VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Vlastimil Lahoda

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

AUTOMATIZOVANÁ TŘÍDICÍ BUŇKA
AUTOMATED CLASSIFICATION CELL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vlastimil Lahoda**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. František Burian, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Vlastimil Lahoda ***ID:*** 158184
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Automatizovaná třídicí buňka

POKYNY PRO VYPRACOVÁNÍ:

Výsledkem práce by měla být automatizovaná buňka pro třídění materiálu osazená kamerou od výrobce Cognex a manipulátorem Universal Robots UR5e. Manipulátor by měl nabírat volně sypané díly, tyto opticky zkontrolovat a podle výsledku kontroly třídit do výstupních pořadačů.

1. Seznamte se s průmyslovými manipulátory firmy Universal Robots a jejich spojení s kamerovými systémy.
2. Navrhněte buňku realizující všechny požadované funkce, současně se zajištěním bezpečnosti obsluhy.
3. Realizujte třídící buňku a otestujte funkčnost jednotlivých periferií.
4. Sjednoťte souřadné systémy robotu a detekované pozice dílu z kamery, otestujte spolehlivé nabírání detekovaného dílu.
5. Realizujte program pro dopravu nabraného dílu do kontrolní kamerové jednotky a zahajte komunikaci s ní.
6. Na základě výsledku z kontrolní kamerové jednotky vložte nabraný díl do příslušného zásobníku.
7. Otestujte funkčnost celého procesu buňky.

Pro úspěšné splnění semestrálního projektu postačuje splnění bodů 1 až 3.

DOPORUČENÁ LITERATURA:

SPONG, Mark W., Seth HUTCHINSON a M. VIDYASAGAR. Robot modeling and control. Hoboken, NJ: John Wiley, c2006. ISBN 978-0471649908.

MARK M., Spong a Vidyasagar M. Robot dynamics and control. India: Wiley, 2008. ISBN 978-8126517800.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. František Burian, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Za účelem optimalizace výroby ve společnosti Tyco Electronics Czech s.r.o. jsem navrhl a zrealizoval automatizovanou třídicí buňku s kolaborativním manipulátorem Universal Robots UR5e a kamerovým systémem Cognex. Třídicí buňka má nabrat volně sypané díly a ty následně opticky otestovat, podle výsledku kontroly díly třídit do výstupních pořadačů.

Nejprve jsem se seznámil s manipulátorem a kamerovým systémem. Následně jsem navrhl a nakreslil elektrické schéma. Podle elektrického schématu jsem zapojil rozvaděč i ostatní periferie. Dále jsem naprogramoval algoritmus ovládající rameno manipulátoru a inspekce obou kamer. Nakonec jsem úspěšně otestoval funkčnost celé třídicí buňky.

KLÍČOVÁ SLOVA

Kolaborativní manipulátor, Universal Robots, UR5e, Cognex, strojové vidění, In-Sight Explorer

ABSTRACT

To optimize production at Tyco Electronics Czech s.r.o. I designed and implemented an automated classification cell with a collaborative manipulator Universal Robots UR5e and a Cognex camera system. The classification cell is supposed to pick up the loose parts and then optically test them, sorting the parts into output binders following the inspection results.

First of all, I had to learn programming language for Universal Robots manipulators and Cognex vision system. Next step was to design and draw an electrical diagram. According to the diagram, I connected the cabinet and other peripherals. I also programmed an algorithm controlling the manipulator arm and inspecting of both cameras. Finally, I successfully tested the functionality of the entire classification cell.

KEYWORDS

Collaborative manipulator, Universal Robots, UR5e, Cognex, machine vision, In-Sight Explorer

LAHODA, Vlastimil. *Automatizovaná třídicí buňka*. Brno, 2020, 102 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. František Burian, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Automatizovaná třídicí buňka“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 1. 6. 2020 .....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Františku Burianovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno 1. 6. 2020 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

S postupným nástupem Průmyslu 4.0 do výrobních závodů stoupá význam kolaborativních manipulátorů. Tyto manipulátory mají oproti klasickým manipulátorům zřejmé výhody. Jak už název naznačuje mohou spolupracovat s lidmi, aniž by je ohrozily. Díky této schopnosti vzniká prostor pro nové formy výroby, poněvadž již nejsou nutné ochranné kryty nebo laserové skenery kolem manipulátorů, proto je možné změnit koncept výrobních linek a šetřit drahocenný prostor uvnitř výrobních závodů [2].

Spolupráce člověka s manipulátorem může mít řadu podob, od sdílení pracovního prostoru bez kontaktu člověka s manipulátorem až po variantu, kdy manipulátor v reálném čase přizpůsobuje svůj pohyb dle pohybu pracovníka. Pravděpodobně nejběžnější variantou využití kolaborativních manipulátorů je sdílená aplikace, kdy manipulátor pracuje spolu s pracovníkem a postupně vykonávají dílčí pracovní operace. Manipulátor vykonává operace, které jsou pro lidského pracovníka neergonomické, namáhavé nebo se neustále opakují [3]. Mohou též vykonávat úkony, které vyžadují přesnost, které by lidský operátor nedosáhl, například pokud by bylo nutné nějakou součástku velice přesně přilepit na určené místo.

Integrace kolaborativních manipulátorů do výrobních závodů nemusí být vždy snadná a nese s sebou řadu problémů, týkajících se zejména bezpečnosti a ES prohlášení o shodě. Společnost Tyco Electronics Czech s.r.o. se sídlem v Kuřimi, pro níž tento projekt zpracovávám, hledá nová řešení ke zvýšení efektivity výroby. Jednou z možností je implementace kolaborativních manipulátorů.

Cílem této práce je navrhnout a následně realizovat první automatizovanou třídicí buňku s kolaborativním manipulátorem v této společnosti. Tato buňka může sloužit jako odrazový můstek pro větší nasazení kolaborativních manipulátorů v závodě. Nejdříve se budu zabývat důvodem vzniku práce a specifikací požadavků na buňku. V teoretické části dle zadaných parametrů vyberu vhodný manipulátor, jeho periferie, kamerový systém a ukáži způsob jejich programování. V kapitole 3 popíši návrh dispozice buňky, návrh a realizaci rozvaděče, dále představím a popíši program manipulátoru a inspekce obou kamer. Nakonec otestuji funkčnost třídicí buňky.

1 Motivace a cíle práce

V této kapitole se budu zabývat důvody pro vytvoření práce a specifikací požadavků. Pro porozumění požadavkům společnosti na automatizovanou třídicí buňku nejdříve uvedu současnou situaci v závodě.

Výrobní závod Tyco Electronics Czech s.r.o. je součástí společnosti TE Connectivity, která je největším světovým výrobcem a dodavatelem kabelové a konektorové techniky. Závod v Kuřimi se zaměřuje svým výrobním programem na automobilový průmysl, vyrábí například konektory, kabeláže pro připojení airbagu, pojistkové skříně, nabíjecí kabely a konektory pro hybridní a elektrická vozidla. Je zde zaměstnáno vice než 2 800 zaměstnanců. Výrobní charakter je diskrétní s procesy, které mají různou úroveň automatizace, od plně automatických linek po ruční pracoviště. V automobilovém průmyslu je nyní patrný trend zvyšování rychlosti a množství nových výrobků. Proto je nutné stále hledat nová řešení pro optimalizaci výroby.

Část výrobních linek si společnost navrhuje a vyrábí sama, zbylé výrobní linky dodávají externí výrobci. Dosud se ve výrobních linkách používaly pouze klasické manipulátory umístěné za bezpečnostními kryty. Jelikož kolaborativní manipulátor dosud ve společnosti nebyl použit, jedná se zatím o testovací projekt. Pokud se osvědčí, bude buňka sloužit jako základ pro testovací linku do výroby. Buňka se může například rozšířit o automatizovaný podavač materiálu anyfeed SX240 [18].

Obr. 1.1: Testovaný výrobek.

Cílem této diplomové práce je navrhnout a zrealizovat automatizovanou třídicí buňku, která volně sypané díly nabere, opticky zkontroluje a dle výsledku kontroly odloží do příslušného pořadače. Buňka by měla být navržena dle interních předpisů společnosti a se zajištěním bezpečnosti obsluhy. Nabíraný výrobek je zobrazen na obrázku 1.1, díl je vyražen z plechu o tloušťce přibližně 1 mm. Jeho rozměry jsou přibližně 45 x 35 mm a obsahuje otvory a drážky. Díl se má nabírat pomocí pneumatických přísavek. Čas na otestování jednoho dílu by neměl být delší než 10 sekund.

Protože se jedná o testovací projekt, jsem velice limitován rozpočtem, z toho důvodu je použito několik komponentů, které nejsou úplně ideální, ale byly zrovna k dispozici a svůj účel splní.

2 Teoretická část práce

2.1 Výběr manipulátoru

V této kapitole se budu zabývat výběrem kolaborativního manipulátoru. V dnešní době má kolaborativní manipulátor v nabídce celá řada výrobců. Například Fanuc má v nabídce celou rodinu kolaborativních manipulátorů lišících se velikostí a nosností [4]. Nebo výrobce Universal Robots má v nabídce pouze kolaborativní manipulátory [5].

K dispozici mám stůl o rozměrech 1100 x 800 mm. Manipulátor bude umístěn v rohu stolu a nabírací místo bude umístěno ve vzdálenosti maximálně 850 mm od manipulátor. To znamená, že manipulátor by měl mít dosah alespoň 850 mm. Nabíraný materiál váží přibližně 10 gramů a konstrukce uchycení přísavek váží přibližně 80 g. Tím pádem není zvláštní požadavek na nosnost manipulátoru. Velikost nabíracího prostoru na dílu a velikost přísavek dovoluje přesnost nabírání až ± 0,5 mm.

2.1.1 Porovnání manipulátorů Fanuc a Universal Robots

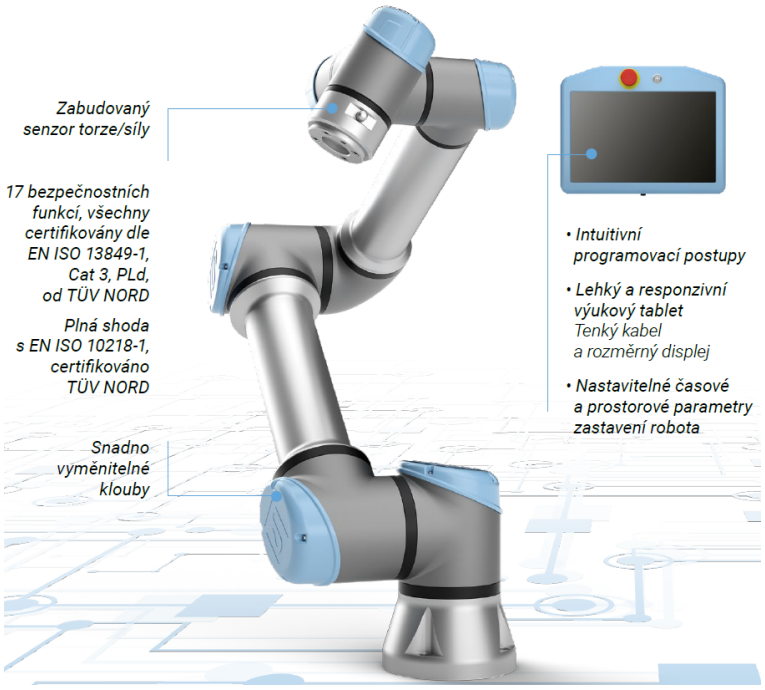
Tab. 2.1: Porovnání manipulátorů dle jejich parametrů [4][5]

Výrobce	Universal Robots		
Model	UR3e	UR5e	UR10e
Dosah	500 mm	850 mm	1300 mm
Opakovatelnost	± 0,03 mm	± 0,03 mm	± 0,03 mm
Nosnost	3 kg	5 kg	10 kg
Hmotnost	11,2 kg	20,6 kg	33,5 kg
Výrobce	Fanuc		
Model	CR-7iA	CR-7iA/L	CR-14iA/L
Dosah	717 mm	911 mm	911 mm
Opakovatelnost	± 0,01 mm	± 0,01 mm	± 0,01 mm
Nosnost	7 kg	7 kg	14 kg
Hmotnost	53 kg	55 kg	55 kg

Od obou výrobců jsem vybral tři manipulátory, porovnání jejich parametru je zobrazeno v tabulce 2.1. Z této tabulky je patrné, že mému zadání vyhovují čtyři manipulátory, dva od výrobce Universal Robots (UR5e a UR10e) a dva od výrobce

Fanuc (CR-7iA/L a CR-14iA/L). Vzhledem k zadaným parametrům jsem jako optimální manipulátor vybral model UR5e, zejména z důvodu nižší hmotnosti. Tím bude eliminována nutnost použít robustnější stůl na kterém bude manipulátor upevněn.

2.1.2 Manipulátor UR5e



Obr. 2.1: Kolaborativní manipulátor UR5e [5].

Pro svoji práci jsem si vybral šestiosý kolaborativní manipulátor UR5e od výrobce Universal Robots, jeho podobu je možné vidět na obrázku 2.1. Jak už bylo řečeno, tento manipulátor má dosah 850 mm, opakovatelnost 0,03 mm, mezi jeho další parametry patří například maximální rychlost kloubů 180 °/s, rychlost nástroje 1 m/s, přesnost senzoru momentu 4 N a frekvence systému 500 Hz [6]. Součástí manipulátoru je řídicí box a ovládací panel (teach pendant) s dotykovým displejem, tlačítkem, kterým se odbrzdí osy manipulátoru, tlačítko Nouzového zastavení a tlačítko pro zapnutí nebo vypnutí celého manipulátoru.

Řídicí box obsahuje šestnáct digitálních vstupů, stejný počet digitálních výstupů, dva analogové vstupy, dva analogové výstupy a čtyři separátní vysokorychlostní digitální vstupy. Dále obsahuje 24 V stejnosměrný zdroj pro napájení výstupů. Řídicí box dále disponuje třemi komunikačními porty, jedním USB 2.0, jedním USB 3.0 a jedním RJ45 konektorem, který podporuje ModbusTCP, ProfiNet a Ethernet/IP komunikační protokol.

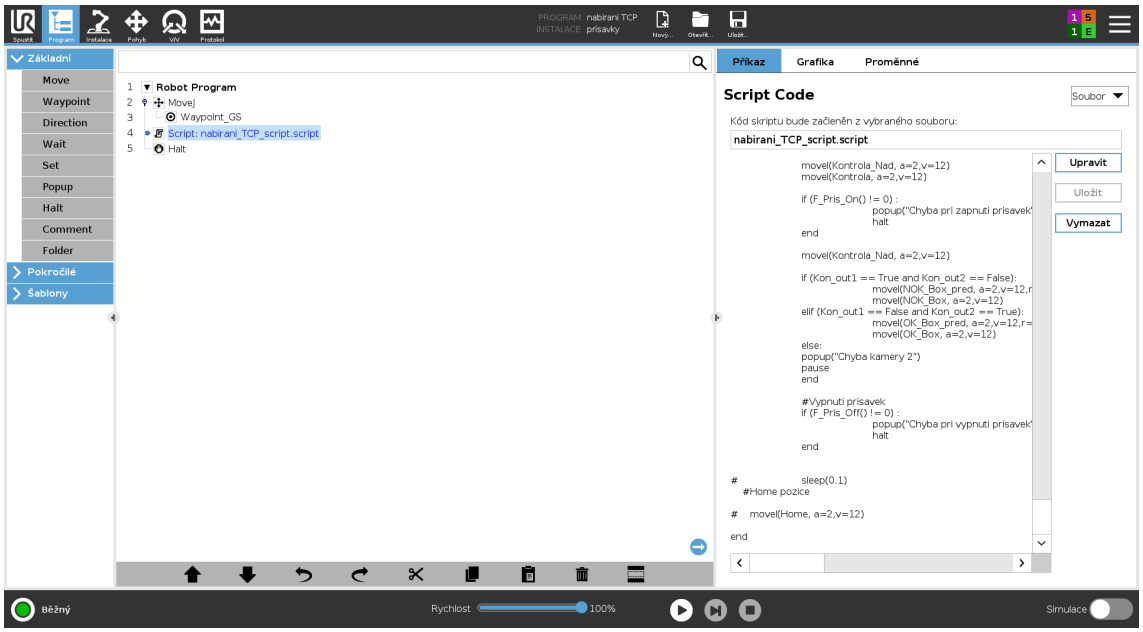
2.1.3 Programování manipulátoru UR5e

Manipulátor je primárně ovládán a programován přes ovládací panel. I když podporuje vzdálené ovládání po síti, tak pro jeho zapnutí a spuštění programu je stále nutný ovládací panel. Programovat je možné na dvou úrovních, na základní grafické úrovní je připravena nabídka funkcí, které se se zadanými parametry skládají pod sebe v pořadí, v jakém se mají vykonat. Programování na této úrovni je snadné, ovšem nabídka funkcí je omezena, tudíž pokud je nutné naprogramovat složitější úkoly, jako je například zpracování ethernetové komunikace, je nutné programovat pomocí skriptů. Program napsán na skriptovací úrovni může být následně volán na základní úrovni funkci *script* jako jediná funkce, nebo jakou část většího programu.

Jelikož bude manipulátor s kamerou komunikovat pomocí Ethernet/IP, musel jsem nastudovat seznam instrukcí pro psaní skriptů [7]. Programovací jazyk se jmenuje URScript a je podobný programovacímu jazyku C nebo Pascal. Pomocí tohoto programovacího jazyka jsem schopný obsloužit komunikaci s kamerovým systémem, který mi odesílá souřadnice nabíraného dílu.

Na ovládacím panelu běží grafické uživatelské prostředí (GUI) s názvem PolyScope. Toto GUI obsahuje ručně ovládané panely kterými je možné ovládat rameno manipulátoru, jeho vstupy/výstupy, dále programovat a spouštět programy. Výrobce vydává poměrně často aktualizace softwaru v nichž opravuje chyby a přidává nové funkce. Při tvorbě této práce jsem používal verzi 5.6.0.90886. Rozhraní PolyScope je rozděleno na tři části, a to na záhlaví, zápatí a pracovní obrazovku. V levé části záhlaví je možné zvolit aktuálně zobrazenou obrazovku, v prostřední části je umístěn správce programů a instalací, kde je možné vidět aktuálně zvolený program a instalaci, dále je možné vytvořit nový program, otevřít jiný program a aktuální program uložit. V pravé části záhlaví je zobrazen kontrolní součet aktuální bezpečnostní konfigurace a nakonec ikona pro zobrazení nabídky možností. Nabídka možností obsahuje položky nápověda, informace, nastavení rozhraní PolyScope a možnost vypnutí manipulátoru. V levé části zápatí je umístěna ikona Inicializace, která jednak značí aktuální stav manipulátoru a jednak slouží k vyvolání nabídky pro inicializaci manipulátoru. Uprostřed zápatí je umístěn posuvník rychlosti jenž v reálném čase ukazuje relativní rychlost ramene manipulátoru, s ohledem na nastavení bezpečnosti. Tuto rychlost je možné pomocí posuvníku ručně měnit. Napravo do posuvníku rychlosti se nachází trojice tlačítek. Jsou to tlačítka *Přehrát*, *Krok* a *Zastavení*. Tlačítka slouží k ovládání běhu programu. V pravém spodním rohu se nachází přepínač Simulace, který slouží k zapnutí simulačního režimu, kdy v tomto režimu nedochází k pohybu ramene manipulátoru, pouze je jeho pohyb zobrazen na obrazovce.

Jak již bylo zmíněno, tak v levém horním rohu je možné vybrat pracovní ob-



Obr. 2.2: GUI PolyScope

razovku, máme na výběr ze šesti obrazovek. První z nich nese název *Spustit* a je rozdělena do třech částí. První část *Program* zobrazuje aktuálně načtený program a jeho stav, dále je možné načíst jiný program. Druhá část *Stáří robota* zobrazuje jak již název napovídá stáří manipulátoru od prvního spuštění. A poslední část nazvaná *Proměnné* zobrazuje aktuální hodnotu proměnných načteného programu.

Druhá obrazovka, kterou je možné zvolit, nese název *Program*. Na této obrazovce se programují programy a její vzhled je možné vidět na obrázku 2.2. Tato obrazovka je rozdělena do tři sloupců. V levém sloupci se nachází instrukce rozdělené do třech kategorií. Z těchto instrukcí se následně v prostředním sloupci skládá program. Program se poté vykonává od první instrukce, která se nachází na druhém řádku, postupně až do poslední instrukce. Ve spodní šedé liště se nachází nástroje pro práci s instrukcemi (přesouvání, vyjmutí, mazaní atd.). V pravém sloupci je poté možné zvolit ze tří karet. Karta *Příkaz* slouží k parametrizaci zvolené instrukce. Například u příkazu *Move* je možné zvolit typ pohybu, středový bod nástroje a prvek (bod, ke kterému se vztahují body trasy), nebo zda se mají použít úhly kloubů. Dále je možné zadat rychlost a zrychlení kloubů, nebo tlačítkem *Obnovit* nastavit rychlost a zrychlení kloubů na základní hodnotu.Funkce a možnosti jednotlivých instrukcí jsou popsány v návodu [6]. Druhá karta s názvem *Grafika* zobrazuje 3D model manipulátoru, který odpovídá aktuální poloze ramena nebo v případě simulace znázorňuje polohu ramena během spuštěného programu. Poslední karta zobrazuje proměnné stejně jako v předchozí obrazovce.

Obrazovka *Instalace* slouží k nastavení vlastností manipulátoru. Opět se v levém

sloupci nachází čtveřice rozbalovacích nabídek. První z nich se věnuje obecným vlastnostem manipulátoru. Jedním z důležitých obecných vlastnosti je středový bod nástroje (*TCP*), který slouží k nastavení parametrů přimontovaného chapadla. Udává se zde jeho poloha, těžiště a váha vzhledem k přírubě. Dále se zde nastavuje způsob montáže manipulátoru. Můžeme zde nastavit vstupy/výstupy, vytvořit instalační proměnné, nastavit akce při spuštění manipulátoru, nastavení plynulého přechodu a nastavení sledování dopravníku. Dále můžeme nastavit šroubování, vstupy/výstupy pro nástroj a nastavení domovské bezpečné polohy manipulátoru. V druhé rozbalovací nabídce je možné nastavit bezpečnostní parametry manipulátoru. Tyto volby jsou chráněny bezpečnostním heslem, aby nemohlo dojit k neodborné či nechtěné manipulaci s nastavenými hodnotami, důsledkem čehož by mohlo dojít k ohrožení zdraví. Hodnoty parametrů v této sekci musí být definovány na základě posouzení rizik. Je zde možné nastavit mezní hodnoty ramene, omezení kloubů, definovat roviny, omezení polohy a směru nástroje. Dále je zde možné nastavit konfigurovatelné bezpečnostní vstupy a výstupy, definovat připojený hardware, určit bezpečnou výchozí pozici a povolit použití třípolohového přepínače. V předposledním rozbalovacím menu je možné definovat prvky, jako jsou body, přímky nebo plochy. Prvky zde nadefinované mohou být poté použity v bezpečnostním nastavení. V posledním rozbalovacím menu se nastavují parametry sběrnic. Pro moji práci je důležité nastavení sběrnice EtherNet/IP, je zde pouze povolení či zakázání této sběrnice a poté činnost programu při ztrátě spojení.

Další obrazovka nazvaná *Pohyb* slouží k ručnímu ovládání ramene manipulátoru. Jsou zde tři možnosti jak manipulátor ovládat. První z nich se nachází v levém sloupci a poloha se mění stisknutím šipky požadovaným směrem vzhledem k vybranému prvku (souřadnému systému), například základně. Zbývající dvě možnosti se nachází v pravém sloupci. Druhou možností ovládání manipulátoru je zadání požadovaných souřadnic nástroje vzhledem ke zvolenému prvku. Poslední možností je zadat přímo požadované polohy kloubů nebo pomocí šipek pohybovat příslušnými klouby. V prostřední části se nachází vizualizace aktuální polohy ramene. Pod vizualizací se nachází tři tlačítka, tlačítko *Home* slouží k přemístění manipulátoru do "domácí" polohy, která je nastavená v sekci Instalace - Obecné - Domů. Prostřední tlačítko slouží k zarovnaní osy Z aktivního TCP s vybraným prvkem. Poslední tlačítko při jeho stisknutí umožní ručně přemístit rameno do požadované polohy.

Na předposlední obrazovce pojmenované *V/V* je možné průběžně sledovat a nastavovat aktuální vstupní/výstupní signály. Je možné vidět aktuální stav binárních i analogových vstupů/výstupů. Výstupy je možné navíc ručně ovládat, binární je možné zapnout nebo vypnout, u analogových je možné nastavit úroveň výstupního napětí či proudu.

Poslední obrazovka slouží k diagnostice hardwaru manipulátoru. V levém hor-

ním rohu se zobrazují informace týkající se ovládací jednotky manipulátoru, zatímco pravá horní strana obrazovky zobrazuje informace o kloubech ramene. U každého kloubu je zobrazena teplota, zatížení kloubů a napětí. Spodní dvě třetiny obrazovky slouží k zobrazení protokolů dat. Zde jsou zobrazeny chybové, varovné nebo informační zprávy s časovým identifikátorem. Může zde být například zobrazeno, že došlo k zastavení manipulátoru, protože narazil do překážky.

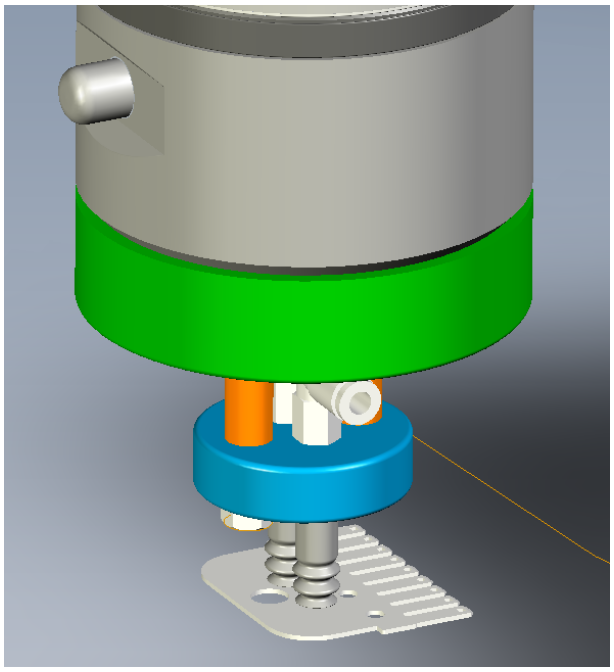
Skripty jsou spouštěny pomocí funkce *script* v Okně *Program* a záložce *Pokročilé*. Po vložení funkce do programu je možné vybrat ze dvou možností, jestli vložíme pouze řádek nebo soubor. Je-li vybrán řádek, je možné pomocí Editoru výrazů zapsat jeden řádek kódu URscript. Při vybrané možnosti soubor je možné vytvořit nový soubor s kódem nebo otevřít již existující soubor. Na výpisu 2.1 je uveden zdrojový kód podmínky if v jazyce URScript.

Výpis 2.1: Příklad implementace podmínky if v jazyce URScript

```
1 # Podmínka if
2 if a < 5:
3     a = a + 2
4 elif b > 5:
5     b = b * a
6 else:
7     c = a + b
8 end
```

2.1.4 Chapadlo manipulátoru

Na zápěstí robota bude připevněno chapadlo (gripper) se dvěma pneumatickými přísavkami, model je možné vidět na obrázku 2.3. Za každou přísavkou je umístěn snímač podtlaku, oba tyto snímače jsou připojeny do vyhodnocovací jednotky a ta je následně připojena na digitální vstupy manipulátoru. Díky těmto snímačům jsem schopný detekovat, jestli je díl správně chycen k přísavkám. Za snímači jsou umístěny ejektory, které vytvářejí podtlak. Následně jsou pneumatické hadice nataženy přes celé rameno manipulátoru až do bloku pneumatických válců, ty jsou ovládány digitálními výstupy manipulátoru. Do pneumatických válců je vzduch přiveden přes úpravnu vzduchu, kde je možné nastavit požadovaný tlak vzduchu, dále obsahuje tlakový spínač a spínací ventil.



Obr. 2.3: 3D model chapadla

2.2 Výběr pneumatických prvků

Jako první bylo nutné vybrat snímače podtlaku a spolu s nimi vyhodnocovací jednotku. Vybíral jsem z řešení od dvou renomovaných výrobců SMC a Festo. Porovnání snímačů a vyhodnocovacích jednotek od obou výrobců je zobrazeno tabulce 2.2. Z této tabulky je patrné, že mezi oběma řešeními jsou jen minimální rozdíly a můžeme zvolit kterékoliv z nich. Já si vybral řešení od SMC, a to z těchto dvou důvodů, Vyhodnocovací jednotka má 4 vstupy, tím pádem tam je rezerva pro případné přidání dalších přísavek, a navíc s tímto řešením máme dobré zkušenosti z jiného stroje.

Obr. 2.4: Vyhodnocovací jednotka PSE201-Ma4C (vlevo) a snímač tlaku PSE541-R04 (vpravo) [8].

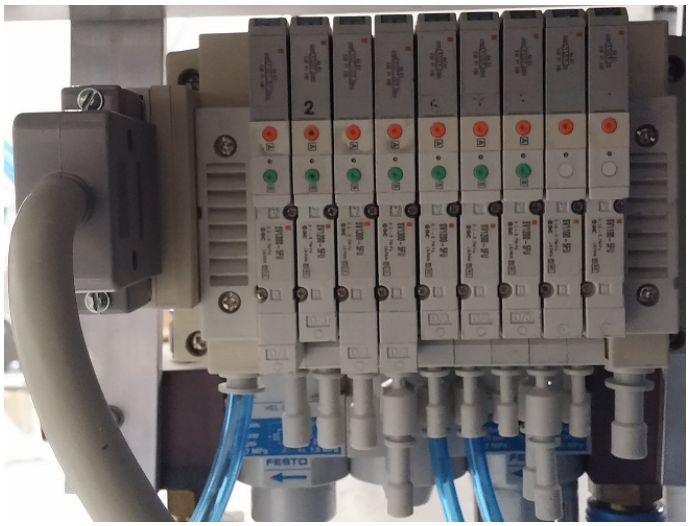
Ejektor spolu s přísavkami vybíral můj kolega z oddělení designu podle firemního

Tab. 2.2: Porovnání snímačů tlaku a vyhodnocovacích jednotek dle jejich parametrů [8][9][10]

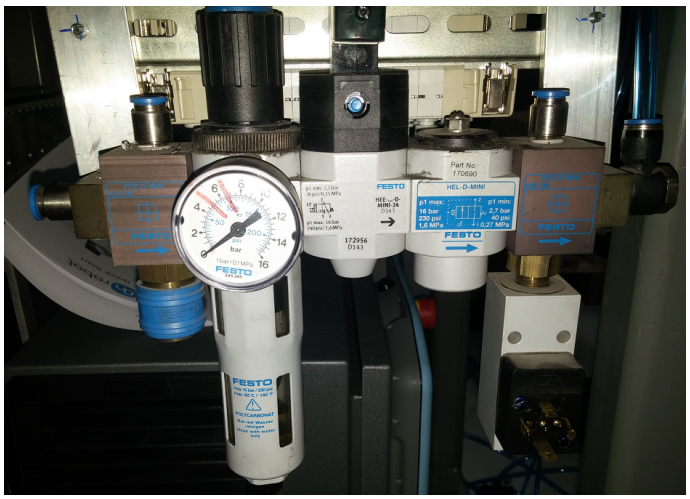
	Snímač tlaku	
Model	SMC PSE 541-R04	Festo SPTE-V1R-S4-V-2.5K
Rozsah jmenovitého tlaku	0 až -101 kPa	0 až -100 kPa
Přesnost	± 2 % z rozsahu	± 3 % z rozsahu
Opakovatelnost	± 0,7 % z rozsahu	± 0,3 % z rozsahu
Napájení	12 až 24 VDC ± 10%	10 až 30 VDC
Výrobce	Vyhodnocovací jednotka	
Model	SMC PSE 201-MA4C	Festo SCDN-V2-EC4-PNLK-L1
Počet vstupů	4	2
Počet výstupů	5 PNP	2 PNP nebo NPN
Napájení	12 až 24 VDC ± 10%	15 až 30 VDC

standardu. Následuje ventilový blok SMC SS5v1-16-0214OBO, jedná se o starší již vyřazený, avšak stále plně funkční ventilový blok. Vzhledem k jeho stáří se mi k němu nepodařilo najít žádnou dokumentaci. Našel jsem dokumentaci k jeho nástupcům jejichž elektrické zapojení je shodné, ze které jsem nastudoval a použil zapojení [11] na straně 66. Ventilový blok se připojuje přes speciální kabel SMC AXT100-DS25-30. Tento pětadvaceti žilový kabel má na jedné straně D-sub konektor, kterým se připojuje k ventilovému bloku, na druhém konci kabelu jsou volné vývody pro připojení do svorkovnice. V katalogu [11] na straně 124 jsem nastudoval i barvy vodičů a jejich zapojení v konektoru, tím jsem získal všechny potřebné informace pro správné zapojení. Tento ventilový blok je možné nahradit například dvěma 5/2 elektromagnetickými ventily SMC SY3201-5U1.

Posledním pneumatickým prvkem je úpravna vzduchu, tu jsem použil taktéž zánovní z již vyřazeného stroje. Úpravna vzduchu je složena ze vstupního rozdělovače, redukčního ventilu s ukazatelem tlaku a filtrem, spínacího ventilu HEE-D-MINI-24 [12], ventilu s pomalým náběhem tlaku HEE-D-MINI-24 [12], výstupního rozdělovače se spínačem tlaku PEV-1/4-B [13], vše od výrobce Festo. Spínací ventil pouští vzduch do ventilového bloku, kde už jsou poté ovládány jednotlivé přísavky. Tlakový spínač slouží k detekci výpadku stlačeného vzduchu. Kompletní pneumatické schéma je v příloze A. Nominální tlak stlačeného vzduchu pro správnou funkci zařízení je 600 kPa.



Obr. 2.5: Ventilový blok SMC SS5V1-16-0214OBO.



Obr. 2.6: Úpravna vzduchu Festo.

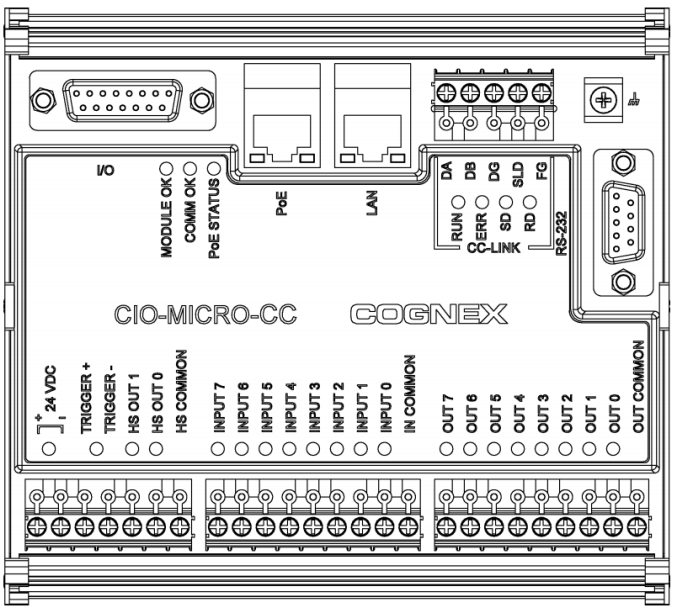
2.3 Výběr kamer

Protože je ve společnosti Tyco s.r.o preferovaným výrobcem kamer firma Cognex, vybíral jsem kamery pouze od tohoto výrobce. Tyto kamery jsou u nás v závodě požity na desítkách zařízení na nespočtu různých aplikací. Implementaci kamerových systémů do výrobních linek, případně úpravám stávajících aplikací dle potřeb výroby se u nás věnuje oddělení Vision Systém. Jelikož spravují spoustu zařízení s různorodými aplikacemi je výhodné požívat výrobky od jednoho výrobce a umět je na vysoké úrovni používat.

Ovšem vzhledem k tomu, že se jedná pouze o testovací projekt, nemohl jsem si kamery vybírat, ale dostal jsem k dispozici dvě kamery Cognex In-Sight 1403C,

Obr. 2.7: Kamera Cognex In-Sight 1403C [14].

které byly ve firmě nevyužité. Jedná se o kameru s rozlišením 1600 x 1200 pixelů s barevným CCD snímačem. Kompletní specifikace je uvedená zde [14] na straně 17. Kamery samotné je nutné připojit do modulu Cognex In-Sight CIO-MICRO nebo CIO-MICRO-CC. Tento modul slouží nejen k napájení pomocí PoE (Power over Ethernet), ale i k připojení vstupů a výstupů pro kameru. Kamera umožňuje komunikaci s PLC nebo manipulátorem pomocí celé řady protokolů, patří mezi ně protokol CC-Link, Ethernet Native, EtherNet /IP, Modbus TCP, PROFINET, UDP a TCP IP. Tato kamera obsahuje vlastní výpočetní jednotku a programuje se každá zvlášť v programu In-Sight Explorer. Tento software obsahuje řadu nástrojů ke zpracování obrazu, použití těchto nástrojů je snadné.



Obr. 2.8: Modul Cognex In-Sight CIO-MICRO-CC [15].

Pokud bych si mohl vybrat kamerový systém sám, volil bych mezi dvěma va-

riantami. Tou první jsou nástupci již starší kamery Cognex 1403C a to například kameru Cognex 8402, která má stejné rozlišení, ale větší výkon. Tudíž by zvládla více operací v kratším čase. Druhou možností je místo těchto kamer s vlastní výpočetní jednotkou, pořídit kamery bez výpočetní jednotky a připojit je k průmyslovému počítači Cognex VC5, který umožňuje připojit až 4 kamery. Druhá možnost by zřejmě byla dražší, ovšem při použití Cognex VC5 se programuje ve SDK Cognex Vision PRO, který umožňuje realizovat složitější úlohy než In-Sight Explorer, je ale složitější. Další výhodou je, že v případě potřeby lze přidat další kameru bez výpočetní jednotky, která je značně levnější než s výpočetní jednotkou čili dodatečné náklady by byly menší.

První z kamer bude umístěna nad nabíracím prostorem a bude sloužit k nalezení dílu a odeslání jeho souřadnic manipulátoru. Aby bylo zajištěno spolehlivé rozpoznání dílu bude nabírací prostor podsvětlen pomocí backlightu LTBC234234-W od výrobce Opto Engineering. Jedná se o polykarbonátovou plochu o velikosti 234 x 234 mm kontinuálně podsvícenou LED diodami. Rozměr nabírací plochy je kompromis mezi výškou kamery a velikostí plochy pro nabírání.

Obr. 2.9: Backlight Opto Engeneering LTBC234234-W [16] (vlevo) a LTPVR070-00-1-W-24V (vpravo) [17].

Druhá kamera bude sloužit ke kontrole nabraného dílu. Implementace kompletní inspekce kontroly dílu není součástí zadání této práce, mělo ji realizovat oddělení

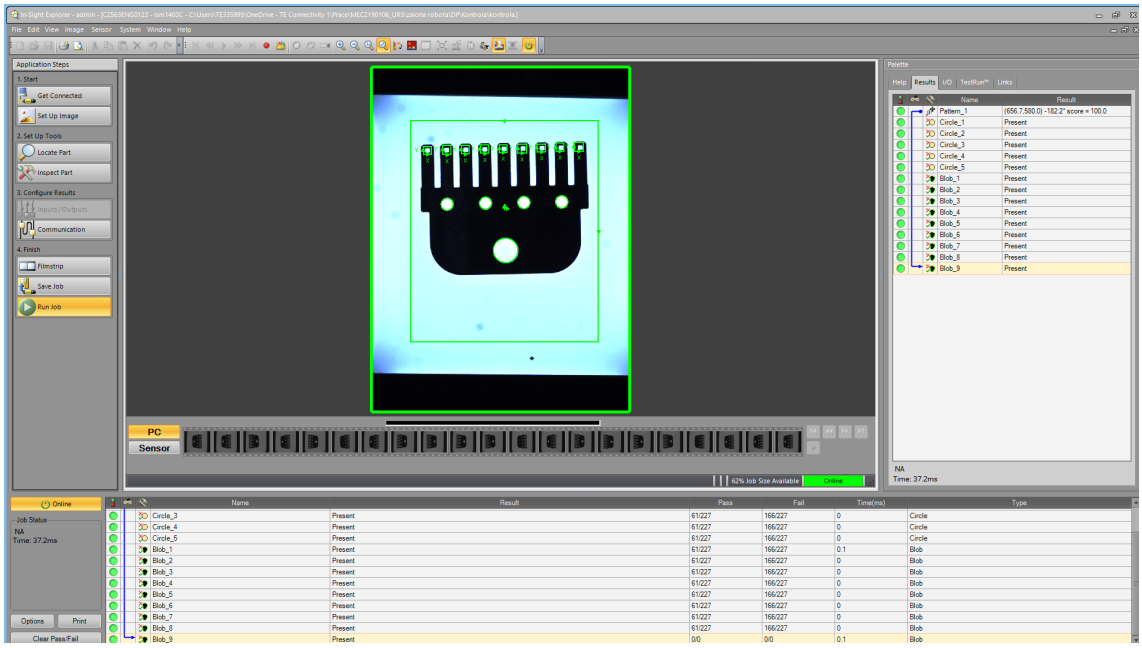
Vision System v našem závodě. Mým úkolem bylo zrealizovat komunikaci mezi kamerou a manipulátorem. Ovšem z důvodů, které jsem nemohl žádným způsobem ovlivnit, to nakonec v termínu nemohli zrealizovat. Proto jsem si návrh scény a základní inspekci vypracoval sám. Jelikož testovaný díl může mít několik různých vad a některé z nich jsou velice složité na implementaci, rozhodl jsem implementovat pouze část z nich. Zaměřil jsem se na kontrolu přítomnosti otvorů, stává se totiž, že otvor není zcela prostřižen. Testovaný díl obsahuje čtrnáct otvorů různého tvaru a velikosti. Scénu jsem zvolil obdobně jako při nabírání a to backlight připevněný k desce stolu a kamera umístěna nad ním. Při tomto rozložení je největší kontrast mezi otvorem a tělem dílu, tudíž kontrola přítomnosti otvorů bude velice spolehlivá. Nevýhoda ovšem je, že manipulátor testovaný díl odloží, odjede ze zorného pole kamery a po otestování díl opět nabere a vloží do příslušného odkládacího boxu. Backlight jsem použil opět od výrobce Opto Engineering model LTPVR070-00-1-W-24V, který má rozměr svítící plochy 70 x 70 mm.

2.3.1 Programovaní kamer

Kamerový systém Cognex In-Sight se programuje a spravuje pomocí vývojového prostředí In-Sight Explorer. Toto vývojové prostředí obsahuje dvě vývojová prostředí s různým přístupem k programování. Jedná se o vývojová prostředí nazvaná EasyBuilder a Spreadsheet, každé z nich je vhodné na jiné typy aplikací. V průběhu vytváření inspekce lze mezi oběma prostředími přepínat. Oběma vývojovým prostředím se budu věnovat níže. Vytvářet inspekce je možné buď přímo u zařízení, to znamená, že počítač je propojen s kamerovým systémem, inspekce se ukládá přímo do paměti kamery a pracuje se s aktuální fotografií. Nebo je možné inspekce vytvářet odkudkoli za pomocí emulátoru, který je součástí vývojového prostředí In-Sight Explorer. Pro použití emulátoru je nutné nejdříve připravit a uložit fotografie na kterých se bude inspekce testovat. Následně je možné programovat inspekce i bez připojeného kamerového systému. Toho lze využít například při vývoji rozsáhlejší výrobní linky, kdy si programátor kamer uloží dostatečný počet fotografií a výrobní linka zůstane k dispozici pro ostatní programátory či mechaniky. Finální odladění je ovšem vhodné provést přímo u výrobní linky.

Vývojové prostředí EasyBuilder obsahuje šablonu aplikačních kroků, která uživatele provede postupem vytváření aplikací strojového vidění. Rozhraní vývojového prostředí EasyBuilder je zaměřeno na práci s obrázkem, kde lze v obrázku pohybovat interaktivními grafickými prvky nástrojů. Tím je umístit na požadované místo a měnit jejich parametry. Rozhraní je intuitivní a přehledné, ovšem jeho možnosti jsou omezené. Při vytváření úlohy v prostředí EasyBuilder jsou ukládány do tabulky v prostředí Spreadsheet specifické funkce, které odpovídají naprogramovaným

aplikačním krokům. Zjednodušeně řečeno prostředí EasyBuilder je prostředníkem mezi uživatelem a prostředím Spreadsheet s úkolem zjednodušit a zrychlit vývoj inspekce. Vývojové prostředí EasyBuilder je vhodný spíše pro jednodušší aplikace, kde se provádí detekce defektů nástroji pro kontrolu přítomnosti bloků, jasu, kontrastu a počtu pixelů. Dále je vhodné pro měřicí aplikace využívající nástroje pro kontrolu vzdálenosti a úhlů [27].



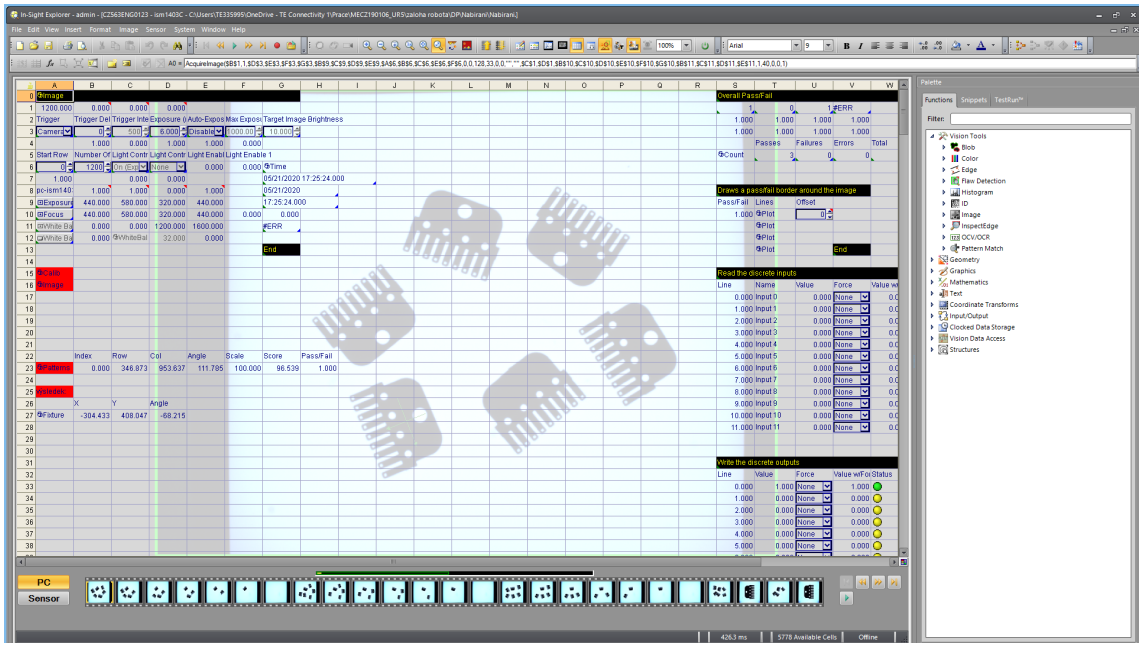
Obr. 2.10: Okno vývojového prostředí EasyBuilder.

Okno vývojového prostředí EasyBuilder je v základním rozložení rozděleno do čtyřech částí, jeho podobu je možné vidět na obrázku 2.10. Horizontálně je rozdělena na dvě části, přičemž horní z nich je dále rozdělena do třech sloupců. V pravém sloupci se nachází podokno *Palette*, které je rozděleno na pět karet. První karta je *Help* a slouží k zobrazení nápovědy k právě vybranému aplikačnímu kroku. Druhá karta *Results* zobrazuje výsledek každého z použitých nástrojů a umožňuje uživateli optimalizovat nastavení úlohy. Třetí karta *I/O* slouží k monitorování stavu vstupů a výstupů. Čtvrtá karta *TestRun* se požívá k vytvoření ověřovacích testů strojového vidění. Poslední, pátá karta *Links* slouží k propojení vlastností vložených inspekcí. V prostředním okně se nachází aktuálně vyhodnocovaný snímek. Snímek je ohraničen červeným nebo zeleným rámem, který znázorňuje výsledek provedené inspekce. Pokud má rám zelenou barvu inspekce proběhla s kladným výsledkem, naopak pokud je rám červený inspekce proběhla s negativním výsledkem. Pod fotografií se nachází filmový pás, který slouží pro výběr snímku uložených buď v počítači nebo v senzoru. V levém sloupci se nachází podokno *Application Steps* to obsahuje již

zmíněnou šablonu aplikačních kroků pro vytváření aplikace strojového vidění nazvané job. Podokno *Application Steps* se skládá celkem z devíti kroků, jednotlivé kroky jsou dále rozděleny do čtyřech kategorií. Po kliknutí na kterýkoli krok se ve spodním podokně zobrazí nabídka možností pro konfiguraci parametrů. Krok *Get Connected* slouží k připojení počítače k senzoru nebo emulátoru a otevření existujícího nebo vytvoření nového jobu. V kroku *Set Up Image* je možné pořídit nebo nahrát snímek, nastavit možnosti získávání snímku a kalibrovat obrázek na jednotky reálného světa. V dalším kroku *Locate Part* je vybrán nástroj pro lokalizaci prvku, který definuje referenční počáteční bod pro ostatní kontrolní nástroje. Krok *Inspect Part* se používá k sestavení a konfiguraci nástrojů počítačového vidění, které budou použity k vytvoření jobu. V dalším kroku *Inputs/Outputs* se definuje nastavení digitálních vstupů a výstupů jak na senzoru tak na I/O modulu. *Communication* krok se používá k exportováni obrázků přes FTP nebo na SD kartu a k nastavení komunikačních sít pro komunikaci s PLC nebo robotem. Krok *Filmstrip* se používá ke konfiguraci nastavení pro přehrávání snímku uložených v počítači nebo pro prohlížení obrázků a výsledků uložených v senzoru. Předposlední krok *Save Job* umožňuje provádět správu souborů a údržby senzoru, zatím co je senzor offline. Jakmile jsou všechny parametry úlohy nastaveny, následuje poslední krok *Run Job*, kde je možné spustit job (přepnou senzor do stavu online) a následně provést kontrolu dosažených výsledků. V případě nevyhovujících výsledku je nutné senzor nejdříve přepnout do stavu offline a následně je možné provádět úpravy.

Spreadsheet je klasické programovací prostředí vývojového prostředí In-Sight Explorer, které využívá poloprůhlednou tabulku pro konfiguraci vision systému In-Sight pro úlohy strojového vidění. Prostředí Spreadsheet poskytuje nejkomplexnější sadu vizuálních a komunikačních funkcí, které jsou umístěny v paletě. Paleta obsahuje rozbalitelnou nabídku funkcí a úryvků, které lze snadno vložit do tabulky. Nabídka a panel nástrojů poskytují přístup k získávání obrázků, vkládání funkcí do buněk a konfiguraci všech ostatních aspektů vision aplikace. Rozhraní SpreadSheet je analogické s jinými tabulkovými aplikacemi (například MS Excel), pokud jde o standardní operace a funkce, jako je manipulace s bloky buněk, úprava buněk, odkazování na buňky a funkce vkládání. Aplikace v rozhraní Spredsheet je konfigurována vždy jen po jedné buňce. Obsah každé buňky je definován vzorcem a každá část informace vložená do buňky, ať už je to jedna číselná hodnota nebo složitá funkce zpracování obrázku, je považována za součást vzorce. Po kliknutí na buňku, dojde k jejímu ohraničení černým rámem a informace obsažené v buňce se zobrazí na liště vzorců. Stejně jako u jiných konvenčních tabulek je pořadí volání buněk dynamicky určováno vztahem a závislostmi mezi funkcemi buněk. Primárně se určuje pořadí provádění buněk podle jejich závislosti a sekundárně podle jejich umístění v tabulce. Výjimku z normálního pořadí provádění buněk tvoří funkce *ReadResult* a *WriteRe-*

sult, které jsou volány ke konci úlohy. Další výjimkou jsou hodinové funkce, které jsou naopak volány na začátku úlohy. Úloha je spuštěna změnou hodnoty buňky se závislostmi, nebo vnějšími událostmi, jako jsou spouštěče pořízení snímku (trigger AcquireImage), sériová data a příchozí pakety od TCPDevice. Grafické ovládací prvky (například tlačítko, zaškrtávací políčko) signalizují spuštění události tabulky. Většina funkcí nástroje Vision Tool závisí přímo nebo nepřímo na funkci *AcquireImage* v buňce A0. Po získání nového obrázku se tyto funkce provedou a vrátí nové hodnoty [28]. Prostředí Spreadsheet je vhodné pro složitější aplikace, které vyžadují komplexnější úpravy v tabulce nebo aplikace kde se kombinují výsledky nástroje pomocí netriviální logiky [27].



Obr. 2.11: Okno vývojového prostředí Spreadsheet.

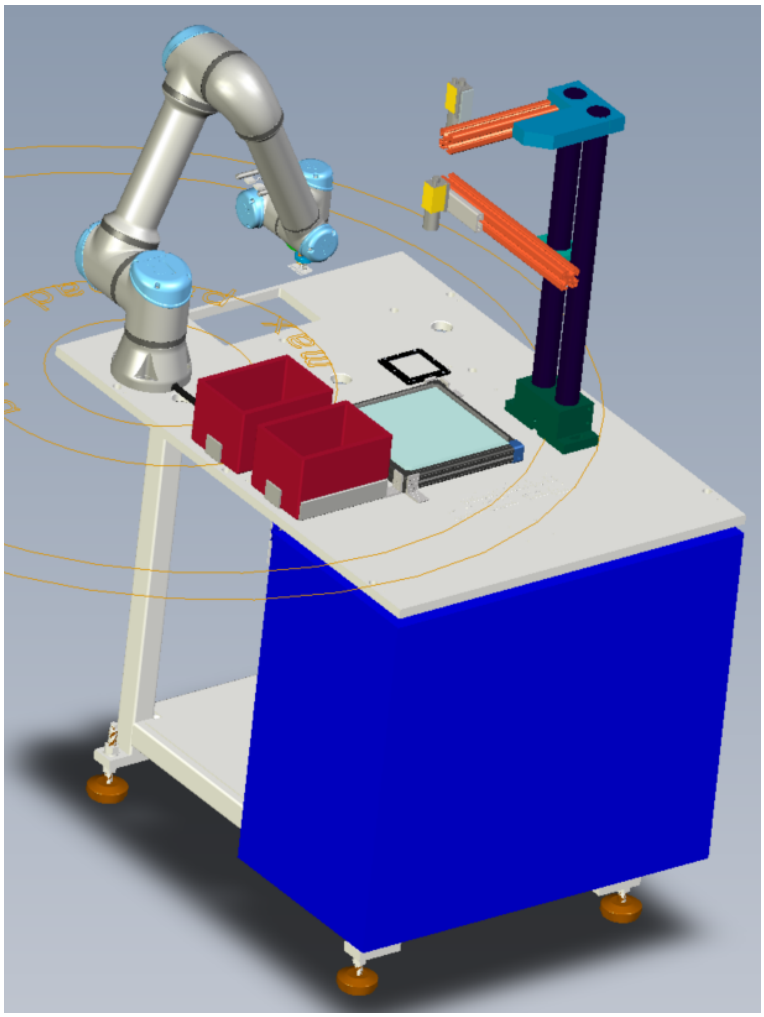
Na obrázku 2.11 je možné vidět okno vývojového prostředí Spreadsheet, které je rozděleno na dvě podokna. Dominantním podoknem je poloprůhledná tabulka na jejímž pozadí je zobrazena fotografie se zvýrazněnými nástroji podobně jako u prostředí EasyBuilder. Průhlednost tabulky je možné měnit dle potřeby tlačítky v liště nástrojů.Na spodní straně tohoto podokna se nachází filmový pás, který plní stejnou funkci jako v prostředí EasyBuilder. Na pravé straně se nachází podokno *Palette*, které je rozděleno do tří karet. První karta *Functions* obsahuje nabídku všech dostupných funkcí, zvolenou funkci lze jednoduše myší přetáhnout na určené místo v tabulce. Po vložení se zobrazí konfigurační okno, kde se nastaví parametry funkce. Poslední karta *TestRun* plní stejnou funkci jako v prostředí EasyBuilder, používá se tedy k vytvoření ověřovacích testů.

3 Praktická část práce

3.1 Dispozice třídicí buňky

V této kapitole se budu věnovat návrhu dispozice buňky. Jak už bylo zmíněno, k dispozici mám stůl s vestavěným rozvaděčem o velikosti pracovní plochy 1100 x 800 mm. Tím je dáno umístění rozvaděče. Dále mám k dispozici manipulátor UR5e, dvě kamery Cognex 1403C, velký a malý backlight.

Manipulátor je umístěn do rohu stolu, aby bylo možné případně v budoucnu nabírat materiál i z nějakého přídavného podavače, například z již zmíněného podavače anyfeed SX240 [18]. Backlight ze kterého se bude nabírat je umístěn diagonálně od manipulátoru a jeho nejvzdálenější hrana je vzdálena 730 mm od manipulátoru.



Obr. 3.1: 3D model třídicí buňky.

Kamera pro nabírání je umístěna nad středem backlightu ve výšce 640 mm od hrany stolu. Kamera pro kontrolu je umístěna nad malým backlightem ve výšce

280 mm, který je umístěn vedle většího backlightu pro nabírání. Před ramenem manipulátoru přibližně ve stejné rovině s větším backligtem je umístěna dvojice odkládacích krabiček, bližší z nich je určena pro dobré díly. Volba pořadí vychází z předpokladu většího podílu dobrých dílů v souboru testovaných výrobků. V pravém dolním rohu se nachází tlačítko nouzového zastavení. Tato poloha nouzového tlačítka byla zvolena, protože je snadno dostupné obsluze při vyprazdňování krabiček, tak i při vkládání dílů pro testování, které předpokládám, že se bude provádět buď ze stejného místa jako vyprazdňování krabiček, nebo ze strany rozvaděče. Rozložení stolu je názorně vidět na obrázku 3.1 na detailním 3D modelu, který vypracoval kolega z oddělení designu. Pro správné umístění bezpečnostních prvků je nutné vypracovat posouzení rizik dle normy ČSN EN ISO 12100 [19], ale to není předmětem této diplomové práce.

3.2 Návrh rozvaděče

Umístění rozvaděče je dáno konstrukcí stolu, nachází se pod deskou stolu a má rozměry 800 x 700 x 400 mm. Na dveřích rozvaděče je umístěn hlavní vypínač elektrické energie. Ten je dle normy ČSN EN 60204-1 ed. 3 [20] umístěn výše než 600 mm od podlahy. Do rozvaděče je nutné umístit a propojit tyto prvky:

- 24 V zdroj
- 2 napájecí jednotky kamer Cognex CIO-MICRO
- Průmyslový switch do rozvaděče Balluff BNI TCP-951-000E028
- Zásuvku do rozvaděče 230 V pro připojení jednotky manipulátoru
- Jističe
- Svorky

3.2.1 Výpočet výkonu 24 V zdroje

Nyní, když jsou známy všechny prvky, které bude rozvaděč obsahovat, je třeba vypočítat velikost potřebného výstupního proud 24V zdroje. Z toho důvodu jsem v manuálech k jednotlivým prvkům, které mají být napájeny 24 V, dohledal jejich maximální příkon a vynesl do tabulky3.1. Hodnoty příkonu jsou v tabulce zaokrouhlené na celé Watty směrem nahoru a s těmito hodnotami i nadále počítám.. Sečtením jednotlivých příkonů jsem dostal minimální výkon 24 V zdroje. Výkon 77 W přibližně odpovídá 3,3 A při 24 V dle vzorce 3.1. Z toho vyplývá, že je možné požít zdroj, který má výstupní proud alespoň 4 A. Já mám k dispozici zdroj Siemens SITOP smart 10 A. Jak už název napovídá, tento zdroj má výstupní proud 10 A a mohu jej tedy použít.

$$P = U * I \qquad [W] \tag{3.1}$$

Tab. 3.1: Tabulka příkonů prvků na 24V [15][16][22][8][12]

Zařízení	Příkon [W]
Napájecí jednotka kamery nakládání	15
Napájecí jednotka kamery kontroly	45
Backlight	21
Switch	3
Vyhodnocovací jednotka tlaku	20
Spínací ventil úpravny vzduchu	3
Suma	**77**

3.2.2 Výpočet odpínacích proudů jističů

V tento okamžik znám všechny zařízení na 230 V, která budou umístěná v rozvaděči, tudíž mohu určit odpínací proudy jističů a průřezy vodičů. Z dokumentace k zdroji Siemens [23] jsem zjistil, že jej doporučují jistit jističem 10 A char. C. Jelikož je za jističem připojena i řídicí jednotka manipulátoru, musím výpočtem ověřit, zda bude stačit jistič 10 A, nebo již budu muset použít jistič 16 A. Budu postupovat stejným způsobem jako při výpočtu výkonu 24V zdroje. Z dokumentace zdroje sem zjistil, že má maximální odběr 2 A při 230 V. Z dokumentace manipulátoru [6] jsem zjistil, že má příkon 570 W, což odpovídá 2,5 A při 230 V. Sečtením těchto dvou proudů získám minimální hodnotu odpínacího proudu jističe 4,5 A. Z toho plyne, že mohu použít výrobcem zdroje Siemens doporučený jistič 10 A char. C.

Abych ochránil 24 V zdroj před zkratem připojím za výstupní svorky jistič s odpínacím proudem 10 A char. C. Velikost odpínacího proudu jsem zvolil podle maximálního proudu zdroje na výstupu.

3.2.3 Výpočet průřezu vodičů

Nyní, když znám i všechny jistící prvky můžu vypočítat průřezy vodičů v rozvaděči. Hlavní přívodní kabel musí mít dle normy [20] průřez nejméně 0,75 mm^2. Dále je nutné vypočítat celkový odběr proudu rozvaděče, z dokumentace jsem zjistil, že manipulátor má maximální příkon 570 W, to odpovídá přibližně 2,5 A a 24 V zdroj má maximální odběr proudu 2A. Sečtením těchto dvou proudů dostanu maximální

odběr proudu rozvaděče tj. 4,5 A. Z tabulky proudového zatížení kabelů [24] ve sloupci *Skupina 2* je možné zjistit, že přívodní kabel s průřezem vodičů 0,75 mm^2 je možné požít pro zatížit až 12 A. Já jsem ovšem použil přívodní kabel 3 x 2,5 mm^2, protože u nás v závodě interní předpis nařizuje použití přívodního vodiče o minimálním průřezu vodičů 2,5 mm^2 a to především z důvodu mechanické odolnosti.

Uvnitř rozvaděče jsem na silové vedení použil vodiče o průřezu 1,5 mm^2, což je minimální hodnota dána interním předpisem. Pro zapojení 24 V napájení jsem použil vodič o průřezu 0,75 mm^2, protože dle tabulky [24] ve sloupci *Skupina 3* je to nejmenší průřez vodiče, který je možný jistit 10 A a zvládne zatížení až 15A, což je dostačující pro 10 A zdroj. Standardně se u nás používá tento vodič i na propojování řídicích obvodů uvnitř rozvaděče, proto jsem to udělal stejným způsobem. Řídicí jednotka manipulátoru je s hlavním rozvaděčem propojena kabelem 25 x 0,25 mm^2, jelikož se jedná pouze o digitální vstupy / výstupy, případně bezpečnostní vstupy výstupy je tento průřez dostatečný, protože digitální výstupy manipulátoru mají maximální proud 1 A a digitální vstupy 15 mA a vodič o průřezu 0,25 mm^2 je možné zatížit 4 A.

3.2.4 Rozmístění prvků v rozvaděči

Vnitřní prostor rozvaděče je rozdělen na čtyři řady, kde je možné na nosnou lištu připevnit součástky. Mezi nosnými lištami se nachází kabelové žlaby. Na vrchní nosnou lištu jsem umístil 24 V zdroj Siemens, switch Balluff a napájecí moduly pro kamery. Na druhou lištu od vrchu jsem umístil svorky svorkovnic *X2, X5, X6, X53 a PE*. Na třetí nosnou lištu jsem upevnil jističe. Skutečné provedení rozvaděče je možné vidět na fotografii v příloze D

3.3 Schéma rozvaděče

Dále bylo nutné nastudovat zapojení všech elektrických prvků a určit jejich vzájemné propojení. V programu WS CAD jsem vytvořil kompletní elektrické schéma třídicí buňky je v příloze C. Na první straně elektrického schématu je znázorněno zapojení od přívodního kabelu přes svorky *X2* dále na hlavní vypínač na jistič a dále na 24 V zdroj a vedení 230 V pokračuje na další list na zásuvku pro manipulátor. Na této straně se ještě nachází zemnící svorky celého manipulátoru označen *PE* a svorky pro napájení 24 V zařízení, které jsou přes jistič připojené k 24 V zdroj Siemens. Svorky jsou dle konvence v našem závodě pojmenovány *X6* pro kladný pól (+24 V) a *X5* pro záporný pól (0 V).

3.3.1 Zapojení kamerového systému

Na druhé straně přílohy C se nachází zapojení kamerového systému pro lokalizaci dílu a na straně třetí systému pro kontrolu dílu. Každý z kamerových systémů je připojen na svoje napájecí a zemnící svorky. Schématické značky pro napájecí jednotku i kameru jsem si musel vytvořit ručně, jelikož pro WS CAD neexistuje knihovna pro tyto výrobky. Kamera samotná je s napájecím modulem spojena dvěma kabely, jeden slouží k napájení a komunikaci s PC či jiným zařízením po rozhraní Ethernet a druhý kabel slouží k ovládání vstupů a výstupů na napájecím modulu. Kamerový systém pro lokalizaci dílu využívá pro komunikaci s manipulátorem protokol EtherNet/IP i digitální vstupy / výstupy, zatímco kamerový systém pro kontrolu dílů využívá pouze digitální vstupy a výstupy. K oběma kamerovým systémům je dále z manipulátoru připojen digitální vstup trigger pro pořízení fotografie a digitální výstup indikující, že je kamerový systém online. U kamerového systému pro lokalizaci je to digitální výstup OUT 0, u kamerového systému pro kontrolu je to digitální výstup OUT 2. U kamerového systému pro kontrolu jsou výsledky testu daného dílu přenášeny digitálními výstupy OUT 0, OUT 1 (OUT 0 v LOG 1 znamená, že díl je špatný a OUT 1 v LOG 1 znamená, že díl je v pořádku)

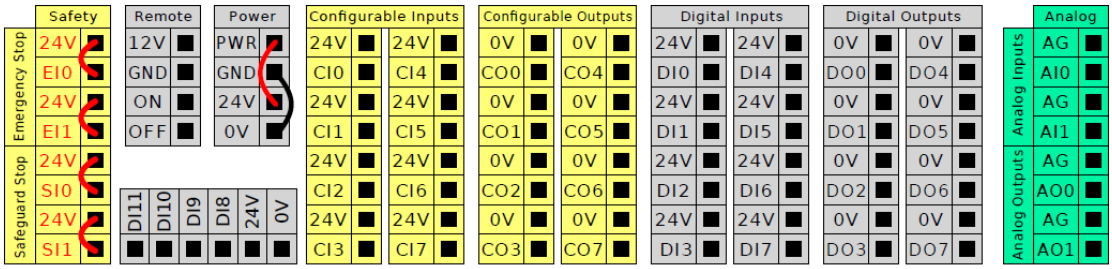
Na čtvrté straně je znázorněno zapojení switche Balluff. Je zde zakresleno připojení napájení a zapojení jednotlivých sítových kabelů, propojení obou kamer s řídicí jednotkou manipulátoru a možnost připojení PC pro programování kamer.

3.3.2 Zapojení tlakových snímačů přísavek

Pátá strana obsahuje schéma zapojení snímačů tlaku a vyhodnocovací jednotky. Je zde zakresleno propojeni snímačů s jednotkou, napájení jednotky a zapojení výstupů z jednotky do řídicí jednotky manipulátoru. K manipulátoru je připojen první kanál a jeho první výstup spolu s kanálem dvě a jeho prvním výstupem. Pomocí manuálu [21] jsem nastavil funkci výstupů tak, že pokud je tlak nižší než nastavená limitní hodnota, tak se příslušný výstup přepne do LOG 1, jinak je v LOG 0. Limit pro indikaci přisátého dílu jsem nastavil zatím experimentálně na -50 kPa. Pokud by se v budoucnu vyskytly problémy s nabíráním dílu bude možné hodnotu upravit. Při dosavadním testování nabírání fungovalo spolehlivě.

3.3.3 Zapojení řídicí jednotky manipulátoru

Na poslední stránce elektrického schématu se nachází nejsložitější část schématu a tou je zapojení manipulátoru. Schématickou značku jsem si opět musel vytvořit ručně. Značka vychází z rozložení svorkovnice na řídicí jednotce manipulátoru, tu je možné vidět na obrázku 3.2. V první řade je zde nakreslena zásuvka pro připojení



Obr. 3.2: Svorky manipulátoru [6]

napájení, dále konektor pro připojení ovládacího panelu a konektor pro připojení Ethernetového kabelu. Dále zde jsou svorky pro bezpečnostní funkce Nouzové zastavení a Bezpečné zastavení s automatickým pokračováním. Ve svorkách pro Nouzové zastavení je připojené tlačítko Nouzového zastavení, které je umístěné na desce stolu. Funkce pro Bezpečné zastavení není využita, proto jsou svorky propojeny pouze vodičem. Následují svorky pro zapnutí / vypnutí vzdáleného ovládání, které nevyužívám.

Vpravo od nich je čtveřice svorek, první dvě PWR a GND jsou výstupní svorky vnitřního 24 V zdroje, který dodá maximálně 2 A. Jestliže je tato hodnota dostatečná, jako v mém případě, je možné svorky PWR a GND připojit na zbylé dvě 24 V a 0 V, které slouží k napájení digitálních vstupů a výstupů. Pokud by ovšem 2 A byly málo, je možné na svorku 24 V a 0 V připojit externí zdroj s maximálním proudem 6 A. Pod svorkovnicemi pro vzdálené ovládání a pro napájení 24 V se nachází svorkovnice digitálních vstupů s číslem 8 až 11 sloužící ke kvadraturnímu kódování sledování dopravníku.

Následují dvě svorkovnice konfigurovatelných vstupů a dvě svorkovnice konfigurovatelných výstupů. Tyto konfigurovatelné vstupy / výstupy je možné použít buď jako bezpečnostní vstupy/výstupy nebo všeobecné vstupy/výstupy, volit mezi režimy je možné v grafickém rozhraní na ovládacím panelu. Svorkovnice pro sledování dopravníku ani konfigurovatelné vstupy / výstupy ve své aplikaci nevyužívám.

Následují dvě svorkovnice obecných digitálních vstupů a dvě digitálních výstupy. Jak je možné vidět v tabulce 3.2 využívám všech osm digitálních vstupů, které jsou přes svorkovnici *X53* připojeny k periferiím. Digitálních výstupů používám sedm a též jsou přes svorkovnici *X53* připojeny k periferiím. Zbývají nevyužité dva analogové vstupy a dva analogové výstupy.

Tab. 3.2: Zapojení digitálních vstupů manipulátoru

Číslo dig. vstupu	Číslo svorky X53	Připojená periferie
DI 0	2	Tlakový spínač na úpravně vzduchu rozpínací kontakt
DI 1	3	Tlakový spínač na úpravně vzduchu spínací kontakt
DI 2	9	Tlakový snímač přísavek 1 kanál
DI 3	10	Tlakový snímač přísavek 2 kanál
DI 4	13	Kamera pro nabíraní výstup 0
DI 5	14	Kamera pro kontrolu výstup 2
DI 6	15	Kamera pro kontrolu výstup 0
DI 7	16	Kamera pro kontrolu výstup 1

Tab. 3.3: Zapojení digitálních výstupů manipulátoru

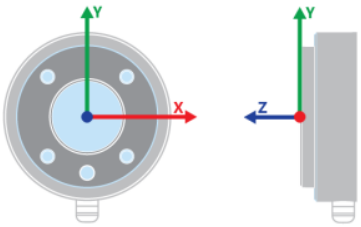
Číslo dig. výstupu	Číslo svorky X53	Připojená periferie
DO 0	5	Pneumatický ventil 1. přísavka - zapnout
DO 1	7	Pneumatický ventil 1. přísavka - vypnout
DO 2	8	Pneumatický ventil 2. přísavka - zapnout
DO 3	11	Pneumatický ventil 2. přísavka - vypnout
DO 4	12	Pneumatický ventil úpravna vzduchu - zapnout / vypnout
DO 5	17	Kamera pro nabíraní trigger +
DO 6	18	Kamera pro kontrolu trigger +

3.4 Programové řešení manipulátoru

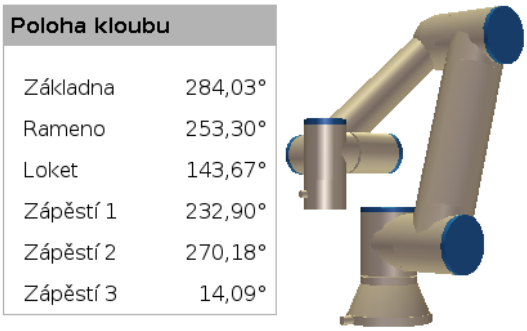
V této kapitole se budu věnovat postupu při programování manipulátoru, popisu instalace a samotného programu. Před zahájením programování je nutné nejdříve nastavit instalaci manipulátoru. Jednak je to proto, aby byl manipulátor před programováním správně nakonfigurován, ale především kvůli bezpečnosti, poněvadž bezpečnostní nastavení je součástí instalace. Při návrhu každého strojního zařízení je nutné vypracovat posouzení rizik dle normy ČSN EN ISO 12100 [19] a dle tohoto posouzení je následně možné nastavit bezpečnostní parametry manipulátoru. Jelikož posouzení rizik není součástí zadání této práce, a protože se jedná o zkušební projekt, který v této podobě nepůjde do provozu, tak posouzení rizik nebylo vypracováno.

3.4.1 Nastavení instalace a bezpečnosti manipulátoru

Z výše zmíněného důvodu jsem použil základní předinstalovanou instalaci a provedl jsem jen drobné úpravy. První parametr, který jsem nastavil je středový bod nástroje (TCP), autor designu chapadla mi poskytl potřebné údaje k vyplnění. Poloha středového bodu je X = 0 mm, Y = 0 mm, Z = 64 mm vzhledem ke středovému bodu příruby. Souřadný systém příruby je znázorněn na obrázku 3.3. Dalším důležitým parametrem chapadla je jeho váha, která činí 640 g a jeho těžiště, které je v bodě CX = 0 mm, CY = 0,22 mm, CZ = 19 mm. Tento středový bod je uložen pod názvem *TCP_prisavky*. Ostatní parametry zůstaly beze změn, ovšem až na pozici polohy domů. Úhly jednotlivých kloubů a grafickou vizualizaci ramene v domovské poloze je možné vidět na obrázku 3.4.



Obr. 3.3: Souřadný systém příruby



Obr. 3.4: Domovská pozice manipulátoru

V bezpečnostním nastavení jsem upravil pouze bezpečnou výchozí pozici, která je stejná jako pozice domů. Mezní hodnoty manipulátoru jsou zobrazeny na obrázku 3.5. Poloha kloubů není nijak omezena, to znamená, že se mohou pohybovat na intervalu [-363 363]°. Maximální rychlost kloubů je 191 °/s. Žádné roviny nejsou definovány, proto nemůže být definován ani směr nástroje. Poloha nástroje je nastavená pro dva prvky, první základní *Tool Flange* je umístěn v bodě [0, 0, 0] mm, druhý *Prisavky* je umístěn v bodě [0, 0, 64] mm, rádius je u obou nástrojů roven nule.

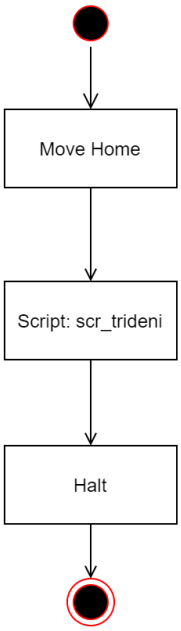
Všechny konfigurovatelné vstupy jsou nevyužity s výjimkou vstupu nula a jedna, ty jsou použity pro resetování bezpečného zastavení. V kartě hardware je nastaveno, že k manipulátoru musí být připojen přenosný terminál, jinak se manipulátor nespustí. Možnost třípolohového spínače je vypnuta. To je z nastavení bezpečnosti vše. Další rozbalovací nabídkou jsou prvky, zde jsou nastavené pouze základna a nástroj. V rozbalovací nabídce *Sběrnice* a kartě *EtherNet/IP* je tento protokol povolen a činnost programu při ztrátě spojení je nastavena na *Žádná*. Instalace je uložena pod názvem *inst_prisavky*.

Mezní hodnota	Běžný
Výkon	300 W
Hybnost	25,0 kg m/s
Čas zastavení	400 ms
Brzdná vzdálenost	500 mm
Rychlost nástroje	1500 mm/s
Síla nástroje	150,0 N
Rychlost lokte	1500 mm/s
Síla lokte	150,0 N

Obr. 3.5: Mezní hodnoty manipulátoru

3.4.2 Hlavní program manipulátor

V okamžiku, kdy je nastavená instalace, je možné začít programovat. První instrukcí v mém programu s názvem *prog_trideni* je *Movej* do pozice *Pozice_Home*, která je stejná jako bezpečná pozice nastavená v instalaci. Tento pohyb se využívá pouze při spuštění programu, kdy před spuštěním samotného programu je uživatel vyzván, aby buď držel tlačítko *Přesunout robota do pozice: 3: Pozice_Home* a manipulátor se sám pokusí přemístit do pozice. Nebo uživatel může manipulátor do pozice navést ručně, to je důležité zejména pokud manipulátor sám nedokáže najít cestu, nebo pokud jsou v blízkosti manipulátoru překážky do kterých by mohl bourat. To, že uživatel musí držet tlačítko při navádění manipulátoru je důležitý bezpečnostní prvek, poněvadž nutí uživatele být u manipulátoru a tudíž si snáze všimne případné kolize a pustí tlačítko čímž zastaví manipulátor. Další volanou instrukcí je instrukce *Script*, která volá soubor s názvem *scr_trideni.script*. Funkci skriptu popíši v následující kapitole. Poslední instrukce je instrukce *Halt*. Pokud program dojde na tuto instrukci jeho vykonávání je zastaveno a rameno manipulátoru se zastaví. Struktura hlavního programu je zobrazena na obrázku 3.6.



Obr. 3.6: Vývojový diagram hlavního programu.

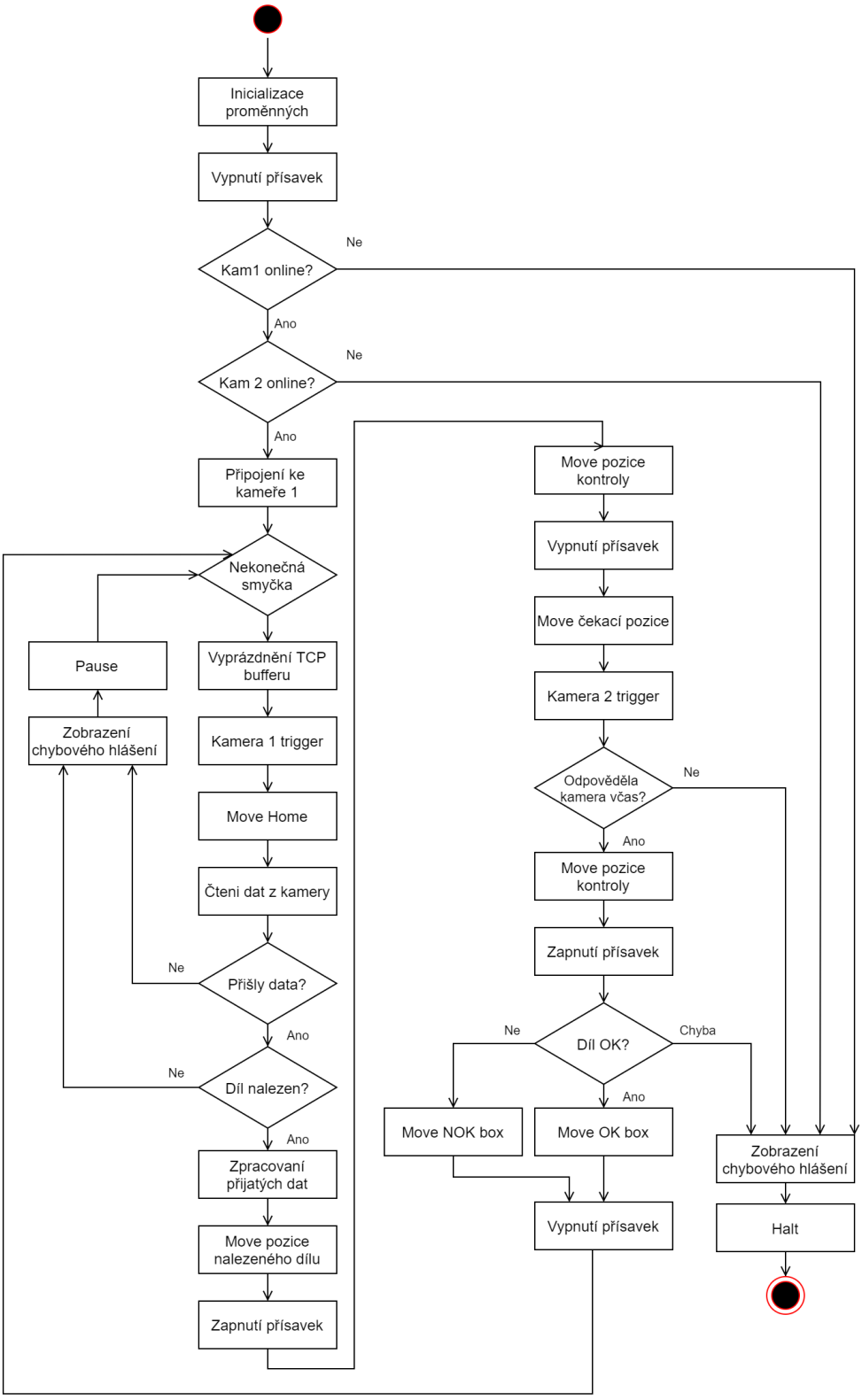
3.4.3 Skript

Jak již bylo dříve zmíněno, jelikož potřebuji obsluhovat komunikaci po sběrnici EtherNet/IP musím využít skriptovací jazyk URScript. Pomocí tohoto skriptovacího jazyku je naprogramována prakticky celá funkcionalita manipulátoru. Na obrázku 3.7 je znázorněn vývojový diagram hlavních částí programu. Vývojový diagram je z důvodu přehlednosti zjednodušen a neobsahuje ošetření chyb při volání funkcí, kdy při výskytu chyby je zobrazeno chybové hlášení a vykonávání programu je zastaveno.

Výpis 3.1: Příklad definice pozice Home

```
1 global Home = get_inverse_kin(p[-0.200, 0.250, 0.170, -3.14, 0, 0])
```

V úvodu skriptovacího programu je umístěna definice pozic, inicializace proměnných a jsou zde nastaveny konstanty pro vstupy a výstupy. Pozice je zadaná ve tvaru X, Y, Z, RX, RY, RZ (kde X, Y, Z jsou souřadnice TCP a RX, RY, RZ je natočení TCP) v souřadném systému základny do funkce *get_inverse_kin*, tato funkce vypočítá inverzí kinematickou úlohu a výsledek uloží do proměnné. Inverzní kinematická úloha vypočítá polohu kloubů dle zadaného koncového bodu ramene. Příklad definice polohy Home je zobrazen ve výpise 3.1. Jelikož do funkcí pracujících se vstupy / výstupy se jako parametr vkládají číselné identifikátory vstupů / výstupů, tak jsem pro přehlednější čtení kódu vytvořil proměnné vhodně pojmenované a vložil do



Obr. 3.7: Vývojový diagram skriptu.

nich příslušné identifikátory oněch vstupů / výstupů. Nyní je možné jako parametr funkce pro práci se vstupy / výstupy vložit proměnnou, z jejíhož názvu je zřetelné k čemu příslušný vstup / výstup slouží.

Následuje definice čtyř mnou vytvořených funkcí. První funkce *F_Pris_Off* slouží k vypnutí přísavek. Parametrem funkce je číslo, které určuje jakou přísavku chceme vypnout. Výchozí hodnota je 0, to znamená, že chceme vypnout obě přísavky, hodnota 1 nebo 2 značí vypnutí 1. resp. 2. přísavky, při jiné vstupní hodnotě funkce končí a vrací hodnotu -1. Filozofie všech tří případů podle zadaného parametru je stejná, pouze je rozdíl v tom jaké výstupy jsou ovládány a vstupy vyhodnocovány. Proto bude funkce popsána pouze pro variantu, kdy se vypínají obě přísavky. Nejdříve se aktivují výstupy pro vypnutí přísavek, následuje 200 ms pauza na odsátí přísavek. Po pauze se vyhodnocují vstupy od tlakových snímačů, pokud jsou oba v LOG 0, tak přísavky nejsou již přisáté a je možné výstupy přísavek deaktivovat. Tím funkce končí a vrací hodnotu nula. Pokud se přísavky neodsají do 5000 cyklů funkce končí a vrací hodnotu -2. Část zdrojového kódu je znázorněna ve výpisu 3.2, kompletní kód funkce lze najít v příloze E. Druhá funkce *F_Pris_On* slouží k zapnutí přísavek a její provedení je obdobné jako u funkce vypnutí přísavek jen s tím rozdílem, že se ovládají jiné výstupy. Funkce končí a vrací hodnotu 0, když vstupy snímačů podtlaku mají hodnotu LOG 1.

Třetí funkce slouží k odeslání triggeru (povelu k pořízení snímku). Vstupním parametrem je opět číslo, je-li funkce zavolána s parametrem 1, kamera pro nabírání dílu pořídí snímek, pakliže se zavolá funkce s parametrem 2, snímek pořídí kamera pro kontrolu dílu. Pokud je parametr jiné číslo než 1 či 2 funkce končí s návratovou hodnotou -1. Jelikož jazyk URScript neobsahuje funkci pro vyslání pulzu na výstup, tak funkce nejdříve aktivuje výstup dle zadaného parametru, po uplynutí 500 ms tentýž výstup deaktivuje a vrací hodnotu 0. Tato funkce nese název *F_Trigger_Kam* a její kód je dostupný v příloze E na řádku 160.

Poslední moji vytvořenou funkcí je funkce na zpracování přijatých dat z kamery po sběrnici EtherNet/IP. Parametrem funkce s názvem *F_Pars_dat* je celý řetězec přijatých dat. Pro přenos dat jsem si zvolil formát "A;X;Y;B", kde A je číslo, které značí jestli byl díl nalezen (A = 1), nebo nebyl nalezen žádný díl (A = 0). X je x-ová souřadnice nalezeného dílu, Y je y-ová souřadnice nalezeného dílu a B je úhel natočení dílu. Celý řetězec přijatých dat může vypadat například nějak takto: "1;300.983215;450.242828;49.721786\r\r\n". Jelikož se délka řetězce může lišit, tak je nejdříve nutné zjistit jeho délku. Následně postupně si vyhledám v řetězci středníky a jejich pozice uložím do proměnných. Poté mohu z řetězce separovat souřadnice a úhel natočení. Dále je nutné jednotlivé separované řetězce převést na číslo. Jelikož kamera odesílá úhel ve stupních a manipulátor pracuje s radiány, tak je nutné přijatý

Výpis 3.2: Část zdrojového kódu funkce pro vypnutí přísavek

```
# Funkce na vypnuti prisavek
# a = 0 - Vypne obe prisavky  (vychozi hodnota)
# a = 1 nebo a = 2 - Vypne prisavku 1 nebo 2
def F_Pris_Off (a = 0):
    Time_Out=0
    if a == 0 :
        #  Zapnuti vystupu pro vypnuti prisavek
        set_standard_digital_out(O_Valec1_Off, True)
        set_standard_digital_out(O_Valec2_Off, True)
        sleep(0.2)
        # Cekani na odsati
        while (not((get_standard_digital_in(I_Tlak_snim_Ch1)  and
            get_standard_digital_in(I_Tlak_snim_Ch2)) ==  False)):
            sync()
            Time_Out=Time_Out+1
            if Time_Out > 5000:
                return -2 # Chyba pri odsati
            end
        end
        # Vypnuti vystupu pro vypnuti prisavek
        set_standard_digital_out(O_Valec1_Off, False)
        set_standard_digital_out(O_Valec2_Off, False)
        return 0 # Vse OK
    elif a == 1 :
        ...
    elif a == 2 :
        ...
    else :
        return -1 # Neplatny vstupni parametr
    end
end
```

úhel ještě převést na radiány. Kamerový systém odesílá souřadnice v milimetrech, ale manipulátor pracuje v metrech, proto se ještě hodnoty souřadnic podělí tisícem. Nakonec se souřadnice v metrech a úhel v radiánech vloží do pole, které následně funkce vrací. Zdrojový kód funkce je vypsán ve výpise 3.3.

Výpis 3.3: Zdrojový kód funkce pro parsování dat

```
# Funkce pro parovani prijatych dat
def F_Pars_dat (data):
    delka = str_len(data)
    strednik1 = str_find(data,";")
    strednik2 = str_find(data,";",(strednik1+1))
    strednik3 = str_find(data,";",(strednik2+1))
    Xpos = str_sub(data,(strednik1+1),(strednik2-strednik1-1))
    Ypos = str_sub(data,(strednik2+1),(strednik3-strednik2-1))
    angle = str_sub(data,(strednik3+1),(delka-strednik3-1-3))
    # Prevod retezce na cislo
    iXpos = to_num(Xpos)
    iYpos = to_num(Ypos)
    iangleD = to_num(angle)
    # Prevod stupnu na Radiany
    iangleR = d2r(iangleD)
    # Prevod na metry
    iXpos = iXpos/1000
    iYpos = iYpos/1000
    pole = [iXpos,iYpos,iangleR]
    return pole
end
```

Po definicích funkcí následuje volání funkce pro vypnutí přísavek, pokud by funkce proběhla neúspěšně, došlo by k zobrazení oznámení uživateli a následně k zastavení vykonávání programu. Toto umístění volání funkce ze voleno pro eliminaci situace, která by mohla nastat při nekorektním ukončení předchozího běhu programu, kdy by mohly přísavky zůstat zapnuté a dokonce držet přisátý díl. Pokud by tato situace nastala, tak díky předchozím příkazům se nyní rameno nachází v pozici Home a zde je možné případně přisátý díl bezpečně odhodit, aniž by narušil běh aktuálně spuštěného procesu testování.

Následně kontroluji, zda jsou oba kamerové systémy připraveny, a tudíž ve stavu online, což signalizují stavem LOG 1 na vstupu manipulátoru. Pokud některá z kamer není připravena dojde k ukončení programu a uživatel je o této skutečnosti informován hlášením. Situace, kdy kamerový systém není online může nastat zejména při jeho spouštění, nebo při poruše.

Sběrnici EtherNet/IP využívá pro komunikaci prostřednictvím Ethernetu standardní TCP/IP protokol, který je určen k aplikacím typu žádost - odpověď mezi dvěma prvky sítě [29]. Proto mohu ke komunikaci využít funkci *socket_open* obsaženou v jazyce URCScript. Tato funkce otevře TCP/IP komunikační soket, jako parametr funkce se vkládá IP adresa a port serveru a pojmenování otevřeného soketu. Funkce vrací hodnotu True pokud se podařilo komunikaci navázat a False pokud navázání selhalo. Na kamerovém systému pro nabírání běží TCP/IP server s IP adresou: 192.168.56.3, na portu 3000. Pokud se připojení ke kameře nezdaří, program končí a uživatel je informován.

Pokud proběhne připojení ke kamerovému systému v pořádku, tak program vstoupí do cyklu while, který funguje jako nekonečné smyčky. Na začátku nekonečné smyčky bylo nutné vyprázdnit buffer TCP/IP protokolu, abych měl jistotu, že budu mít aktuální data. Programovací jazyk URScrip ovšem neobsahuje funkci na vyprázdnění bufferu, tudíž jsem si musel pomoci cyklem while a funkcí *socket_read_string*. Funkce *socket_read_string* vyčte data ze soketu, který je vložený jako parametr a vrátí data jako řetězec. Druhý parametr funkce je čas v sekundách, který bude funkce čekat na přečtení dat. Pokud do zadaného času data nepřijdou funkce vrací prázdný řetězec. Pro vyprázdnění bufferu se tedy pokusím přečíst z něj data, pokud jsem nějaké přečetl, tak se pokouším číst data tak dlouho dokud se nevrátí prázdný řetězec. Zdrojový kód pro otevření soketu a následné vyčtení dat z něj je zobrazeno ve výpise 3.4.

Výpis 3.4: Zdrojový kód otevření soketu a čtení dat z něj

```
1 #Pripojeni ke kamere
2 if (socket_open("192.168.56.3",3000,"socket_kamera") == False):
3     popup("Nepodarilo␣se␣pripojit␣ke␣kamere")
4     halt
5 end
6
7 #Cteni dat ze socketu
8 global Sock_Data=socket_read_string("socket_kamera",timeout=5)
```

Nyní můžu předpokládat, že žádná data nečekají na přečtení a budu mít tedy k dispozici pouze data aktuální.Následně je odeslán trigger do kamery pro nabírání. Jelikož zpracování snímku a odeslání souřadnic trvá přibližně 1,5 s, tak je manipulátor v mezičase přesouván do polohy Home. To má samozřejmě uplatnění až od druhého testovacího cyklu, kdy je trigger vyslán v okamžiku, když je rameno manipulátoru ještě nad vykládací bednou. Kamera fotografii zpracovává během přesouvání ramene a tím se odstranilo čekání v pozici Home na příchod dat. Jakmile

manipulátor dojede do polohy Home, jsou čteny data ze soketu. Jestliže data nabyla přijata, uživatel je o této skutečnosti informován a běh programu je pozastaven, dokud uživatel nerozhodne, zda se vykonávání programu ukončí, nebo jestli se bude pokračovat ve vykonávání programu od začátku testovacího cyklu.

Pokud byl řetězec dat přijat, provede se vyhodnocení zda byl nalezen díl či nikoliv na základě první číslice přijatého řetězce. Pokud díl nebyl nalezen opět je o tom obsluha informována a program čeká na jeho rozhodnutí, zda ukončit program nebo pokračovat od začátku testovacího cyklu. Jestliže byl díl nalezen, přijatý řetězec dat je předán funkci *F_Pars_dat*. Z výsledného vektoru po zpracování dat jsou dále rozděleny jeho složky do proměnných.

Nyní jsou k dispozici veškeré potřebné údaje pro nabrání dílu. Nejdříve manipulátor zamíří nad nalezený díl podle souřadnic X a Y, následně zamíří směrem dolů na dotek s dílem a zároveň přetočí přísavky do správného úhlu. Aby se osa třetího zápěstí nemusela otáčet někdy až téměř o 180°, tak je algoritmus navrhnut tak, že díl může být nabrán i otočený o 180°. Kamerový systém kontroly je připraven na to, že díl může být ve dvou orientacích. Tím je ušetřen čas cyklu a zároveň zjednodušen program. Po 200 ms jsou zapnuty přísavky a manipulátor se již s nabraným dílem přesune do výšky 10 cm nad desku stolu odkud směřuje na pozici kontroly dílu. Po 200 ms jsou přísavky vypnuty, kontrolovaný díl zůstává na backlightu a manipulátor je přesunut na pozici, kde nepřekáží kamerovému systému kontroly. Následně manipulátor čeká až kamerový systém nastaví příslušný vstup na LOG 1 (podle toho je-li díl vadný nebo ne) a druhý zůstane v LOG 0. Pokud se tak nestane do 5000 cyklů, program je ukončen. Pakliže vše proběhne v pořádku, hodnoty obou vstupů jsou uloženy do proměnných na základě kterých bude následně díl zapnutím přísavek nabrán a odložen do boxu pro dobré nebo špatné díly. Po té se program vrací na začátek nekonečného testovacího cyklu.

Rychlosti a zrychlení kloubů při jednotlivých pohybech jsou ve většině případů při pohybu *Movel* je $a = 2m/s^2$ a $v = 12m/s$ a při pohybu *Movej* $a = 10rad/s^2$ s $v = 20rad/s$. Jen u pohybu nabírání nalezeného dílu je rychlost a zrychlení $a = 5m/s^2$ a $v = 5m/s$, protože během testování se ukázalo, že pokud byla rychlost vyšší, docházelo k nechtěnému posuvu dílu před nabráním. Tyto rychlosti jsem určil tak, aby se manipulátor přemisťoval dostatečně rychle, ale zároveň nebyl příliš nebezpečný. Jelikož nebylo provedení posouzení rizik, nevím jaké rychlosti jsou ještě bezpečné. Jelikož není tlak na rychlost testování rychlosti mohou být z preventivních důvodů nižší.

3.5 Programové řešení kamerového systému

V této kapitole se budu věnovat postupu při programování kamerových systémů a sjednocení souřadných systému manipulátoru s kamerovým systémem lokalizace.

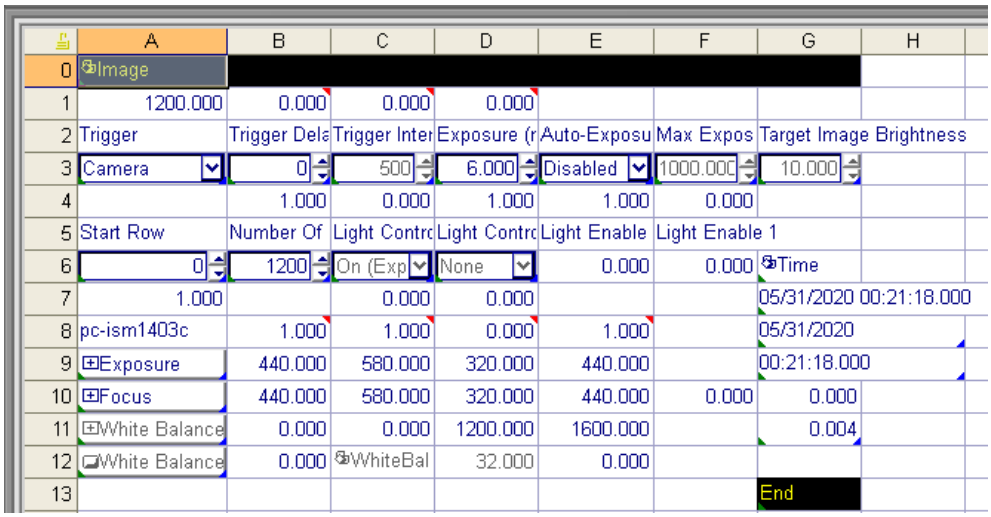
3.5.1 Kamerový systém lokalizace

Jelikož kamerový systém lokalizace využívá složitější funkce, zejména pak funkce na sjednoceni souřadných systému manipulátoru s kamerovým systémem, musel jsem použít vývojové prostředí Spreadsheet. Nejdříve bylo nutné nastavit základní parametry obrázku. V políčku A0 se nachází nejdůležitější funkce *AcquireImage*, která specifikuje parametry pro pořízení snímku a jeho přenosu do procesní paměti. Funkce obsahuje řadu nastavitelných parametrů, pro nás je důležitý parametr jakým způsobem se bude provádět trigger. Jelikož chci využít vstupy určené přímo pro tuto funkci zvolil jsem možnost *Camera*. Existují další způsoby provedení triggeru, například ten kdy kamerový systém bude neustále pořizovat a vyhodnocovat snímek, nebo může přijít povel po některé ze sběrnic. Dále jsem nastavil čas expozice na 6 ms, proto na fotografii je pozadí světle modré místo bílého. Je to z toho důvodu abych byl schopen odlišit pozadí od případné saturace, která by mohla vzniknout při zpracovávání obrázku. Další parametry jsem nechal v základním nastavení. Volání funkce je zobrazeno ve výpise 3.5 a část tabulky, která s touto funkcí souvisí je na obrázku 3.8.

Výpis 3.5: Volání funkce AcquireImage

```
1 AcquireImage($B$1,1,$D$3,$E$3,$F$3,$G$3,$B$9,$C$9,$D$9,$E$9,$A$6,
2 $B$6,$C$6,$E$6,$F$6,0,0,128,33,0,0,"","",$C$1,$D$1,$B$10,$C$10,
3 $D$10,$E$10,$F$10,$G$10,$B$11,$C$11,$D$11,$E$11,1,40,0,0,1)
```

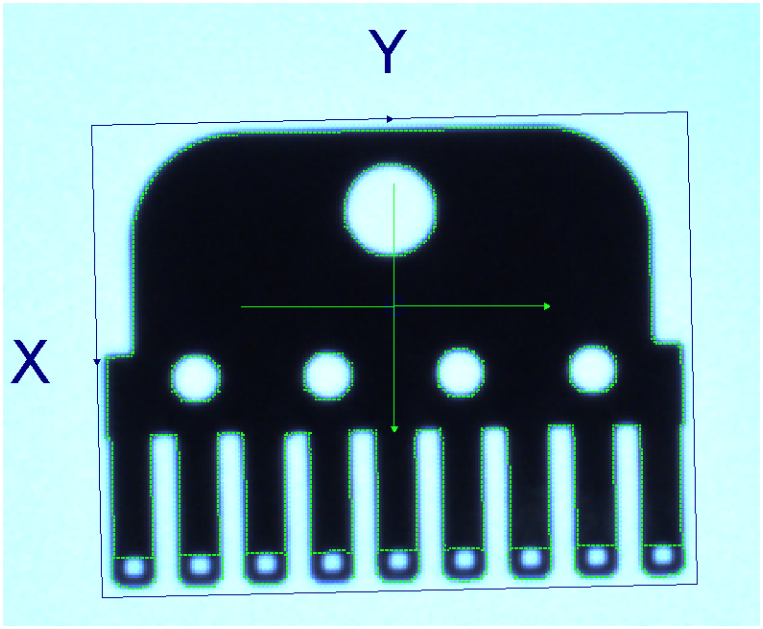
Funkce pro vyhledání dílu se nazývá *FindPatterns* a skládá ze dvou částí. První část funkce extrahuje oblast nebo vzor založený na plošné nebo hranové detekci, zatímco druhá část funkce hledá obraz dříve natrénovaného modelu. Nejdříve na referenčním obrázku vybereme vzor, který chceme vyhledávat, funkce se vzor naučí a vyhledává jej v obrázku. Učený vzor má tvar obdélníku, který se snažím přesně umístit kolem referenčního dílu, tak aby ohraničení vzoru bylo rovnoběžné s hranami dílu. Pro naučení používám hranový model se střední hrubostí a přesností. Je to kompromis mezi rychlostí zpracování a přesností výsledku. Jelikož funkce hledání vrací souřadnice středu hledaného dílu, kde jsou vylisovány otvory, přísavky by se tam nebyly schopny přisát. Z toho důvodu jsem nastavil offset -21 řádků a tím posunul hledaný bod výš, kde už se přísavky přisají. Na obrázku 3.9 je znázorněn



Obr. 3.8: Kompletní parametry funkce *AcquireImage*.

naučený model se zvýrazněným hranovým modelem a nalezeným bodem. Další důležité parametry, které je třeba nastavit jsou *AngleRange* a *Thresh: Accept*. Prvně zmíněný parametr udává maximální úhel natočení dílu, který bude ještě nalezen. V mém případě chci hledat díly, které jsou otočené jakkoli, čili zadaná hodnota je 180°. Druhý parametr určuje minimální přijatelné skóre vzoru. Hodnotu jsem po řadě pokusů nastavil na 60%, protože díky tomu dokáže vyhledat i částečně překryté díly. Při nižší hodnotě byl sice díl nalezen, ale nepodařilo se ho přísavkami nabrat. Nyní z této funkce musíme získat informace, které budeme dále potřebovat. Funkcemi *GetRow, GetCoil, GetAngle, GetScore* získám řádek, sloupec, úhel a skóre nalezeného dílu. Dále bylo nutné získat informaci, zda se díl podařilo vyhledat, k tomu slouží funkce *GetNFound*, která vrátí 1 pokud se vyhledání podařilo a 0 pokud nikoli. Všechny tyto funkce mají jako parametr buňku s funkcí *FindPatterns*.

Nyní, když mám naprogramované pořízení obrázku a vyhledání dílu, musím souřadnice udané v pixelech převést na souřadnice manipulátoru. Postupu sjednocení souřadnicových systému kamery a manipulátoru se budu podrobněji věnovat v další kapitole. Pro tuto chvíli postačí, že po kalibraci dostanu hodnotu souřadnic X a Y v souřadném systému základny manipulátoru a natočení dílu ve stupních od osy X. V tento okamžik již nastal čas odeslat souřadnice do manipulátoru. V buňce A27 je vložená funkce *TCPDevice("",3000,0,0,1000,0)*, která slouží ke spuštění TCP serveru. Prvním parametrem funkce má být IP adresa serveru pokud by kamera byla klient, pokud IP adresa zůstane nevyplněná, tak se kamera chová jako server. Druhým parametrem je číslo portu TCP/IP serveru. Třetí parametr určuje, že se jedná o TCP/IP protokol (jiný protokol není na výběr), čtvrtý parametr definuje formát datového řetězce, nechal jsem základní nastavení. Pátý a šestý parametr nemají pro server význam. Buňka B27 obsahuje funkce *ReadDevice*, která má jako parametr



Obr. 3.9: Naučený hranový model.

odkaz na buňku se serverem a slouží ke čtení dat ze serveru. Jelikož neočekávám, že by na server chodily nějaké zprávy není tato funkce nutná, slouží pouze pro ošetření situace kdyby přece jen nějaká data dorazila, nebudou tak plnit buffer. Ovšem důležitá funkce se nachází v buňce C27 a je to funkce *WriteDevice*, která slouží k odesílání dat přes TCP server. Funkce má tři parametry, jako první parametr se vkládá spouštěč, v mém případě buňka A0 (pořízení obrázku), druhým parametrem je odkaz na buňku se serverem a poslední parametr je řetězec, který se bude odesílat. Pro formátování řetězce jsem použil funkci *FormatString* a v jejím dialogovém okně nastavil oddělovač na středník a vložil odkazy na čtyři buňky, které chci odeslat. První buňka je F17, čili informace o výsledku hledání, další parametry potom jsou B23, C23 a D23 což jsou souřadnice a úhel nalezeného dílu. Ovšem je tu problém v tom, že pokud díl není nalezen, tak funkce vrátí hodnotu #ERR, ale já potřebuji, do manipulátoru odeslat řetězec obsahující nulu. Z toho důvodu jsem v buňce D26 použil příkaz *if* a pokud byl díl nalezen, do této buňky se nakopíruje dříve nakonfigurovaný řetězec, jestliže se díl nepodaří najít vloží se do této buňky 0. Tento řetězec je poté vložen do funkce *WriteDevice* a odeslán do manipulátoru.

Poslední funkce, kterou bylo nutné implementovat, má za úkol na výstupu číslo 0 držen úroveň LOG 1 pokud je kamera ve stavu online. K tomu jsem použil funkci *GetOnline*, která vrací hodnotu 1 je-li kamera online a hodnotu 0 je-li kamera offline. Tuto funkci jsem umístil do buňky T33, hodnota v této buňce se překlápí na výstup. Funkce *GetOnline* si vytvořila pomocnou funkce *Event*, která je umístěna v buňce R33 a je předávána jako parametr funkci *GetOnline*. Funkce *Event* slouží k

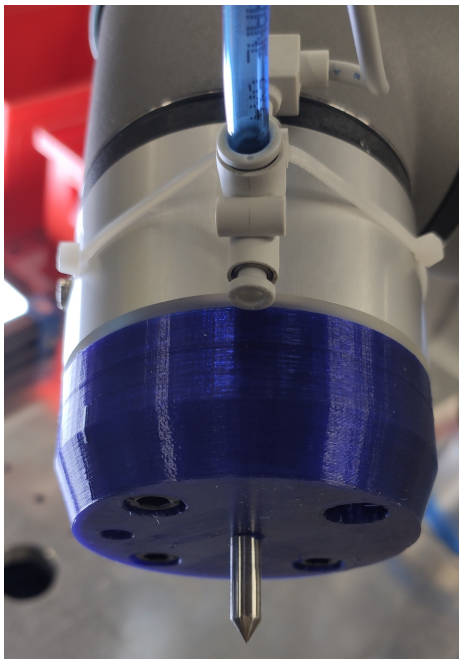
obnovení tabulky při změně stavu online/offline. Část inspekce pojmenovaná *Overall Pass/Fail* slouží jako počítadlo inspekcí. Část umístěná pod ním slouží k vykreslení zeleného resp. červeného rámečku kolem fotky podle toho zda byl díl nalezen resp. nebyl nalezen. Část kódu *Read the discrete inputs* jak již název napovídá slouží k obsluze vstupů. Kompletní zobrazení inspekce se nachází v příloze F.

3.5.2 Sjednocení souřadných systémů kamery a manipulátoru

Jsou dvě možnosti, jak sjednotit souřadné systémy. První z nich, kterou jsem v této práci implementoval je, že se sjednocení provede v kamerovém systému a do manipulátoru se odesílají přímo cílové souřadnice. Při druhém způsobu se v manipulátoru vytvoř nový souřadný systém, který koresponduje se souřadným systémem kamery. Vybral jsem první možnost, protože pokud by se vyměnila nebo přemístila kamera, musel by se upravovat jak program v manipulátoru tak v kameře. Jelikož je plán projekt rozšířit o automatický podavač, tak je úprava kamery pravděpodobná.

Jak jsem již zmínil, sjednocení systémů neboli kalibraci provádím v kameře. Použil jsem pro to funkci *CalibrateAdvanced*, tato funkce vytváří 2D transformaci pro převod mezi souřadnými systémy pixelů a reálného světa (v našem případě manipulátoru). Transformaci provádí na základě jednoho až třiceti dvou párů zadaných bodů a bere při ní v úvahu lineární i nelineární zkreslení objektivu. Jako parametry funkce se vkládají páry bodů, kdy první souřadnice z páru je v pixelech a druhá například v milimetrech. Pokud to aplikuji na můj případ, tak si potřebuji určit souřadnice nějakého bodu na obrázku a k němu odpovídající souřadnice manipulátoru. Abych byl schopen souřadnice těchto bodů získat, musel jsem si navrhnout přípravek. Kamera obsahuje mimo jiné i kalibrační funkci *CalibrateGrid*, jedná se o nejpřesnější kalibrační funkci, která ke kalibraci využívá kalibrační mřížku. Ovšem jelikož je proces kalibrace automatizován a provádí se na kalibrační mřížce o známých vlastnostech, není možné ji použít na sjednocení souřadných systémů. Využil jsem ale možnost, že funkce nabízí vytisknout kalibrační mřížku, kterou využiji. Šachovnicovou kalibrační mřížku, o velikosti čtverce 10 mm, jsem si vytiskl a připevnil na průhledné polykarbonátové sklo, tak by hrana skla byla rovnoběžná s hranou mřížky. polykarbonátové sklo má takový tvar, že přesně zapadne do kovového rámu který obklopuje průhlednou plochu backlightu, tím je zajištěno, že kalibrační mřížka je umístěna přesně souměrně s backlightem, ale hlavně půjde opakovaně umístit na přesně stejné místo. Nyní mám v zorném poli kamery řadu bodů, jež můžu využít ke kalibraci. Body představují rohy jednotlivých čtverců. V tento okamžik jsem řešil problém, jak přesně určit souřadnice manipulátoru ve zvoleném bodě kalibrační mřížky. Vymyslel jsem tedy, že místo chapadla s přísavkami se na přírubu manipulátoru připevní špičatý trn, který bude mít špičku hrotu umístěnou přesně

mezi přísavkami (ve středu příruby). A poté budu schopen rameno manipulátoru přesně navést na zvolený kalibrační bod. Opět mi pomohl kolega z oddělení designu a toto chapadlo mi zrealizoval. Kalibrační trn je zobrazen na obrázku 3.10. Tím je kalibrační přípravek hotov.

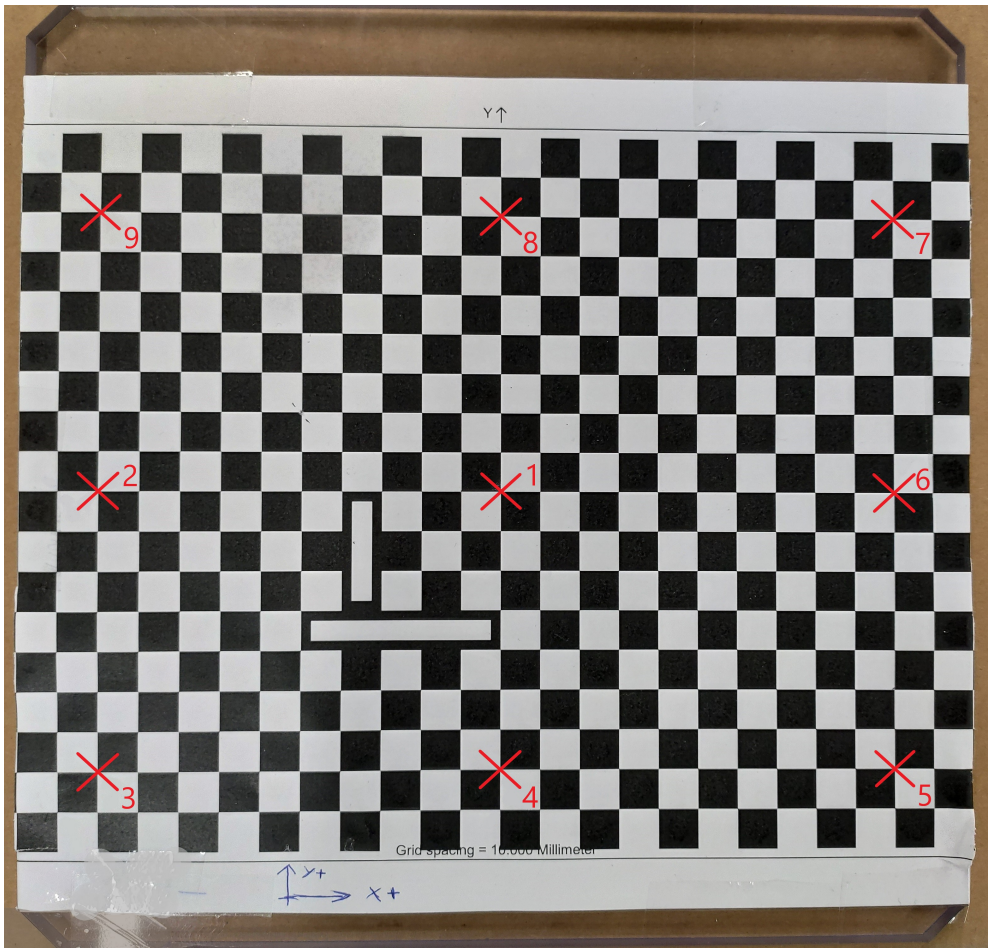


Obr. 3.10: Kalibrační hrot.

Na kalibrační mřížce jsem si vybral celkem devět rovnoměrně rozmístěných bodů, na kterých budu provádět kalibraci. Na obrázku 3.11 je zobrazena kalibrační mřížka s vyznačenými a očíslovanými body. Počet devíti bodů se ukázal jako dostatečný, pokryji s nimi celý nabírací prostor a s přesností nabírání nebyl žádný problém. Pokud bych přidal více bodů pravděpodobně by se tím zpomalilo již tak dlouhé zpracování obrázku a zvýšení přesnosti by nemělo moc velký význam. Pro určení souřadnic bodů kamery v pixelech jsem opět využil funkci *CalibrateGrid*, která v dialogovém okně v záložce *Pose* automaticky vyhledá všech 368 bodů a vypíše jejich souřadnice do tabulky. Z tabulky jsem si vypsal souřadnice dříve zvolených devíti bodů. Na obrázku 3.12 je znázorněno určování souřadnic bodu.

Souřadnice manipulátoru jsem získal tak, že jsem rameno manipulátoru přemístil do pozice, aby byla špička hrotu přesně v určeném bodě. Názorně je to vidět na obrázku 3.13. Následně jsem si zapsal souřadnice v souřadném systému základny. Souřadnice kalibračních bodu jsem vynesl do tabulky 3.4.

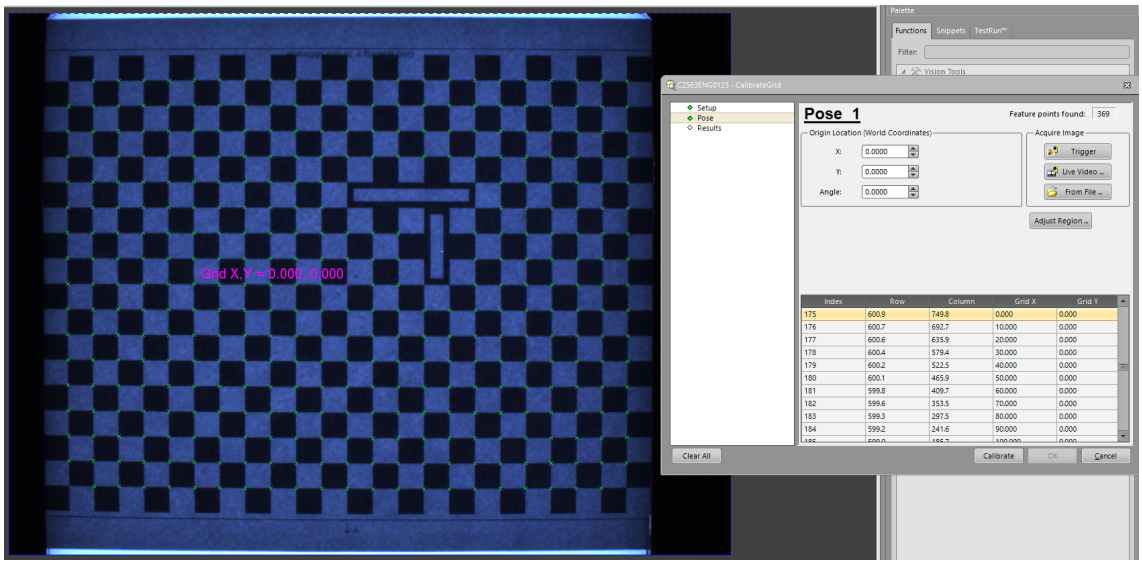
Souřadnice získaných bodů jsem vložil ve dvou řádcích do prostředí Spreadsheet. Do dvou řádků je to rozděleno z důvodu přehlednosti. Kalibrační body jsem následně vložil jako parametry do funkce *CalibrateAdvanced*. V tomto okamžiku mám vypočtenou kalibrační strukturu. Pomocí funkce *TransFixtureToWorld* je možné převést



Obr. 3.11: Kalibrační mřížka.

Tab. 3.4: Souřadnice kalibračních bodů

Číslo bodu	Kamera Row [px]	Kamera Columun [px]	Robot X [mm]	Robot Y [mm]
1	602.7	747.0	-266.15	451.82
2	602.9	1314.0	-366.15	455.75
3	207.0	1312.1	-368.80	385.7
4	205.1	746.1	-268.75	381.75
5	207.1	183.6	-168.58	377.79
6	601.1	181.9	-166.11	447.71
7	994.0	184.7	-163.46	517.34
8	999.1	747.1	-263.39	521.48
9	997.3	1311.7	-363.33	525.54



Obr. 3.12: Kalibrace pomocí *CalibrateGrid*.

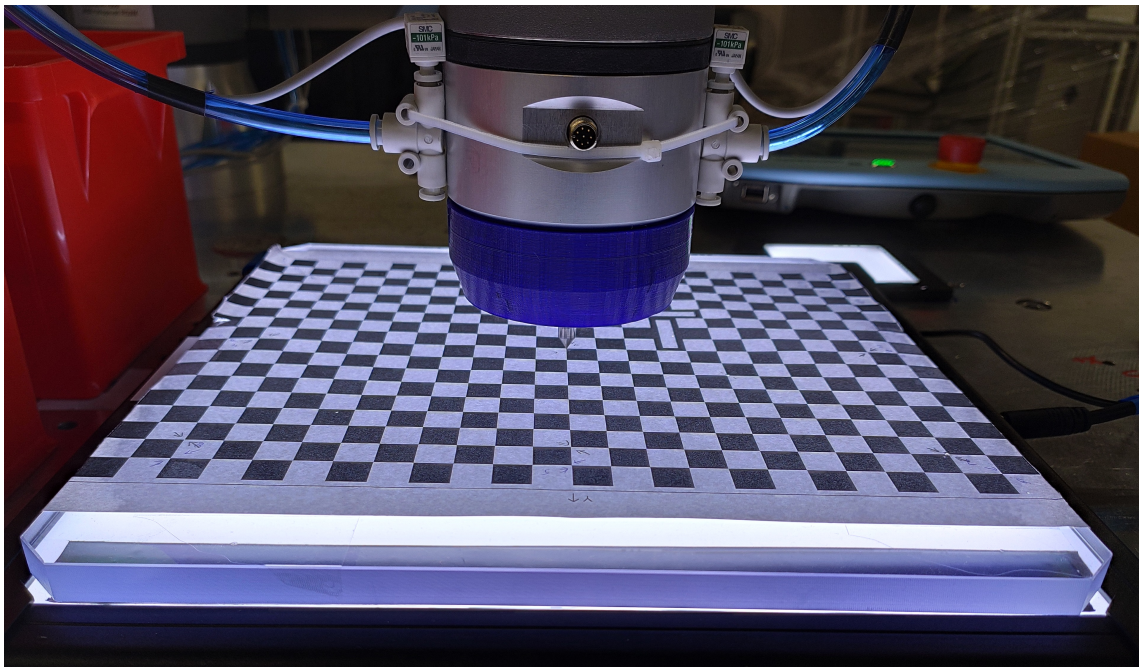
souřadnice nalezeného dílu v pixelech na souřadnice manipulátoru v milimetrech. Funkce má pět parametrů, prvním z nich je buňka s kalibrační strukturou, druhým parametrem je číslo řádku nalezeného dílu a třetím parametrem je číslo sloupce nalezeného dílu. Čtvrtým parametrem je úhel natočení, ale ten nepotřebuji transformovat, čili parametr je 0 a posledním parametrem se pouze zapíná nebo vypíná zobrazení hledaného bodu na obrázku. Tato funkce je umístěna v buňce A23. Hodnoty přepočtených souřadnic získám funkcemi *GetRow* a *GetCol*, které jsou umístěné v buňce B23 resp. C23. V buňce D23 se nachází výpočet úhlu pro nabrání dílu. Jak již bylo řečeno, tak manipulátor dva díly otočené o 180° odebírá se stejným úhlem natočení. Toho je docíleno výpočtem, který je zobrazený ve výpise 3.6. Vím, že úhel nalezeného dílu, který je umístěn v buňce D17, nabývá hodnot v intervalu [-180°; 180°]. Pokud je nalezený úhel v intervalu [-90°; 90°], do manipulátoru se odešla tato hodnota, jestliže je v intervalu [90°; 180°] od úhlu se odečte 180° a jestliže je úhel v intervalu [-90°; -180°] je k tomuto úhlu přičteno 180°.

Výpis 3.6: Přepočet úhlu

```
1 If(D17<90 && D17>-90,D17, If(D17>90 && D17<180,D17-180,180+D17))
```

3.5.3 Kamerový systém kontroly

Kameru na kontrolu jsem programoval jak v prostředí EasyBuilder tak v prostředí Spreadsheet. V prostředí EasyBuilder jsem realizoval nástroje pro kontrolu dílu a



Obr. 3.13: Určování souřadnic manipulátoru při kalibraci.

v prostředí Spreadsheet jsem implementoval vyhodnocení kontroly a odeslání výsledku na výstup. V prostředí EasyBuilder a kartě *Set Up Image* se nastavují stejné parametry jako u funkce *AcquireImage* a jsou zde nastaveny podobně jako u kamery lokalizace. Jen expozice je nyní 2,4 ms. Nachází se zde i možnost kalibrace, kterou jsem provedl pomocí stejné kalibrační mřížky jaku o u kamery lokalizace, nyní ovšem pomocí rozhraní EasyBuilder. Jelikož díl může být kontrolován ve dvou pozicích otočených o 180° a nabírání přísavkami též není moc přesné, musím implementovat funkcí, která najde střed dílu a až poté mohu kontrolovat přítomnost otvorů. Proto jsem v kartě *Locate Part* zvolil nástroj *Pattern*, tento nástroj je vlastně stejný jako funkce *FindPatterns* použitá v kameře lokalizace. Tudíž nástroj má naučený vzor, který vyhledává v obraze a vrací jeho střed. Nastavení je totožné s kamerou lokalizace, až na *Accept Threshold*, který je nastaven na 85 %. Předpokládám, že pokud by byla odchylka větší, tak díl může být hodně zdeformovaný, nebo nastal jiný problém a díl vyhodnotím jako špatný. V následující kartě *Inspect Part* se již vkládají nástroje pro inspekci dílu. Pro kontrolu kruhových otvorů jsem použil nástroj *Circle* z kategorie *Presence/Absence Tool*. Tento nástroj vyhodnocuje, zda se v definované oblasti nachází kruhový přechod barev. Oblast hledání je ohraničena jak z vnější, tak z vnitřní strany. Nástroj *Circle* je spojen s nástrojem *Pattern*, a proto i když je díl otočen o 180°, tak se otvor hledá na místě odpovídajícím tomuto otočení. Tímto způsobem jsou spojeny i ostatní nástroje. Ke kontrole malých čtvercových otvorů jsem použil nástroj *Blob* z kategorie *Presence/Absence Tool*, který

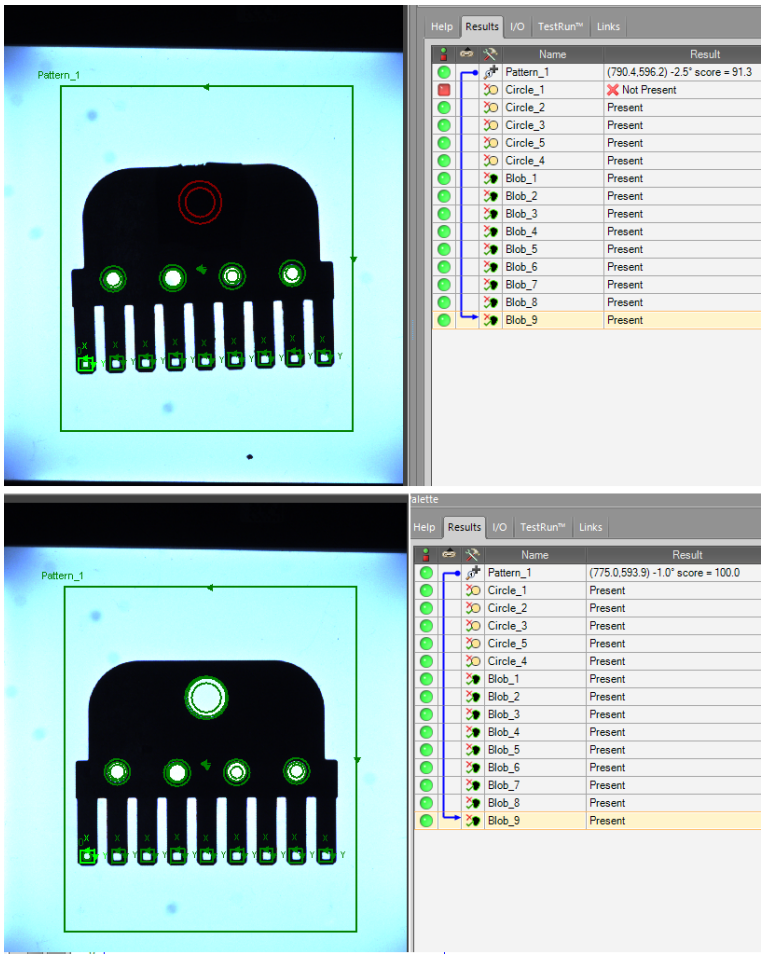
ve zvolené oblasti hledá skupinu spojených bílých pixelů. Tento nástroj nerozlišuje, jaký má nalezená skupina tvar, jen je možné nastavit minimální a maximální počet pixelů ve skupině. Ale pro můj účel je tento nástroj dostatečný, protože kontroluji jen přítomnost otvoru.

Ve vývojovém rozhraní Spreadsheet je vidět kolik práce mi ušetřilo využití prostředí EasyBuilder, protože nástroj *Circle* byl nastaven na pár kliknutí, ale v prostředí Spreadsheet ho reprezentuje několik funkcí na přibližně stovce buněk. Samozřejmě se jedná o univerzální nástroj, tudíž bude vždy složitější, než pokud bych si vytvořil nástroj pouze na jednu činnost. K odeslání výsledku na výstup jsem využil toho, že každý nástroj obsahuje buňku, která udává výsledek inspekce. Výsledek je vždy zobrazen v buňce pod buňkou s názvem *Pass/Fall*. Čili do buňky T34 sloužící k ovládání výstupu č. 1 jsem vložil funkci *And* a jako parametry jsem vložil odkazy na buňky s výsledky všech inspekcí. Do buňky T34, sloužící k ovládání výstupu č. 0 jsem vložil negaci buňky T34. V tento okamžik pokud všechny nástroje proběhnou v pořádku na výstupu č. 0 bude LOG 0 a a na výstupu č. 1 bude LOG 1, pokud bude díl špatný výstupy budou mít opačnou hodnotu. Na obrázku 3.14 je zobrazeno vyhodnocení špatného a dobrého dílu.

3.6 Test funkčnosti třídicí buňky

V této kapitole budu prezentovat zrealizovanou třídicí buňku a funkčnost jednotlivých periferií. Třídicí buňka je zobrazena na obrázku 3.15 a dále pak v příloze B. Je zde vidět rozvaděč za kterým je pod stolní deskou umístěn řídicí modul manipulátoru, vedle něj se nachází úpravna vzduchu a ventilový blok. Na desce stolu se nalézá manipulátor s malým a velkým backlightem a konzole s kamerami. Dále se na desce stolu nalézají odkládací krabičky a tlačítko nouzového zastavení.

Správnou funkci třídicí buňky jsem otestoval nejdříve tak, že jsem na nabírací backlight vložil jeden dobrý díl a vyzkoušel jsem několik cyklů, jestli bude spolehlivě nabrán, otestován a odložen. Díl jsem umisťoval do různých pozic s různým natočením. Přitom jsem sledoval výsledky kamer a pokud se nějaký výsledek lišil od reality, tak jsem upravil příslušné parametry příslušného nástroje. Především u kamery pro kontrolu jsem ze začátku upravoval převážně oblasti hledání otvorů. Až byly výsledky uspokojivé, přidal jsem více dílů a pokračoval v testování. Tím jsem otestoval především kameru nabírání. Dále jsem přidal i špatné díly a tím jsem otestoval obě kamery, u kamery nabírání, jestli dokáže špatné díly spolehlivě lokalizovat a u kamery kontroly, zda dokáže spolehlivě rozpoznat špatný díl od správného. Nakonec jsem zkusil překrývat různými způsoby a různou plochou dva díly a zkoumal jsem, zda dojde k úspěšnému nabrání dílu. Pokud se díly překrývaly přibližně do

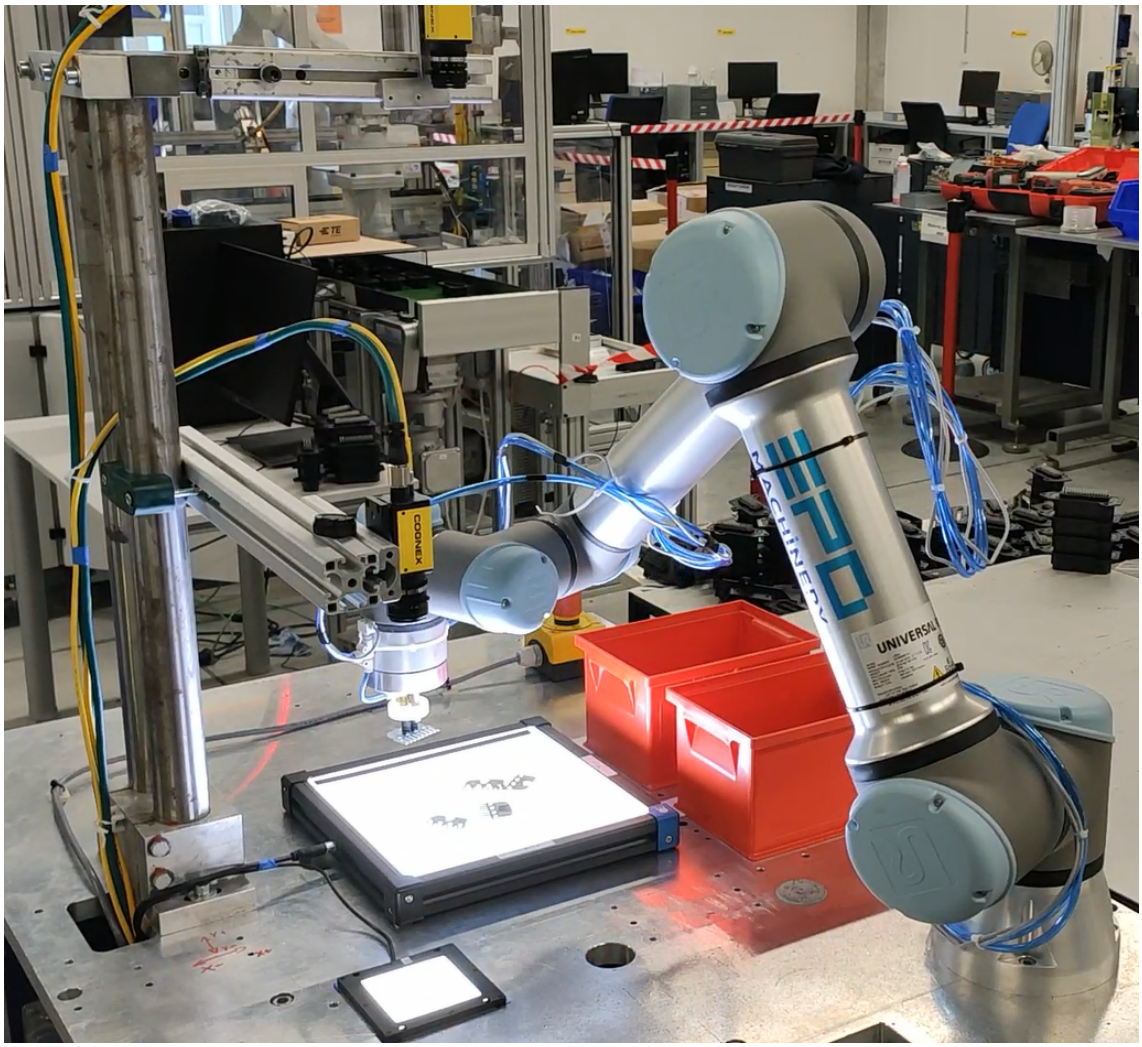


Obr. 3.14: Ukázka vyhodnocení špatného (nahoře) a dobrého (dole) dílu

dvou třetin, tak bylo nabírání spolehlivé. Záleželo ale i na vzájemném natočení obou dílů.

Dále jsem otestoval bezpečnostní prvky buňky. Zejména obě tlačítka nouzového zastavení, jedno umístěné na ovládacím panelu a druhé na desce stolu. Otestoval jsem též ošetření mezních stavů, například některá z kamer není online, nebo nedošlo ke správnému přisátí dílu, což jsem nasimuloval rozpojením vedení vzduchu k přísavce. Dále jsem na počítači spustil TCP server, kameru odpojil a manipulátorem jsem se připojil k serveru na počítači, do manipulátoru jsem poslal několik řetězců dat a otestoval funkci vyprázdnění bufferu. Rovněž jsem otestoval ošetření případu kdy kamera ve stanovené době neodešle výsledek. Ve všech těchto případech testy proběhly úspěšně.

Posledním z testů, které jsem provedl byl test na ověření doby testovacího cyklu. Na nabírací backlight jsem náhodně rozmístil šest dílů, pět dílů bylo v pořádku a jeden byl špatný. Se spuštěním testovacího programu jsem začal měřit čas a měření jsem ukončil, jakmile manipulátor po odhození posledního dílu dojel do domácí



Obr. 3.15: Třídicí buňka

polohy. Naměřený čas byl 49 s, to znamená, že čas testování jednoho dílu je 8,2 s. Vytyčený cíl byl, aby čas testovaného cyklu nebyl delší než 10 s, což je splněno.

Závěr

Ve své diplomové práci jsem se ve spolupráci se společností Tyco Electronics s.r.o., Kuřim zabýval návrhem a realizací automatické třídicí buňky s kolaborativním manipulátorem Universal Robots UR5e a kamerovým systémem od výrobce Cognex. Navrženou třídicí buňku jsem zrealizoval a úspěšně otestoval její funkčnost.

V teoretické části jsem se zabýval výběrem vhodného manipulátoru, dle zadaných parametrů jsem jako optimální manipulátor vybral model UR5e od výrobce Universal Robots. Následně jsem vybral vhodné pneumatické prvky pro ovládání přísavek, a nakonec jsem vybral vhodný kamerový systém od výrobce Cognex sloužící k nalezení a kontrole testovaného dílu. Dále jsem se zabýval způsoby programování manipulátoru a kamer.

V hlavní části práce jsem se nejdříve věnoval návrhu dispozice třídicí buňky. Následně jsem se věnoval návrhu rozvaděče. Dle vybraných slaboproudých periferií jsem určil minimální potřebný výkon 24 V zdroje. Ze znalostí parametrů manipulátoru a 24 V zdroje jsem určil odpínací proudy jističů. Všechny prvky jsem vhodně umístil do rozvaděče.

Jakmile jsem znal všechny elektrické prvky, které budou pro funkci třídicí buňky potřebné, již nic nebránilo návrhu elektrického schématu. Elektrické schéma jsem nakreslil pomocí programu WS CAD a následné dle schématu rozvaděč a periferie zapojil.

Po úspěšném zapojení a zprovoznění rozvaděče a všech periferií manipulátoru jsem mohl začít tvořit software. Algoritmus manipulátoru komunikuje s kamerovými systémy pomocí digitálních vstupů / výstupů a s kamerovým systémem pro nabírání i pomocí sběrnice EtherNet/IP. Manipulátor odebere díl dle souřadnic obdržených od kamerového systému lokalizace. Následně je díl odložen na testovací plochu a rameno manipulátoru se přemístí mimo zorné pole kamerového systému kontroly. Dojde k optické kontrole dílu a na základě jeho výsledku je díl odložen do příslušného odkládacího boxu.

V kamerovém systému lokalizace je implementováno kromě samotné lokalizace dílu a komunikace s manipulátorem i sjednocení souřadných systémů. V kamerovém systému kontroly je implementována inspekce pro kontrolu dílu s následným odesláním výsledku do manipulátoru. Zadání předpokládalo spolupráci s oddělením Vision, které mělo vytvořit inspekci na kontrolu dílu, ale jelikož k tomu nedošlo, musel jsem si inspekci v omezené formě implementovat sám.

Na závěr jsem otestovat funkčnost celé buňky řadou testů s uspokojivým výsledkem. Otestoval jsem spolehlivost lokalizace dílu v různých pozicích, nabírání dílu přísavkami a rozpoznání špatného a dobrého dílu. Dále jsem otestoval ošetření mezních stavu programu manipulátoru. Průměrný čas testovacího cyklu je přibližně 8

sekund, což je lepší výsledek než jsem si na začátku stanovil.

Vypracování této diplomové páce bylo pro mě velmi přínosné pro svojí komplexnost a zásahem do různých oblastí průmyslové automatizace. Seznámil jsem se s kolaborativními manipulátory, počítačovým viděním a jejich propojením, návrhem a zapojením rozvaděče. Také bylo zajímavé zjistit, že v praxi někdy musíme pracovat s tím co máme k dispozici i když to třeba není úplně ideální. Díky možnosti pracovat na tomto projektu jsem získal cenné zkušenosti, které se mi do budoucna budou určitě hodit.

Literatura

[1] LAHODA, Vlastimil. *Automatizovaná třídicí buňka* [online]. Brno, 2020 [cit. 2020-05-26]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/122437. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce František Burian.

[2] Kolaborativní robot KUKA šroubuje opěrky rukou u dodavatele automobilového průmyslu. *KUKA* [online]. Augsburg: KUKA, c2019 [cit. 2020-01-05]. Dostupné z: https://www.kuka.com/cs-cz/odv%C4%9Btv%C3%AD/solutions-database/2018/11/yanfeng_cobot-%C5%A1roubuje-op%C4%9Brky-rukou.

[3] FANUC CZECH S.R.O. Demystifikace kolaborativních průmyslových robotů. *Technický týdeník* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.technickytydenik.cz/rubriky/automatizace-robotizace/demystifikace-kolaborativnich-prumyslovych-robotu_47275.html.

[4] *Experience more: Collaborative robots for a wide range of applications.* Echternach Luxembourg, 2019. Dostupné také z: https://www.fanuc.eu/~/media/files/pdf/products/robots/brochures/mbr-02361-ro-collaborative-robot-v4/collaborative-robot-brochure-en.pdf?la=cs.

[5] *E-Series OD UNIVERSAL ROBOTS.* Praha, 2018.

[6] *Universal Robots řady e Návod k použití: UR5e.* Verze 5.5. Praha, 2019. Dostupné také z: https://s3-eu-west-1.amazonaws.com/ur-support-site/55430/99446_UR5e_User_Manual_cs_Global.pdf.

[7] *The URScript Programming Language.* Verze 5.5. Dánsko, 2019. Dostupné také z: https://s3-eu-west-1.amazonaws.com/ur-support-site/61791/scriptManual.pdf.

[8] *Remote Type Pressure Sensors/Pressure Sensor Controllers: Series PSE.* Tokyo. Dostupné také z: https://www.smcpneumatics.com/pdfs/PSE.pdf.

[9] *Převodníky tlaku SPTE.* Praha, 2018. Dostupné také z: https://www.festo.com/cat/cs_cz/data/doc_cs/PDF/CZ/SPTE_CZ.PDF.

[10] *Převodníky signálů SCDN.* Praha, 2017. Dostupné také z: https://www.festo.com/cat/cs_cz/data/doc_cs/PDF/CZ/SCDN_CZ.PDF.

[11] *5 Port Solenoid Valve: SV 1000/2000/3000/4000 Series.* Tokyo. Dostupné také z: https://content2.smcetech.com/pdf/SV.pdf.

[12] *Spínací ventily/ventily s pomalým náběhem tlaku HE/HEE/HEP/HEL, řada D.*Praha, 2013. Dostupné také z: https://www.festo.com/cat/cs_cz/data/doc_cs/PDF/CZ/D-START-UP-EXHAUST-VALVES_CZ.PDF.

[13] *Tlakové spínače PEV, mechanické.*Praha, 2006. Dostupné také z: https://www.festo.com/cat/cs_cz/data/doc_cs/PDF/CZ/PEV_CZ.PDF.

[14] *Cognex In-Sight Micro Series Vision System: Installation Manual.* Revision 5.7.4.3. Natick, 2019. Dostupné také z: https://support.cognex.com/docs/is_574/ISE/EN/Manuals/isMicroinst.pdf.

[15] *Cognex In-Sight CIO-MICRO and CIO-MICRO-CC I/O Modules: Installation Manual.* Revision 5.8.0.1. Natick, 2019. Dostupné také z: https://support.cognex.com/docs/is_580/ISE/EN/Manuals/IOMicroinst.pdf.

[16] LTBC234234-W: Continuos LED backlight, 234x234 illumination area, white. *Opto Engineering* [online]. Mantova: Opto Engineering SEO and consulting by Dipiemme Studio, c2015-2019 [cit. 2019-12-29]. Dostupné z: https://www.opto-e.com/products/ltbc-model-LTBC234234-W.

[17] LTPVR070-00-1-W-24V: Flat side-emitting LED backlight, 70X70 mm illumination area, white, 24V. *Opto Engineering* [online]. Mantova: Opto Engineering SEO and consulting by Dipiemme Studio, c2015-2020 [cit. 2020-05-22]. Dostupné z: https://www.opto-e.com/products/ltbfc-model-LTPVR070-00-1-W-24V.

[18] Anyfeed SX240. *Flexfactory* [online]. Dietikon: flexfactory, c2019 [cit. 2019-12-31]. Dostupné z: https://www.flexfactory.com/en/products/feeders/sx240/.

[19] ČSN EN ISO 12100 (833001). *Bezpečnost strojních zařízení – Všeobecné zásady pro konstrukci – Posouzení rizika a snižování rizika.* Praha: Úřadem pro technickou normalizaci, metrologii a státní zkušebnictví, 2011.

[20] ČSN EN 60204-1 ED. 3 (332200). *Bezpečnost strojních zařízení – Elektrická zařízení strojů.* Ed. 3. Praha: Česká agentura pro standardizaci, 2019.

[21] *Installation & Maintenance Manual: Multi Channel Pressure Sensor Controller Series PSE200 / PSE201.* Tokyo. Dostupné také z: https://www.smc.eu/smc/Net/EMC_DDBB/ce_documentation/data/attachments/IMM_PSE20x_TFI57GB-B.pdf.

[22] *Industrial Ethernet Switch.* Neuhausen auf den Fildern. Dostupné také z: https://assets.balluff.com/WebBinary1/MAN_BNI_TCP_951_000_E028_E_1406_DOK_914485_00_000.pdf.

[23] *Siemens: SITOP smart 2,5A, SITOP smart 5A, SITOP smart 10A.* Wien, 2010. Dostupné také z: https://cache.industry.siemens.com/dl/files/569/22324569/att_90007/v1/BA_6EP133x-2xAxx_C98130-A7559-A1-12-6419_201012.pdf.

[24] Proudové zatížení pro flexibilní kabely. *DDA* [online]. Brno: FRONK Design, c2006 [cit. 2020-01-03]. Dostupné z: http://www.kabelyvodice.cz/technicka-podpora/tech_proud_zat.php.

[25] SPONG, Mark W., Seth HUTCHINSON a M. VIDYASAGAR. *Robot modeling and control.* Hoboken, NJ: John Wiley, c2006. ISBN 978-0471649908.

[26] SMARK M., Spong a Vidyasagar M. *Robot dynamics and control.* India: Wiley, 2008. ISBN 978-8126517800. 978-0471649908.

[27] In-Sight Explorer Development Environments. *Cognex* [online]. Natick: Cognex Corporation, 2020 [cit. 2020-05-18]. Dostupné z: https://support.cognex.com/docs/is_590/web/EN/ise/Content/GettingStarted/DevEnvironment.htm?tocpath=Getting%20Started%7C_____1

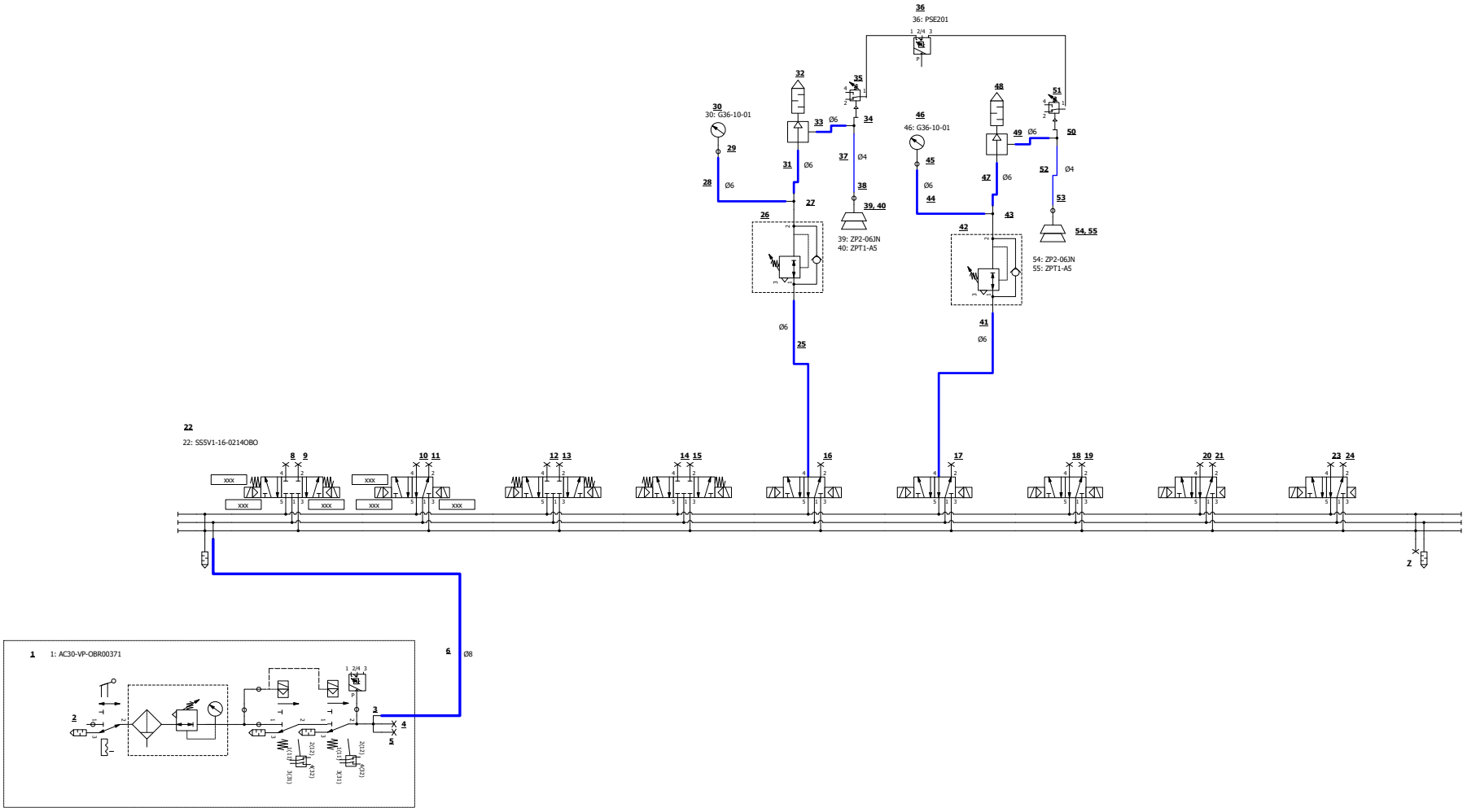
[28] Spreadsheet Cell Execution and Editing Reference. *Cognex* [online]. Natick: Cognex Corporation, 2020 [cit. 2020-05-18]. Dostupné z: https://support.cognex.com/docs/is_590/web/EN/ise/Content/User_Interface/CellLogicReferences.htm?TocPath=Spreadsheet%20View|Using%20the%20Spreadsheet|_____1

[29] Průmyslový Ethernet IX: EtherNet/IP, EtherCAT. *Automa – časopis pro automatizační techniku, s. r. o.* [online]. Ústí nad Labem: Automa, ©2016 [cit. 2020-05-18]. Dostupné z: https://automa.cz/cz/casopis-clanky/prumyslovy-ethernet-ix-ethernet/ip-ethercat-2008_10_37910_6510/

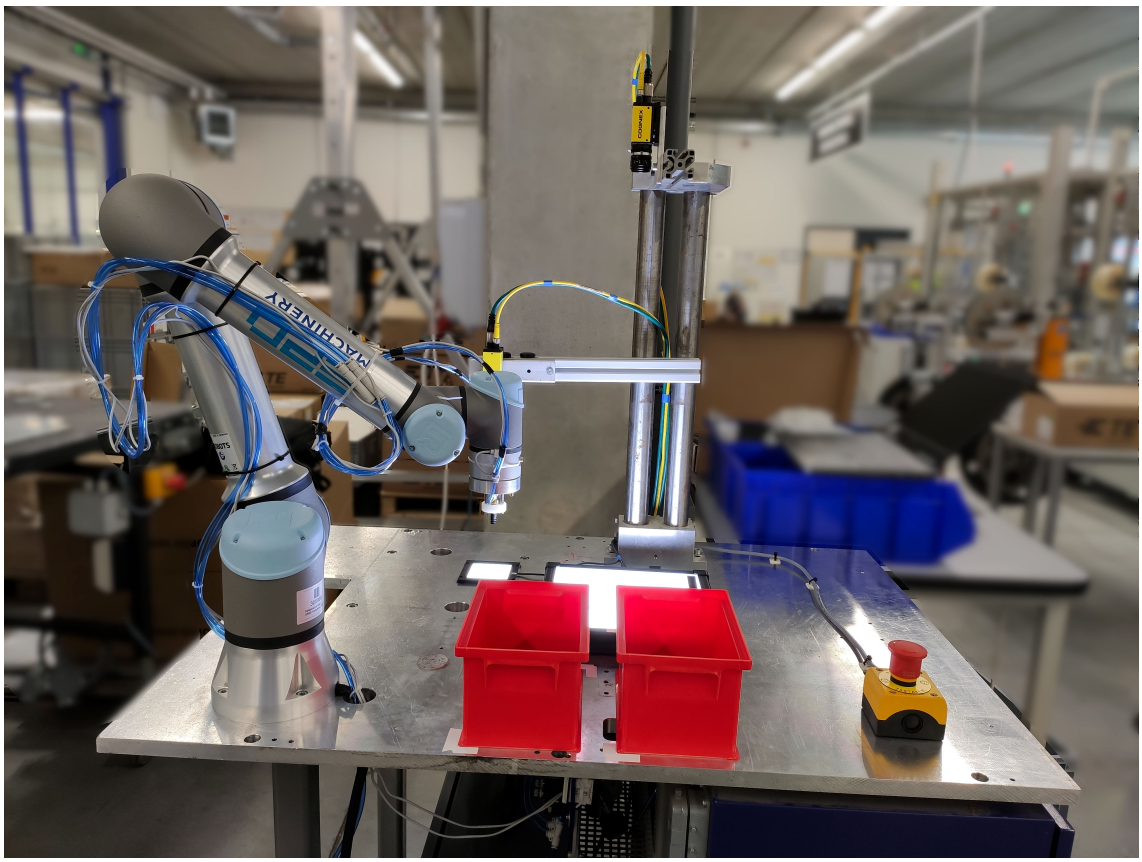
Seznam příloh

A Pneumatické schéma

Pneumatické schéma se nachází na další straně. Schéma je současně uloženo i v přiloženém archivu Prilohy.zip


36
36: PSE201
32
35
30
30: G36-10-01
33
Ø6
34
48
51
46
46: G36-10-01
49
Ø6
50
29
31
Ø6
37
Ø4
45
47
Ø6
52
Ø4
28
Ø6
38
Ø6
44
53
27
39, 40
43
26
42
54, 55
39: ZP2-06JN
40: ZPT1-A5
54: ZP2-06JN
55: ZPT1-A5
Ø6
41
Ø6
25
22
22: SS5V1-16-0214OBO
8 9
10 11
12 13
14 15
16
17
18 19
20 21
23 24
xxx
Z
1
1: AC30-VP-OBR00371
6
Ø8
2
3
4
5

B Skutečná podoba třídicí buňky

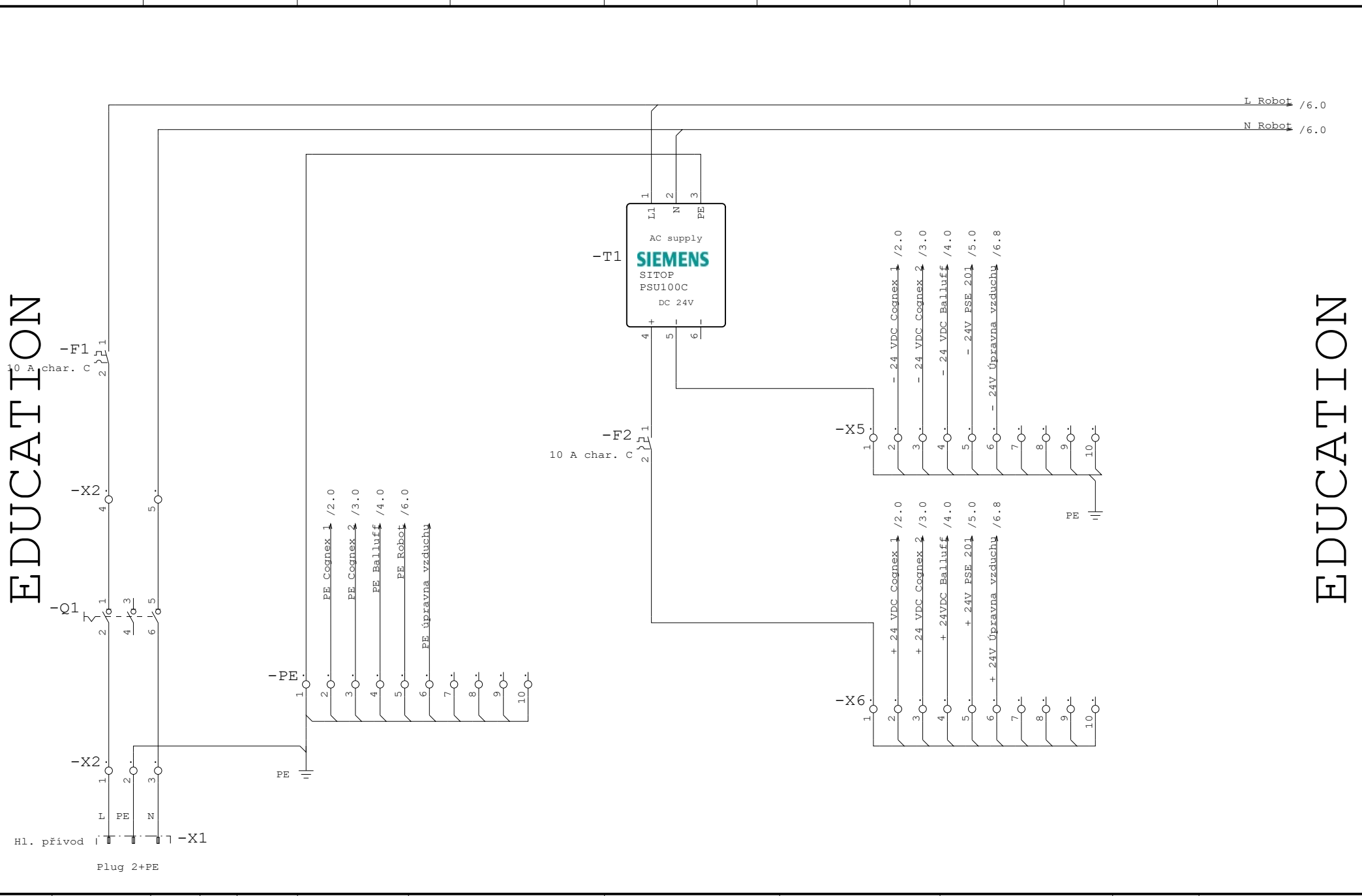


Obr. B.1: Třídicí buňka

Obr. B.2: Třídicí buňka

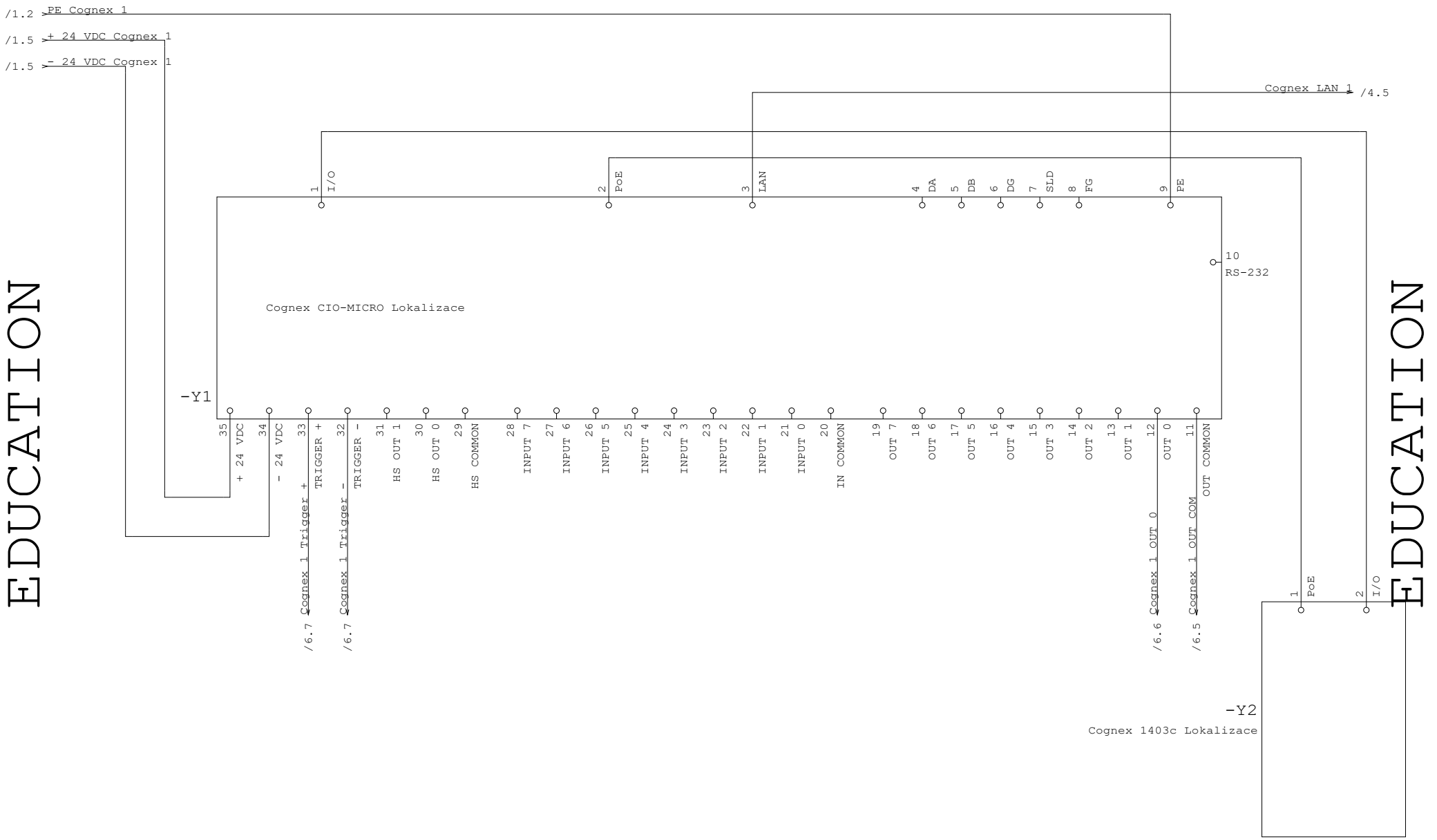
C Elektrické schéma

Elektrické schéma začíná na další straně. Schéma je současně uloženo i v přiloženém archivu Prilohy.zip



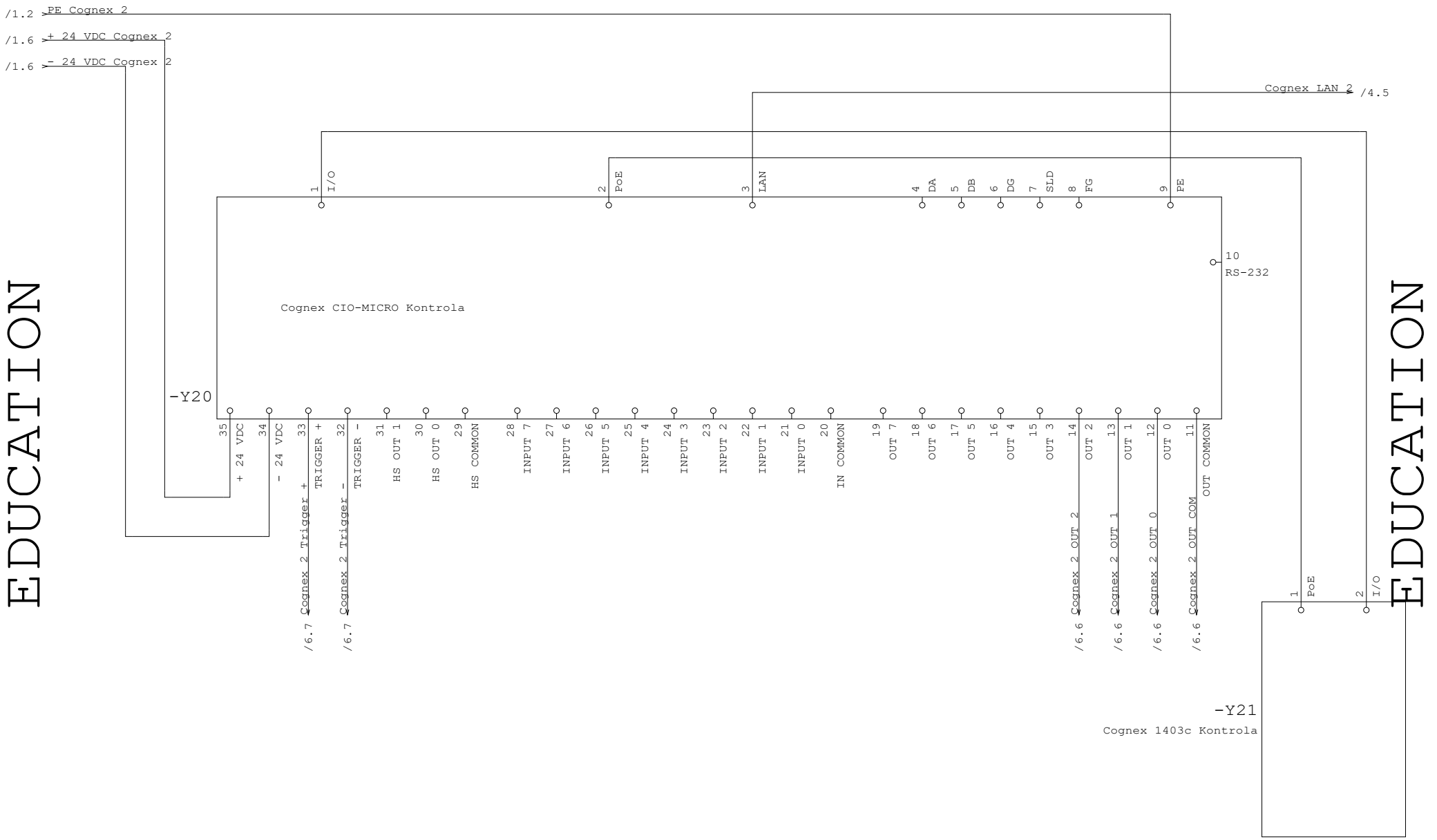
				Date	23.10.19			Project number	Unit	=
				Work			Vlastimil Lahoda	Diplomová práce	Field	+
				Check					Drawing number	sheet 1
Status	Rev.	Date	Name	Norm	DIN 81346	Created for	Created by			of 6

WS CAD ELECTRICAL ENGINEERING

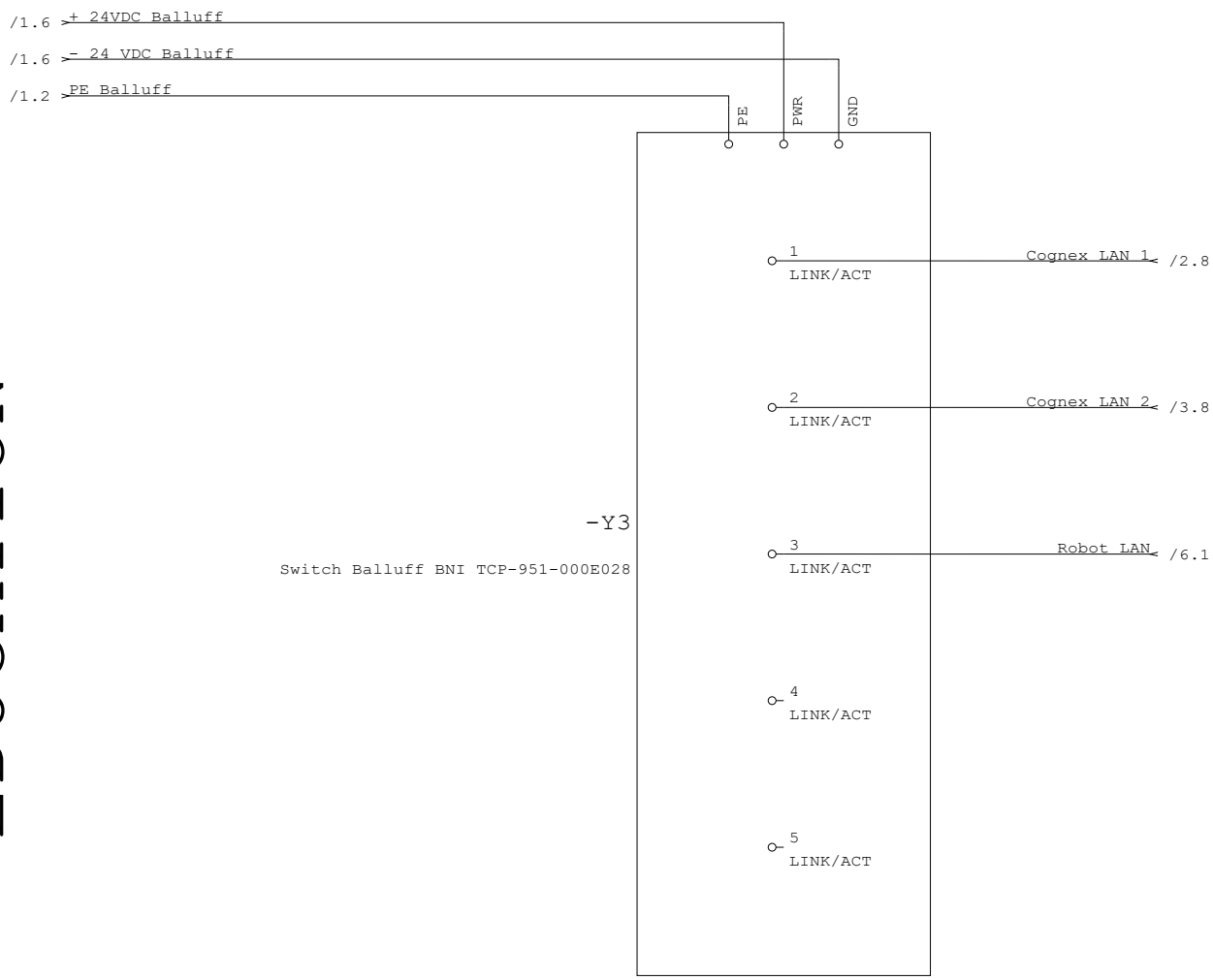


				Date	23.10.19			Project number	Unit	=
				Work			Vlastimil Lahoda	Diplomová práce	Field	+
				Check					Drawing number	sheet 2
Status	Rev.	Date	Name	Norm	DIN 81346	Created for	Created by			of 6

WS CAD ELECTRICAL ENGINEERING



				Date	23.10.19			Project number	Unit	=
				Work			Vlastimil Lahoda	Diplomová práce	Field	+
				Check						
Status	Rev.	Date	Name	Norm	DIN 81346	Created for	Created by		Drawing number	sheet 3 of 6

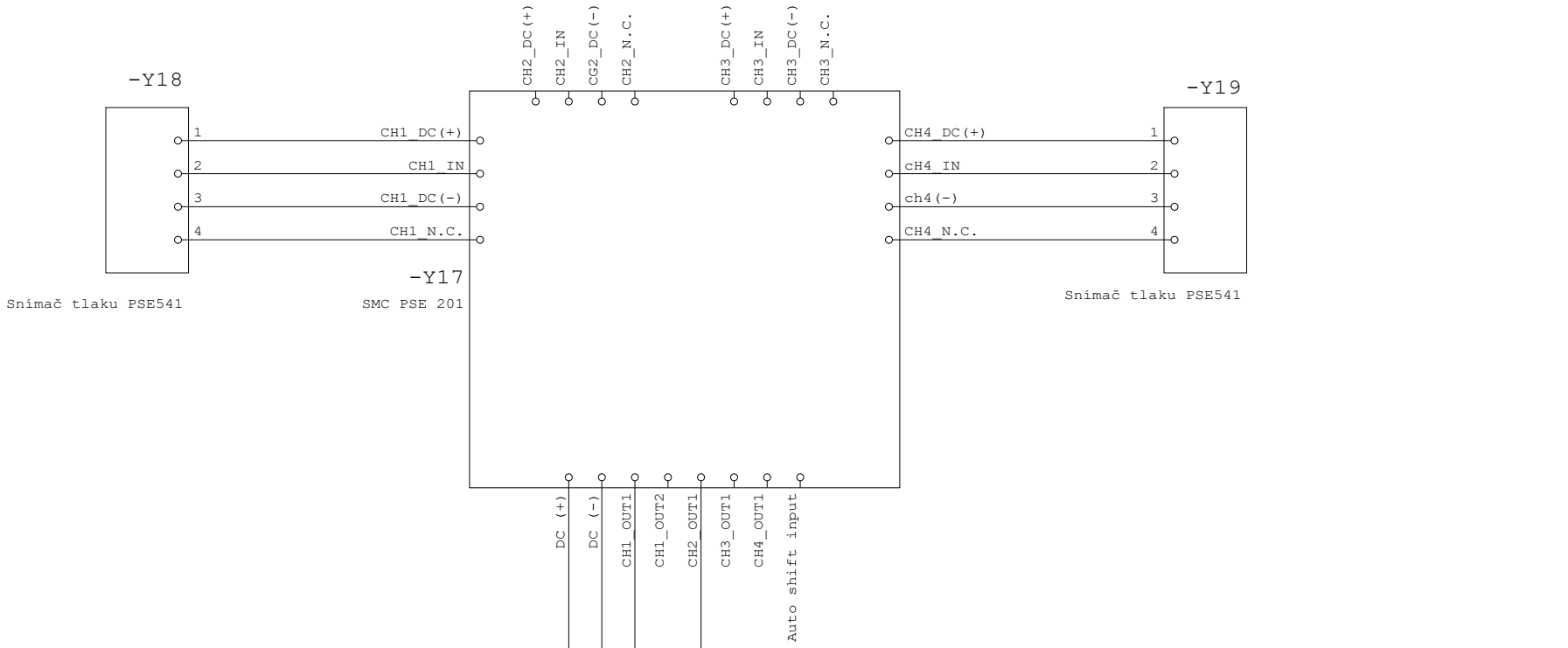


EDUCATION

EDUCATION

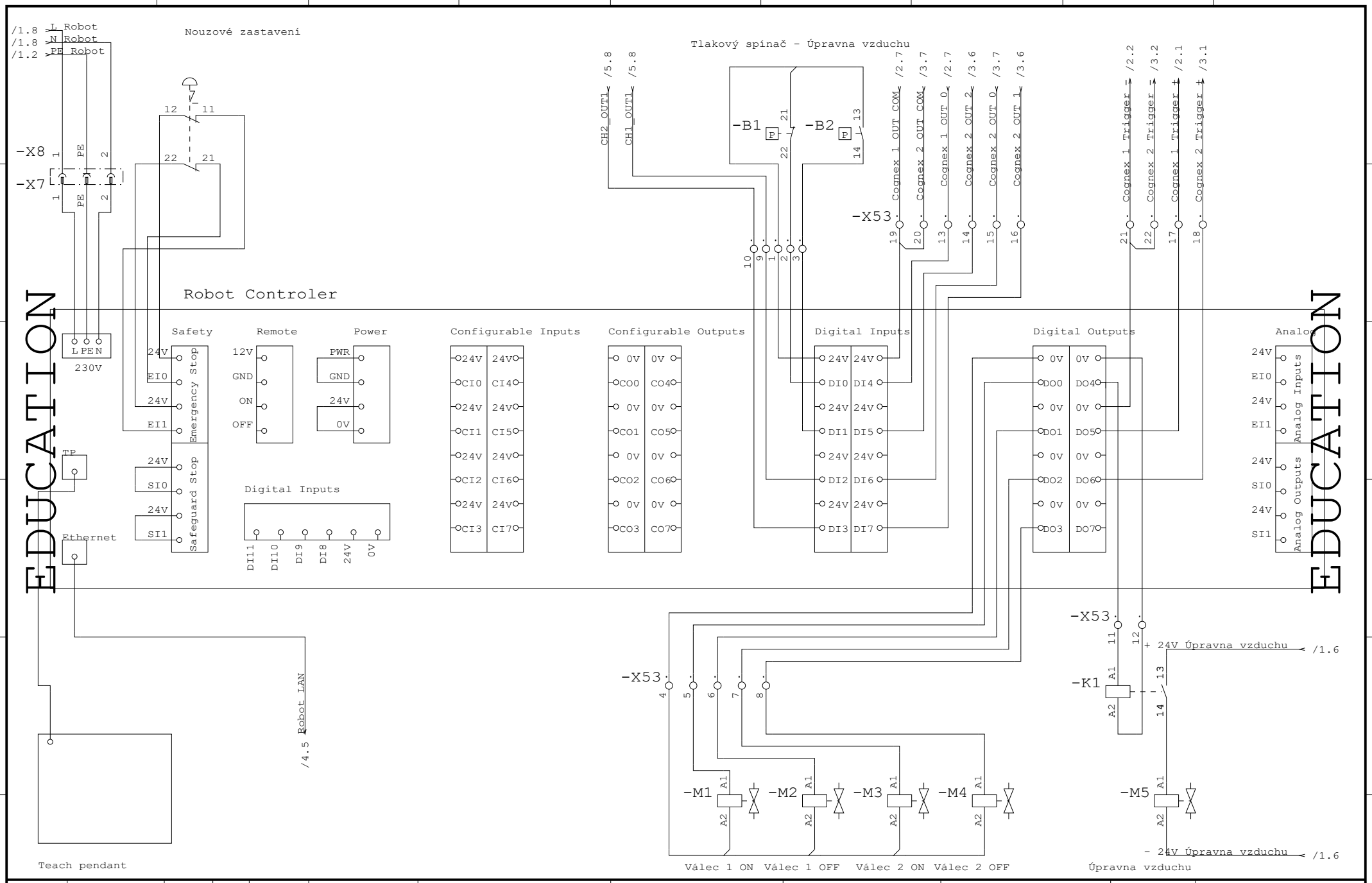
				Date	23.10.19			Project number	Unit	=
				Work			Vlastimil Lahoda	Diplomová práce	Field	+
				Check						
Status	Rev.	Date	Name	Norm	DIN 81346	Created for	Created by		Drawing number	sheet 4 of 6

EDUCATION



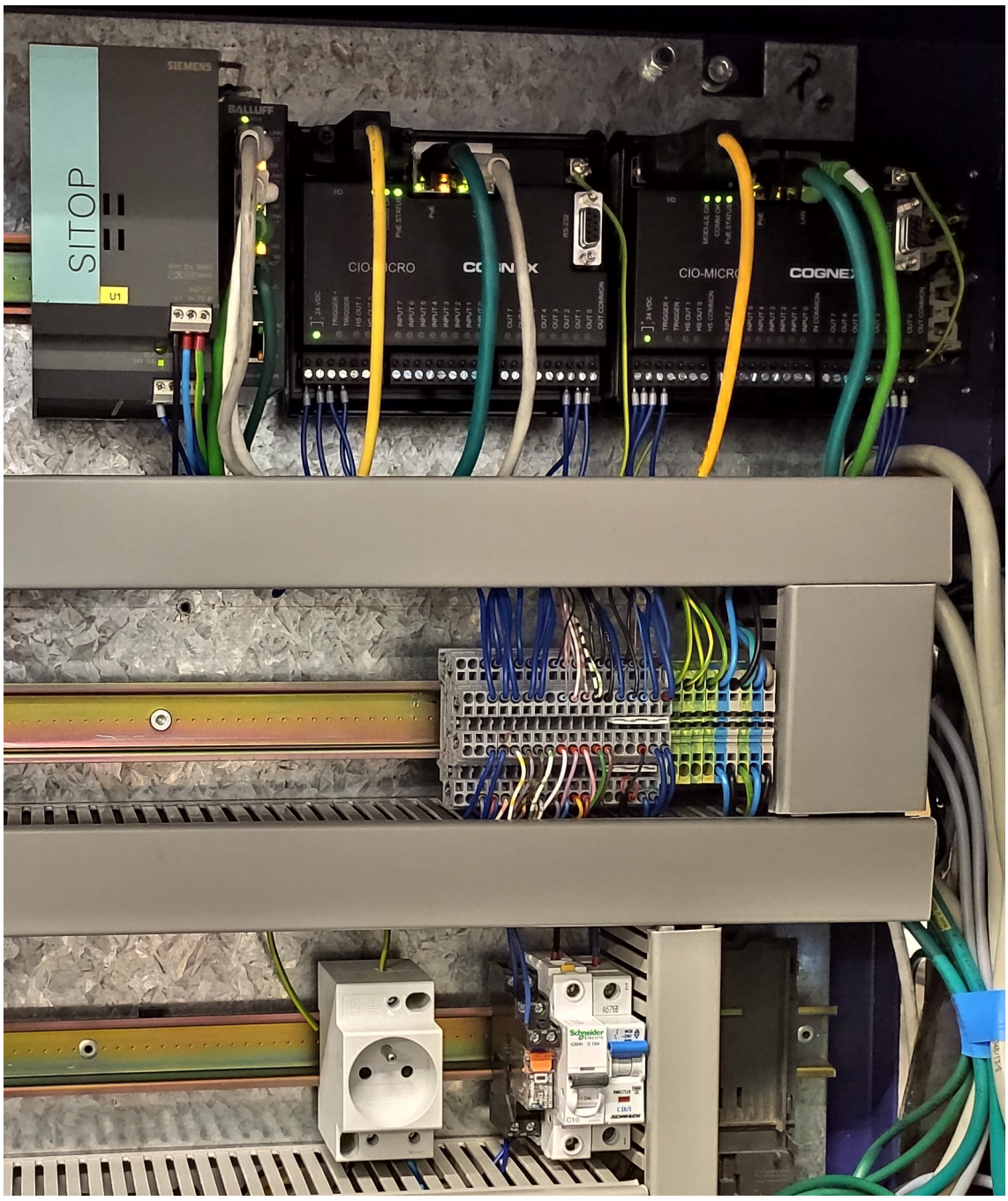
EDUCATION

				Date	23.10.19		WS CAD ELECTRICAL ENGINEERING	Vlastimil Lahoda	Project number Diplomová práce / Unit = / Field +
				Work					
				Check					Drawing number / sheet 5 / of 6
Status	Rev.	Date	Name	Norm	DIN 81346	Created for	Created by		



				Date	23.10.19			Project number	Unit	=
				Work			Vlastimil Lahoda	Diplomová práce	Field	+
				Check					Drawing number	sheet 6
Status	Rev.	Date	Name	Norm	DIN 81346	Created for	Created by			of 6

D Fotografie rozvaděče



Obr. D.1: Skutečné provedení rozvaděče

E Skript manipulátoru

Skript manipulátoru začíná na další straně. Schéma je současně uloženo i v přiloženém archivu Prilohy.zip

```
# Definice bodu
global Home = get_inverse_kin(p[-0.200, 0.250, 0.170,-3.14,0,0])
global Kontrola = get_inverse_kin(p[-0.43808, 0.22637, 0.00545,-3.14,0,0])
global Kontrola_Nad = get_inverse_kin(p[-0.43808, 0.22637, 0.01,-3.14,0,0])
global Kontrola_Cekani = get_inverse_kin(p[-0.36728, 0.13284,
0.01448,-3.14,0,0])
global NOK_Box = get_inverse_kin(p[-0.005, 0.500, 0.135,-3.14,0,0])
global NOK_Box_pred = get_inverse_kin(p[-0.135, 0.500, 0.135,-3.14,0,0])
global OK_Box = get_inverse_kin(p[-0.005, 0.320, 0.135,-3.14,0,0])
global OK_Box_pred = get_inverse_kin(p[-0.135, 0.320, 0.135,-3.14,0,0])

# Definice jmen vystupu
global O_Valec1_On = 0
global O_Valec1_Off = 1
global O_Valec2_On = 2
global O_Valec2_Off = 3
global O_UpravnaVz = 4
global O_Kam1_trig = 5
global O_Kam2_trig = 6
global O_DO7 = 7

# Definice jmen vstupu
global I_Tlak_spinac_UV_NC = 0
global I_Tlak_spinac_UV_NO = 1
global I_Tlak_snim_Ch1 = 2
global I_Tlak_snim_Ch2 = 3
global I_Kam1_Out_0_ON = 4
global I_Kam2_Out_2_ON = 5
global I_Kam2_Out_0_NOK = 6
global I_Kam2_Out_1_OK = 7

# Nastaveni promennych pred startem
global job = True
global data =""
set_standard_digital_out(O_Valec1_On, False)
set_standard_digital_out(O_Valec1_Off, False)
set_standard_digital_out(O_Valec2_On, False)
set_standard_digital_out(O_Valec2_Off, False)
set_standard_digital_out(O_UpravnaVz, False)
set_standard_digital_out(O_Kam1_trig, False)
set_standard_digital_out(O_Kam2_trig, False)
set_standard_digital_out(O_DO7, False)


# Definice funkci

# Funkce na vypnuti prisavek
# a = 0 - Vypne obe prisavky  (vychozi hodnota)
# a = 1 nebo a = 2 - Vypne prisavku 1 nebo 2
def F_Pris_Off (a = 0):
    Time_Out=0
    if a == 0 :
        #  Zapnuti vystupu pro vypnuti prisavek
        set_standard_digital_out(O_Valec1_Off, True)
        set_standard_digital_out(O_Valec2_Off, True)
        sleep(0.2)
        # Cekani na odsati
        while (not((get_standard_digital_in(I_Tlak_snim_Ch1) and
                    get_standard_digital_in(I_Tlak_snim_Ch2)) == False)):
            sync()
            Time_Out=Time_Out+1
            if Time_Out > 5000:
            return -2 # Chyba pri odsati
            end
```

```
64            end
65            # Vypnuti vystupu pro vypnuti prisavek
66            set_standard_digital_out(O_Valec1_Off, False)
67            set_standard_digital_out(O_Valec2_Off, False)
68            return 0 # Vse OK
69        elif a == 1 :
70            #  Zapnuti vystupu pro vypnuti prisavek
71            set_standard_digital_out(O_Valec1_Off, True)
72            # Cekani na odsati
73            while (get_standard_digital_in(I_Tlak_snim_Ch1) == True):
74                sync()
75                Time_Out=Time_Out+1
76                if Time_Out > 5000:
77                return -2 # Chyba pri odsati
78                end
79            end
80            # Vypnuti vystupu pro vypnuti prisavek
81            set_standard_digital_out(O_Valec1_Off, False)
82            return 0 # Vse OK
83        elif a == 2 :
84            #  Zapnuti vystupu pro vypnuti prisavek
85            set_standard_digital_out(O_Valec2_Off, True)
86            # Cekani na odsati
87            while (get_standard_digital_in(I_Tlak_snim_Ch2) == True):
88                sync()
89                Time_Out=Time_Out+1
90                if Time_Out > 5000:
91                return -2 # Chyba pri odsati
92                end
93            end
94            # Vypnuti vystupu pro vypnuti prisavek
95            set_standard_digital_out(O_Valec2_Off, False)
96            return 0 # Vse OK
97        else :
98            return -1 # Neplatny vstupni parametr
99        end
100   end
101
102   # Funkce pro zapnuti prisavek
103   # a = 0 - Zapne obe prisavky  (vychozi hodnota)
104   # a = 1 nebo a = 2 - Zapne prisavku 1 nebo 2
105   def F_Pris_On (a = 0):
106       Time_Out=0
107       if a == 0 :
108           #  Zapnuti vystupu pro vypnuti prisavek
109           set_standard_digital_out(O_Valec1_On, True)
110           set_standard_digital_out(O_Valec2_On, True)
111           # Cekani na odsati
112           while (not(get_standard_digital_in(I_Tlak_snim_Ch1) and
113                      get_standard_digital_in(I_Tlak_snim_Ch2) == True)):
114               sync()
115               Time_Out=Time_Out+1
116               if Time_Out > 5000:
117               return -2 # Chyba pri prisati
118               end
119           end
120           # Vypnuti vystupu pro vypnuti prisavek
121           set_standard_digital_out(O_Valec1_On, False)
122           set_standard_digital_out(O_Valec2_On, False)
123           return 0 # Vse OK
124       elif a == 1 :
125           #  Zapnuti vystupu pro vypnuti prisavek
126           set_standard_digital_out(O_Valec1_On, True)
127           # Cekani na odsati
```

```
128             while (get_standard_digital_in(I_Tlak_snim_Ch1) == False):
129                 sync()
130                 Time_Out=Time_Out+1
131                 if Time_Out > 5000:
132                 return -2 # Chyba pri přisati
133                 end
134             end
135             # Vypnuti vystupu pro vypnuti prisavek
136             set_standard_digital_out(O_Valec1_On, False)
137             return 0 # Vse OK
138         elif a == 2 :
139             #  Zapnuti vystupu pro vypnuti prisavek
140             set_standard_digital_out(O_Valec2_On, True)
141             # Cekani na odsati
142             while (get_standard_digital_in(I_Tlak_snim_Ch2) == False):
143                 sync()
144                 Time_Out=Time_Out+1
145                 if Time_Out > 5000:
146                 return -2 # Chyba pri přisati
147                 end
148             end
149             # Vypnuti vystupu pro vypnuti prisavek
150             set_standard_digital_out(O_Valec2_On, False)
151             return 0 # Vse OK
152         else :
153             return -1 # Neplatny vstupni parametr
154         end
155     end
156
157     # Funkce pro odeslani triggeru kamere
158     # cislo = 1 trigger kamera 1
159     # cislo = 2 trigger kamera 2
160     def F_Trigger_Kam (cislo):
161         if cislo == 1 or cislo == 2:
162             set_standard_digital_out(4 + cislo,True)
163                 #Cekani 0.5s
164                 sleep(0.5)
165             set_standard_digital_out(4 + cislo,False)
166
167             return 0 # Vse OK
168         else:
169             return -1 # Neplatny vstupni parametr
170         end
171     end
172
173     # Funkce pro parovani prijatych dat
174     def F_Pars_dat (data):
175         delka = str_len(data)
176         strednik1 = str_find(data,";")
177         strednik2 = str_find(data,";",(strednik1+1))
178         strednik3 = str_find(data,";",(strednik2+1))
179         Xpos = str_sub(data,(strednik1+1),(strednik2-strednik1-1))
180         Ypos = str_sub(data,(strednik2+1),(strednik3-strednik2-1))
181         angle = str_sub(data,(strednik3+1),(delka-strednik3-1-3))
182
183         # Prevod retezce na cislo
184         iXpos = to_num(Xpos)
185         iYpos = to_num(Ypos)
186         iangleD = to_num(angle)
187
188         # Prevod stupnu na Radiany
189         iangleR = d2r(iangleD)
190
191         # Prevod na metry
```

```
192         iXpos = iXpos/1000
193         iYpos = iYpos/1000
194         pole = [iXpos,iYpos,iangleR]
195         return pole
196     end
197 
198     #Vypnuti prisavek
199     if (F_Pris_Off() != 0) :
200         popup("Chyba pri vypnuti prisavek")
201         halt
202     end
203 
204     # Kontrola zda je kamera nabirani online
205     if (get_standard_digital_in(I_Kam1_Out_0_ON) != True ):
206         popup("Kamera nabirani neni online")
207         halt
208     end
209 
210     # Kontrola zda je kamera kontroly online
211     if (get_standard_digital_in(I_Kam2_Out_1_OK) != True):
212         popup("Kamera kontroly neni online")
213         halt
214     end
215 
216     #Pripojeni ke kamere
217     if (socket_open("192.168.56.3",3000,"socket_kamera") == False):
218         popup("Nepodarilo se pripojit ke kamere")
219         halt
220     end
221 
222     while True:
223 
224         #Vyprázdnění TCP bufferu
225         global S_data = socket_read_string("socket_kamera",timeout=0.2)
226         while str_empty(S_data)==False:
227             S_data = socket_read_string("socket_kamera",timeout=0.2)
228         end
229 
230         #Kamera 1 trigger
231         if (F_Trigger_Kam(1) != 0) :
232             popup("Chyba pri triggeru")
233             halt
234         end
235 
236         #Home pozice
237         movej(Home, a=10,v=20)
238 
239         #Cteni dat ze socketu
240         global Sock_Data=socket_read_string("socket_kamera",timeout=5)
241 
242         #Prisly data?
243         if (str_empty(Sock_Data)) :
244             popup("Nedosli data")
245             pause
246             continue
247         end
248 
249         job_stat=str_sub(Sock_Data,0,1)
250 
251         #Dil nalezen?
252         if job_stat == "0" :
253             popup("Dil nebyl nalezen")
254             pause
255             continue
```

```
256         end
257
258
259         global vektor = F_Pars_dat(Sock_Data)
260
261         global X = vektor[0]
262         global Y = vektor[1]
263         global A = vektor[2]
264
265         #Pozice nad nalezenym dilem
266         global dest = get_inverse_kin(p[X, Y, 0.050,-3.14,0,0])
267         movel(dest, a=2, v=12.0)
268
269         #Nabiraci pozice nalezeneho dilu
270         global feed=p[0,0,(0.05-0.0339),0,0,(-1*A)]
271         global pose_feed = pose_trans(get_forward_kin(),feed)
272         movel(pose_feed,a=5,v=5,r=0)
273         sleep(0.2)
274
275
276         #Zapnuti prisavek
277         if (F_Pris_On() != 0) :
278             popup("Chyba pri zapnuti prisavek")
279             halt
280         end
281
282         #Pozice nad dilem
283         global feed2=p[0,0,(0.033-0.1),0,0,0]
284         global pose_feed2 = pose_trans(get_forward_kin(),feed2)
285         movel(pose_feed2,a=2,v=12)
286
287         #Pozice pro kontrolu
288         movel(Kontrola_Nad, a=2,v=12)
289         movel(Kontrola, a=2,v=12)
290
291         #Vypnuti prisavek
292         if (F_Pris_Off() != 0) :
293             popup("Chyba pri vypnuti prisavek")
294             halt
295         end
296         sleep(0.2)
297
298         #Cekaci pozice
299         movel(Kontrola_Nad, a=2,v=12)
300         movel(Kontrola_Cekani, a=2, v=12)
301
302         if (F_Trigger_Kam(2) != 0) :
303             popup("Chyba pri triggeru")
304             halt
305         end
306
307         Time_Out=0
308         while (not ((get_standard_digital_in(I_Kam2_Out_0_NOK) == True and
            get_standard_digital_in(I_Kam2_Out_1_OK) == False)or
309         (get_standard_digital_in(I_Kam2_Out_0_NOK) == False and
            get_standard_digital_in(I_Kam2_Out_1_OK) == True))):
310             sync()
311             Time_Out=Time_Out+1
312             if Time_Out > 5000:
313                 popup("Kamera kontroly neodpovida")
314                 halt
315             end
316         end
317
```

```
318         global Kon_out1=get_standard_digital_in(I_Kam2_Out_0_NOK)
319         global Kon_out2=get_standard_digital_in(I_Kam2_Out_1_OK)
320 
321         movel(Kontrola_Nad, a=2,v=12)
322         movel(Kontrola, a=2,v=12)
323 
324         if (F_Pris_On() != 0) :
325             popup("Chyba pri zapnuti prisavek")
326             halt
327         end
328 
329         movel(Kontrola_Nad, a=2,v=12)
330 
331         if (Kon_out1 == True and Kon_out2 == False):
332             movel(NOK_Box_pred, a=2,v=12,r=0.1)
333             movel(NOK_Box, a=2,v=12)
334         elif (Kon_out1 == False and Kon_out2 == True):
335             movel(OK_Box_pred, a=2,v=12,r=0.1)
336             movel(OK_Box, a=2,v=12)
337         else:
338         popup("Chyba kontrolni kamery")
339         halt
340         end
341 
342         #Vypnuti prisavek
343         if (F_Pris_Off() != 0) :
344             popup("Chyba pri vypnuti prisavek")
345             halt
346         end
347     end
348 
```

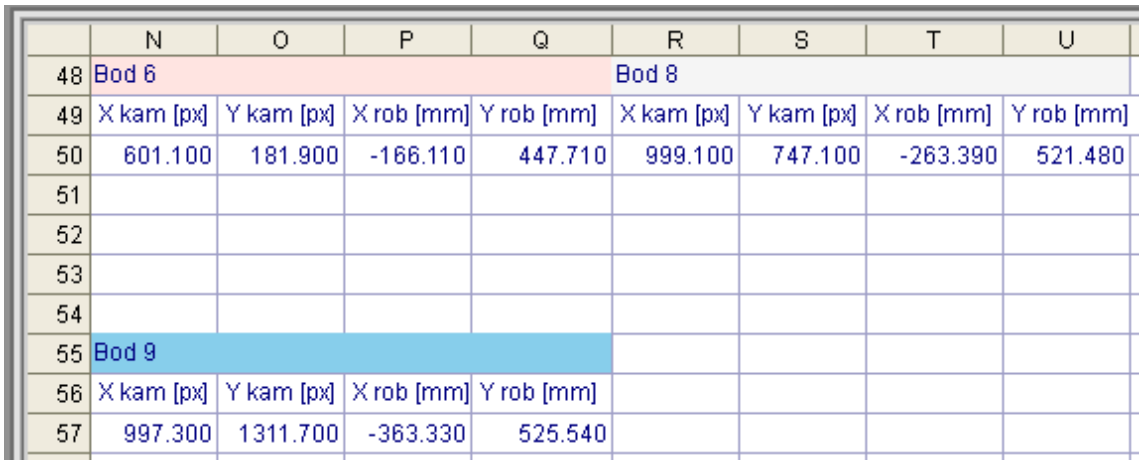
F Tabulka inspekce kamery nabírání

	A	B	C	D	E	F	G	H	I
0	Image								
1	1200.000	0.000	0.000	0.000					
2	Trigger	Trigger Del	Trigger Inte	Exposure (i	Auto-Exposu	Max Exposu	Target Image Brightness		
3	Camera	0	500	6.000	Disabled	1000.00	10.000		
4		1.000	0.000	1.000	1.000	0.000			
5	Start Row	Number Of	Light Contr	Light Contr	Light Enable	Light Enable 1			
6	0	1200	On (Exp	None	0.000	0.000	Time		
7	1.000		0.000	0.000			05/31/2020 00:21:18.000		
8	pc-ism1403c	1.000	1.000	0.000	1.000		05/31/2020		
9	Exposure	440.000	580.000	320.000	440.000		00:21:18.000		
10	Focus	440.000	580.000	320.000	440.000	0.000	0.000		
11	White Balance	0.000	0.000	1200.000	1600.000		0.004		
12	White Balance	0.000	WhiteBal	32.000	0.000				
13							End		
14									
15									
16	Patterns	Row	Col	Angle	Score	Pass/Fail			
17		796.942	456.745	-152.332	63.750	1.000			
18									
19	Calib								
20									
21	výsledek:								
22		X	Y	Angle					
23	Fixture	-213.294	484.187	27.668					
24									
25	TCP communication with robot								
26			1;-213.293	1;-213.293686;484.187164;27.667969					
27	Device	Read	Write						

Obr. F.1: Tabulka kamery nabírání 1. část

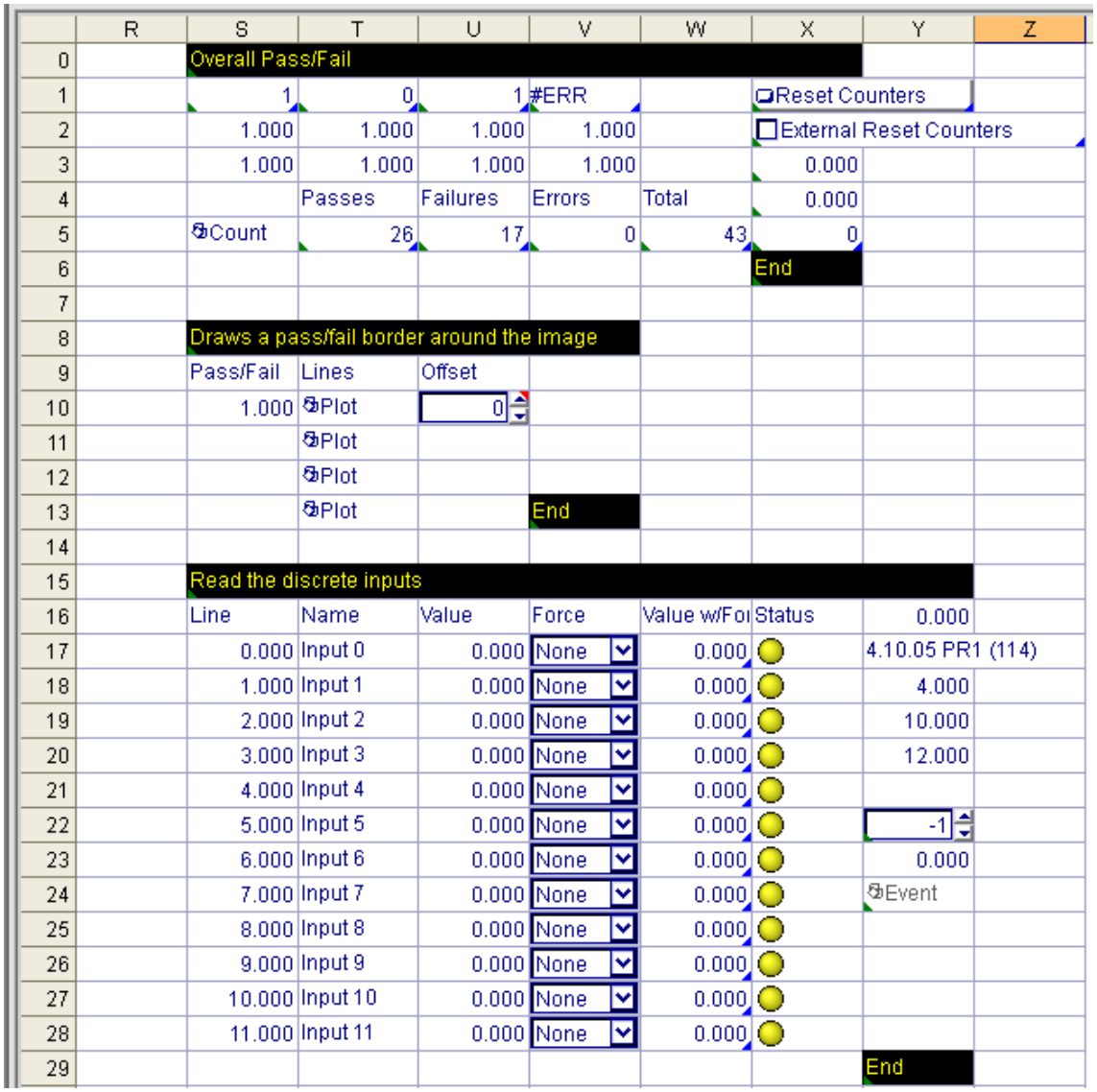
	B	C	D	E	F	G	H	I	J	K	L	M
48	Bod 1				Bod 2				Bod 4			
49	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]
50	602.700	747.000	-266.150	451.820	602.900	1314.000	-366.150	455.750	205.100	746.100	-268.750	381.750
51												
52												
53	Calibracni body 2											
54												
55	Bod 3				Bod 5				Bod 7			
56	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]
57	207.000	1312.100	-368.800	385.700	207.100	183.600	-168.580	377.790	994.000	184.700	-163.460	517.340

Obr. F.2: Tabulka kamery nabírání 2. část



	N	O	P	Q	R	S	T	U
48	Bod 6				Bod 8			
49	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]
50	601.100	181.900	-166.110	447.710	999.100	747.100	-263.390	521.480
51								
52								
53								
54								
55	Bod 9							
56	X kam [px]	Y kam [px]	X rob [mm]	Y rob [mm]				
57	997.300	1311.700	-363.330	525.540				

Obr. F.3: Tabulka kamery nabírání 3. část



	R	S	T	U	V	W	X	Y	Z
0		Overall Pass/Fail							
1		1	0	1	#ERR		Reset Counters		
2		1.000	1.000	1.000	1.000		External Reset Counters		
3		1.000	1.000	1.000	1.000		0.000		
4			Passes	Failures	Errors	Total	0.000		
5		Count	26	17	0	43	0		
6							End		
7									
8		Draws a pass/fail border around the image							
9		Pass/Fail	Lines	Offset					
10		1.000	Plot	0					
11			Plot						
12			Plot						
13			Plot		End				
14									
15		Read the discrete inputs							
16		Line	Name	Value	Force	Value w/Fo	Status	0.000	
17		0.000	Input 0	0.000	None	0.000		4.10.05 PR1 (114)	
18		1.000	Input 1	0.000	None	0.000		4.000	
19		2.000	Input 2	0.000	None	0.000		10.000	
20		3.000	Input 3	0.000	None	0.000		12.000	
21		4.000	Input 4	0.000	None	0.000			
22		5.000	Input 5	0.000	None	0.000		-1	
23		6.000	Input 6	0.000	None	0.000		0.000	
24		7.000	Input 7	0.000	None	0.000		Event	
25		8.000	Input 8	0.000	None	0.000			
26		9.000	Input 9	0.000	None	0.000			
27		10.000	Input 10	0.000	None	0.000			
28		11.000	Input 11	0.000	None	0.000			
29								End	

Obr. F.4: Tabulka kamery nabírání 4. část

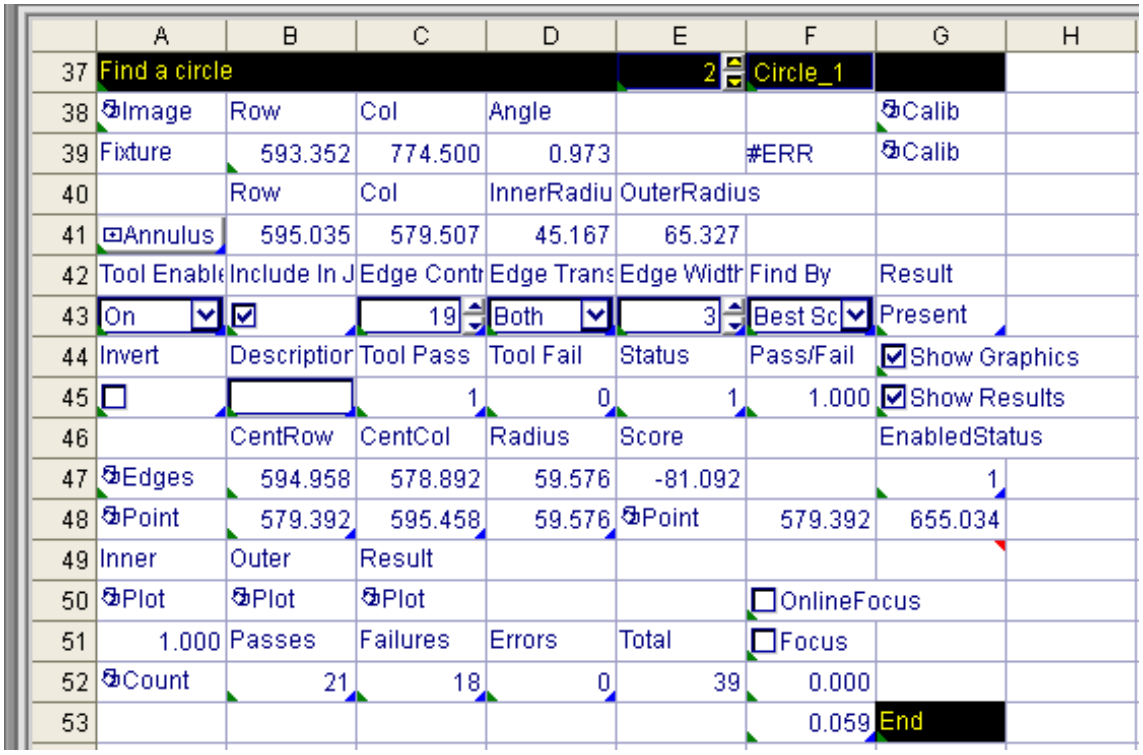
	R	S	T	U	V	W	X	Y	Z
31		Write the discrete outputs							
32		Line	Value	Force	Value w/For	Status	0.000	2.000	
33	Event	0.000	0.000	None	0.000		0.000	4.10.05 PR1 (114)	
34		1.000	0.000	None	0.000		0.000	4.000	
35		2.000	0.000	None	0.000		0.000	10.000	
36		3.000	0.000	None	0.000		0.000	14.000	
37		4.000	0.000	None	0.000		0.000		
38		5.000	0.000	None	0.000		0.000		
39		6.000	0.000	None	0.000		0.000		
40		7.000	0.000	None	0.000		0.000		
41		8.000	0.000	None	0.000		0.000		
42		9.000	0.000	None	0.000		0.000		
43		10.000	0.000	None	0.000		0.000		
44		11.000	0.000	None	0.000		0.000		
45		12.000	0.000	None	0.000		0.000		
46		13.000	0.000	None	0.000		0.000		
47								End	

Obr. F.5: Tabulka kamery nabírání 5. část

G Část tabulky inspekce kamery kontroly

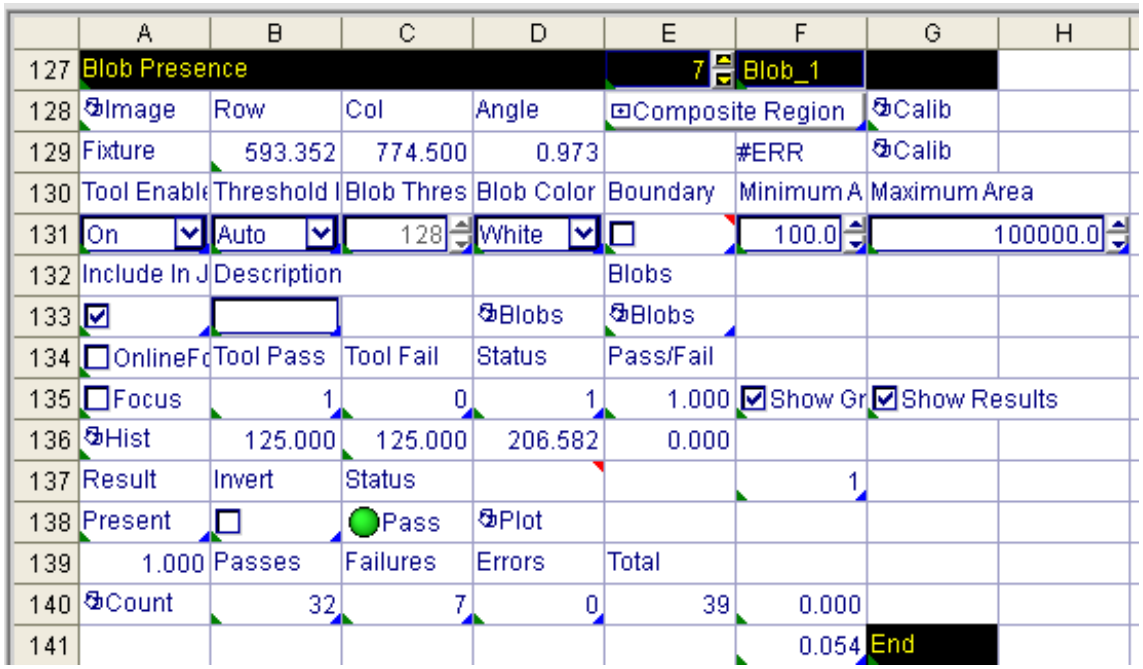
	A	B	C	D	E	F	G	H	I
0	Image								
1	1200.000	0.000	0.000	0.000					
2	Trigger	Trigger Del	Trigger Inte	Exposure (I	Auto-Expos	Max Exposu	Target Image Brightness		
3	Camera	0	500	2.400	Disable	1000.00	10.000		
4		1.000	0.000	1.000	1.000	0.000			
5	Start Row	Number Of	Light Contr	Light Contr	Light Enabl	Light Enable 1			
6	0	1200	On (Exp	None	0.000	0.000	Time		
7	1.000		0.000	0.000			06/01/2020 01:56:23.000		
8	pc-ism140:	1.000	1.000	0.000	1.000		06/01/2020		
9	Exposure	440.000	580.000	320.000	440.000		01:56:23.000		
10	Focus	440.000	580.000	320.000	440.000	0.000	0.000		
11	White Ba	0.000	0.000	1200.000	1600.000		0.001		
12	White Ba	0.000	WhiteBal	32.000	0.000				
13							End		
14									
15	Find a Pattern					1	Pattern_1		
16	Image	Row	Col	Angle			Patterns	Calib	
17	Fixture	0.000	0.000	0.000			Patterns		
18	Show Mc	Row	Col	High	Wide	Angle	Curve	Calib	
19	Model	174.398	356.423	797.671	628.871	359.227	#ERR	#ERR	
20		174.398	356.423	797.671	628.871	359.227	0.000		
21	Search	145.534	249.363	872.638	1029.262	0.000	0.000		
22	Train	External Tra	Train Input	Event					
23	Train		None	0.000	12.536	Image	#ERR		
24	Tool Enable	Include In Job Pass		Accept Thre	Rotation To	Scale Toler	Horizontal (	Vertical Offset	
25	On			85	180		0.000	0.000	
26	Enabled Status			Model Type	Accuracy	Timeout	Result	Description	
27	1	Trained	0.000	Edge M	Medium	5000	(775.0,593.		
28	Patterns	Row	Col	Angle					
29	Point	593.352	774.500	0.973	Plot	593.352	774.500	0.973	
30	Fixture	Fixture	775.000	593.852	-0.973	100.000			
31		Tool Pass	Tool Fail	Status	Pass/Fail				
32	Focus	1	0	1	1.000	Show Gr	Show Results		
33	1.000	Passes	Failures	Errors	Total			Plot	
34	Count	32	7	0	39	0.000			
35						28.919	28.957	End	

Obr. G.1: Tabulka kamery nabírání 1. část



	A	B	C	D	E	F	G	H
37	Find a circle				2	Circle_1		
38	Image	Row	Col	Angle			Calib	
39	Fixture	593.352	774.500	0.973		#ERR	Calib	
40		Row	Col	InnerRadiu	OuterRadius			
41	Annulus	595.035	579.507	45.167	65.327			
42	Tool Enable	Include In J	Edge Contr	Edge Trans	Edge Width	Find By	Result	
43	On	☑	19	Both	3	Best Sc	Present	
44	Invert	Description	Tool Pass	Tool Fail	Status	Pass/Fail	☑ Show Graphics	
45	☐		1	0	1	1.000	☑ Show Results	
46		CentRow	CentCol	Radius	Score		EnabledStatus	
47	Edges	594.958	578.892	59.576	-81.092		1	
48	Point	579.392	595.458	59.576	Point	579.392	655.034	
49	Inner	Outer	Result					
50	Plot	Plot	Plot			☐ OnlineFocus		
51	1.000	Passes	Failures	Errors	Total	☐ Focus		
52	Count	21	18	0	39	0.000		
53						0.059	End	

Obr. G.2: Tabulka kamery kontroly 2. část



	A	B	C	D	E	F	G	H
127	Blob Presence				7	Blob_1		
128	Image	Row	Col	Angle	Composite Region		Calib	
129	Fixture	593.352	774.500	0.973		#ERR	Calib	
130	Tool Enable	Threshold I	Blob Thres	Blob Color	Boundary	Minimum A	Maximum Area	
131	On	Auto	128	White	☐	100.0	100000.0	
132	Include In J	Description			Blobs			
133	☑			Blobs	Blobs			
134	☐ OnlineFc	Tool Pass	Tool Fail	Status	Pass/Fail			
135	☐ Focus	1	0	1	1.000	☑ Show Gr	☑ Show Results	
136	Hist	125.000	125.000	206.582	0.000			
137	Result	Invert	Status			1		
138	Present	☐	Pass	Plot				
139	1.000	Passes	Failures	Errors	Total			
140	Count	32	7	0	39	0.000		
141						0.054	End	

Obr. G.3: Tabulka kamery kontroly 3. část

H Obsah archivu Prilohy.zip

- `/` ..... kořenový adresář
  - `Prilohy` ..... Složka s přílohami
    - `Manipulator` ..... Složka se zdrojovými soubory manipulátoru
      - `inst_prisavky.installation` ..... Instalace manipulátoru
      - `inst_prisavky.variables` ..... Pomocný soubor instalace manipulátoru
      - `prog_trideni.script` ..... Pomocný soubor programu manipulátoru
      - `prog_trideni.txt` ..... Pomocný soubor programu manipulátoru
      - `prog_trideni.urp` ..... Program manipulátoru
      - `scr_trideni.script` ..... Skriptovací soubor manipulátoru
    - `Elektricke_schema.pdf` ..... Elektrické schéma
    - `Kamera_Kontrola.job` ..... Inspekce kontrolní kamery
    - `Kamera_Nabirani.job` ..... Inspekce kamery nabírání
    - `Pneumaticke_schema.pdf` ..... Pneumatické schéma
    - `scr_trideni.pdf` ..... Zdrojový kód skriptu se zvýrazněnou syntaxí