 2024-05-10]. Dostupné z: https://python-snap7.readthedocs.io/en/latest/index.html#

[14] ASHTARI, Hossein. Big Endian vs. Little Endian: Key Comparisons. Spiceworks [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://www.spiceworks.com/tech/tech-general/articles/big-endian-vs-little-endian/

[15] HEMA, D. a S. KANNAN. Interactive Color Image Segmentation using HSV Color Space [online]. Science and Technology Journal, 2020 [cit. 2024-05-10]. 2321-3388. Dostupné z: https://mzu.edu.in/wp-content/uploads/2020/05/Interactive-Color-Image-Segmentation-using-HSV-Color-Space.pdf

[16] MAMDOUH, Tarek. Color spaces (RGB vs HSV) - Which one you should use? HubPages [online]. 2020 [cit. 2024-05-10]. Dostupné z: https://discover.hubpages.com/technology/Color-spaces-RGB-vs-HSV-Which-one-to-use

[17] Virtuální zprovoznění - budoucnost průmyslu. SIEMENS Visions [online]. 2018 [cit. 2024-05-10]. Dostupné z: https://www.visionsmag.cz/virtualni-zprovozneni-meni-budoucnost-prumyslu

[18] KUKA.Sim. KUKA [online]. c2024 [cit. 2024-05-10]. Dostupné z: https://www.kuka.com/cs-cz/produkty,-slu%C5%BEby/robotick%C3%A9-syst%C3%A9my/software/pl%C3%A1nov%C3%A1n%C3%AD-projektov%C3%A1n%C3%AD-servis-bezpe%C4%8Dnost/kuka,-d-,sim

[19] AIT ALI YAHIA, Redouane. An Introduction to PLCSim Advanced. SolisPLC [online]. [cit. 2024-05-10]. Dostupné z: https://www.solisplc.com/tutorials/plcsim-advanced