VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ABB ROBOTSTUDIO - VZOROVÉ PŘÍKLADY V C#
ABB ROBOTSTUDIO - EXAMPLES IN C#

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Mittaš**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. et Ing. Stanislav Lang, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Tomáš Mittaš**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**Ing. et Ing. Stanislav Lang, Ph.D.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

ABB RobotStudio – vzorové příklady v C#

Stručná charakteristika problematiky úkolu:

Student se v rámci práce seznámí s prostředím RobotStudio, v němž vytvoří scénu s robotem a bude programovat jeho pohyby. Využito bude i programování pohybů v jazyce C#. Veškeré postupy pak zpracuje do stručných návodů. Vytvořené návody budou tvořit přílohu práce a budou sloužit pro podporu výuky.

Cíle bakalářské práce:

Proveďte stručnou rešerši v oblasti robotiky (zaměřte se na stacionární roboty).
Nastudujte řešení průmyslové robotiky v pojetí společnosti ABB.
Vytvořte scénu s robotem v prostředí RobotStudio (přiložte vlastní návod).
Vytvořte jednoduché sekvence pohybů v prostředí RobotStudio (přiložte vlastní návod).
Ve svých příkladech využijte i programování pohybů v jazyku C# (přiložte vlastní návod).
Vybrané pohybové programy otestujte i na reálném robotu (v laboratoři).
Zhodnoťte použité prostředí z pohledu uživatelské přívětivosti.

Seznam doporučené literatury:

ABB Group [online]. [cit. 2018-09-11]. Dostupné z: https://new.abb.com/cz

B&R Automation [online]. [cit. 2018-09-10]. Dostupné z: http://www.br-automation.com/

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Táto bakalárska práca sa zaoberá programovaním pohybov robota od spoločnosti ABB v jazyku C#. Cieľom je ukázať aj netradičné spôsoby použitia jazyka C#. V úvode sú v krátkosti popísané prostriedky programovania a simulácie robota. Ďalšie kapitoly sa venujú histórii robotov, ich rozdeleniu, senzorike a programovaniu a taktiež predstaveniu spoločnosti ABB a ich produktov, ktoré sa v práci používajú. Na záver je porovnanie a zhodnotenie oboch používaných prostredí. Práca obsahuje aj návody na prípravu aplikácie a simulácie a tiež demonštračné video fungujúceho demo projektu v simulácii aj na reálnom robotovi.

ABSTRACT

This bachelor thesis is concerned about programming movements of a robot developed by company ABB in a programming language C#. The aim of thesis is to show unconventional ways of using C# programming language. In the introduction are briefly described the instruments for programming and robot simulation. Following chapters are devoted to history of robots, their categories, sensors and programming and also to introduction of company ABB and their products, which are used in this thesis. In the conclusion is comparison and analysis of both used environments. Thesis also contains instructions for making an application and simulation and also a demonstration video of functional demo project in simulation and on real robot.

KLÚČOVÉ SLOVÁ

robot, ABB, ABB RobotStudio, Microsoft Visual Studio, jazyk C#, aplikácia, prostredie, programovanie, RAPID

KEYWORDS

robot, ABB, ABB RobotStudio, Microsoft Visual Studio, C# language, application, environment, programming, RAPID

ÚSTAV AUTOMATIZACE A INFORMATIKY

BIBLIOGRAFICKÁ CITÁCIA

MITTAŠ, T. *ABB RobotStudio - vzorové příklady v C#.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2019. 57 s. Vedoucí bakalářské práce Ing. et Ing. Stanislav Lang, Ph.D..

POĎAKOVANIE

Chcel by som poďakovať svojmu vedúcemu práce Ing, et Ing. Stanislavovi Langovi Ph.D za pomoc, ochotu, rady a vrelý prístup pri zostavovaní tejto práce. Ďalej by som chcel poďakovať pánovi Ing. Romanovi Parákovi za konzultácie a jeho rady. Na záver by som chcel poďakovať svojej rodine a priateľke za ich podporu.

ÚSTAV AUTOMATIZACE A INFORMATIKY

ČESTNÉ PREHLÁSENIE

Prehlasujem, že táto práca je mojím pôvodným dielom, že som ju spracoval samostatne pod vedením Ing. et Ing. Stanislava Langa Ph.D a s použitím literatúry uvedenej v zozname použitej lteratúry.

V Brne dňa 24.5.2019 .....

Tomáš Mittaš

OBSAH

1 ÚVOD

V dnešnej dobe patrí robotika medzi najdôležitejšie odvetvia priemyslu. Jednou z hlavných otázok v oblasti robotiky je dnes aj programovanie robotov. Existuje mnoho spôsobov či už to klasických alebo netradičných. V spoločnosti ABB je jedným z netradičných spôsobov programovania použitie jazyka C# od firmy Microsoft. Použitie tohoto vysokoúrovňového objektovo orientovaného programovacieho jazyka môže pre veľa programátorov znamenať priehľadnejšie, efektívnejšie a netradičnejšie riešenie pri vývoji robotických aplikácii. Ďalšou možnou výhodou je prepojenie produktov s firmou Microsoft, ktorá je jednou s vedúcich spoločností v oblasti informatiky.

Cieľom tejto bakalárskej práce je ukázať rozmanitosť jazyka C# aj v aplikáciách pre tento jazyk netradičných ako je robotika. Prvá časť práce bude venovaná robotike, jej histórii, rozdeleniu robotov, senzorike robotov a ich programovaniu. Pojednávať bude hlavne o stacionárnych robotoch. Ďalej bude v práci v krátkosti predstavená firma ABB a jej divízia Robotiky, vrátane jej produktov, medzi ktoré patrí aj robot IRB120 a jeho kontrolér IRC5, ktorý bude slúžiť na praktickú demonštráciu programu. Spomenutá bude aj divízia Priemyselnej automatizácie.

Spoločnosť ABB a jej ABB RobotStudio používa na podporu programovania v jazyku C# rozšírenie PC SDK, vrátane rady knižníc, ktoré uľahčujú prácu s robotom. Samotné programovanie robota v jazyku C# bude prebiehať vo vývojovom prostredí Microsoft Visual Studio za pomoci knižníc z modulu PC SDK.

Ďalej sa táto práca bude venovať samotnému ABB RobotStudio a práci s ním. Bude ukázané ako vytvoriť scénu s robotom v prostredí ABB RobotStudio a taktiež bude popisovať tvorbu jednoduchých sekvencií pohybov v ABB RobotStudio.

Posledná časť tejto práce sa bude venovať zhodnoteniu prostredia ABB RobotStudio a porovnaniu práce s robotom za použitá tohoto prostredia oproti použitiu jazyka C# a Microsoft Visual Studio.

2 ROBOTIKA

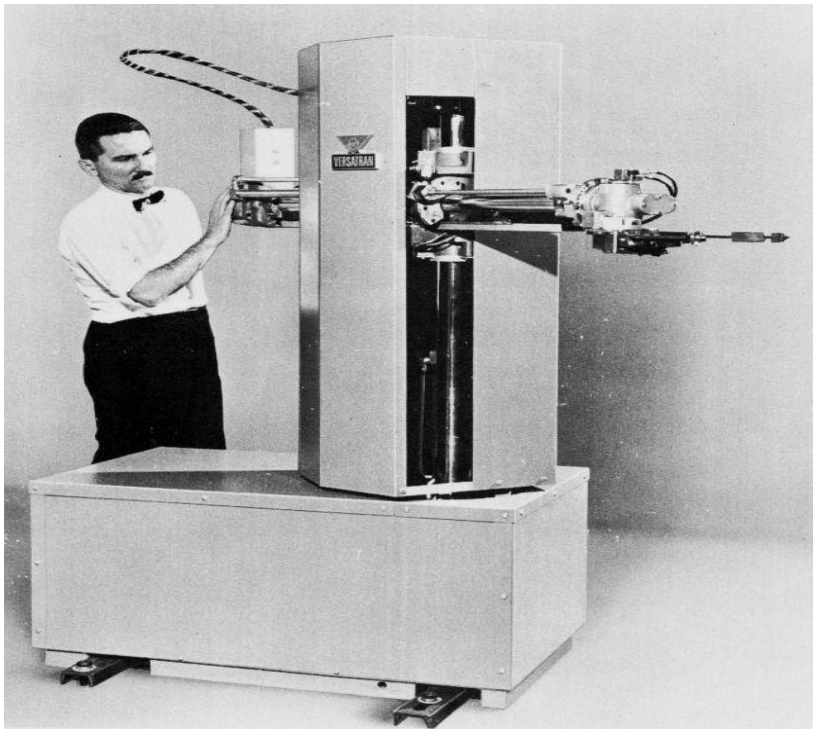
Robotika je dnes jedným z najrýchlejšie sa rozvíjajúcich odborov. V súčasnosti sa s robotmi môžeme stretnúť skoro v každom odvetví priemyslu. Roboty sa používajú napríklad všade tam, kde nahrádzajú stereotypnú ľudskú činnosť, alebo v prostredí pre človeka nebezpečnom. Robotiku môžeme rozdeliť na: [2]
Teoretickú – táto časť sa zaoberá teóriou, senzorikou, navigáciou, simuláciou a umelou inteligenciou.
Technickú – zahŕňa výpočty, návrhy, prevádzku, údržbu, no tiež vývoj a výskum subsystémov robotov.
Aplikačnú – v tejto časti sa rieši nasadzovanie robotov do výroby a ich efektívnosť.

2.1 Stručná história robotiky

Po prvýkrát bolo slovo robot v histórii použité českým spisovateľom Karlom Čapkom v roku 1920 v jeho hre R.U.R. Prvé počiatky robotiky však môžeme hľadať už v starej Číne a Grécku v podobe samohybných mechanizmov, ktoré však nemali reálne využitie. V 1. storočí nášho letopočtu Herón Alexandrijský navrhol a skonštruoval samočinné javiskové zariadenia a taktiež parný stroj, ktorý slúžil na dávkovanie vody po vhodený mince. Jedným z ďalších veľkých priekopníkov počiatkov robotiky je Leonardo da Vinci, ktorý zostrojil mechanického leva, ktorý bol schopný po zdvihnutí laby zarevať. V Európe môžeme v časoch stredoveku badať automaty v podobe mechanických hodín a orlojov. Aj keď sa v priemyselnej výrobe hojne využívajú stroje, ktoré plnia určité ľudské funkcie, tak ich nenazývame robotmi ale automatmi. Dôvod je taký, že automaty sa veľmi málo podobajú na človeka a ich funkcie sú dosť úzko špecializované. Takéto automaty vo výrobe napríklad používal už Henry Ford pri svojej pásovej výrobe automobilov. Firma Unimation v roku 1958 prišla s robotom UNIMATE. [1,8]

Obr. 1 Priemyselný robot typu UNIMATE od firmy Unimation [8]

Americká firma American Machine and Foundry Corporation prišla v roku 1961 na trh s viacúčelovým automatom, ktorý nazvali priemyselný robot VERSATRAN. [9]



Obr. 2 Priemyselný robot typu VERSATRAN [9]

V bývalom Československu nebola robotika na rozdiel od kybernetiky považovaná za pavedu, a preto bola podporovaná od začiatku výskumu. Napríklad v Prešove bol výskum a vývoj priemyselných robotov VUKOV Prešov. Ďalšie podniky boli napríklad ČZM Strakonice alebo URSST Piešťany. Po privatizácií a rozpade republiky sa trhu chopili medzinárodní výrobcovia ako ABB, REIS, KUKA, FANUC, PANASONIC, UNIMATE a ďalší. [1]

2.2 Definícia robota

Robot ako pojem predstavuje zariadenie, ktoré má väčšinu z nasledujúcich vlastností a teda: [1,2]

1. Manipulačná schopnosť – schopnosť uchopovať objekty, prenášať ich a to aj v zmysle práce s nimi, teda pracovať ako výrobný robot.
2. Univerzálnosť – zariadenie neslúži iba na jeden účel, ale pokiaľ zmeníme program, chápadlá alebo nástroje, je možné použiť zariadenie aj na iné účely na inom pracovisku.
3. Väzba s prostredím – možnosť vnímania pomocou senzorov napodobňujúcich ľudské zmysly. Napríklad vizuálna väzba, ktorá odpovedá ľudskému zraku, akustická odpovedajúca sluchu a podobne.
4. Autonómnosť chovania – schopnosť vykonávať automaticky zložitú postupnosť úloh daného programu. Dôležitý je hlavne prípad, kedy tento program nie je pevný (daný konštrukciou), ale je voliteľný buď človekom alebo automaticky vlastným zariadením.
5. Priestorová sústredenosť jednotlivých zložiek (integrovanosť - teda opak voči stavebnicovej konštrukcii) pokiaľ je to možné do jedného celku. Toto sa však

netýka riadiaceho systému, hlavne teda počítače, ktorý môže robota riadiť aj bezdrôtovo. Pokiaľ je robot integrovaný do jedného celku dá sa ľahšie transportovať. Niekedy sa dá požadovať aby bol robot mobilný.

6. Označenie „robot“ je teda vhodné pre manipulačné mechanizmy vykonávajúce úlohy blížiace sa typickým úlohám človeka a tieto úlohy predvádza s „ľudskou“ obratnosťou. Robotom môžeme označiť aj mechanizmy manipulačného charakteru riadené za pomoci počítača.

2.3 Kritéria pre posudzovanie robotov

Pri posudzovaní manipulačných zariadení, priemyselných robotov a robotov ako takých sa uplatňuje rada aspektov a to predovšetkým:

- Morfológia robota
- Veľkosť a hmotnosť
- Veľkosť obsluhovaného priestoru
- Hmotnosť bremena
- Dosahovaná presnosť
- Rýchlosť pohybu
- Spôsob pohonu
- Druh servopohonov
- Spôsob a rozsah vnímania
- Spôsob riadenia a komunikácia s okolím

Morfológia je odvodená od kinematickej štruktúry robota v závislosti na použitých konštrukčných prvkoch. Počet stupňov voľnosti je tiež významnou veličinou robota, obyčajne majú roboty 5 až 6 stupňov voľnosti. *Veľkosť a hmotnosť* je daná stavbou priemyselného robota a jeho použitím, snahou je aby bola čo najnižšia pri čo najvyššej pevnosti a tuhosti. *Veľkosť obsluhovaného priestoru* závisí od veľkosti priemyselného robota a tiež od jeho kinematickej štruktúry. Obsluhovaný priestor o veľkosti jedného metra kubického sa považuje za základnú veľkosť. *Hmotnosť bremena* určuje, na akú prácu môžeme použiť robota. Započítava sa sem aj hmotnosť úchopného mechanizmu, takže hmotnosť čistého bremena bude nižšia. *Dosahovaná presnosť* je z veľkej časti závislá na zaťažení robota. To, že väčšina robotov sú otvorené kinematické mechanizmy spôsobuje, že ich výsledná presnosť je vždy nižšia. *Rýchlosť pohybu* je opäť z veľkej časti závislá na zaťažení robota, preto zaťaženie treba vždy kontrolovať. Ďalším aspektom, od ktorého sa odvíja rýchlosť pohybu je druh použitých pohonov. Dosiahnuť maximálnu rýchlosť a presnosť je tým náročnejšie, čím väčšie je okamžité zaťaženie robota. Žiadúce je dosiahnuť čo najvyššiu rýchlosť pri zachovaní garantovanej presnosti. *Spôsob pohonu* závisí od potreby a možností užívateľa. Pri priemyselných robotoch a manipulátoroch sa používajú pohony mechanické, pneumatické, hydraulické, elektrické a kombinované. *Druh servopohonov* a spôsob odmerania záležia od optimálnej voľby pohonov a sú dôležité hlavne pre konštruktérov. *Spôsob a rozsah vnímania* alebo vnímací subsystém predstavuje vybavenie robotov potrebnými senzormi tak, ako to odpovedá jednotlivým uvedeným generáciám robotov. *Spôsob riadenia a komunikácia* s okolím je závislá na hardware a software riadiaceho systému, ktorý komunikuje s akčným systémom vrátane podsystému. Autonómnosť robota je výsledkom komplexného prepojenia všetkých úrovní prvkov akčného aj kognitívneho systému. [1]

2.4 Rozdelenie robotov

Roboty môžeme klasifikovať podľa rôznych kritérií. Kedysi bolo rozdelenie robotov postavené na odlišnostiach manipulátorov a robotov z hľadiska programovania a riadenia. Používali sa (a naďalej sa ešte aj používajú) nasledujúce pojmy: [2]

- Manipulátor (jednoúčelový, s pevne daným programom)
- Synchrónny manipulátor (master - slave manipulátor)
- Robot (manipulátor obsahujúci premenný program)
- Adaptívny robot (reaguje na zmeny pracovnej scény)
- Kognitívny robot (obsahujúci určitú úroveň umelej inteligencie)

Dnes klasifikujeme podľa rôznych kritérií: [1,2]

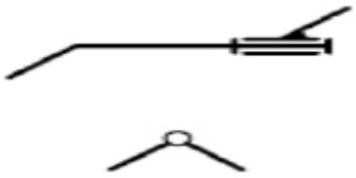
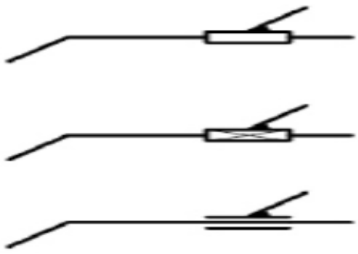
1. Počet stupňov voľnosti:

- Univerzálny robot – má šesť stupňov voľnosti, ktoré jednoznačne vymedzujú polohu a orientáciu manipulácie v kartézskom súradnicovom systéme.
- Redundantný robot – disponuje viac ako šiestimi stupňami voľnosti, dokáže sa lepšie pohybovať v stiesnenom priestore a má väčšiu voľnosti pri obchádzaní prekážok.
- Deficitný robot – tento robot má menej ako šesť stupňov voľnosti (napríklad roboty Scara, ktoré obsahujú 3 až 4 stupne voľnosti).

2. Kinematická štruktúra:

- Sériová
- Paralelná
- Hybridná

Priemyselné roboty najčastejšie využívajú posuvné a rotačné kinematické dvojice. Kinematické dvojice sú vlastne dva geometrické prvky alebo dve sústavy prvkov, v ktorých sa dochádza k styku povrchov dvoch členov mechanizmu.

Kinematická dvojica	Počet stupňov voľnosti	Značenie	Trieda dvojice	Zobrazenie
rotačná	1	R	5	
posuvná	1	T	5	

Obr. 3 Kinematické dvojice priemyselných robotov a manipulátorov [2]

Roboty so sériovou kinematikou majú otvorený kinematický reťazec manipulátoru, ich akčný systém pozostáva z viacerých binárnych členov, ktoré sú za sebou zostavené prostredníctvom kinematických dvojíc a sú navzájom viazané. Sériové reťazce majú pomerne jednoduchú kalibráciu a riadenie, pretože dokážeme riadiť jednotlivé osi

samostatne, avšak majú vyššiu hmotnosť pohybujúcich sa častí, nižšiu rýchlosť a zrýchlenie. Ďalšou nevýhodou je, že majú pomerne nákladnú a zložitú výrobu. [1,2]
Roboty s kinematikou paralelnou majú uzavretý kinematický reťazec manipulátoru. Obrábací nástroj je uložený na plošinu, ktorá je zavesená na dĺžkovo premenlivých a kĺbovo uchytených závesoch umožňujúcich natáčanie vzhľadom k obrobku. Paralelné reťazce majú vysoké zrýchlenie a rýchlosť nižšiu hmotnosť pohybujúcich sa častí, no majú komplikovanejšiu kalibráciu a ich riadenie je komplikovanejšie, keďže sa musí mechanizmus riadiť ako celok. [1,2]
Hybridná kinematika kombinuje oba typy kinematických reťazcov. [1,2]

Obr. 4 Kolaboratívny robot YuMi od firmy ABB(sériová kinematika) [6]

Obr. 5 robot IRB 360 FlexPicker (paralelná kinematika) [7]

3. Podľa druhu pohonov:
- Hydraulické
- Pneumatické
- Elektrické
- Kombinované

Pri *hydraulických pohonoch* sú výhodami veľká účinnosť a spoľahlivosť, vysoká tuhosť, výhodné dynamické vlastnosti vďaka nízkej hmotnosti pohyblivých častí a možnosť plynulého riadenia a dosiahnutia žiadanej polohy. Sú používané pre vysoké hmotnosti. Medzi nedostatky patrí napríklad zmena viskozity pracovných kvapalín v dôsledku zmeny teploty a tiež horľavosť týchto kvapalín. Pri tomto druhu pohonov je vyžadovaný samostatný a oddelený energetický blok. [1,2]
Pneumatické pohony sú vhodné pre manipulátory a roboty menších výkonov (do 1kW). Sú čistejšie a bezpečnejšie ako hydraulika. Majú vysokú rýchlosť, no nie vždy využiteľnú kvôli problémom s brzdením. Používajú sa tu predovšetkým pneumatické valce, čo sú vlastne motory s priamočiarym pohybom. [1,2,27]
V súčasnosti prevažujúce sú *elektrické pohony*, pretože sú spoľahlivé a dobre regulovateľné, vysoko presné a rýchle. Efektívnosť pri riadení pohybu je zaistená pomocou uzavretej regulačnej slučky, obsahujúcej polohovú, rýchlostnú a prúdovú spätnú väzbu. Pri jednoduchých manipulátoroch je možnosť práce v otvorenej regulačnej slučke, napríklad narážkovým riadením. To znamená, že elektromotor nie je riadený, a rozsah pohybu je určený pomocou koncových snímačov. [1,2]

Spojenie rôznych druhov pohonov je označované ako *kombinované*. Patrí sem napríklad elektrohydraulický pohon, čo je spojenie elektrického riadenia a výhodných vlastností hydromotoru. [1]

4. Podľa vykonávaných činností a oblastí nasadenia: [1,2]

- Priemyselné roboty – slúžia pre činnosti spojené s výrobou rôznych produktov.
- Servisné roboty – používané pre obslužné činnosti v domácnosti, zdravotníctve, stavebníctve a podobne. Rýchlo rastúci dopyt. Typickou vlastnosťou je ich mobilita.

5. Podľa geometrie pracovného priestoru: [1,23,24]

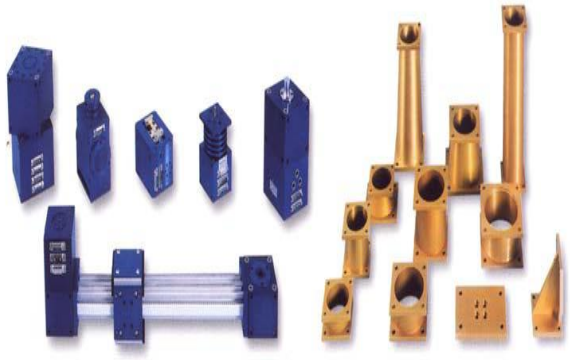
- Kartézske – pracovný priestor robota má tvar hranola. Kinematické štruktúry konajú pohyb po troch lineárnych osiach kolmých na seba.
- Cylindrické – vytvárajú také kinematické štruktúry, kedy jedna os je rotačná a dve slúžia na lineárne posuny vo dvoch osiach. Tento priestor má tvar podkovy s obdĺžnikovým alebo štvorcovým prierezom.
- Sférické – tvorené kinematickými štruktúrami definovanými dvoma osami rotačnými a jednou osou pre lineárny posun.
- Angulárne – kinematické štruktúry sú tu definované rotačným pohybom vo všetkých troch osiach. Pracovný priestor má nepravidelný tvar.

6. Podľa kompaktnosti konštrukcie a funkčnej autonómnosti pohybových jednotiek robotov: [1,2,24]

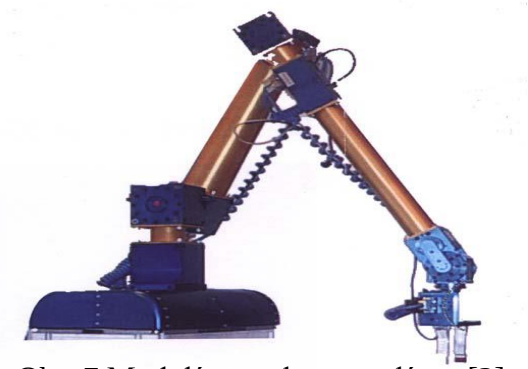
- Univerzálne – možnosť nasadenia na veľkú triedu úloh. Pozostávajú z navzájom previazaných pohybových jednotiek, ktoré nie sú autonómne funkčné. Tieto roboty sú konštrukčne zložité a drahé.

Obr. 6 Univerzálny robot IRC 2400 [5]

- Modulárne – každá polohovacia jednotka je samostatne funkčná a pomocou viacerých jednotiek dokážeme zložiť štruktúru prispôsobenú na danú úlohu. Výrazne lacnejšie. Polohovacie jednotky vyrábané vo väčších sériách.



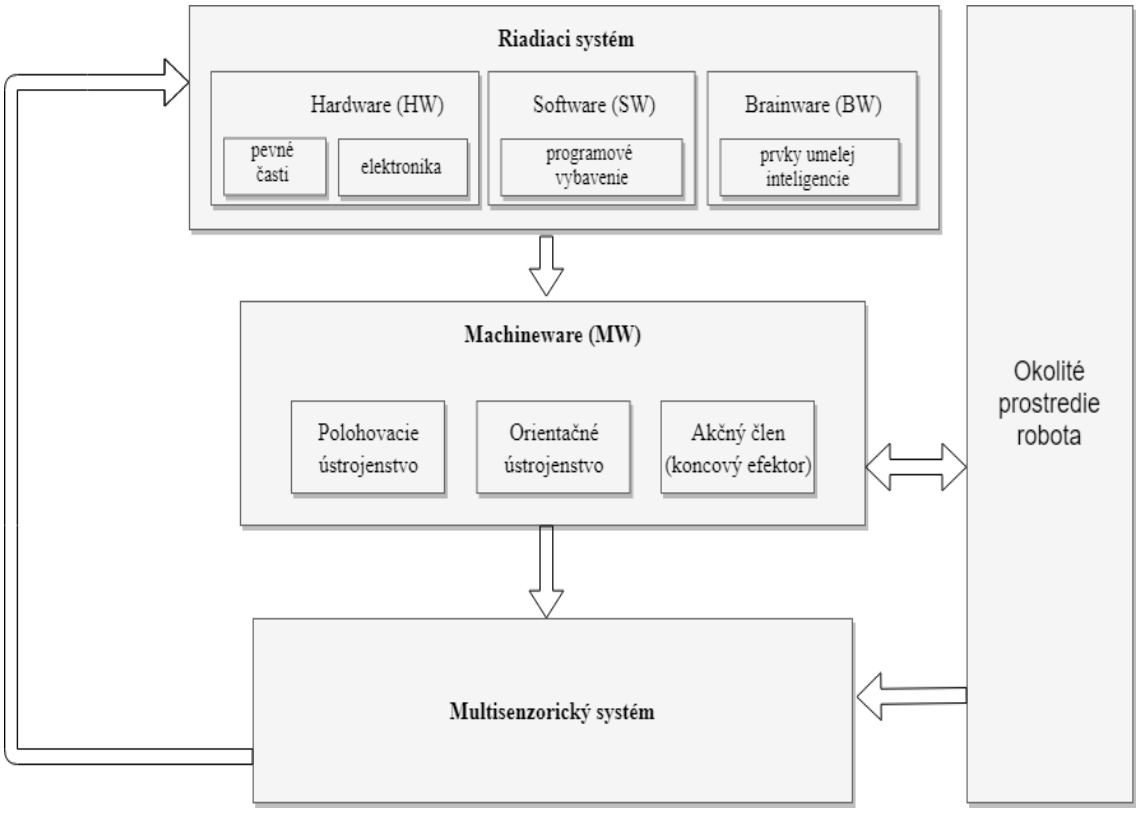
Obr. 6 Moduly pre vytvorenie štuktúry [2]



Obr. 7 Modulárny robot angulárny [2]

2.5 Senzorika robotov

Keďže táto práca bude pojednávať o programovaní priemyselného robota, bolo by dobré spomenúť ako robot získava dáta, ktoré využívame pri programovaní. Samotná problematika senzorov pri priemyselných robotoch je veľmi obsiahla, a tak tu bude popísaná len v krátkosti. Pokiaľ teda chceme, aby robot reagoval na zmenu prostredia a parametrov, ktoré sleduje, musíme ho vybaviť okrem riadiaceho systému aj senzormi, ktoré tomuto systému poskytujú informácie o sledovaných udalostiach, parametroch a prostredí. Pohyby robota sú teda riadené programom v riadiacom systéme za využívania informácii z jednotlivých senzorov. Robota, ktorý je schopný reagovať takýmto spôsobom na zmeny nazývame adaptívnym robotom. Adaptivita robota je charakterizovaná hlavne kvalitou jeho senzorického vybavenia a taktiež schopnosťami riadiaceho systému reagovať na zmeny okolia. Môžeme povedať, že každý adaptívny priemyselný robot sa skladá z troch hlavných častí a to z riadiaceho systému obsahujúceho programové vybavenie, z mechanického systému, ktorý je charakterizovaný napríklad akčnými členmi a zo senzorického systému, ktorý sleduje stav samotného priemyselného robota a jeho okolie. Informácie, ktoré nadobudne senzorický systém sú posielané riadiacemu systému za pomoci spätnej väzby. [1]



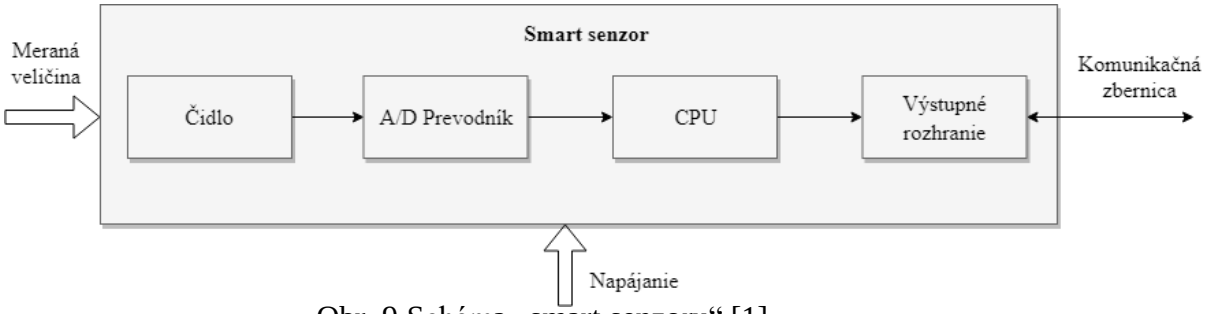
Obr. 8 Adaptívny priemyselný robot ako sústava konštrukčných uzlov, skupín a súčastí [1]

Môžeme teda povedať, že aby bol priemyselný robot schopný sledovať svoje okolie a aj svoje vlastné pohyby, potrebuje niečo, ako ľudské zmysly. Týmito zmyslami sú pri priemyselnom robotovi senzory. Senzory sú omnoho presnejšie ako ľudské zmysly a dokážu snímať informácie vo väčšom spektre (napríklad optickom či akustickom). Nevýhodou oproti ľudským zmyslom je však to, že daný senzor dokáže snímať iba veličinu, pre ktorú bol zostrojený. Preto treba robota vybaviť množstvom potrebných senzorov, aby bol schopný zbierať čo najviac informácii. [1]

2.5.1 Definícia a delenie

Vstupným blokom meracieho reťazca, ktorý je v priamom styku s meraným prostredím je senzor. Citlivá časť senzoru, pomocou ktorej sa sníma meraná veličina (často neelektrická) sa nazýva čidlo. Toto čidlo veličinu nie len sníma, ale aj ju vo väčšine prípadov prevádza na elektrickú, ktorá je vhodnejšia pre ďalšie spracovanie. Senzor sa skladá z troch častí : [1]

- Vstupná časť – umožňuje vstup meranej veličiny, ktorú prevádza na elektrický signál. Zároveň senzor chráni proti pôsobeniu nežiaducich rušení.
- Vnútorná časť – spracúva vstupný elektrický signál a kompenzuje vplyv okolia. Patrí sem prevodník A/D a D/A, pamäť, komparátor, generátor a mikroprocesor.
- Výstupná časť – tu sa rieši komunikácia senzoru s riadiacou jednotkou.



Obr. 9 Schéma „smart senzoru“ [1]

S dnešným rozvojom mikroprocesorov vznikla možnosť integrovať ich do snímačov. Tieto „smart snímače“ majú výhodu v presnosti merania a odľahčujú riadiaci systém, pretože preberajú veľkú časť jeho činnosti. Umožňujú tiež pripojenie k internetu. Nevýhodou je však vyššia cena a obmedzené použitie v ťažkých podmienkach.

Senzory podľa umiestnenia delíme na vnútorné a vonkajšie. Vnútorné súvisia so samotným robotom a poskytujú informácie o zrýchlení, polohe a rýchlosti pohyblivých častí, o prúde vstupujúcom do pohonu a tak podobne. Nazývajú sa tiež propriocepčné. Vonkajšie sa sústredia na prostredie, v ktorom je robot umiestnený. Môžu byť realizované ako koncové efektory, ktoré snímajú prítomnosť objektu, alebo sú uložené mimo robota samostatne, napríklad ako kamera, ktorá sníma prostredie okolo robota. Inak sa nazývajú ako exterocepčné snímače. [1,10]

Podľa styku s meraným objektom poznáme senzory dotykové, bezdotykové, aktívne a pasívne. *Dotykové* nazývané aj taktilné, potrebujú k snímaniu dotyk s meraným objektom, sú to napríklad koncové spínače. *Bezdotykové* alebo proximitné ,nepotrebujú dotyk, sú to napr. optické a indukčné. *Aktívne* snímače sa chovajú ako zdroj energie pri pôsobení neelektrickej veličiny, teda vysielajú energiu do prostredia a merajú jej zmenu v rámci kontaktu s prostredím. Radia sa sem napríklad piezoelektrické alebo indukčné snímače. *Pasívne* prijímajú energiu z prostredia, takže pôsobením neelektrickej veličiny menia nejaký svoj parameter. Patria sem napríklad odporové snímače alebo kapacitné snímače. [1,10]

Podľa výstupného signálu delíme snímače na analógové a digitálne. *Analógové* snímače sú také, ktoré udržujú nepretržitý spojitý signál a sú vhodné pri meraní teploty alebo tlaku. Pri *digitálnych* sa signál mení skokovo a diskrétne v čase, vhodné sú napr. pri enkodéroch. [1,10]

Krátky popis niektorých dôležitých snímačov uplatňujúcich sa pri robotoch:
Indukčné snímače slúžia na snímanie kovových objektov na malé vzdialenosti, majú dlhú životnosť, sú odolné voči rušeniu, vysokým teplotám a vplyvom prostredia a sú bezdotykové. *Kapacitné senzory* sú charakterizované dlhou životnosťou a odolnosťou voči rušeniu a teplotným výkyvom. Sú bezdotykové. Pri vodivých látkach dosahujú

väčšiu vzdialenosť snímania ako pri nevodivých. *Magnetické snímače* rozlišujeme jazýčkové alebo Hallove. Využívajú sa na meranie a detekciu pohybu, priblíženia a umiestnenia. Bývajú nahrádzané indukčnými, pretože je problém ich zminimalizovať. Hallove majú väčšiu odolnosť, životnosť a presnosť ako jazýčkové. *CCD senzory* dokážu rozpoznávať tvar a orientáciu súčastí, alebo môžu vykonávať optickú kontrolu. *Elektromechanické snímače* často fungujú ako tlačidlo alebo spínač, sú jednoduché a bez prevodníkov. *Tenzometre* fungujú na princípe činnosti na zmene odporu, sú určené na sledovanie deformácii. *Inkrementálne a absolútne rotačné snímače*, nazývané tiež *enkodéry*, sa používajú ako spätná väzba pri riadení servopohonov a robotov a na zistenie uhlu natočenia. Využívajú kotúč s Grayovým kódom. [1,10]

2.6 Programovanie robotov

Táto práca sa zameriava na programovanie robota ABB v prostredí RobotStudio. Táto kapitola by mala byť braná ako menší úvod do problematiky programovania a bude pojednávať o programovaní robotov.
V súčasnosti je mnoho spôsobov ako programovať priemyselné roboty. Dnes najpoužívanejšia metóda je *online metóda*. Pri tejto metóde môže obsluha programovať robota priamo na pracovisku pomocou ovládacieho panelu (*teach-pendant*, *pendant*). Princíp spočíva v tom, že programátor pomocou uživatelského rozhrania navádza robota alebo programuje konkrétnu aplikáciu priamo na pracovisku, kde je robot fyzicky prítomný. Užívatelské rozhranie je vyrábané v dvoch variantách a to na výšku a na šírku. [1,22]

Obr. 10 Uživatelské rozhranie (*teach pendant*) na výšku (vpravo) a na šírku (vľavo) [11]

Ďalším typom programovania je *offline programovanie*, ktoré spočíva v práci so software systémami. Tie umožnia vytvoriť 3D návrh robotizovaného pracoviska vo virtuálnom prostredí založenom na kinematickom, poprípade aj dynamickom

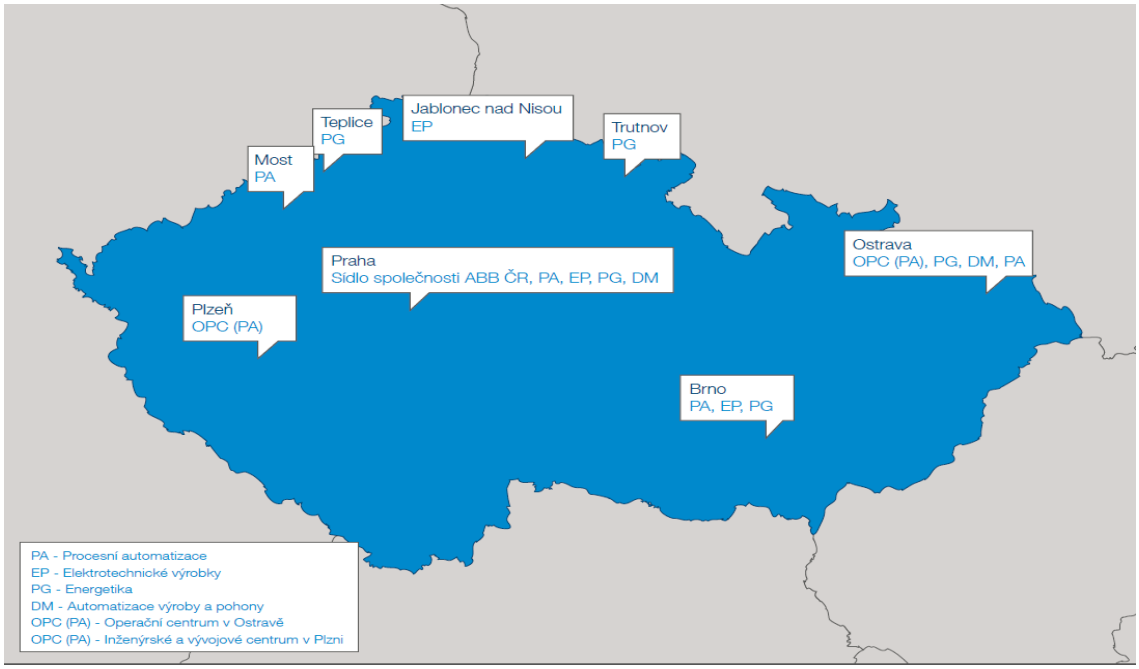
simulačnom modely vybraného robota. Ďalšou charakteristickou vlastnosťou tejto metódy je možnosť zapisovať body, definovať dráhy a iné činnosti robota s ohľadom na konkrétnu aplikáciu. Chod robotickej bunky dokážeme neskôr optimalizovať na základe simulačných činností. Je tu aj možnosť sledovať celkový pracovný čas robota, vykresľovať pracovné priestory robota, testovať dosah robota vzhľadom k bodom, dráham, komponentom a podobne. Otázka bezpečnosti je tu riešená za pomoci automatickej detekcie kolízii. Vytvorené body a dráhu robota je možné exportovať do programu, ktorý v plnom rozsahu rešpektuje syntax programovacieho jazyka daného robota (napríklad pre ABB jazyk RAPID alebo pre KUKA roboty jazyk KRL). V praxi býva väčšinou offline program modifikovaný na pracovisku aby presne odpovedal prostrediu a rozmiestneniu rôznych komponentov. Jeho použitie je vhodné pre frézovanie, brúsenie, leštenie, lakovanie a podobne. Menej často sa používa napríklad pri zváraní či manipulácii. [1,22]

3. SPOLOČNOSŤ ABB

Spoločnosť ABB je svetovou firmou, ktorá pôsobí v oblasti energetiky a automatizácie. Je členená do niekoľkých divízii a teda Elektrotechnické výrobky, Robotika a pohony, Priemyselná automatizácia a Energetika. Táto práca sa bude zameriavať predovšetkým na divíziu Robotiky a pohonov a v krátkosti bude predstavená aj divízia Priemyselnej automatizácie. [4]

3.1 Stručná história firmy

História tejto spoločnosti je charakterizovaná zlučovaním viacerých firiem. Najväčším míľnikom bol rok 1988 kedy sa zlúčili dve veľké firmy ASEA (Allmänna Svenska Elektriska Aktiebolaget) a BBC (Brown, Boveri & Cie). Tým sa začína podoba firmy ako ju poznáme dnes. Prvého v rámci ABB zostrojila vtedy ešte samostatná firma ASEA v roku 1986. Čo sa týka pôsobenia firmy v Českej republike formálne tu vznikla v roku 1992 . Česká republika sa čoskoro stala vedúcou krajinou v rámci európskych krajín, kde ABB pôsobí ako Slovensko, Ukrajina či Maďarsko. V roku 2017 sa k firme ABB pričlenila aj spoločnosť B&R Industrial Automation. Spoločnosť kladie veľký dôraz na výskum a vývoj. [3,4]



Obr. 11 Mestá v Českej republike, kde sídli spoločnosť ABB [4]

3.2 Divízia Robotiky

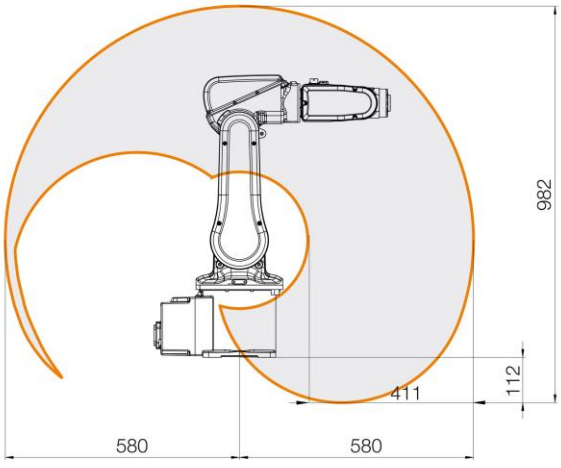
Spoločnosť ABB je vedúcim dodávateľom priemyselných robotov. Ponúka širokú škálu robotov na rôzne typy úloh. Sídlo divízie Robotiky sa nachádza v Prahe. Roboty spoločnosti ABB sú známe svojou kvalitou a bezpečnosťou. Za zmienku stoja aj ich kolaboratívne roboty ako je YuMi. Táto práca sa bude sústrediť na robota IRB 120, na ktorom bude aj demonštrovaná praktická časť. [29]

3.2.1 Robot IRB 120

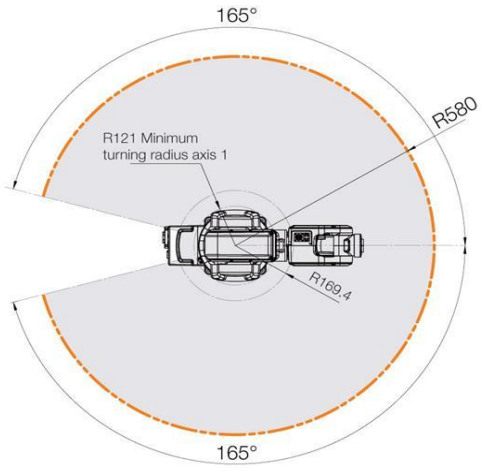
Jedná sa o najmenšieho šesťosého viacúčelového robota, ktorého ABB ponúka. Je to najnovší prírastok k 4 generácii robotov. Vďaka jeho kompaktnému dizajnu môže byť inštalovaný takmer akokoľvek, na ľubovoľnom mieste a s ľubovoľným uhlom, napríklad vo vnútri klietky, na vrch stroja alebo blízko k iným robotom. Taktiež je ľahko prenosný a jednoducho integrovateľný vďaka jeho nízkej hmotnosti iba 25 kilogramov. Hodí sa na použitie v elektronickom, potravinárskom, farmaceutickom či medicínskom priemysle. Všade, kde je potrebná veľká rýchlosť a zrýchlenie vysoká presnosť a obratnosť. Má hliníkovú štruktúru a tiež spĺňa hygienickú normu ISO 5 na prácu s jedlom a nápojmi vďaka minimálnemu úniku časticových emisii z robota. Jeho ďalšou výhodou je, že dokáže pracovať pod jeho základňou. Rameno má dosah 580 mm v horizontálnej osi a dokáže manipulovať s bremenom až 3 kilogramy, v zvislej osi až 4 kilogramy. Za zmienku tiež stojí jeho varianta IRB 120T, ktorá je rýchlejšia v osách 4, 5 a 6 a celkovo dosahuje o 25 percent vyššiu rýchlosť. [12,13]

Obr. 12 Priemyselný robot IRB 120 [12]

Robot je vybavený kontrolérom IRC5 a má operačný systém Robotware. Oba tieto prvky budú popísané v osobitných podkapitolách.



Obr. 14 Pracovný priestor robota IRB 120 pohľad zboku [13]



Obr. 15 Pracovný priestor robota IRB 120 pohľad zhora [13]

3.2.2 Kontrolér IRC5

Priemyselné roboty od ABB sú ovládané kontrolérmi. Kontrolér IRC5 je založený na pokročilom dynamickom modelovaní, takže automaticky optimalizuje prácu robota tým, že skracuje časové cykly (QuickMove) a poskytuje vysokú presnosť cesty (TrueMove). Vďaka IRC5 je pohyb robota predvídateľný a precízny. Veľkým benefitom IRC5 je bezpečnosť, ktorá spĺňa všetky potrebné regulácie a je výsledkom zabezpečenia SafeMove2, ktoré mimo iného podporuje kolaboráciu medzi robotmi a ľuďmi. Tento kontrolér od ABB je kompatabilný s rôznymi napäťovými úrovňami a je odolný voči nepriaznivým vplyvom prostredia. Podporuje väčšinu priemyselných vstupne-výstupných rozhraní a má možnosť vzdialeného prístupu. Obsahuje tiež vbudovanú autodiagnostiku, ktorá zaisťuje rýchle obnovenie prevádzky po výpadku a diaľkový monitoring ABB Ability Connected Services. Jedným kontrolérom IRC5 je možné ovládať až 4 robotov pomocou systému MultiMove, pričom pre každého robota navyše sa pridáva kompaktný pohonný modul. IRC5 je modulárne a dodáva sa v rôznych variantách, dajú sa skladať na seba, vedľa seba, alebo do bunky. Medzi jeho varianty patrí jednoskriňové prevedenie a kompaktné prevedenie. Špeciálnym prevedením určeným na farbiace procesy je IRC5P. Kontroléry obsahujú operačný systém RobotWare. [14,25,26]

Obr. 16 IRC5 v prevedení kompakt (vpravo), skriňovom (stred) a IRC5P (vpravo) [14]

Na programovanie kontrolérov sa používa ABB RobotStudio, alebo rovno na pracovisku môže operátor programovať robota ABB pomocou *teach pendant* od ABB nazývaného FlexPendant. [25]

Obr. 17 FlexPendant od ABB [22]

3.2.3 OmniCore kontroléry

V krátkosti určite stojí za zmienku nová generácia kontrolérov OmniCore. Sú menšie a majú ľahšiu inštaláciu ako IRC5 kontroléry. Podobne ako IRC5, na ktorých odkaze sú navrhnuté, obsahujú ABB Ability digital platform a ABB Abilitiy Connected Services a majú vbudovaný systém SafeMove2 vďaka čomu môžu tak byť použité pre kolaboratívne roboty. Sú flexibilné a môžu byť vybavené napríklad fieldbus protokolmy alebo vizualizáciou. Ich operačným systémom je RobotWare 7 a sú programovateľné cez nový FlexPendant. Sú najlepšou voľbou pre ovládanie pohybu a presnosti robotov. Prvé vydanie by malo podporovať robotov YuMi, ostatní roboti budú pridaní samozrejme neskôr tiež. [15,16]

Obr. 18 OmniCore kontrolér spolu s novým typom FlexPendant položeným na ňom [16]

3.2.4 RobotWare a programovací jazyk RAPID

Základom všetkých kontrolérov od ABB je software RobotWare. Poskytuje vysokú prispôsobivosť a presnosť, krátke časové cykly a je spoľahlivý a bezpečný. Má tiež vbudovanú kontrolu funkčnosti procesov. Jeho možnosti sú celkom rozsiahle, preto tu budú spomenuté len vo všeobecnosti. V príručke pre RobotWare sú popísané dôkladnejšie.

- Koordinácia pohybu
- Udalosti pohybu
- Funkcie pohybu
- Kontrola pohybu
- Komunikácia
- Kontrola I/O
- Kontrola servomotorov
- Nástroje diagnostiky
- Možnosti aplikácii

Koordinácia pohybu umožňuje ovládanie niekoľkých robotov z jedného kontroléru a koordináciu ich pohybov alebo ich vzájomnú nezávislosť, nastavovanie osí a synchronizáciu senzorov. *Udalosti pohybu* slúžia na definíciu zón rôznych tvarov (napríklad takých, kam robot nesmie vstúpiť) a fixných udalostí pohybu (keď sa robot

dostane do určitej pozície). *Funkcie pohybu* poskytujú možnosť vytvoriť nezávislé osi, resetovať osi, obnoviť cestu po výpadku alebo chybe, sledovanie cesty robota. *Kontrola pohybu* detekuje kolízie a chráni voči poškodeniu od kolízii. Možnosť *komunikácie* slúži na spojenie a výmenu dát medzi počítačom a robotom. Ďalej zaisťuje rozbehnutie aplikácii z Microsoft Visual Studio na FlexPendant, Fieldbus komunikáciu, protokol prenosu súborov a prístup k nim na sieť a komunikáciu cez sériové kanály. *Kontrola I/O* vykonáva multitasking, prerušenia analógového signálu, logické prepojenia (na princípe funkcie programovateľného automatu), rozhranie pre senzory, platformy pre aplikácie. *Kontrola servomotorov* slúži na kontrolu a sledovanie servo nástrojov a elektricky spätých motorov. *Nástroje diagnostiky* sledujú a spravujú operačný čas, kalendár, pokročilé algoritmy na výpočty prevodov, predpovedajú servisné zastavenia pre veľké roboty a taktiež sa starajú o alarmy. *Možnosti aplikácii* umožňujú špecializovať roboty na technologické procesy ako zváranie, striekanie, kontrolovanie nástrojov. [19]

Jadro tohoto operačného systému je naprogramované v programovacom jazyku RAPID, ktorý umožňuje vytvárať komplexné riešenia, ale aj jednoduché pohyby. Je to vyšší programovací jazyk. Pozostáva z množstva inštrukcii, ktoré popisujú prácu robota. Takže existujú špecifické inštrukcie na rôzne príkazy (napríklad na pohyb). Každá inštrukcia má určitý počet argumentov, ktoré menia vlastnosti inštrukcie (napríklad rýchlosť pohybu, presnosť pohybu a tak ďaľej).

Ďalej pri tomto jazyku poznáme tri typy rutín a teda procedúry, funkcie a trap routine. Procedúry sa používajú ako podprogramy. Funkcie vracajú nejakú hodnotu špecifického typu a môžu obsahovať cykly alebo podmienky. Trap rutina alebo „rutina pasce“, slúži na zachytávanie prerušení a na reagovanie na tieto prerušenia. Môže byť spojená so špecifickým prerušením (napríklad keď je nejaký vstup zopnutý, čo má značiť prerušenie, tak sa automaticky spustí). [17,18]

Informácie môžu byť uložené v dátach, ktoré sa delia na konštantné (constant), premenné (variable) a pretrvávajúce (persistent). Konštantné reprezentujú statické hodnoty a nová hodnota im môže byť priradená iba manuálne. Premenné môžu mať hodnotu menenú počas behu programu. Pri pretrvávajúcich sa pri uložení programu do inicializačnej hodnoty priradí okamžitá hodnota pretrvávajúcej premennej. [17,18]

Ďalšími vlastnosťami jazyka sú rutinne parametre, aritmetické a logické výrazy, automatické zaobchádzanie s chybami, modulárne programy, multitasking a mnohé iné vlastnosti. [17,18]

Ako už bolo spomenuté, RAPID sa používa na programovanie kontrolérov od ABB a tým aj robotov. Samotný jazyk je však veľmi obsiahly a preto v tejto práci nebude nejako hlbšie popísaný.

```
MODULE MainModule
  VAR num length;
  VAR num width;
  VAR num area;

  PROC main()
    length := 10;
    width := 5;
    area := length * width;
    TPWrite "The area of the rectangle is " \Num:=area;
  END PROC
ENDMODULE
```

Obr. 19 Jednoduchý program napísaný v programovacom jazyku RAPID [17]

3.3 Divízia Priemyselnej automatizácie

Aj keď sa táto práca zameriava na robotiku, bude v nej pár vetami popísaná aj divízia automatizácie, pretože automatizácia a robotika sú veľmi úzko prepojené. Táto divízia sa zameriava hlavne na automatizáciu, reguláciu a optimalizáciu priemyselných procesov. V rámci tejto divízie ABB dodáva predovšetkým distribuované riadiace systémy. Ich software na vývoj aplikácii pre PLC, Safety PLC, kontrolných panelov a motorov sa nazýva Automation Builder. Divízia Priemyselnej automatizácie dodáva množstvo produktov ako programovateľné logické automaty, kontrolné panely, motory, pohony a podobne. Významnú pozíciu v rámci priemyselnej automatizácie firmy ABB zastáva spoločnosť B&R Industrial Automation. [20,21]

Obr. 20 Programovateľné automaty od firmy ABB, rôzne prevedenia [20]

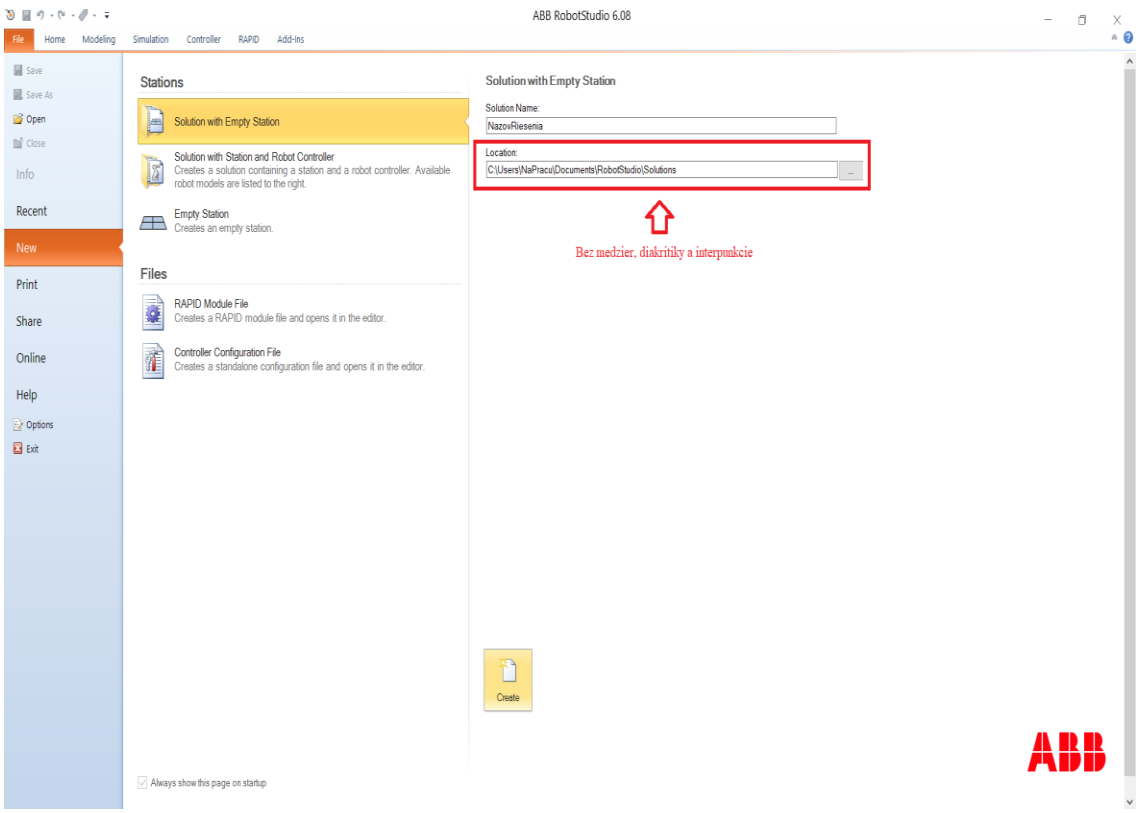
4. TVORBA SCÉNY V ABB ROBOTSTUDIO

Nasledujúca kapitola bude popisovať ako vytvoriť scénu v ABB RobotStudio. Scénou sa myslí vytvorenie nového riešenia, pridanie robota, kontroléru (robotického systému), nástrojov a prostredia.

Najskôr sa však zoznámime s vývojovým prostredím ABB RobotStudio a prácou s ním. V prvej podkapitole bude prebraté ako vytvoriť riešenie s prázdnou stanicou a ako sa v prostredí pohybovať a otáčať s kamerou. Ďalej bude ukázané ako pridať do riešenia robota, nástroje a robotický systém, teda kontrolér. Nasledujúca podkapitola ukáže ako pridať a vytvoriť geometriu a objekty v ABB RobotStudio. Posledná podkapitola sa bude venovať nástrojom a funkciám ABB RobotStudia, ktoré neskôr budeme používať.

4.1 Tvorba nového riešenia a práca s ABB RobotStudio

Po spustení ABB RobotStudio zvolíme možnosť pri staniciach vytvoriť riešenie s prázdnou stanicou (*Solution with Empty Station*). Túto možnosť zvolíme, aby sa vytvoril adresár, kam sa nám budú ukladať knižnice, geometria, stanica a všetky ostatné elementy súvisiace. Pokiaľ by sme zvolili možnosť vytvoriť riešenie so stanicou a robotickým kontrolérom, bol by nám do riešenia automaticky pridaný robot, ktorého by sme si pri tvorbe vybrali, spolu s virtuálnym kontrolérom, ktorého verziu systému RobotWare by sme si tiež zvolili. Pre názornosť riešenia je ale lepšie, aby sme si tieto atribúty pridali sami a manuálne. Zadáme meno a lokáciu, kam chceme riešenie uložiť. Je nutné, aby v ceste lokácie kam sa má riešenie uložiť neboli žiadne medzery, diakritika ani interpunkcia, inak nebude možné pre dané riešenie spustiť kontrolér.

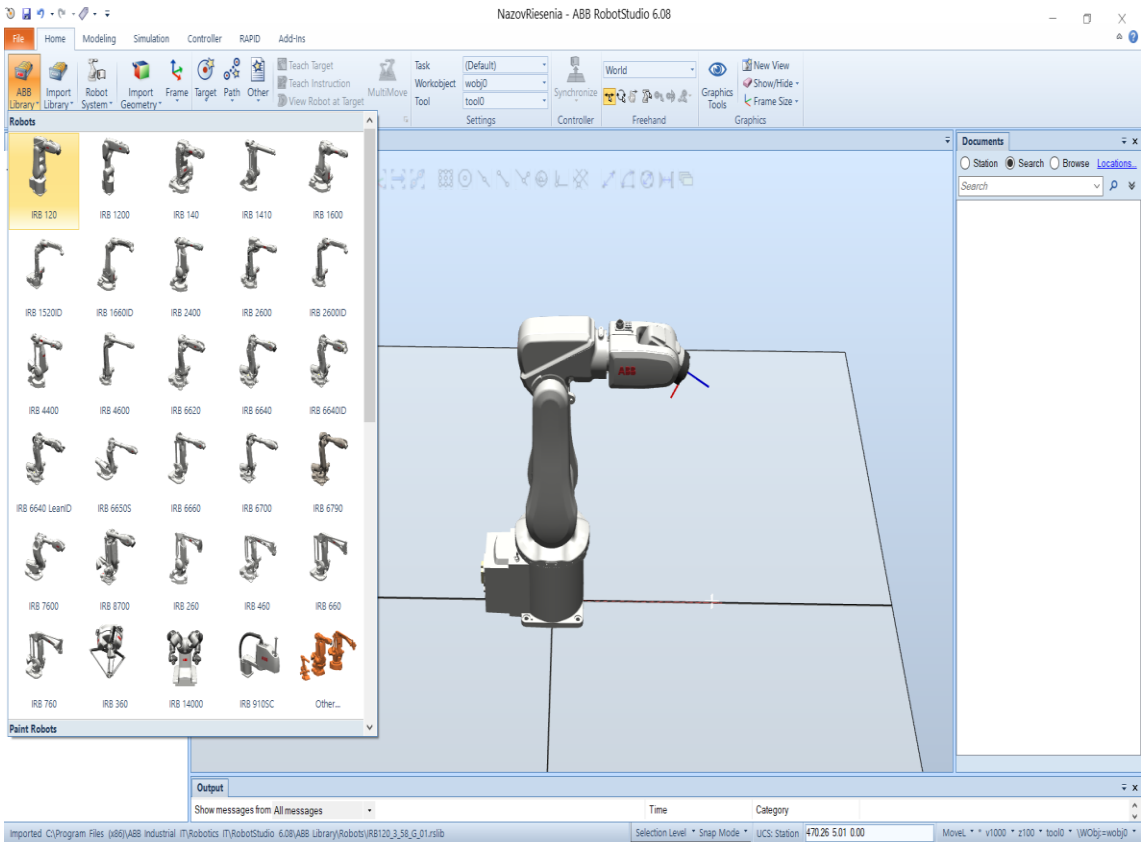


Obr. 21 Tvorba riešenia s prázdnou stanicou

Po vytvorení riešenia sa nám zobrazí prázdna stanica. V prostredí sa lineárne pohybujeme pomocou počítačovej myši a stlačeného tlačidla CTRL. Obraz sa dá natáčať pomocou myši a stlačených tlačidiel CTRL + Shift. Spolu s obrazom sa natáča aj súradnicový systém. Pomocou koliečka na myši pohľad približujeme a odďaľujeme. Približovať sa dá aj po stlačení koliečka a následným pohybom myši doľava alebo doprava.

4.2 Pridanie robota, nástrojov a robotického systému

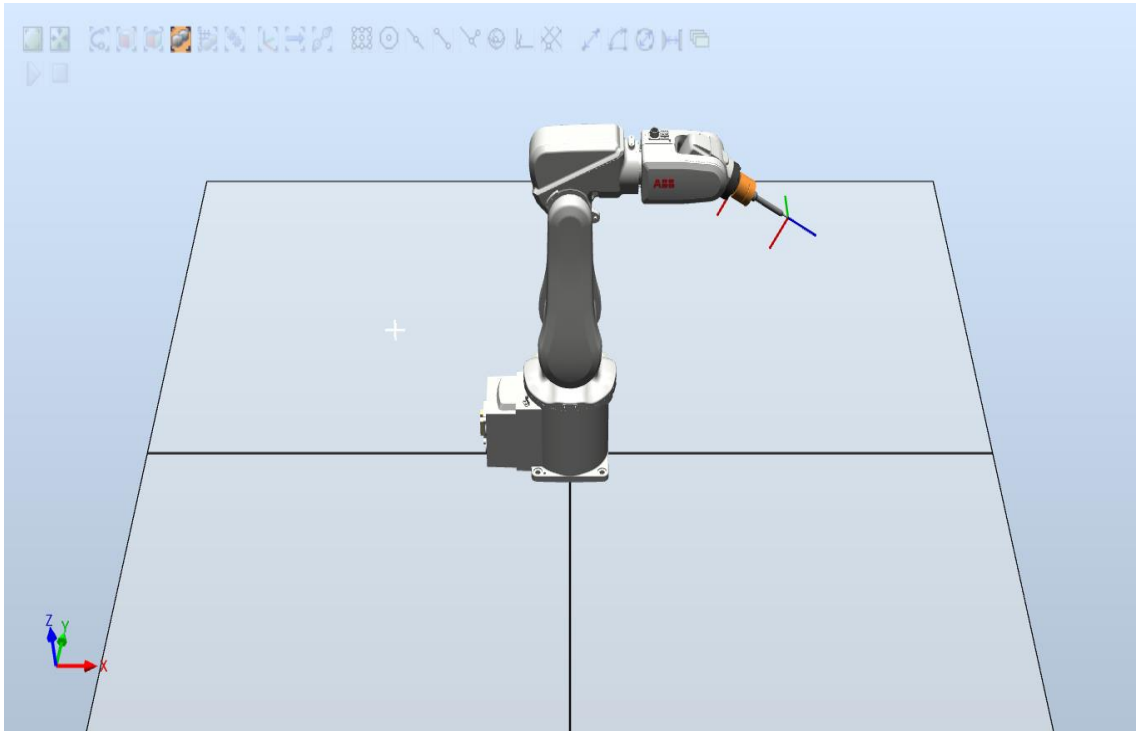
Keď máme pripravenú prázdnu stanicu, môžeme do nej z knižnice ABB pridať robota. V knižnici je mnoho robotov, farbiacich robotov, dopravníkov, polohovačov a motorov. Ďalšie potrebné stroje sa dajú dodatočne stiahnuť. Po stlačení tlačidla *ABB Library* v pravom hornom rohu (karta *Home* v toolbare) sa objaví ponuka produktov. Pre túto prácu bude zvolený robot *IRB 120* popísaný v predošlej kapitole. Po zvolení robota *IRB 120* dostaneme na výber zvoliť variantu (obyčajnú, turbo alebo vhodnú na prácu v hygienickom prostredí). Zvolíme variantu obyčajnú. Ak boli dodržané všetky kroky, mala by stanica vyzerať ako na obrázku 22.



Obr. 22 Pridanie robota do prázdnej stanice

Ďalej sa presunieme k pridaniu a pripojeniu nástroja na robota. Nástroje sa nachádzajú pod možnosťou *Import Library*. Po zvolení tejto možnosti sa nám ponúka niekoľko variant. Pod možnosťou *User Library* by sme našli nástroje, ktoré boli vymodelované pomocou *CAD Software* a môžu byť do ABB RobotStudio pridané. V možnosti *Equipment* sú predvolené nástroje a iné prostriedky, ktoré sú v RobotStudio integrované. Opäť je možné k ním niektoré chýbajúce prostriedky stiahnuť. *Solution Library* poskytuje *CAD objekty*, ktoré máme uložené v našom adresári riešenia. Importovaná môže byť aj externá knižnica. Pre naše riešenie zvolíme *Browse for*

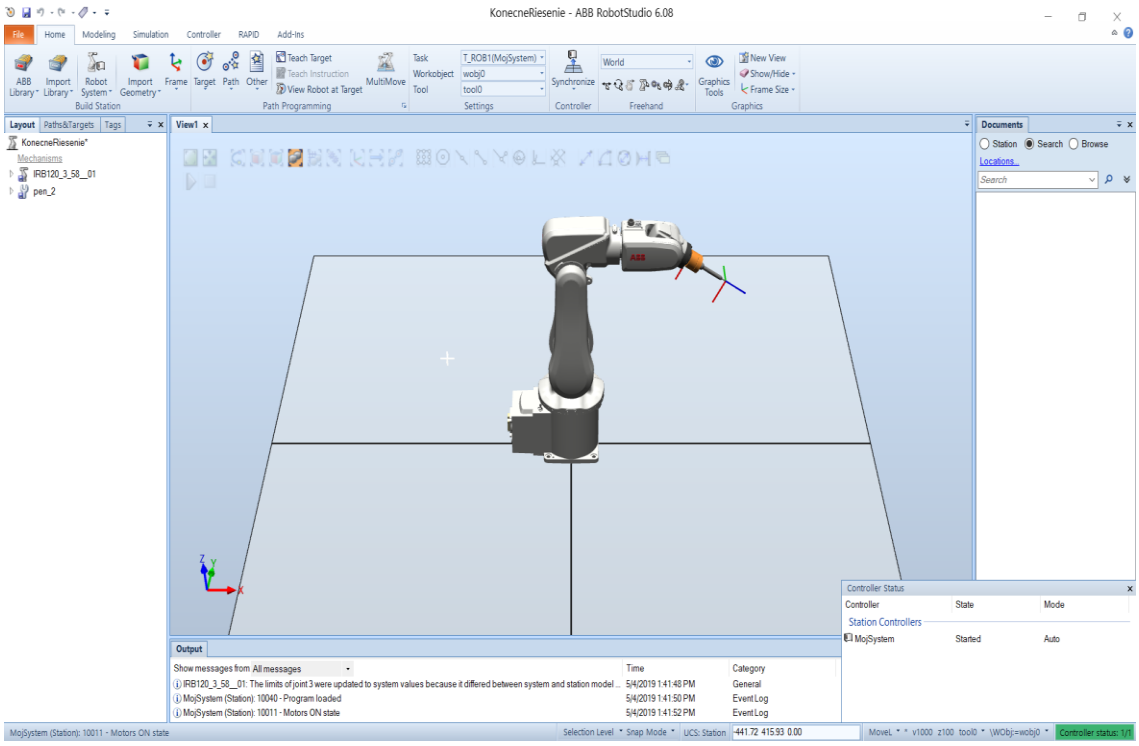
library a zvolíme nástroj *Pen*, ktorého model bol stiahnutý z náučného videa [32]. Po tejto akcii by sa nám mal v strede stanice objaviť nástroj *Pen*. Tento nástroj je možné na robota pripevniť. V ľavej časti prostredia zvolíme možnosť *Layout*. Tam vidíme mechanizmy, ktoré sa v stanici nachádzajú, medzi nimi náš robot *IRB 120* a náš nástroj *Pen*. Pravým kliknutím myši na nástroj *Pen* sa nám zobrazí panel možností. Zvolíme možnosť *Attach to* a zvolíme nášho robota *IRB 120*. Systém sa nás opýta či chceme upraviť polohu nástroja *Pen*. Zvolíme áno, aby bol nástroj automaticky pridaný na koniec ramena robota. Pri dodržaní všetkých krokov by mala naša stanica vyzerať ako na obrázku 23.



Obr. 23 Pridanie nástroja na robota

Naším ďalším krokom je pridanie robotického systému do našej stanice. RobotStudio používa virtuálne kontroléry na riadenie robotov. Tieto kontroléry majú úplne rovnaký software ako ten, ktorý používajú reálne kontroléry. Vďaka nim môžeme simulovať situácie na počítači a tak ušetriť materiál, opotrebenie robota a nástrojov, poprípade poškodenie zariadení pri testovaní. Pre pridanie robotického systému zvolíme záložku *Home* a v nej nájdeme časť *Build Station*. Tu zaklikneme možnosť *Robot System*. Na výber máme z troch možností. Prvou je vytvoriť robotický systém na základe plánu, predispozície. Túto možnosť zvolíme my. Zadáme meno systému, uloží sa nám do nášho adresáru riešenia. Ďalej zvolíme verziu *RobotWare*. Verzie sa dajú do ABB RobotStudio dodatočne stiahnuť. Keďže máme len jedného robota v stanici, môžeme rovno stlačiť tlačidlo *Finish*. Pokiaľ by sme mali viac robotov v našej stanici, tak po stlačení tlačítka *Next* by sme mohli vidieť, ktoré mechanizmy budú súčasťou nášho systému. Na jeden systém môžu byť maximálne 4 roboty. Po ďalšom stlačení tlačidla *Next* sa nám zobrazí zosumarizovanie nášho systému. Opäť by sme zvolili *Finish*. Ďalšou možnosťou je vytvoriť si nový robotický systém na nami vybraného robota. Týchto systémov môžeme vytvoriť na stanici niekoľko a sú špecifikované na typ robota. Treťou možnosťou je použiť existujúci systém, ktorý sme už vytvorili. Tvorba systému a pripojenie virtuálneho kontroléru chvíľu trvá. Po dokončení by mal byť stav kontroléru zapnutý a mal by byť v móde *auto*.

Ak boli dodržané všetky kroky, mal by byť stav kontroléru zobrazený ako na obrázku 24.



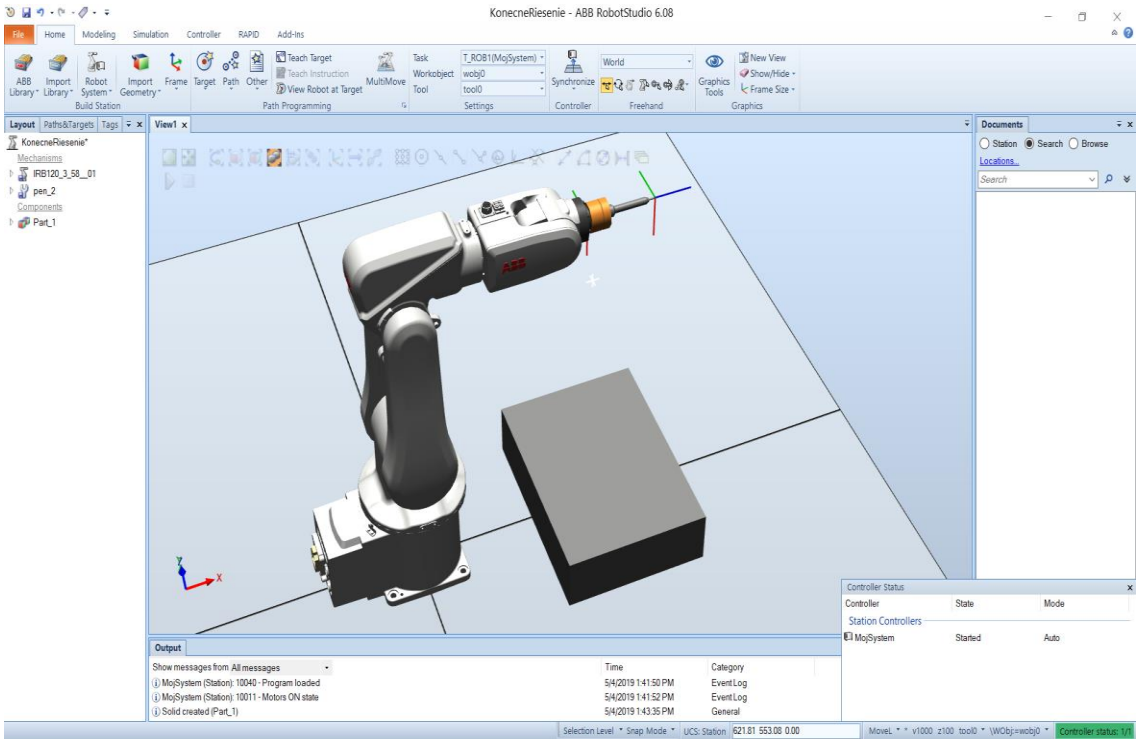
Obr. 24 Tvorba robotického systému

Po spustení viruálneho kontroléru môžeme s robotom skúsiť manuálne pohybovať a vidieť, že nástroj bol na neho skutočne pripevnený. V záložke *Home* nájdeme sekciu Freehand, kde sa nachádzajú nástroje na manuálne hýbanie s robotom. Ďalej je možné tu nastaviť súradnicový systém, voči ktorému sa bude robot pohybovať. Pre ukážku zvolíme súradnicový systém *World*. *Jog Joint* dokáže pohybovať s jednotlivými časťami robota tak, že na niektorú časť klikneme, a stlačením ľavého tlačidla myši a jej pohybom sa daný kĺb rozpohybuje. Ďalšou možnosťou je pohybovať robotom ako celkom lineárne. Zvolíme možnosť *Jog Linear* a klikneme na ktorúkoľvek časť robota. Objaví sa súradnicový systém so šípkami. Po potiahnutí jednej zo šípok sa robot pohne. Robota dokážeme aj natáčať pomocou nástroja *Jog Reorient*. Postup je rovnaký ako pri lineárnom pohybe.

4.3 Geometria a objekty

Dôležitou súčasťou každej stanice sú aj objekty, ktoré sa v nej nachádzajú. Môžu to byť napríklad stoly alebo súčasti na opracovanie. Veľkou výhodou ABB RobotStudio je možnosť importovať geometriu. To znamená, že pokiaľ máme nejakú súčiastku alebo objekt navrhnutú v *CAD software*, je možné ich do stanice pridať, pokiaľ sú v správnom formáte. Toto je možné pomocou možnosti *Import Geometry*. Pokiaľ by sme chceli vytvoriť nejaký objekt v ABB RobotStudio, presunieme sa do záložky *Modeling*. Tu nájdeme sekciu *Create*. V nej zvolíme možnosť *Solid*. Objaví sa nám výber geometrických tvarov. V našom prípade zvolíme napríklad *box*. Po zakliknutí sa nám objaví okno *Create Box*, v ktorom môžeme zadať dĺžku, výšku a šírku objektu a taktiež na ktorý súradnicový systém sa má odkazovať. Po zadaní veľkostí a ponechaní súradnicového systému *World* sa nám vytvorí objekt, s ktorým môžeme pohybovať

pomocou nástrojov *Move* a *Rotate* v sekcii *Freehand*. Pri splnení všetkých krokov by naša stanica mohla vyzerať napríklad ako na obrázku 25.



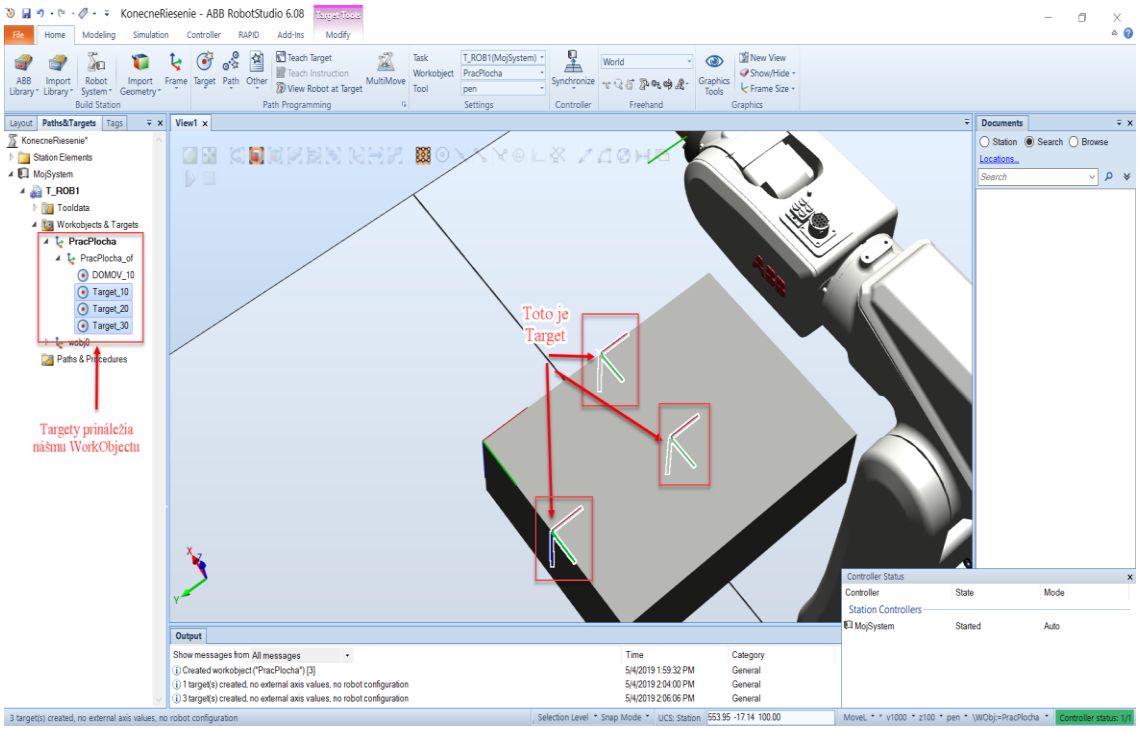
Obr. 25 Tvorba geometrického objektu

4.4 Nástroje a funkcie

V tejto podkapitole budú popísané niektoré funkcie RobotStudio, ktoré budeme neskôr používať. Jednou z najdôležitejších funkcii je tvorba pracovného objektu (*Workobject*). Pracovný objekt je súradnicový systém, ktorý popisuje pozíciu pracovných bodov na našom objekte (súčiastke napríklad). Skladá sa z dvoch rámov (*frame*) a to *User frame* a *Object frame*. Na vytvorenie pracovného objektu zvolíme záložku Home a nájdeme sekciu *Path programming*. Tu rozklikneme možnosť *Other* a v nej zvolíme možnosť *Create Workobject*. *Workobject* definujeme cez *User Frame*. V tomto prípade bude zvolená metóda určenia pomocou bodov. Zaklikneme *Frame by points* a zvolíme možnosť *Three-point*. Klikneme na *posistion* a na objekt, kde chceme *Workobject* vytvoriť. Presnejšie a efektívnejšie vytvorenie *Workobject* dosiahneme pomocou nastavenia povrchovej selekcie (*Surface selection*) a *Snap object* módu. Keď máme vytvorený *Workobject*, môžeme ho otočiť tak, aby súradnice odpovedali nášmu objektu alebo súčiastke. Pravým kliknutím myši na náš *Workobject* zvolíme možnosť *Rotate*. Podľa toho, ako potrebujeme súradnicový systém otočiť zvolíme os x, y alebo z a doplníme, o koľko stupňov ho chceme otočiť.

Ďalej sa môžeme zamerať na vytvorenie terčov (*Target*) v našom *Workobject*. Opäť zvolíme záložku *Home* a vyhľadáme sekciu *Path Programming*. Tu zvolíme možnosť *Target* a po rozkliknutí možnosť *Create Target*. Zvolíme referenciu na náš *Workobject* a zaklikneme pozície, kde chceme aby sa terče nachádzali. Po zvolení pozície sa nám pridá bod. Takto môžeme pridávať ľubovoľný počet bodov. Stlačíme tlačidlo *Create* a terče sa vytvoria. Za pomoci terčov môžeme vytvoriť cestu, ktorá sa dá použiť v inštrukcii na pohyb. Tento postup bude prebratý v nasledujúcej kapitole.

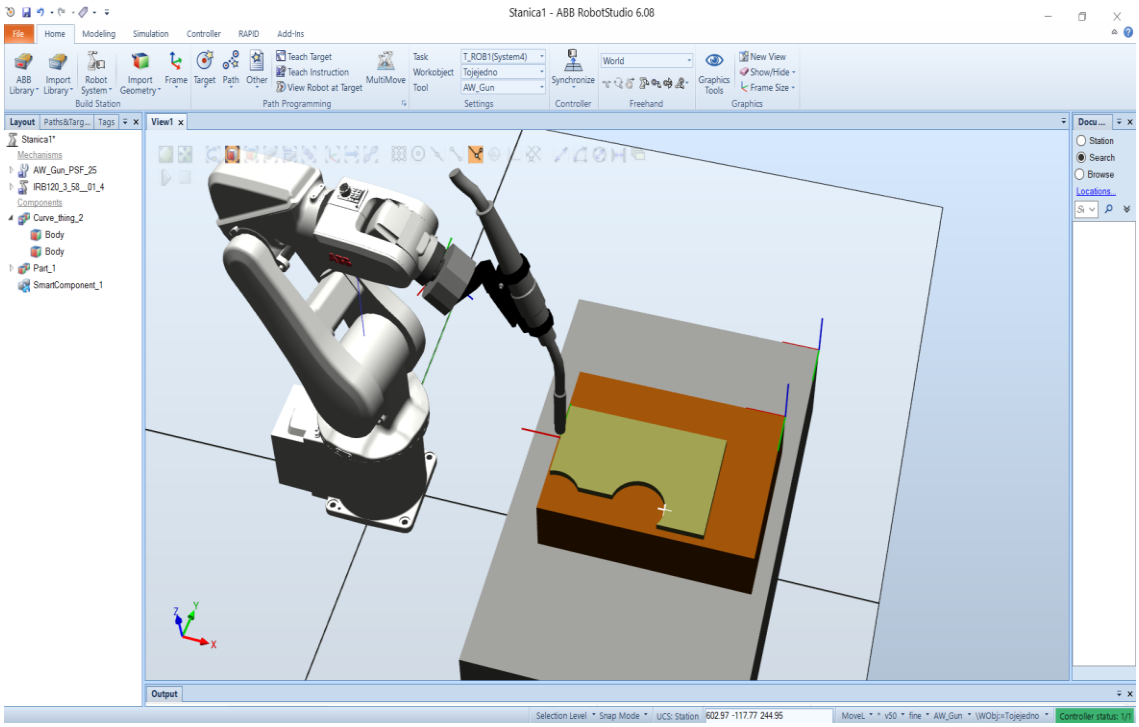
Pokiaľ sme dodržali všetky kroky môže naša stanica a jej pracovný priestor vyzerať napríklad ako na obrázku 26.



Obr. 26 Tvorba terčov a pracovných objektov

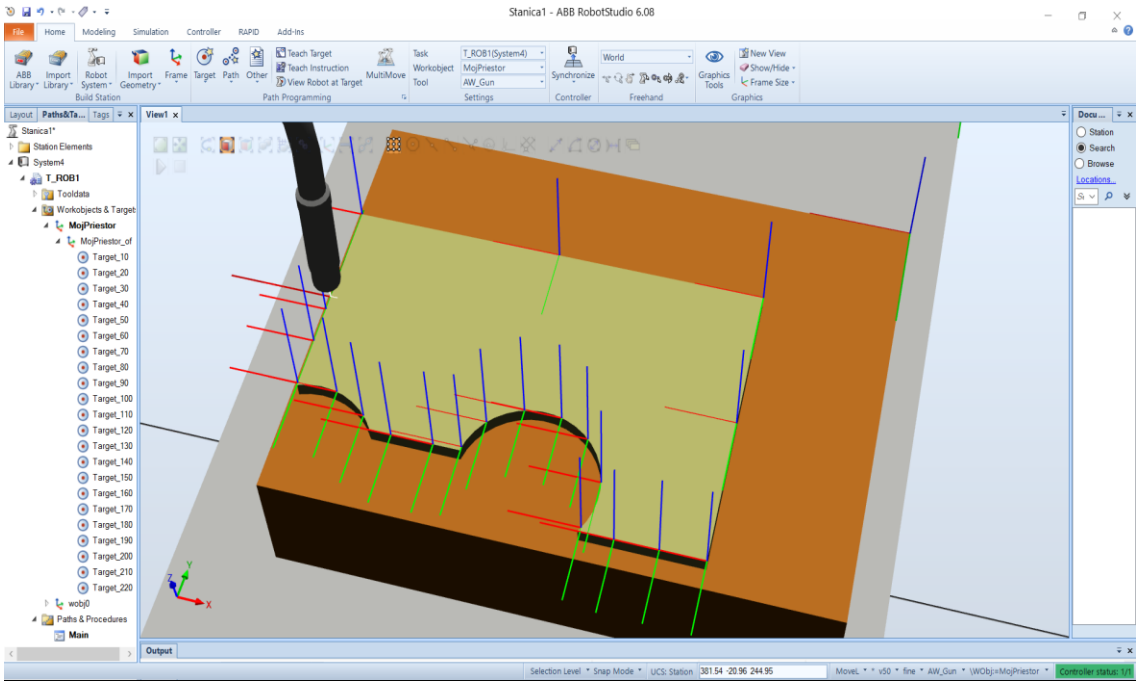
5. Tvorba sekvencii pohybov v ABB RobotStudio

V predošlej kapitole bolo spomenuté, ako robota rozhýbať manuálne. V tejto kapitole sa zameriame na prácu s ABB RobotStudiom a jeho vlastnosťou programovať roboty. Podľa postupu v predošlej kapitole si pripravíme scénu. Scéna môže vyzerať napríklad ako na obrázku 27.



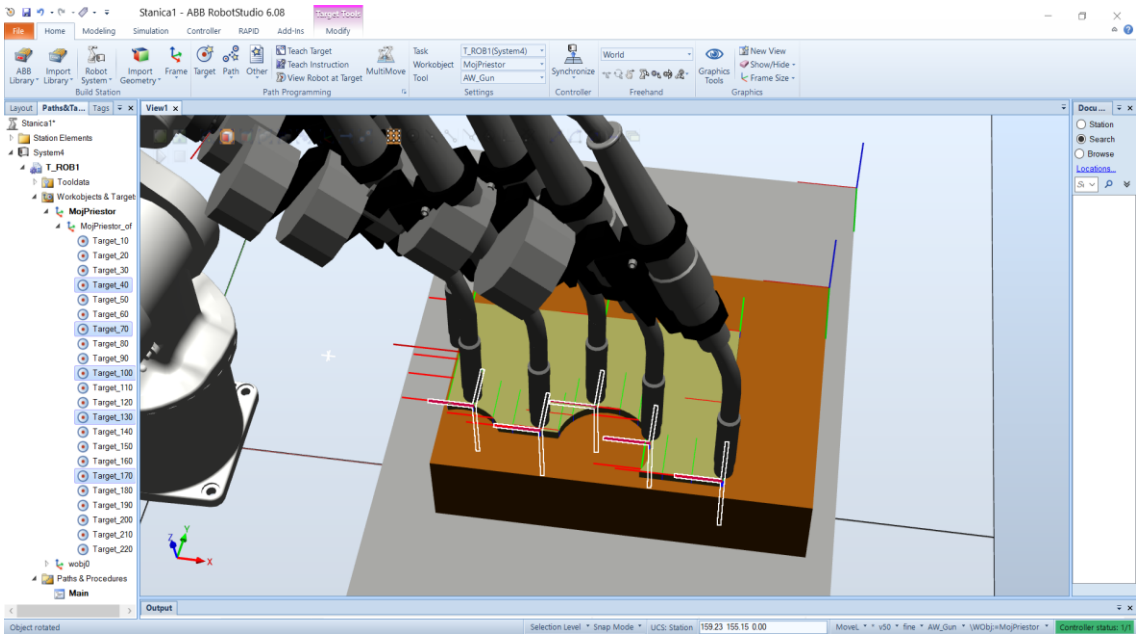
Obr. 27 Pripravená scéna na programovanie

Predstavme si, že chceme robota naprogramovať na zváranie nejakej súčiastky. Na ploche, ktorá má byť zváraná mu musíme vytýčiť *Targety*. Postup tvorby *Targetov* popisuje predošlá kapitola. *Targety* môžu byť zvolené napríklad ako na obrázku 28.



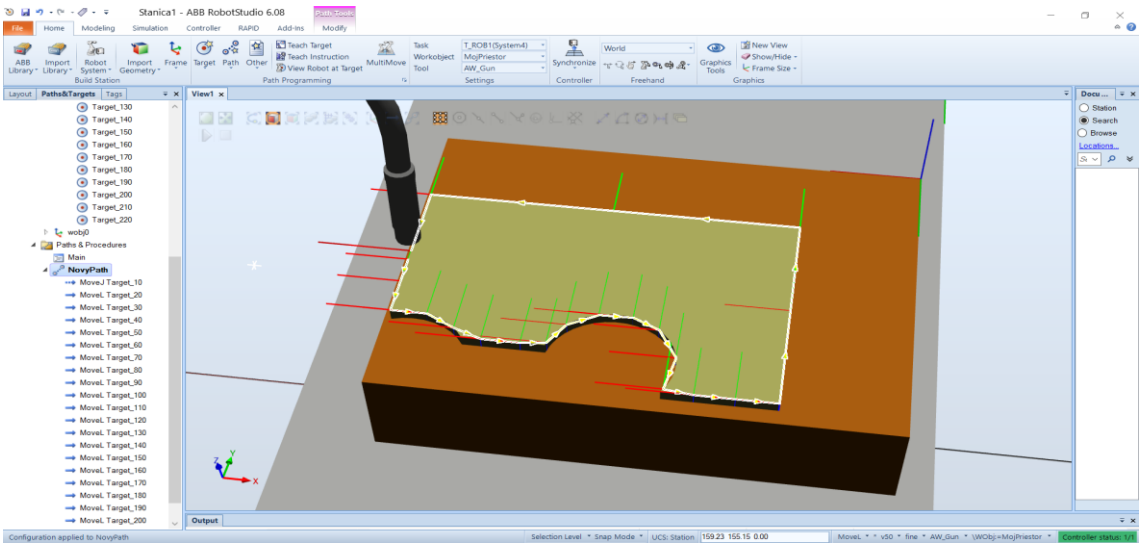
Obr. 28 Pripravené *Targety* na programovanie

Počet *Targetov* závisí na danom probléme. Tam, kde je vyžadovaná vysoká presnosť bude *Targetov* viac, naopak na rovných plochách bez kriviek ich môžeme dať menej. Orientáciu *Targetov* voči nástroju dokážeme jednoducho skontrolovať. Označíme všetky *Targety* a po pravom kliknutí myši zvolíme možnosť *View Tool at Target* a označíme náš nástroj (v prípade viacerých nástrojov v stanici je možné zvoliť rôzne nástroje). Pokiaľ orientácia voči robotovi a nástroju nesedí, *Targety* otočíme pomocou možnosti *Modify Target* a *Rotate*. Na obrázku 29 je ukázaná správna orientácia Targetov voči nástroju.



Obr. 29 Orientácia Targetov

Keď máme *Targety* hotové, vytvoríme cestu. Označíme všetky *Targety* a zvolíme možnosť *Add to new path*. Po vytvorení cesty si ju môžeme rozkliknúť a vidíme množstvo inštrukcii na lineárny pohyb. Prvý pohyb, keď sa robot približuje k pracovisku, je lepšie nastaviť na kĺbový. Typ pohybu zmeníme tak, že po rozkliknutí inštrukcie zvolíme možnosť *Edit instruction* a následne vyberieme typ pohybu *Motion type Joint*. RobotStudio umožňuje autokonfiguráciu cesty. Označíme cestu a zvolíme možnosť *Autoconfiguration* a *All move instructions*. Cesta sa nám po jej označení ukáže.



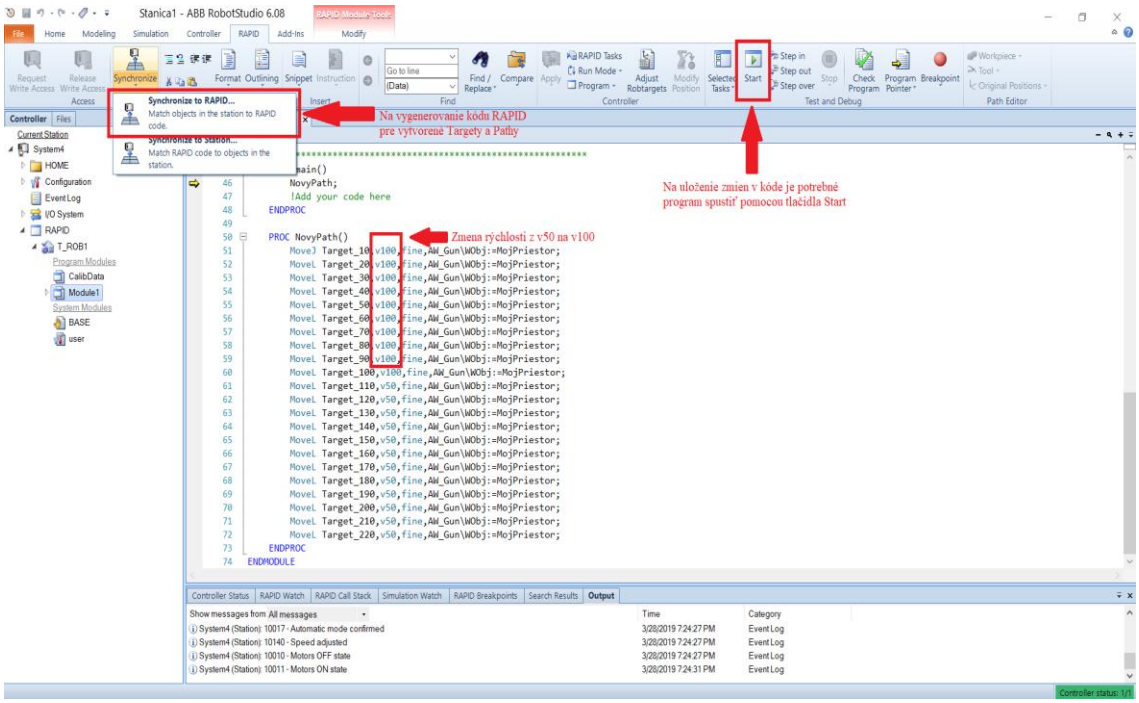
Obr. 30 Vyznačená cesta

Našu cestu môžeme odskúšať. Po označení cesty zvolíme možnosť *Move Along Path*. Pokiaľ je niektorý z bodov nedosiahnuteľný, RobotStudio ohlási chybu a robot cestu nevykoná. Teraz našu cestu musíme vložiť do procesu *Main*. Zvolíme možnosť *Path* a vyberieme *Empty path*. Po vytvorení premenujeme tento *path* na *Main*. Vyznačíme ho a zvolíme možnosť *Insert Procedure Call* a vyberieme našu vytvorenú cestu. Naše *Targety* a *Pathy* musíme zosynchronizovať s kódom RAPID, aby sa nám vygeneroval kód. Dosiahneme toho pomocou tlačítka *Synchronize*. Simulácia môže byť spúšťaná v karte *Simulation*.

RobotStudio k našim akciám automaticky vygenerovalo kód v jazyku RAPID. Tento kód je možné zobraziť v záložke RAPID. Kód sa automaticky uložil do programového modulu *Module1*. V *CalibData* sa nachádzajú definície súradníc nášho nástroja a *WorkObjectu*. Náš kód sa nachádza v *Module1*. Základnou inštrukciou v našom programe je inštrukcia *Move*. Táto inštrukcia môže mať mnoho podôb v závislosti na pohybe, ktorý predstavuje (napríklad *MoveJ* pre kĺbový, *MoveL* pre lineárny, *MoveC* pre krúživý). Preberá niekoľko argumentov :

1. *Target*, do ktorého sa pohyb uskutoční.
2. Rýchlosť, ktorou robot do daného *Targetu* pôjde.
3. Zóna, ktorá určuje presnosť pohybu a prípadné preloženie bodov krivkou
4. Nástroj, ktorým sa pohyb uskutoční spolu s *WorkObjectom*, v ktorom sa pohyb odohrá.

Všetky argumenty inštrukcie *Move* môžu byť v kóde zmenené. Po zmene sa musí kód spustiť pomocou tlačidla *Start* v sekcii *Test and Debug* aby sa zmeny uložily. Pokiaľ by sme chceli napríklad zmeniť rýchlosť pohybu pre prvých deväť *Targetov*, vyzeralo by to ako na obrázku 31.



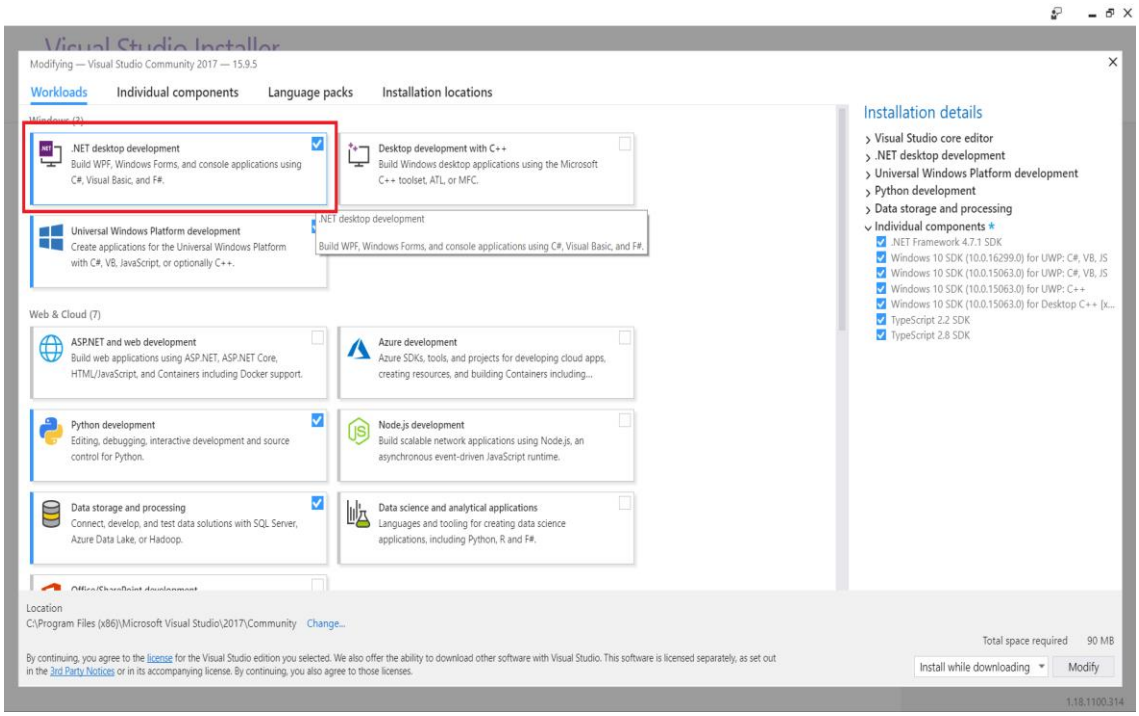
Obr. 31 Zmena v kóde RAPID

6. PROGRAMOVANIE POHYBOV V JAZYKU C#

V tejto kapitole bude prebraté, ako ovládať robota pomocou programovacieho jazyka C#. Okrem práce s ABB RobotStudio budeme používať aj vývojové prostredie jazyka C# Microsoft Visual Studio. Prvá kapitola bude venovaná zoznámeniu sa s týmto prostredím a práci s ním. Nasledujúca kapitola sa bude venovať ABB RobotStudio, kde si pripravíme scénu a ďalšie náležitosti, ktoré budeme potrebovať na komunikáciu s Microsoft Visual Studio. V poslednej kapitole bude vysvetlené, ako navrhnúť a naprogramovať aplikáciu v jazyku C# a ako túto aplikáciu spojiť so stanicou v ABB RobotStudio.
Vytvorená aplikácia je inšpirovaná študentským projektom ABB drawing robot [30,31].

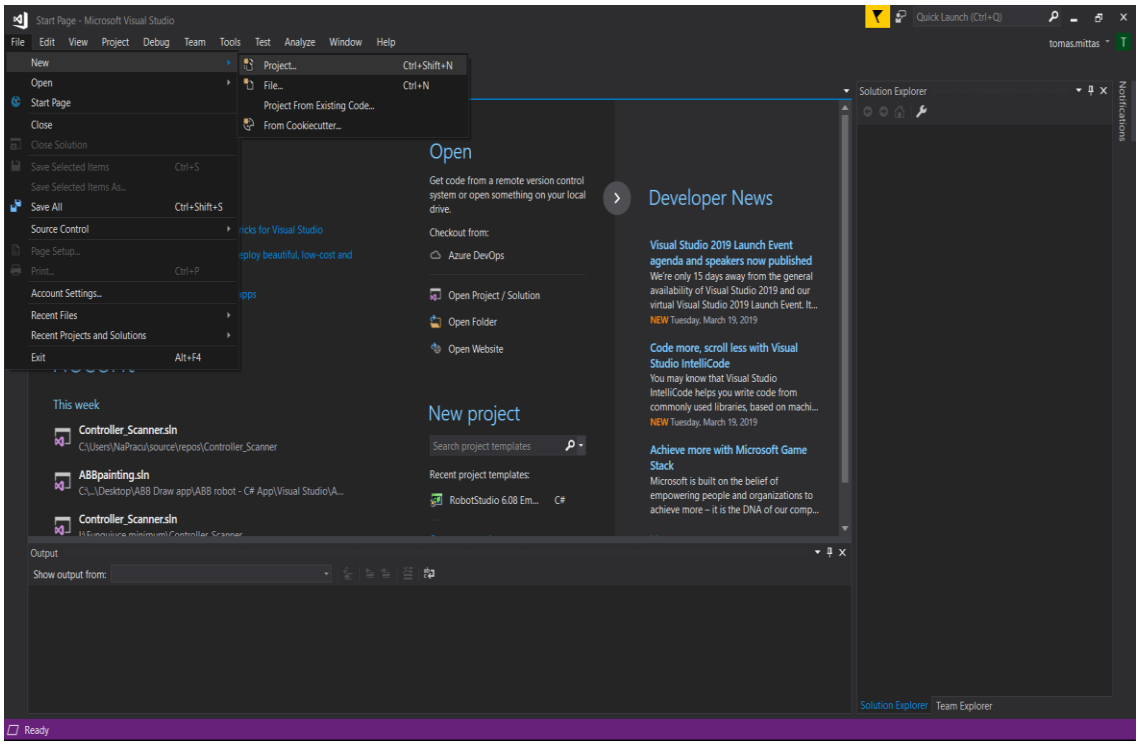
6.1 Práca s Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostredie aj pre programovací jazyk C#. Je vyvíjané spoločnosťou Microsoft Corporation a poskytuje nástroje pre vývoj rôznych druhov aplikácii. Pre študentov je dostupná bezplatná verzia tohto software. Visual Studio sa dá stiahnuť z oficiálnych stránok Microsoft. Okrem vývoja aplikácii slúži aj na ladenie, testovanie, profilovanie a diagnostiku aplikácii. Je možné do neho pridať radu rozšírení pomocou Visual Studio installer. Pred tým než sa zoznámime s prostredím si stiahneme rozšírenie .NET desktop development. .NET je software framework vyvinutý spoločnosťou Microsoft, ktorý obsahuje veľké množstvo knižníc a pozostáva z dvoch celkov a teda Framework Class Library a Common Language Runtime, teda virtual machine, ktorý funguje ako garbage collector a tak sa nám stará o prácu s pamäťou. Súčasťou podpory platfotmy .NET je aj jazyk C#. Je to objektovo orientovaný jazyk, ktorý vychádza z jazykov C++ a Java. .NET framework a jazyk C# sú svojou komplexnosťou veľmi obsiahle a sú tak nad rámec tejto práce. Nebudú preto hlbšie popísané. Rozšírenie .NET desktop development stiahneme pomocou Visual Studio installeru.



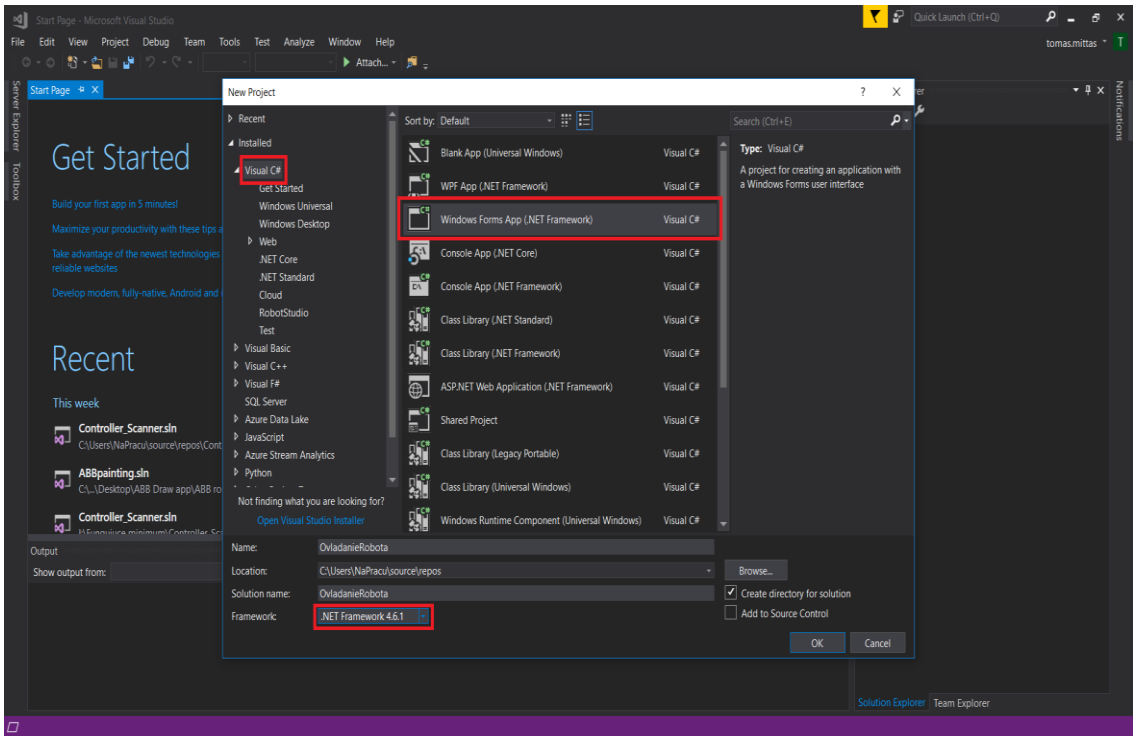
Obr.32 Visual Studio Installer

Naša aplikácia bude typu Windows Forms application, takže toto rozšírenie budeme potrebovať. Po tom ako sme stiahli rozšírenie môžeme prejsť k vytvoreniu aplikácie. Postupujeme ako na obrázku 33.



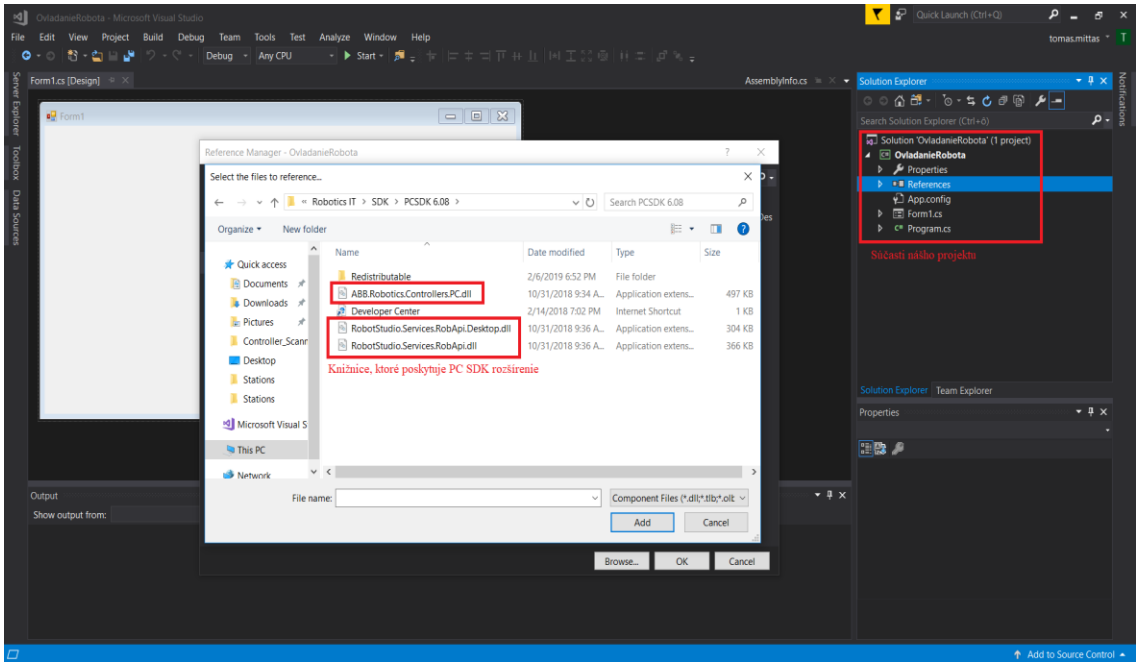
Obr. 33 Tvorba nového projektu

Ďalej zvolíme jazyk C# a *Windows Forms App*. Skontrolujeme, či máme vybratú najnovšiu verziu .NET platformy. To nám zaistí podporu najnovších knižníc. Zvolíme názov nášho projektu. Názov by nemal obsahovať medzery ani diakritiku.



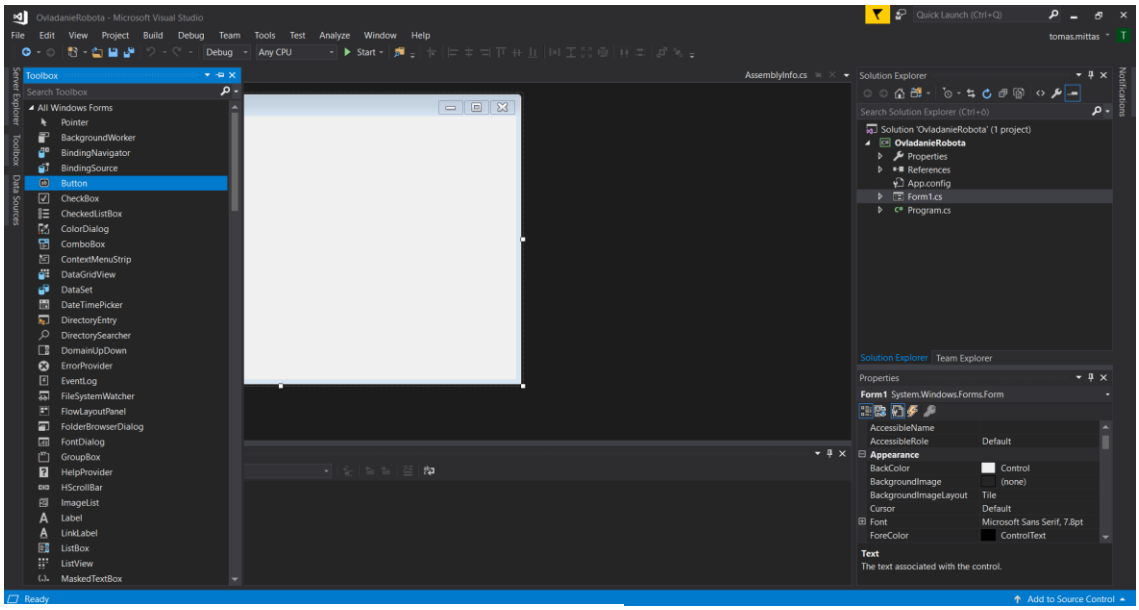
Obr. 34 Tvorba *Windows Forms App*

Po vytvorení projektu vidíme súčasti nášho projektu. Po kliknutí na jednotlivé súčasti sa nám zobrazí kód im prináležiaci. Na začiatok máme vytvorenú triedu *Program* a *Form1*. Ďalej tu máme *Properities*, *References* a *App.config*. Nás budú zaujímať *References*, pretože budeme potrebovať pridať rozšírenie PC SDK od firmy ABB na vývoj. Do *References* sa pridávajú knižnice, ktoré nie sú obyčajne zahrnuté pri vytvorení projektu. Klikneme na *References* a *Add reference*. Nájdeme priečinok, kde máme nainštalované rozšírenie PC SDK a označíme knižnice.



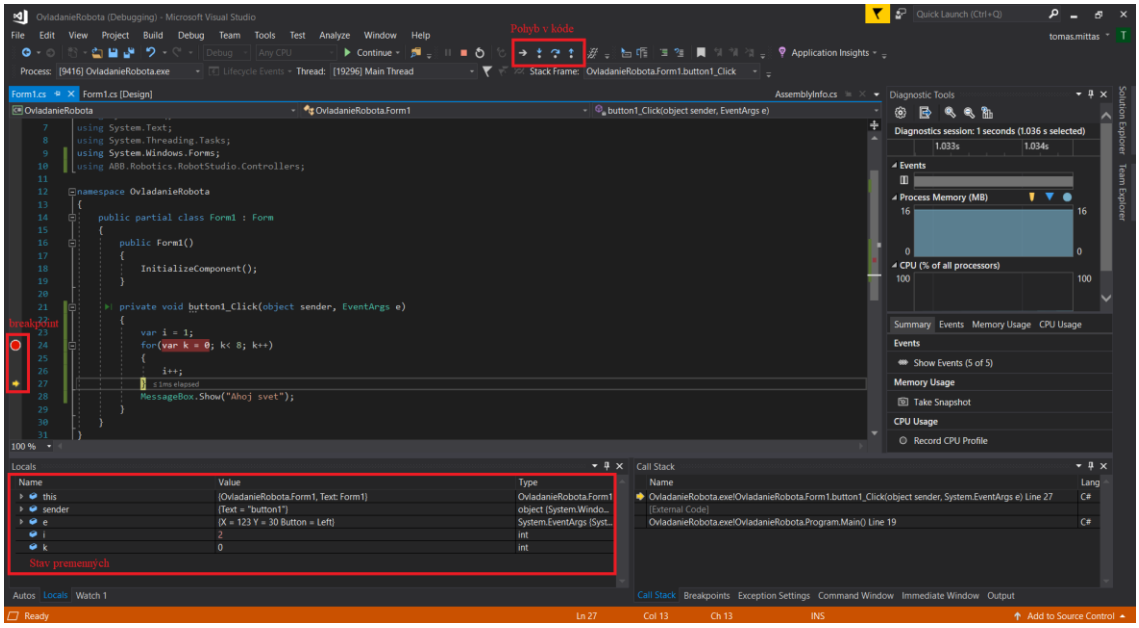
Obr. 35 Pridávanie referencii

ABB poskytuje aj iné rozšírenie ako PC SDK. Toto rozšírenie je RobotStudio SDK a slúži na tvorbu špeciálnych typov aplikácii, ktoré po vytvorení slúžia ako Add-in nástroje pre RobotStudio. Pre toto rozšírenie je taktiež podpora jazyka C#. V tejto práci ale používané nebude. Pri tvorbe našej aplikácie budeme potrebovať uživateľské rozhranie. Označíme si *Form1* a na ľavej časti obrazovky vyberieme *Toolbox*. Tu môžeme vidieť množstvo nástrojov.



Obr. 36 *Toolbox*

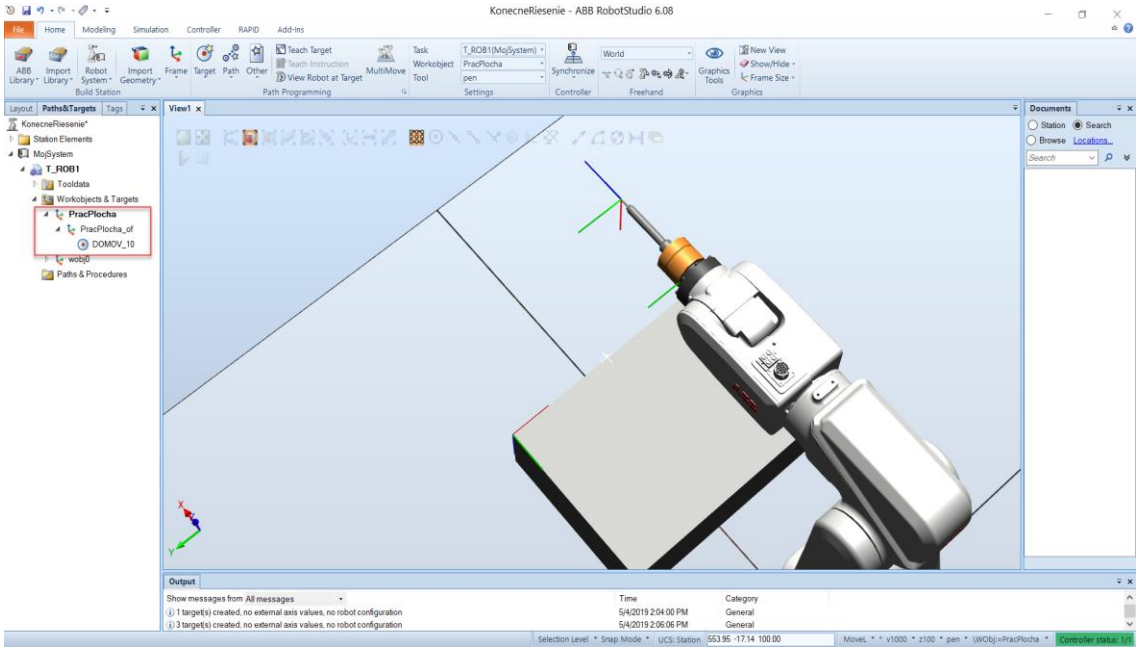
Po zakliknutí vytvoreného nástroja môžeme nastavovať jeho vlastnosti a udalosti. Obe možnosti neskôr použijeme. Našu aplikáciu môžeme testovať a debugovať, pokiaľ by sme potrebovali zistiť ako sa chová. Po zvolení *breakpointu* a spustení kódu môžeme sledovať zmenu premenných a pohybovať sa medzi funkciami a triedami.



Obr. 37 Debugovanie a testovanie

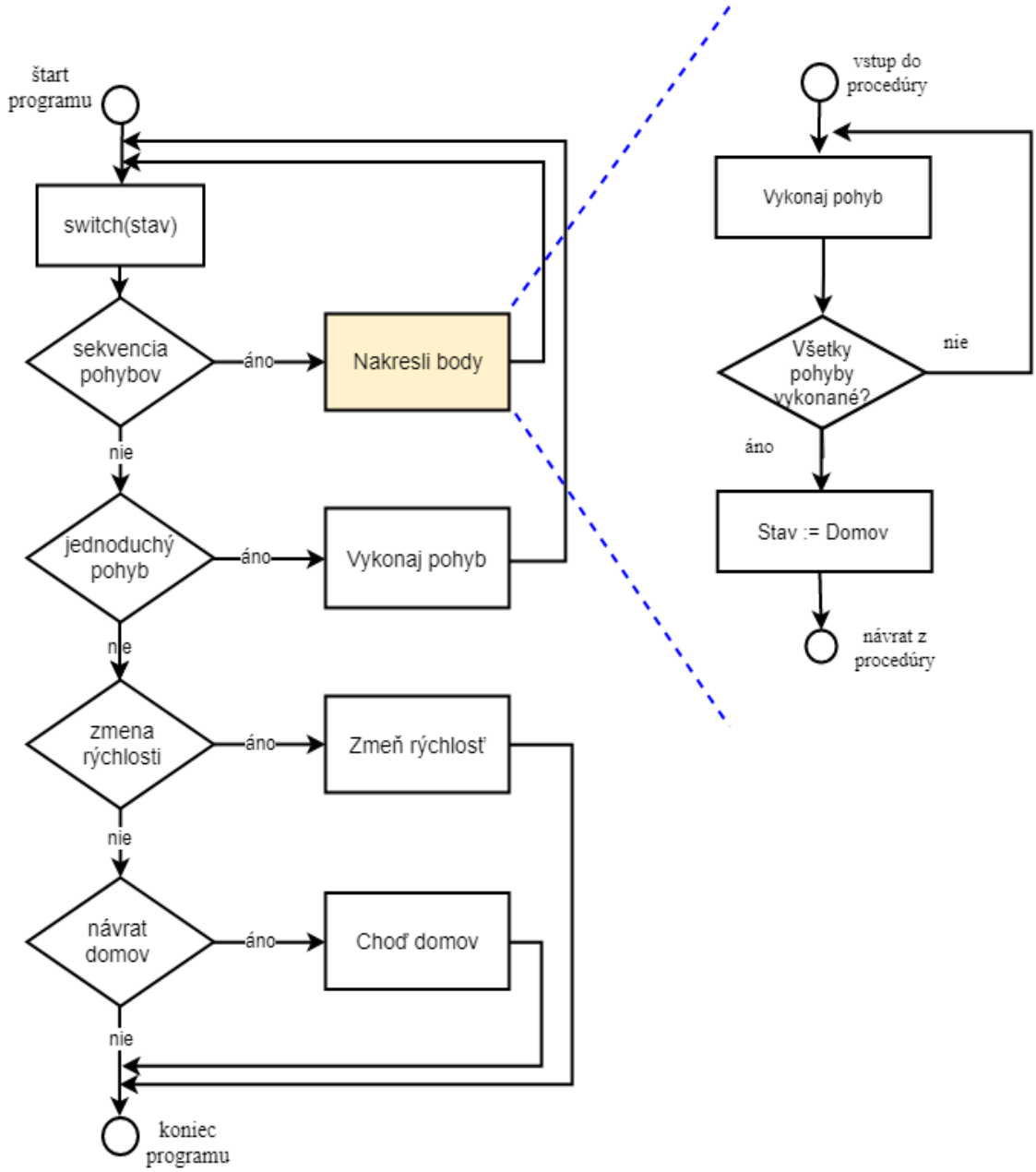
6.2 Nastavenie na strane ABB RobotStudio

Podľa postupu z predchádzajúcej kapitoly vytvoríme scénu s robotom a pripravíme si prostredie na prácu. Prostredie bude „plátno“ na kreslenie, ktoré vhodne umiestnime. Nástroj pre robota zvolíme pero. Na plátne vhodne zvolíme *WorkObject*. Pripravíme si *Target* na začiatočnú pozíciu robota a nazveme ho DOMOV. Zvolíme robota *IRB 120*. Pri dodržaní postupu z predošlej kapitoly by naša scéna mala vyzerať tak, ako na obrázku 38.



Obr. 38 Pripravená scéna

Robot bude vykonávať pohyby na základe súradníc, ktoré bude dostávať zo C# aplikácie. Ďalej bude na strane aplikácie C# možné nastavenie rýchlosti robota. S týmito náležitosťami by sme mali počítať, keď budeme písať zdrojový kód. Nasledujúci vývojový diagram stručne charakterizuje algoritmus použitý na strane ABB RobotStudio. Switch CASE formula beží v cykle. Cyklický beh programu je nastavený v programe C#

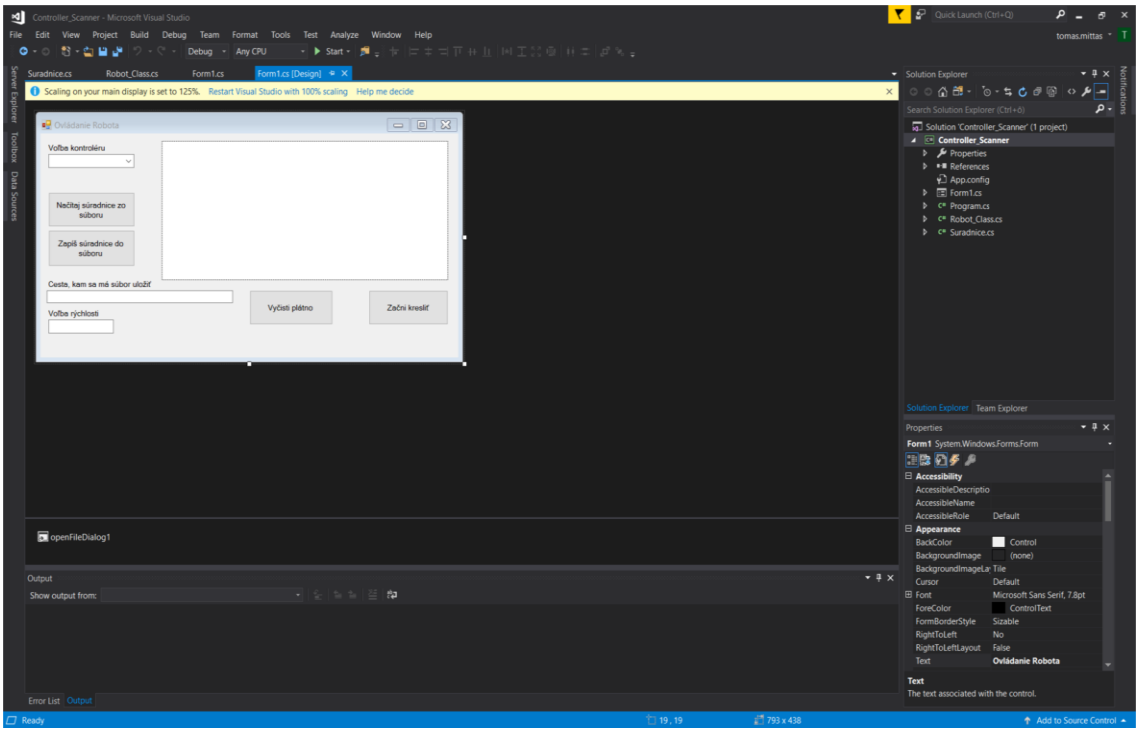


Obr. 39 Vývojový diagram použitého algoritmu

Podľa daného algoritmu môžeme vidieť jeho princíp. Dáta zadefinované na začiatku kódu budú menené pomocou aplikácie napísanej v jazyku C#. Na základe zmeny dát sa zmení stav podmienok, ktoré nám umožnia vykonať pohyb robota. Je dôležité zosynchronizovať stanicu s RAPID kódom, aby sa nám vygenerovali dáta pre nástroj, prostredie a *Targety*, ktoré sme si vytvorili. Pred tým, než je možné ovládať stanicu robota pomocou jazyka C#, je potrebné nastaviť *Program pointer* v RAPID kóde do procesu main().

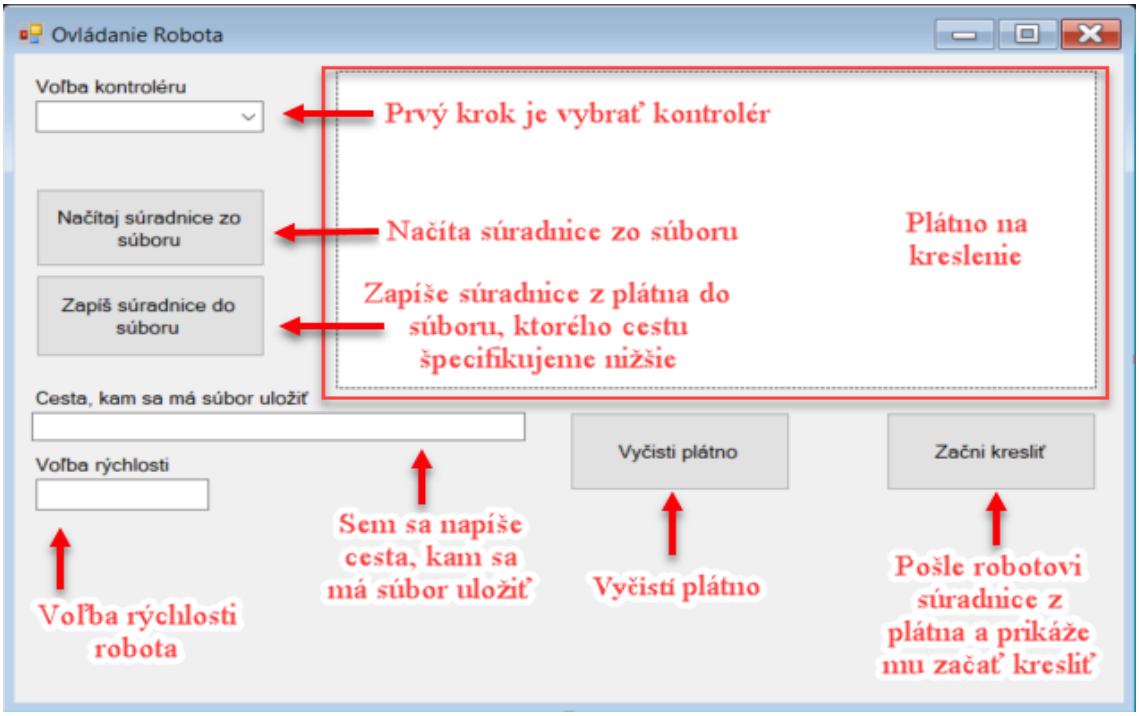
6.3 Aplikácia v jazyku C#

Keď máme všetko pripravené na strane ABB RobotStudio, začneme s prácou na aplikácii. Vytvoríme si *Windows Forms App* podľa inštrukcii v predošlej kapitole. Uživatelské rozhranie našej aplikácie by malo vyzerať následovne.



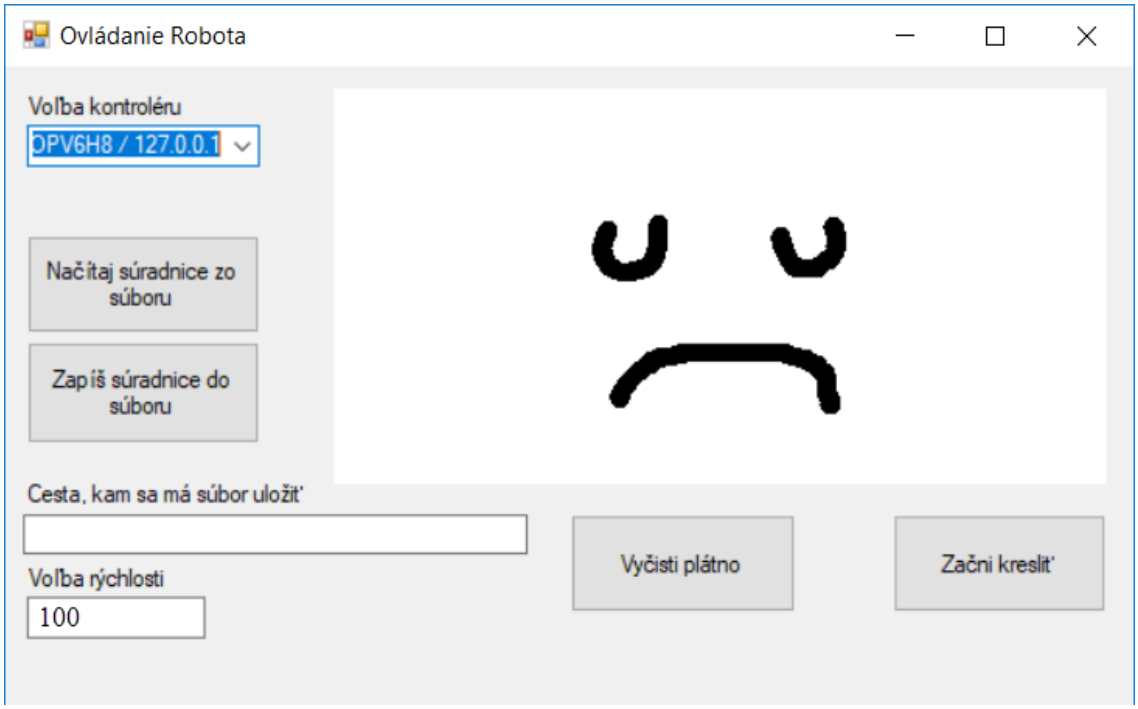
Obr.40 Uživatelské rozhranie aplikácie

Pred tým, než môžeme posielať dáta do ABB RobotStudio a tým rozhýbať robota, potrebujeme v sieti nájsť kontrolér a začať s ním komunikáciu. Tohto dosiahneme pomocou comboboxu *Voľba kontroléru*. Po jeho rozkliknutí sa nám zobrazia kontroléry dostupné v sieti. Ten, ktorý vyberieme bude sprostredkúvať naše dáta robotovi a zmenou dát v RAPID kóde kontroléru budeme schopný robota ovládať. Pokiaľ chceme súradnice iba zapísať do súboru, nie je potrebné mať vybratý kontrolér. Detailný popis kódu je pripojený v prílohe.
V aplikácii sa nachádzajú tlačidlá, ktoré majú rôznu funkciu. Tlačidlo *„Načítaj súradnice zo súboru“* načíta súradnice z vybraného súboru, ktorého formát je v tomto prípade *.txt*, aby bolo jednoduchšie čítať jeho dáta vo forme dátového typu *string*. Tieto súradnice musia byť zapísané v presne definovanom formáte, inak toto tlačidlo nebude fungovať. Taktiež vykreslí obrazec súradníc zo súboru na plátno. Pomocou tlačidla *„Zapíš súradnice do súboru“* môžeme zapísať súradnice z plátna do daného súboru. Súradnice sú zapisované vo formáte špecifickom pre daný algoritmus, aby sa dali ľahšie čítať. Cesta súboru je špecifikovaná v textboxe Funkcia tlačidla *„Vyčisti plátno“* je zrejmá z jeho názvu. Po jeho stlačení nám vyčistí čokoľvek sa na plátne momentálne nachádza. Vymaže aj aktuálnu pamäť aplikácie. Tlačidlo *„Začni kresliť“* nahrá súradnice z plátna do ABB RobotStudio, kde pomocou nich vytvoríme *Targety*. Samotné nahrávanie istú chvíľu trvá. Po ich načítaní sa vykreslia všetky obrazce uložené v súbore (do počtu *Targetov* 500). V aplikácii môžeme robotovi nastaviť aj rýchlosť, akou má kresliť.



Obr. 41 Popis nástrojov aplikácie

Pri spustení a používaní môže aplikácia vyzerať napríklad ako na obrázku 42.



Obr. 42 Rozbehnutá aplikácia

Aj keď sa obrazce vykreslené na plátne môžu zdať trochu kostrbaté, presnosť pohybov robota je zaručená pomocou argumentu *zone* v inštrukcii *MoveL* v ABB RobotStudio.

7. ZHODNOTENIE A POROVNANIE

Ovládanie robota pomocou jazyka C# sa môže zdať ako netradičné riešenie. Dalo by sa argumentovať, že takéto ovládanie je zbytočné, pretože ABB RobotStudio má vysoko prepracované inštrukcie, prostredie, možnosti a nástroje na prácu s robotom. V tejto kapitole budú vypísané prednosti a rozdiely jednotlivých prostredí a prístupov.

7.1 ABB RobotStudio

ABB RobotStudio dokazuje kvality prvenstva spoločnosti ABB v rebríčku svetovej robotiky. Práca s ním je uživateľsky prijateľná. Nasledovať bude popísanie jeho najdôležitejších vlastností.
Medzi hlavné výhody ABB RobotStudio patrí bohatá podpora knižníc pre prácu s robotmi ako aj simuláciu a vizualizáciu a taktiež množstvo inštrukcii na pohyb robota a jeho konfiguráciu. Okrem iného sa v ňom dá simulovať fyzika objektov, vytvoriť prostredie a používať nástroje, s ktorými roboty reálne aj pracujú. Veľkou výhodou je tiež, že si môžeme importovať *CAD objekty*.
Keďže je RobotStudio určené na prácu s robotmi a ich programovanie, je veľmi obtiažne nájsť veľké nedostatky. Vytknúť by sa mu ale dalo horšie debugovanie a testovanie, slabšia prehľadnosť kódu RAPID a nemožnosť používania objektovo orientovaného programovania (čo v konečnom dôsledku často netreba pretože RobotStudio to dokáže nahradiť inými spôsobmi, avšak za zmienku to stojí).

7.2 Microsoft Visual Studio a jazyk C#

Microsoft Visual Studio je vývojové prostredie určené na vývoj rôznych druhov aplikácii na rôznych úrovniach. Podporuje radu programovacích jazykov medzi nimi aj C#. V tejto podkapitole budú v krátkosti popísané jeho klady aj zápory v rámci problému robotiky.
Podpora robotiky v Microsoft Visual Studio je zaistená predovšetkým pomocou rozšírení, ako je PC SDK a RoboSDK od firmy ABB určených pre ich produkty. Vďaka týmto rozšíreniam máme k dispozícii radu knižníc a metód, pomocou ktorých sa dá pomerne efektívne pracovať s dátami a nástrojmi RobotStudio. Hlavnými výhodami tohto prostredia je väčšia prehľadnosť kódu, lepšie nástroje na spravovanie kódu a efektívne, prehľadné a jednoduchšie debugovanie a testovanie. Za zmienku tiež stojí, že do Visual Studio je možné stiahnuť programy, ktoré pomáhajú s písaním kódu (napríklad Resharper od spoločnosti Jet Brains, ktorý je určený pre jazyk C#). Na strane jazyka C# je výhodou možnosť objektovo orientovaného programovania, *garbage collector* a bohatá podpora knižníc na prácu s dátami, procesmi, konverziou a podobne. Výhodou tiež je tvorba externej aplikácie schopnej ovládať robota (napríklad ako istá forma *HMI - human machine interface*).
Medzi hlavné nevýhody patrí neschopnosť používania inštrukcii na pohyb robota vo Visual Studio za použitia jazyka C# a niekedy problematická konverzia dátových typov. Nedostatok metód na prácu s robotom nás často limituje na zmenu dát na strane C# aplikácie a následné posielanie týchto dát do RAPID kódu v RobotStudio.

8. ZÁVER

Cieľom tejto práce bolo ukázať multifunkčnosť jazyka C# a jeho použitie v aplikáciách pre tento jazyk netradičných. Na začiatku sa práca zaoberá robotikou ako vednou oblasťou, jej históriou, rozdelením a časťami robotiky. V krátkosti sa práca venuje aj snímačom. Ďalšia časť práce sa venuje spoločnosti ABB a jej produktom, ktoré sú v riešenej problematike využívané.
Na začiatok riešenia daného problému bolo potrebné stiahnuť a nainštalovať používaný software. PC SDK je voľne dostupné zo stránok spoločnosti ABB, no na ABB RobotStudio a Microsoft Visual Studio je potrebné mať licenciu (napríklad školskú alebo študentskú). Ďalší krok bol zoznámiť sa s jednotlivými prostrediami, minimálne na uživateľskej úrovni.
Nasledoval postup práce pri vytvorení stanice s robotom a prostredím a jeho rozhýbanie pomocou nástrojov, ktoré RobotStudio ponúka. Riešenie ukázalo výhodnosť použiť RobotStudio pre ovládanie robota. Je možné skonštatovať, že RobotStudio je vysoko sofistikovaný nástroj určený na prácu v oblasti robotiky a poskytuje solídne prostriedky na prácu s robotmi. Detailný postup tvorby krok po kroku sa nachádza v prílohe 1.
Jadrom práce a hlavnou časťou bolo ovládanie robota pomocou jazyka C#, čo je znázornené v kapitole 6. Môžeme zhodnotiť, že C# svoju prácu odviedol prijateľne, aj napriek tomu že robot nie je ovládaný volaním metód alebo iným spôsobom z aplikácie C#. Podstatou riešenia je fakt, že pomocou aplikácie dokážeme meniť hodnoty dát v RobotStudio a vhodnou voľbou podmienok, inštrukcii a deklarácii dát dokážeme vytvoriť funkčnú stanicu ovládanú externe pomocou aplikácie naprogramovanej v jazyku C#. Presnejší popis kódu a jeho funkcii je v prílohe 2.
Je dosť zložité porovnávať ABB RobotStudio a Microsoft Visual Studio spolu s jazykom C# v rámci problému robotiky. Vo viacerých ohľadoch si RobotStudio vedie omnoho lepšie, čo je však prihliadajúc k jeho funkcii pochopiteľné. Aj napriek tomu dokázalo Microsoft Visual Studio spolu s jazykom C# ukázať svoje kvality, hlavne v prehľadnosti testovania a debugovania aplikácie. Poukazujúc na fakt, že Visual Studio, rovnako ako jazyk C# sa obyčajne používajú na iné oblasti, je možné povedať, že práca s nimi bola v tomto ohľade bez väčších problémov. Určite je možné skonštatovať, že takéto spojenie dvoch prostredí a vyššieho jazyka ako je C# má zmysel a v budúcnosti offline programovania robotov má potenciál.
Na elektronickom nosiči je dodané video, kde je aplikácia testovaná na reálnom robotovi v laboratóriu a tiež video, na ktorom je ukázané fungovanie aplikácie v spojení so simuláciou v ABB RobotStudio.

9. ZOZNAM POUŽITEJ LITERATÚRY

[1] KOLÍBAL, Zděnek. *Roboty a robotizované výrobní technologie*. Brno: Vysoké učení technické v Brně – nakladatelství VUTIUM, 2016, ISBN 978-80-214-4828-5
[2] SKAŘUPA, Jiří. *Průmyslové roboty a manipulátory* [online]. Ostrava: Vysoká škola báňská - Technická univerzita, [2008] [cit. 2019-04-11]. ISBN 978-80-248-1522-0.
[3] ABB. History[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/about/abb-in-brief/history>
[4] ABB. Základné údaje[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/cz/o-nas/zakladni-udaje>
[5] ABB. Průmyslové roboty *IRB 2400*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/robotics/cs/prumyslove-roboty/irb-2400>
[6] ABB. Průmyslové roboty *YUMI*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/robotics/cs/prumyslove-roboty/yumi>
[7] ABB. Roboty *IRB 360*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/robotics/sk/roboty/irb-360>
[8] Robot *Unimate*[obrázok]. Sutori.com, 2019, [cit. 2019-03-31]. Dostupné z: <https://www.sutori.com/item/1961-general-motors-puts-the-first-industrial-robot-unimate-into-use>
[9] Robot *Versatran*[obrázok]. Cyberneticzoo.com, 2013,[cit. 2019-03-31.]. Dostupné z: <http://cyberneticzoo.com/early-industrial-robots/1958-62-versatran-industrial-robot-harry-johnson-veljko-milenkovic/>
[10] SMC. *Sensors Technology*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://smctraining.mrooms.net/mod/scorm/player.php?a=203&currentorg=TOC1&scoid=406&sesskey=ztyjSVFgqR&display=popup&mode=normal>
[11] *Teach-pendant*[obrázok]. Daincube.com, 2018, [cit. 2019-03-31]. Dostupné z: <http://www.daincube.com/>
[12] ABB. *Industrial robots IRB 120*[online]. 2019, [cit. 2019-03-31]. Dostupné z : <https://new.abb.com/products/robotics/industrial-robots/irb-120>
[13]ABB. *Product specification IRB 120*[PDF dokument]. 2018, [cit. 2019-03-31]. Dostupné z: <https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC035960-001&LanguageCode=en&DocumentPartId=&Action=Launch>
[14] ABB. Řídící systémy[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/robotics/cs/ridici-systemy>
[15] ABB. *Omnicore controller data sheet*[PDF dokument]. 2018, [cit. 2019-03-31]. Dostupné z: <https://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK107046A3855&LanguageCode=en&DocumentPartId=&Action=Launch>
[16] ABB. *Omnicore controller*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/robotics/controllers/omnicore>
[17] ABB. *Introduction to RAPID*[PDF dokument]. 2007, [cit. 2019-03-31]. Dostupné z: <http://www.oamk.fi/~eeroko/Opetus/Tuotantoautomaatio/Robotiikka/Introduction_to_RAPID_3HAC029364-001_rev-_en.pdf>
[18] ABB. *RAPID Technical reference manual*[PDF dokument]. 2010, [cit. 2019-03-31]. Dostupné z: <https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf>

[19] ABB. *RobotWare data sheet*[PDF dokument]. 2004, [cit. 2019-03-31]. Dostupné z: <https://library.e.abb.com/public/c351e246eeb20a19c12570b90039d5dc/RobotWare%20data%20sheet.pdf>
[20] ABB. *Programmable logic controllers*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/plc/programmable-logic-controllers-plcs>
[21]. ABB. *Industrial Automation Division*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/about/our-businesses/industrial-automation-division>
[22]. ABB. ABB *Teach-pendant*[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/3HAC028357-001/teach-pendant>
[23] BOŽEK, Pavol, BARBORÁK, Oto, NAŠČÁK, Ľubomír, ŠTOLLMANN, Vladimír. Špecializované robotické systémy. 1. vyd, ÁMOS STU Bratislava, 2011, ISBN 978-80-904766-8-4. Dostupné z: <http://www.uiam.mtf.stuba.sk/predmety/srs/1-Ucebnica/#_Toc310244205>
[24] VACHÁLEK, Ján, TAKÁCS, Gergély. *Robotika*. V Bratislave: Slovenská technická univerzita v Bratislave, 2014. Edícia vysokoškolských učebníc (Slovenská technická univerzita). ISBN 978-80-227-4163-7
[25] ABB. *IRC5*[PDF dokument]. 2019, [cit. 2019-03-31]. Dostupné z: <https://search-ext.abb.com/library/Download.aspx?DocumentID=ROB0295EN&LanguageCode=en&DocumentPartId=&Action=Launch>
[26] ABB. *IRC5 datasheet*[PDF dokument]. 2019, [cit. 2019-03-31]. Dostupné z: <https://library.e.abb.com/public/c13e1c5490c61230c125796000515137/IRC5%20datasheet%20PR10258%20EN_R13.pdf?fbclid=IwAR3V3A34Cyz6XLKtWRjC2wqcPq49TI--U75bSGsWjVEGtci9KkaCi4FIOdQ>
[27] SMC. *Principles of pneumatics*[online]. 2019, [2019-03-31]. Dostupné z: <https://smctraining.mrooms.net/mod/scorm/player.php?a=199&currentorg=TOC1&scoid=398&sesskey=Ggt2NlIHxW&display=popup&mode=normal>
[28] ABB. *PC SDK Application manual*[PDF dokument]. 2012, [2019-03-31]. Dostupné z: <https://library.e.abb.com/public/124d6b59313ed85fc125793400410c5b/3HAC036957-en.pdf>
(opat 28 nie je citovany v texte, ale informacie z neho som pri praci pouzival)
[29] ABB. Robotic[online]. 2019, [cit. 2019-03-31]. Dostupné z: <https://new.abb.com/products/robotics/>
[30] RÁCZ, Ervin. ABB drawing robot – Windows Forms App – PC SDK [online]. 2017, [cit. 2019-03-31]. Dostupné z: < https://www.youtube.com/watch?v=a_2hTYFHBGo >
[31] RÁCZ, Ervin. *ABB robot - C# App* [software]. 2017, [cit. 2019-03-31]. Dostupné z: <https://drive.google.com/file/d/0By6anwacYsC3QnFUTE1LZ20xdGM/view?usp=drive_open>
[32] BOLMSJÖ, Gunnar. *RobotStudio Drawing on Paper 1 – basic tutorial* [CAD objekt]. 2017, [cit. 2019-03-31]. Dostupné z : <https://app.box.com/s/od57k7pvbdl17x1knt2dlsi99bd41ouc/folder/38943459125>

10. ZOZNAM PRÍLOH

10.1 PRÍLOHA 1. : Popis funkcii triedy Robot_Class v aplikácii C#

```
private Controller controller = null;// náš kontrolér
private Num prechood;// pomocná premenná pri načítaní počtu targetov
public List<Suradnice> NaNahratie = new List<Suradnice>();//Na vykreslovanie
//obrazcu z načítaného súboru
private Num rychlost;//pre nastavenie rýchlosti
private Num stavRych;//stav na zmenu rýchlosti
private Num stavPoh;//stav na začatie pohybu kreslenia

//-----------------------------------------------------------------
//Priradí kontrolér a jeho informácie
//-----------------------------------------------------------------
//V konštruktore priraď kontrolér, ktorý je nájdený scannerom
//v connect_Click comboboxe. Pred každou prácou aplikácie s dátami
//v ABB RobotStudio bude táto metóda volaná.
//-----------------------------------------------------------------
public Robot_Class(Controller controller)
{
   this.controller = controller;
   this.controller.Logon(UserInfo.DefaultUser);
}
//-----------------------------------------------------------------------
//Spustí RAPID program v RobotStudio
//-----------------------------------------------------------------------
//Ak je kontroler v auto mode, nie manual tak požiadaj o mastership
//aby sa mohol spustiť RAPID program. Túto metódu budeme volať vždy,
//keď budeme chcieť uložiť zmeny v RAPID kóde a spustiť náš program.
//-----------------------------------------------------------------------
public void StartRapid()
{
   try
   {
     if(controller.OperatingMode == ControllerOperatingMode.Auto)
      {
        using (Mastership m = Mastership.Request(controller.Rapid))
        {
          controller.Rapid.Cycle = ExecutionCycle.Forever;
          controller.Rapid.Start(true);
        }
       }
   }
   catch(System.Exception ex)
   {
     MessageBox.Show(ex.Message);
   }
}
```

```
//---------------------------------------------------------------------
//Zapíše súradnice z aplikácie do RobotStudio
//---------------------------------------------------------------------
//Táto metóda nám bude slúžiť na posielanie súradníc z plátna
//v aplikácii do ABB RobotStudio, kde z nich vytvoríme Targety.
//Najskôr získame dáta z kontroléru a priradíme im naše hodnoty.
//Potom založíme premennú pozícia, ktorej priradíme dáta na základe
//súradníc z nášho plátna. Nakoniec tieto pozície po jednej zapíšeme
//do premennej v RAPID programe.
//---------------------------------------------------------------------
public void PosliSuradnice(List<Suradnice> targety)
{
  try
  {
    RapidData pocetCyklov = controller.Rapid.GetRapidData("T_ROB1",
"Module1", "pocitadlo");
    int pocetTarg = targety.Count;
    prechood.FillFromString2(pocetTarg.ToString());
    using (Mastership mstr = Mastership.Request(controller.Rapid))
    {
      pocetCyklov.Value = prechood;
    }
    RapidData naPoslanie = controller.Rapid.GetRapidData("T_ROB1", "Module1",
"tgPos");
    Pos pos;
    for (int i = 0; i < targety.Count; i++)
    {
      pos = new Pos();
      pos.X = targety[i].X;
      pos.Y = targety[i].Y;
      pos.Z = targety[i].Z;
      using (Mastership m = Mastership.Request(controller.Rapid))
      {
        naPoslanie.WriteItem(pos, i);
      }
    }
  }
  catch(Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}
//---------------------------------------------------------------------------
//Načíta súradnice zo súboru
//---------------------------------------------------------------------------
//Načíta súradnice zo súboru, ktorého cesta je špecifikovaná ako vstupný
//parameter. Tiež nastaví počet pozícii na počet bodov, nahratých zo súboru
//Taktiež načíta body do pamäte na vykreslenie na plátno.
//---------------------------------------------------------------------------
public void NacitajSubor(string cesta)
{
  string content = File.ReadAllText(cesta);
  List<string> nacitaj = content.Split(';').ToList();
  RapidData pocetCyklov = controller.Rapid.GetRapidData("T_ROB1", "Module1",
"pocet_pozicii");
  int pocetTarg = (nacitaj.Count)/3;
  prechood.FillFromString2(pocetTarg.ToString());
  using (Mastership mstr = Mastership.Request(controller.Rapid))
  {
     pocetCyklov.Value = prechood;
  }
```

```
    RapidData naPoslanie = controller.Rapid.GetRapidData("T_ROB1", "Module1",
    "tgPos");
    Pos pos;
    var k = 0;
    for (int i = 0; i < ((nacitaj.Count) / 3); i++)
    {
      pos = new Pos();
      pos.X = float.Parse(nacitaj[k]);
      pos.Y = float.Parse(nacitaj[k + 1]);
      pos.Z = float.Parse(nacitaj[k + 2]);
      NaNahratie.Add(new Suradnice(Int32.Parse(nacitaj[k ]),
      Int32.Parse(nacitaj[k]), Int32.Parse(nacitaj[k+2])));
      NaNahratie[i].X = Int32.Parse(nacitaj[k]);
      NaNahratie[i].Y = Int32.Parse(nacitaj[k+1]);
      NaNahratie[i].Z = Int32.Parse(nacitaj[k+2]);
      k = k + 3;
      using (Mastership m = Mastership.Request(controller.Rapid))
      {
        naPoslanie.WriteItem(pos, i);
      }
    }
}


//-----------------------------------------------------------
//Povolí kreslenie v RobotStudio
//-----------------------------------------------------------
//Táto funkcia nám po zavolaní nastaví kľúčovú hodnotu na
//hodnotu, ktorá nám umožňuje začať kresliť. Bez volania
//tejto funkcie by aj po nahraní súradníc do RobotStudio
//proces kreslenia nezačal.
//-----------------------------------------------------------
public void Zacni(List<Suradnice> suradnices)
{
  stavPoh.Value = 1;
  RapidData stav = controller.Rapid.GetRapidData("T_ROB1", "Module1",
"stav");
  using (Mastership m = Mastership.Request(controller.Rapid))
  {
    stav.Value = stavPoh;
  }
}
//-------------------------------------------------------------
//Zmení rýchlosť vykreslovania
//-------------------------------------------------------------
//Nasledujúce funkcie slúžia na zmenu rýchlosti robota.
//Data z textboxu zapíšeme do premennej z RAPID kódu.
//Pokiaľ je textbox prázdny, naplň ho hodnotou 100, pokiaľ
//je hodnota priveľmi veľká alebo priveľmi malá, ohraničju
//zhora hodnotou 500 a zdola hodnotou 20.
//-------------------------------------------------------------
public void zmena_rychlost(string rych)
{
  if(rych == string.Empty)
  {
   rych = "100";
  }
  var hodnota = Int32.Parse(rych);
  if (hodnota > 500)
  {
   hodnota = 500;
  }
```

```
  if (hodnota < 0)
  {
     hodnota = 20;
  }
 RapidData menena_rychlost = controller.Rapid.GetRapidData("T_ROB1", "Module1",
 "menena_rychlost");
 RapidData stavR = controller.Rapid.GetRapidData("T_ROB1","Module1","stav");
 rychlost.Value = hodnota;
 stavRych.Value = 3;
 using (Mastership m = Mastership.Request(controller.Rapid))
 {
   menena_rychlost.Value = rychlost;
   stavR.Value = stavRych;
 }
}
```

10.2 PRÍLOHA 2. : Popis funkcii triedy Form v aplikácii C#

```
private NetworkScanner scanner = null; // skener na najdenie kontroleru v sieti
private Robot_Class mojRobot = null; //
private List<Suradnice> pozicia;
private SolidBrush farbaKreslenia;
private bool kresli = false
private bool Nahrate = false;

//-------------------------------------------------------------
//Nájde kontrolér a nastaví default rozhranie aplikácie
//-------------------------------------------------------------
//Pri spustení aplikácie sa nastaví hodnota comboBoxu rýchlosti
//na normálnu. Ďalej je sieť zoskenovaná pre aktívne kontroléry
//a pokiaľ sú nájdené kontroléry tak nech ich priradí do comboboxu
//na výber pomocou ich mena a IP adresy.
//-------------------------------------------------------------
private void Form1_Load(object sender, EventArgs e)
{
  RychlostHodnota.Text = "100";
  this.scanner = new NetworkScanner();
  this.scanner.Scan();
  ControllerInfoCollection controllers = scanner.Controllers;
  foreach (ControllerInfo info in controllers)
  {
    controller_comboBox.Items.Add(info.ControllerName + " / " +
info.IPAddress.ToString());
  }
  pozicia = new List<Suradnice>();
}
//-------------------------------------------------------
//Zapíše súradnice do súboru
//-------------------------------------------------------
//Pokiaľ dôjde k stlačeniu tlačidla na zápis, zavolaj
//funkciu na zápis do súboru. Kedže ZapisDoSuboru je
//metódou našej triedy Robot_Class, je potrebné vytvoriť
//objekt.
//-------------------------------------------------------
private void Zapis_Click(object sender, EventArgs e)
{
  mojRobot = new Robot_Class();
  Zapis_Body();
  pozicia = new List<Suradnice>();
}
//Zavolané pri stlačení tlačidla vyčisti plátno, vyčistí nám plátno.
private void VycistiPlatno_Click(object sender, EventArgs e)
{
  Graphics graphics = panel1.CreateGraphics();
  graphics.Clear(panel1.BackColor);
  pozicia = new List<Suradnice>();
}
```

```
//---------------------------------------------------------------------
//Načítanie súradníc zo súboru
//---------------------------------------------------------------------
//Najskôr si založíme objekt kontroléry, do ktorých priradíme info
//o kontroléroch nachádzajúcich sa v siet. Potom testujeme, či kontrolér
//zvolený v comboBoxe sa zhoduje s názvom kontroléru v sieti, a pokiaľ áno
//či je prístupný. Pokiaľ toto spĺňa, založíme objekt robota, priradíme mu
//rýchlosť na základe hodnoty comboBoxu na rýchlosť a zavoláme metódu na
//načítanie súradníc zo súboru. Po načítaní nastavíme hodnotu kľúčovej
//premennej
//flag pomocou metódy Zacni na hodnotu, ktorá povoľuje kreslenie. Nakoniec
//spustíme RAPID kód pomocou metódy StartRapid.
//---------------------------------------------------------------------
private void NacitajZoSuboru_Click(object sender, EventArgs e)
{
  DialogResult vysledok = openFileDialog1.ShowDialog();
  ControllerInfoCollection controllers = scanner.Controllers;
  foreach (ControllerInfo info in controllers)
  {
    if (controller_comboBox.Text.Equals(info.ControllerName + " / " +
info.IPAddress.ToString()))
    {
      if (info.Availability == Availability.Available)
      {
        mojRobot = new Robot_Class(ControllerFactory.CreateFrom(info));
        if( vysledok == DialogResult.OK)
        {
              string cesta = openFileDialog1.FileName;
              try
              {
              mojRobot.NacitajSubor(cesta);
              }
              catch (IOException)
              {
              }
        }
        List<Suradnice> Nahravaci = mojRobot.NaNahratie;
        for (var i = 0; i < Nahravaci.Count; i++)
         {
           farbaKreslenia = new SolidBrush(Color.Black);
           Graphics graphics1 = panel1.CreateGraphics();
           graphics1.FillEllipse(farbaKreslenia, Nahravaci[i].X, Nahravaci[i].Y,
           10 10);
           graphics1.Dispose();
         }
         Nahrate = true;
      }
    }
  }
}
```

```
//-----------------------------------------------------------
//Spustenie kreslenia, nastavenie rýchlosti
//-----------------------------------------------------------
//Podobné predošlej metóde, jediným rozdielom je volanie
//metódy PosliSuradnice objektu mojRobot, ktorá slúži na
//poslanie súradníc z plátna do robota. Po stlačení tlačidla
//Začni kresliť teda robot začne kresliť to, čo je na plátne.
//-----------------------------------------------------------
private void ZacniKreslit_Click(object sender, EventArgs e)
{
  ControllerInfoCollection controllers = scanner.Controllers;
  foreach (ControllerInfo info in controllers)
  {
    if (controller_comboBox.Text.Equals(info.ControllerName + " / " +
info.IPAddress.ToString()))
    {
      if (info.Availability == Availability.Available)
      {
        mojRobot = new Robot_Class(ControllerFactory.CreateFrom(info));
        if(RychlostHodnota.Text == string.Empty)
         {
           RychlostHodnota.Text = "100";
         }
        if (RychlostHodnota.Text == "100")
        {
          mojRobot.zmena_rychlost("100");
        }
        mojRobot.Zacni();
        if (pozicia.Count != 0)
        {
          mojRobot.PosliSuradnice(pozicia);
          mojRobot.StartRapid();
        }
        if (Nahrate)
        {
          mojRobot.StartRapid();
          Nahrate = false;
        }
      }
    }
  }
}
//-------------------------------------------------------------
//Zapísanie súradníc do súboru
//-------------------------------------------------------------
//Po zavolaní zapíše súradnice z plátna do súboru, ktorého path
//špecifikujeme v textboxe.
private void Zapis_Body()
{
  for (int j = 0; j < pozicia.Count; j++)
  {
    File.AppendAllText(@"" + textBox1.Text + ".txt", pozicia[j].VratSur());
  }
}
```

```
//-------------------------------------------------------------
//Zapísanie prvej súradnice a povolenie kreslenia
//-------------------------------------------------------------
//Po stlačení myši nastavíme kresli na jedna a založíme nový list
//nášho objektového typu Suradnice. Prvá pozícia sa zapíše.
//-------------------------------------------------------------
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
  kresli = true;
  pozicia.Add(new Suradnice(e.X, e.Y, 0));
}
//-------------------------------------------------------------
//Vykreslovanie obrazca na plátne a zapisovanie súradníc do listu
//-------------------------------------------------------------
//Po pohnutí myši na plátne sa vykreslí čiernou čiarou elipsa
//na každej pozícii, kade prejdeme myšou. Tieto pozície sa v
//podobe súradníc zapíšu do nášho listu. Následne sa zbavíme
//grafiky vykreslenej na plátne (no nevymažeme dáta v liste
//a ani nakreslený obrazec, iba jeho resources.
//-------------------------------------------------------------
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
  if (kresli)
  {
    farbaKreslenia = new SolidBrush(Color.Black);
    Graphics graphics1 = panel1.CreateGraphics();
    graphics1.FillEllipse(farbaKreslenia, e.X, e.Y, 10, 10);
    pozicia.Add(new Suradnice(e.X, e.Y, 0));
    graphics1.Dispose();
  }
}
//-------------------------------------------------------------
//Zapísanie poslednej súradnice a zastavenie kreslenia
//-------------------------------------------------------------
//Po odkliknutí myši nastav kreslenie na false, čiže už sa na plátne
//pri pohybe myši nebude nič zobrazovať, a zapíš poslednú súradnicu
//kde bol kurzor pred odkliknutím. -30 je v súradnici  kvôli offsetu
//aby pri kreslení ďalšieho obrazca robot neťahal pero po podložke.
//-------------------------------------------------------------
private void panel1_MouseUp(object sender, MouseEventArgs e)
{
kresli = false;
pozicia.Add(new Suradnice(e.X, e.Y, -30));
}
```

```
//-----------------------------------------------------------------
//Zmena rýchlosti kreslenia
//-----------------------------------------------------------------
//Po prepísaní hodnoty v textboxe zmení hodnotu rýchlosti, akou robot
//pracuje na hodnotu v textboxe. Neodporúča sa meniť rýchlosť počas
//práce robota.
//-----------------------------------------------------------------
private void RychlostHodnota_TextChanged(object sender, EventArgs e)
{
  ControllerInfoCollection controllers = scanner.Controllers;
  foreach (ControllerInfo info in controllers)
  {
   if (controller_comboBox.Text.Equals(info.ControllerName + " / " +
   info.IPAddress.ToString()))
   {
     if (info.Availability == Availability.Available)
     {
     mojRobot = new Robot_Class(ControllerFactory.CreateFrom(info));
     mojRobot.zmena_rychlost(RychlostHodnota.Text);
     mojRobot.StartRapid();
     }
   }
  }
}
```

10.3 Popis funkcii v aplikácii ABB RobotStudio

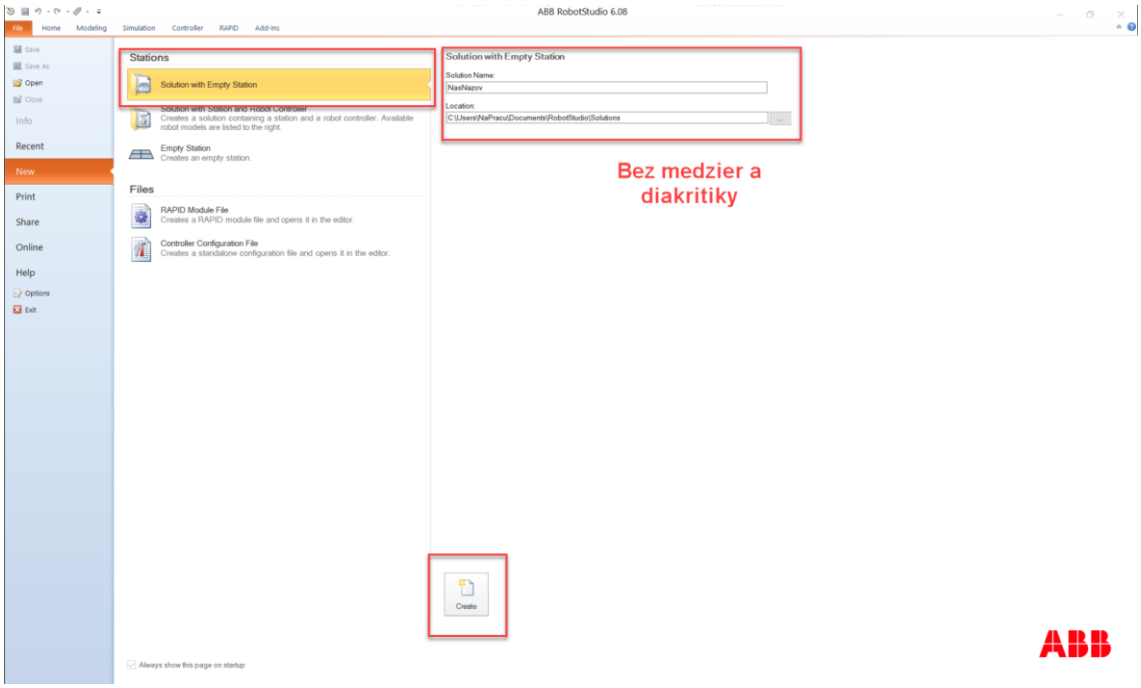
Pred ovládaním programom C# je potrebné nastaviť program pointer do procesu main() v RAPID kóde našej stanice.

```
VAR num pocet_pozicii := 1; !Počet pozícii na pohyb
VAR num stav := 0; !Stav, ktorý má robot vykonať
CONST num KRESLIT :=1; !Hodnota stavu na kreslenie
CONST num DOMOV := 2; !Hodnota stavu na pohyb domov
CONST num ZMENENA_RYCHLOST := 3; !Hodnota stavu na zmenu rýchlosti
CONST num CHOD_DO_TARGETU := 4; !Hodnota stavu na pohyb do targetu,
nachystané pre rozšírenie programu
VAR speeddata rychlost := v100; !Pôvodná hodnota rýchlosti
VAR num menena_rychlost := 100; !premenná na menenie rýchlosti
VAR pos tgPos{500};!Pole pozícii nahratých zo C#
 VAR pos premenna_pozicia; !Pozícia na vytvorenie targetu pri jednoduchom pohybe
VAR robtarget premenny_target; !Target pre jednoduchý pohyb
VAR robtarget bod; !Bod, do ktorého sa vykoná pohyb v cykle FOR

!Tento proces nám slúži na zmenu rýchlosti robota. Zmena sa deje
!na základe dát, ktoré nám posiela aplikácia v C#, teda nastavuje
!hodnotu menit a tak mení rýchlosť.
PROC ZmenaRychlosti(num menit)
       rychlost := [menit,500,5000,1000];
ENDPROC

!Proces na vykonanie pohybu do jedného targetu. Ovládaný samostatnou aplikáciou a
!použití v inej stanici. V tejto stanici slúži pre možné rozšírenie programu. Po pohybe
!stav nastavený na DOMOV, aby sa robot vrátil do domovskej pozície.
PROC ChodDoTargetu()
    premenny_target :=
[premenna_pozicia,[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    MoveJ premenny_target,v100,z80,pen\WObj:=NovePlatno;
    WaitTime 0.5;
    stav := DOMOV;
ENDPROC
!Proces Vykresli body v sebe obsahuje cyklus na pohyb robota. Pri každom jeho
!volaní máme v tgPos nahraté pozície, z ktorých postupne vytvárame target a
!následne sa do tohto targetu robot pohne. Po všetkých pohyboch sa stav nastaví
!na DOMOV aby sa robot vrátil do domovskej pozície.
PROC VykresliBody()
     ConfJ \Off;
     FOR i FROM 1 TO pocet_pozicii DO
       bod := [tgPos{i},[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
          MoveJ bod,rychlost,z100,pen\WObj:=NovePlatno;
     ENDFOR
     stav := DOMOV;
   ENDPROC
!Proces, ktorý nás dostane do domovskej pozície
PROC ChodDomov()
       MoveAbsJ  [ [ 0, 0, 0, 0, 0, 0], [ 0, 9E9, 9E9, 9E9, 9E9, 9E9] ],v100,z100,pen;
ENDPROC
```

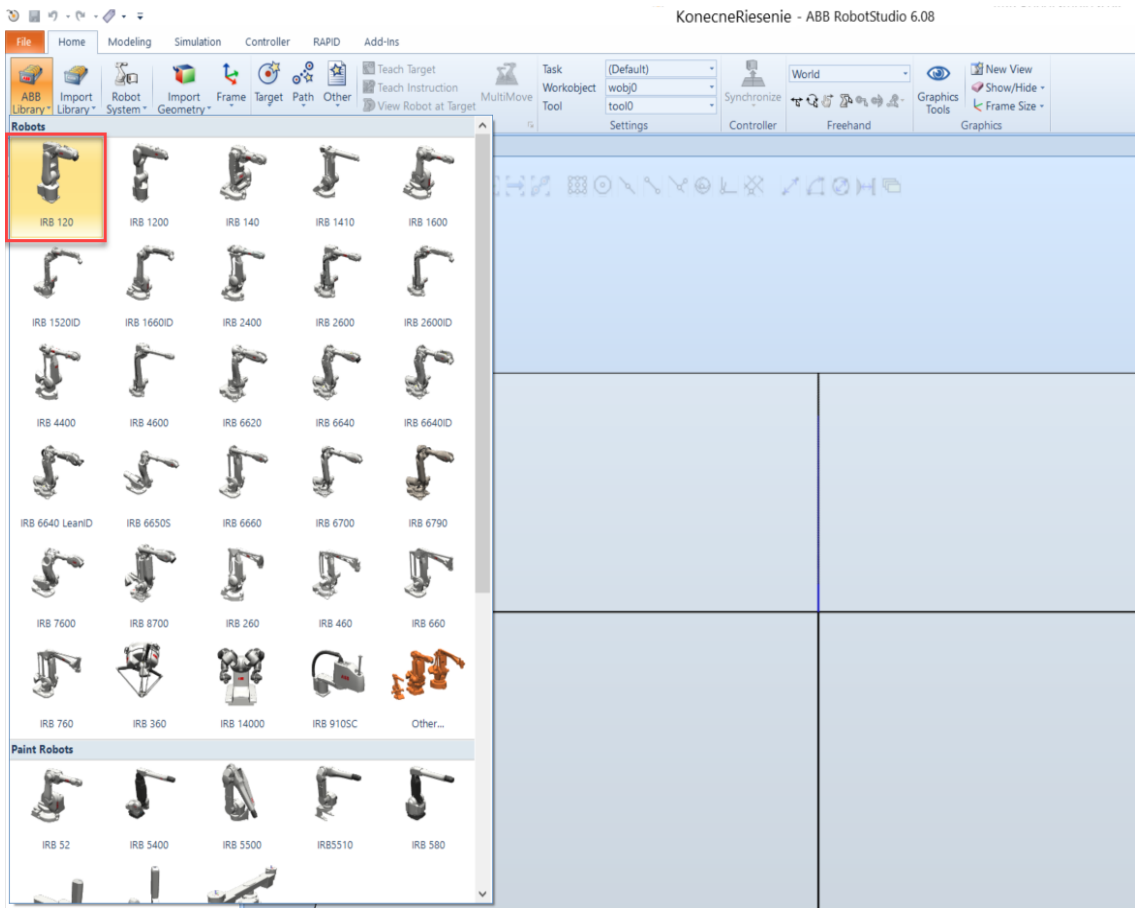
```
!Priebeh programu je v C# aplikácii nastavený na cyklický. V main procese voláme
!naše vlastné procesy. Kód a priebeh je štylizovaný ako stavový automat. Stop slúži na
!zastavenie Tasku, aby nám v tgPos neostávali predošlé hodnoty.
PROC main()

    TEST stav
    CASE KRESLIT :
      VykresliBody;
    CASE ZMENENA_RYCHLOST :
      ZmenaRychlosti(menena_rychlost);
      Stop;
    CASE CHOD_DO_TARGETU:
      ChodDoTargetu;
    CASE DOMOV :
      ChodDomov;
      Stop;
    ENDTEST

  ENDPROC
```

10.4 Postup tvorby aplikácie na strane ABB RobotStudio

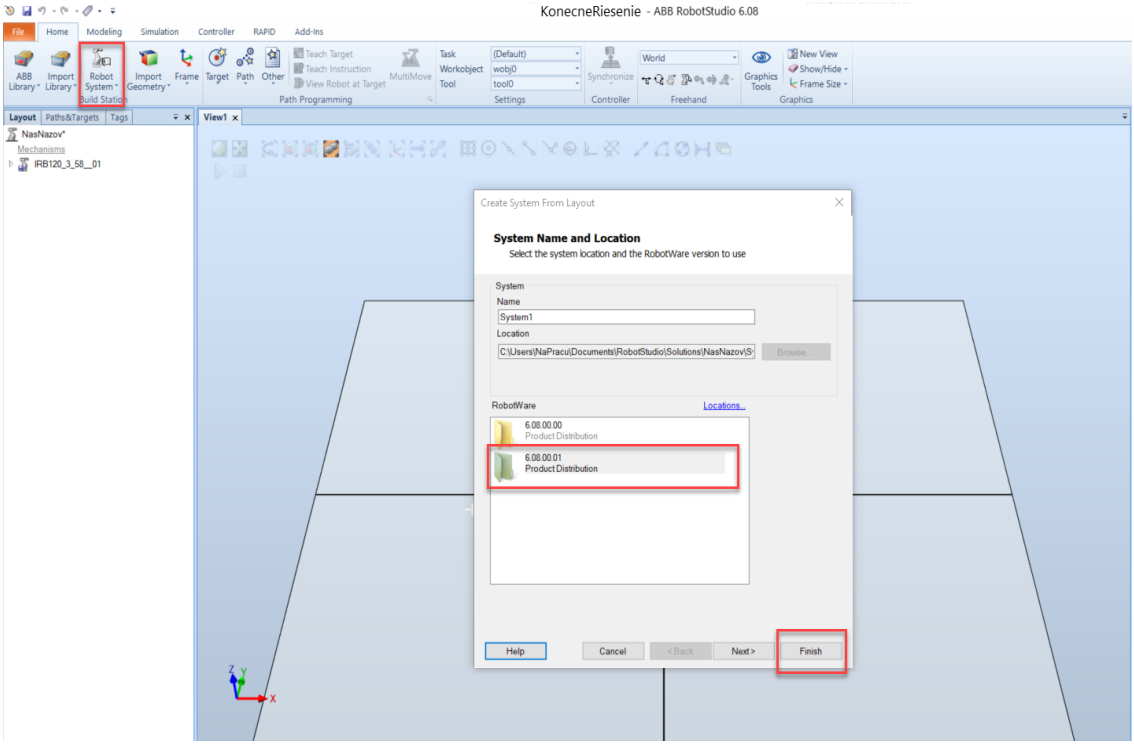
1.Spustite ABB RobotStudio.
2.Vyberte možnosť *Solution with Empty Station*
3.Zvolte cestu uloženia



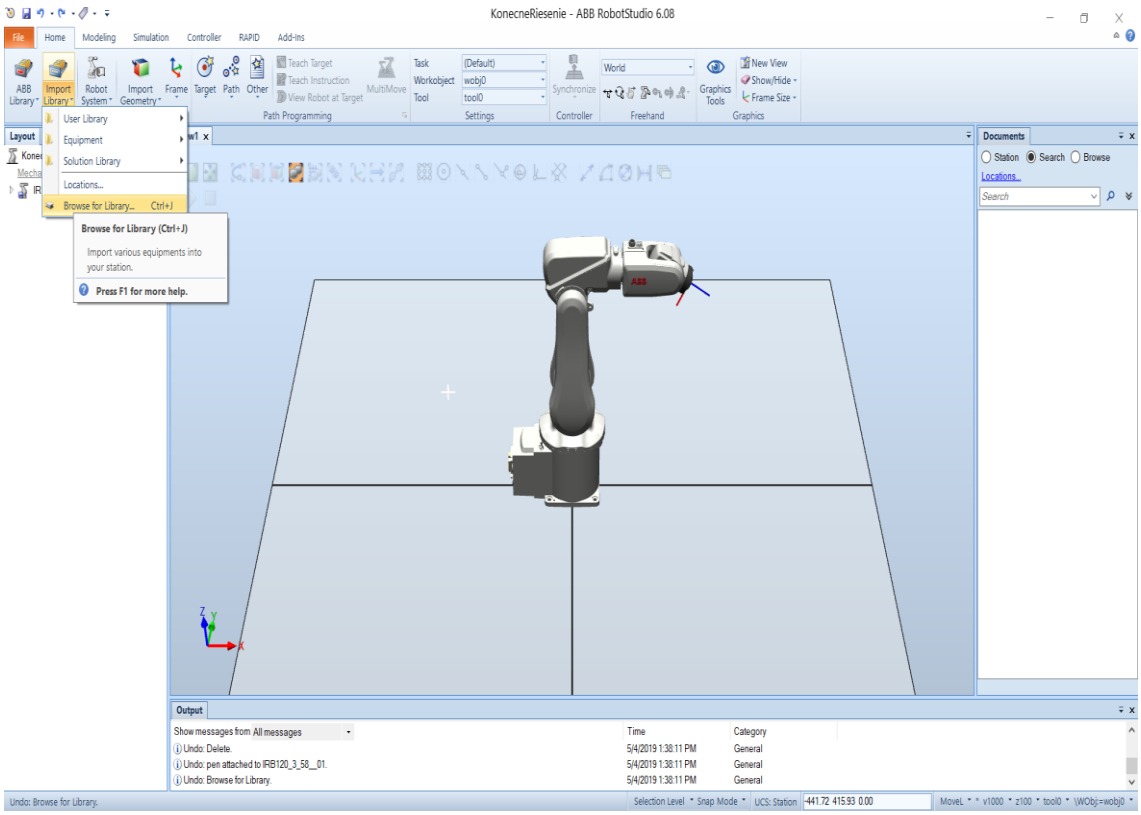
4.V záložke *Home -> ABB Library -> Zvolíme IRB 120*



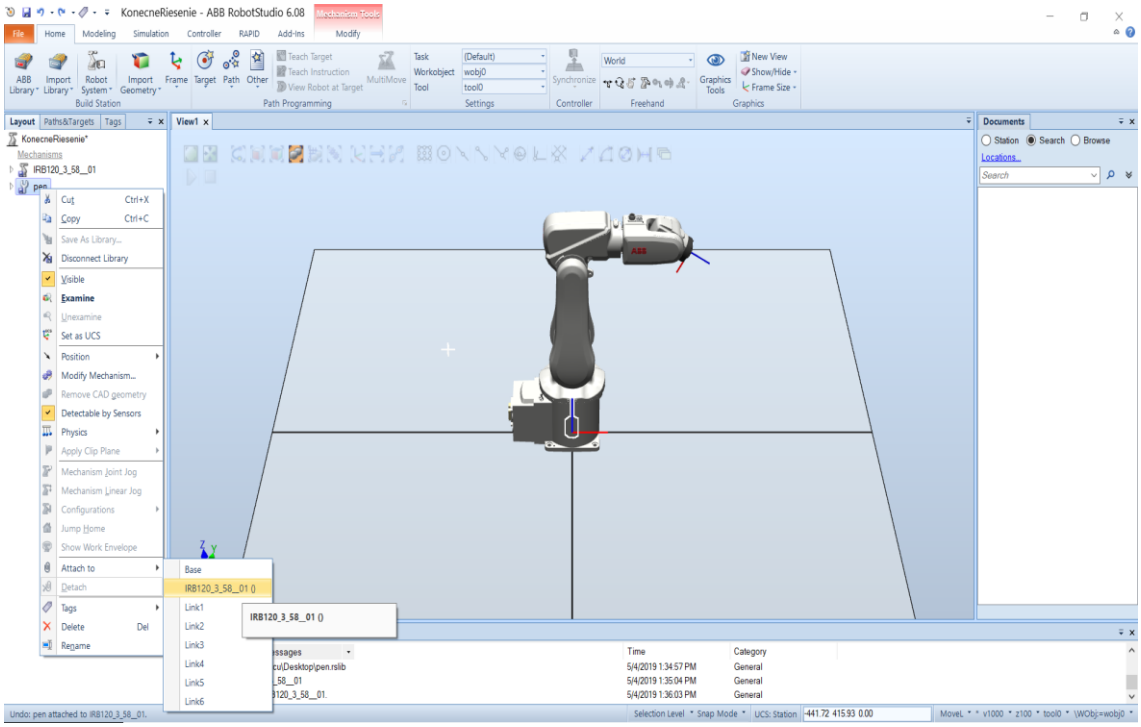
5.V záložke *Home -> Robot System -> From Layout* vytvoríme kontrolér, použijeme najnovšiu verziu RobotWare, klikneme Finish



6.Pridáme nástroj. V záložke *Home -> Import Library -> Browse for Library-> -> Pen [32]*.

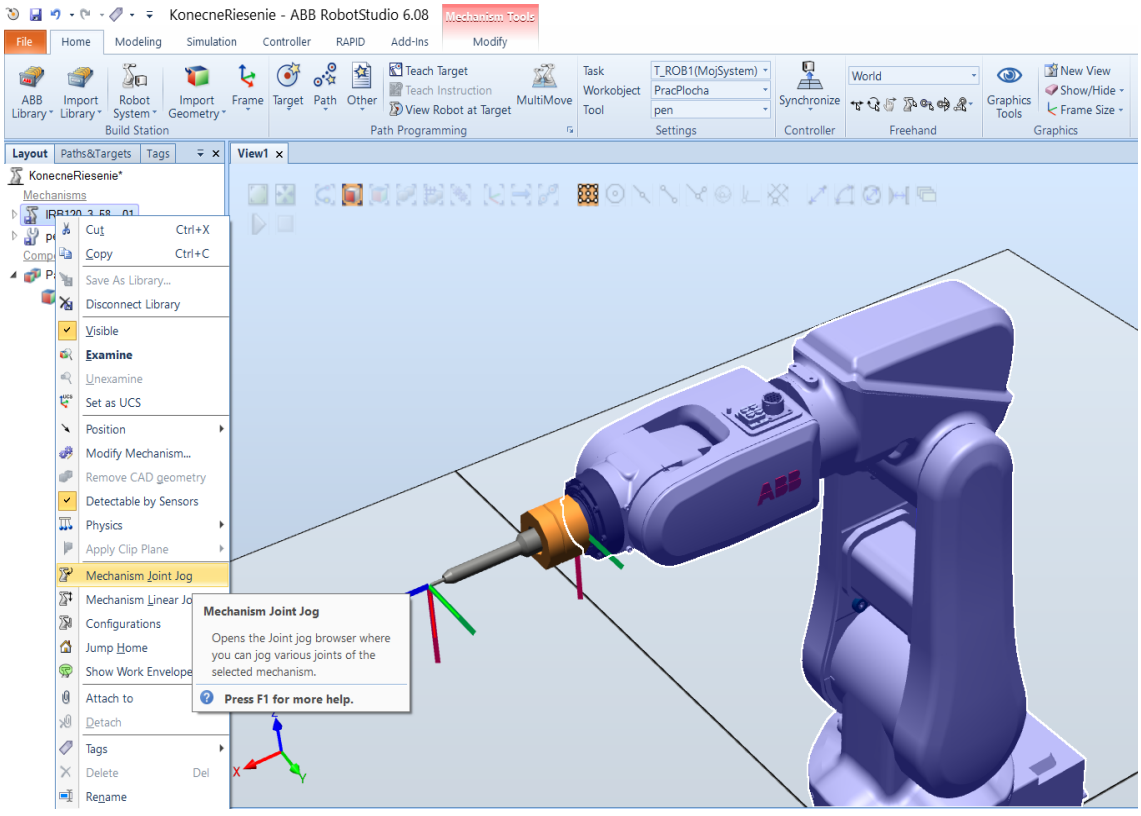


7.V záložke *Home -> Layout -> Pen*. Vyberieme a zvolíme *Attach to -> IRB 120*. Po opýtaní zvolíme *Update position -> Yes*.

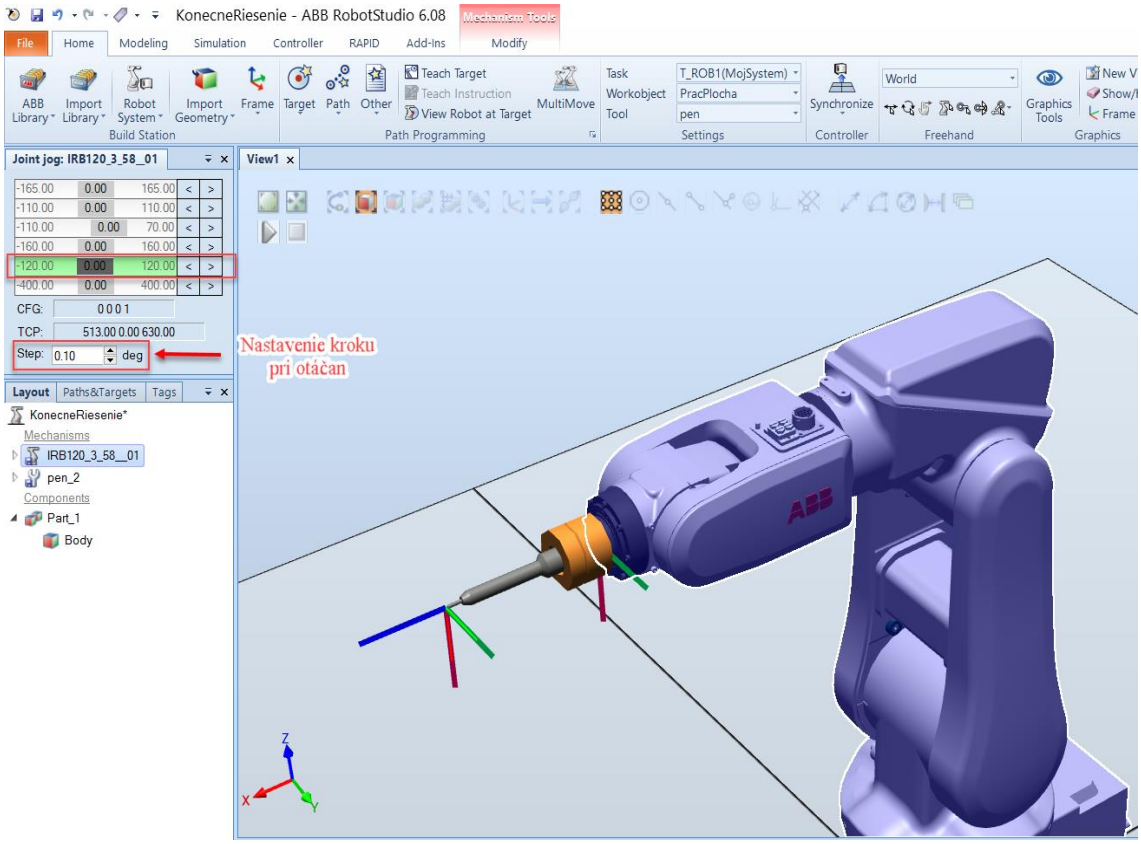


8.V záložke *Home -> Import Library -> Equipment* vyberieme *Curve Thing*

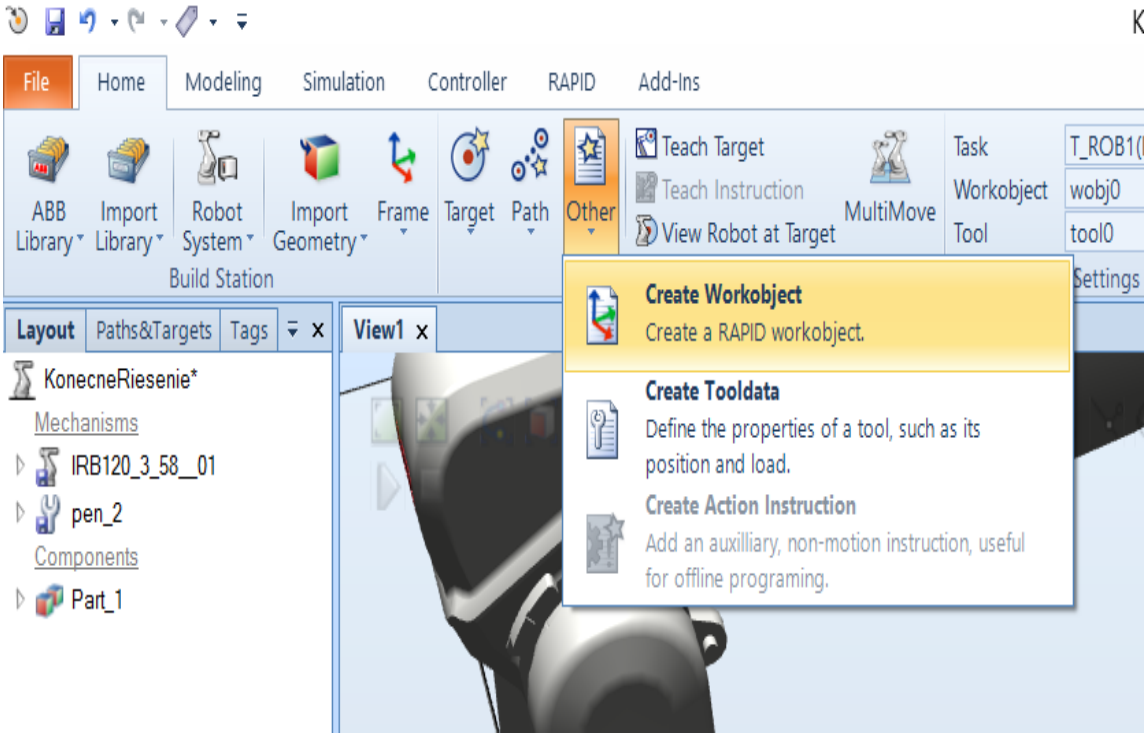
9.Pomocou nástrojov v záložke *Home -> Layout -> Mechanisms -> Mechanism Joint Jog* otočíme robota do začiatočnej polohy.



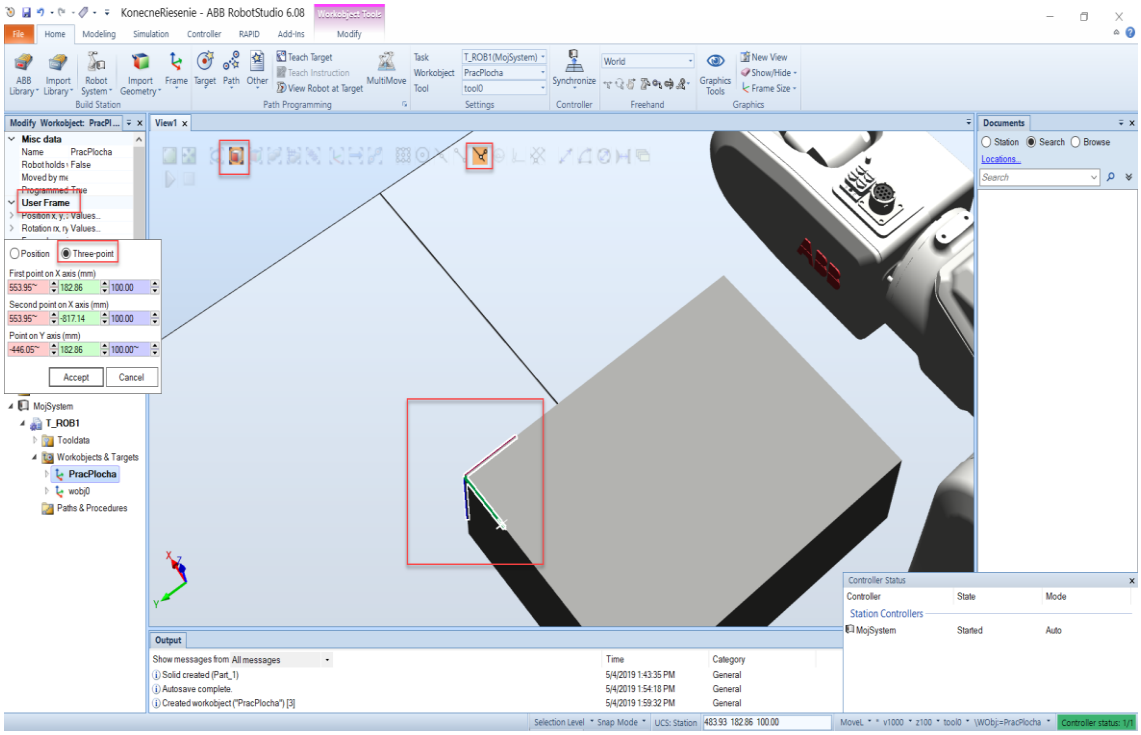
10. Otočíme robota



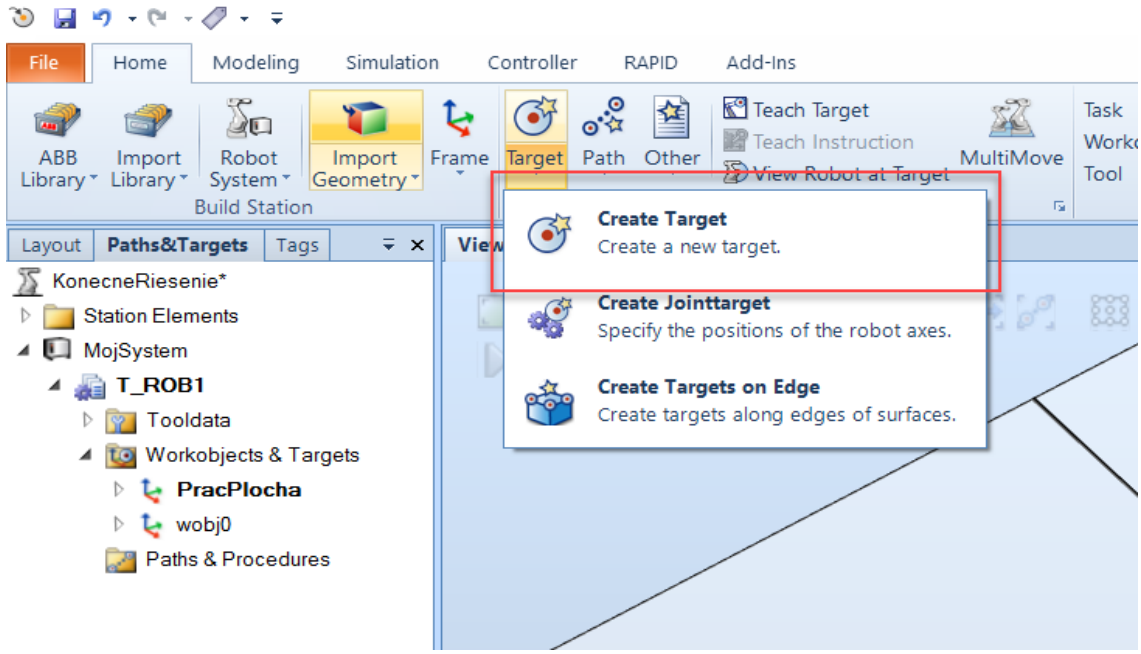
11. V záložke vyberieme *Home -> Other -> Create WorkObject*. Dáme mu názov.



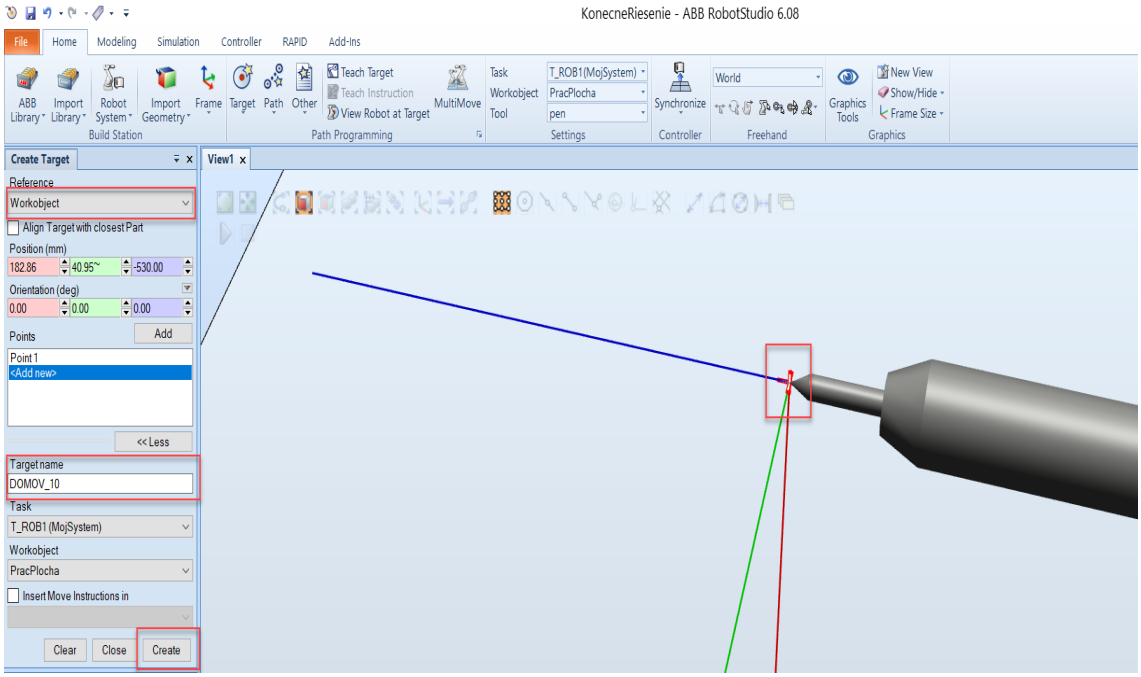
12. V záložke *Create WorkObject* zvolíme *User Frame -> Frame by points-> Three-point*. Zaklikneme *Surface selection* a *Snap Edge*. Vytvoríme v rohu *WorkObject*.



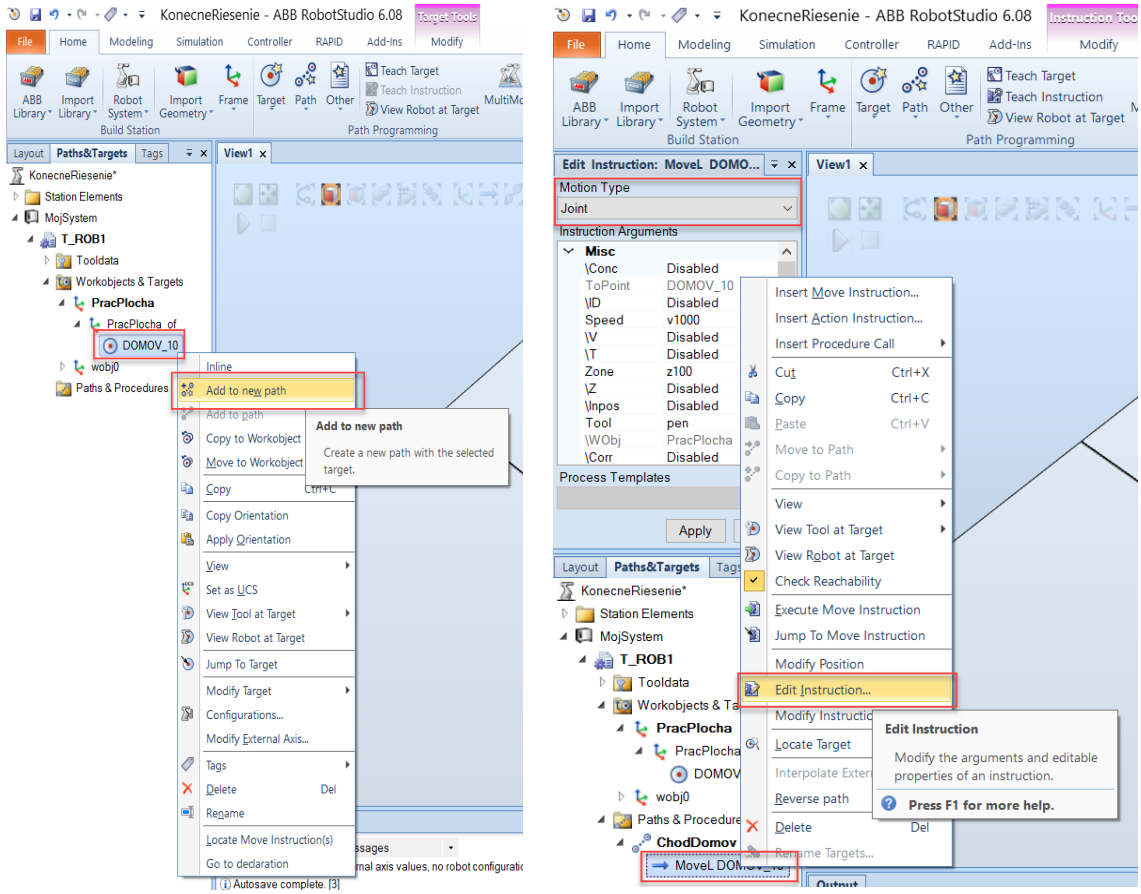
13. Vytvoríme domovský *Target* v záložke *Home -> Target -> Create Target.*



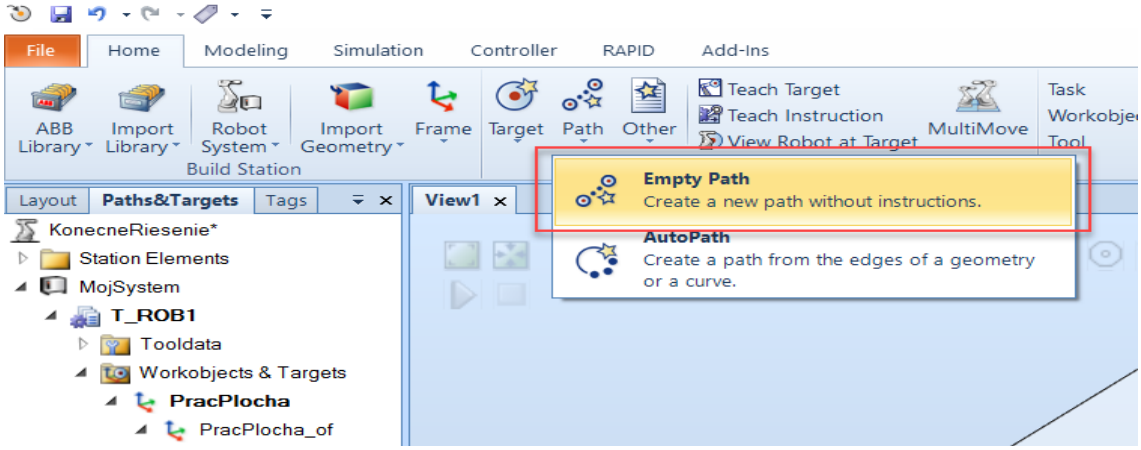
14. V karte *Create Target* zvolíme *Add new* a vyberieme koniec pera. *Target* nazveme DOMOV. *Referenciu* zvolíme na *WorkObject* a stlačíme *Create*.



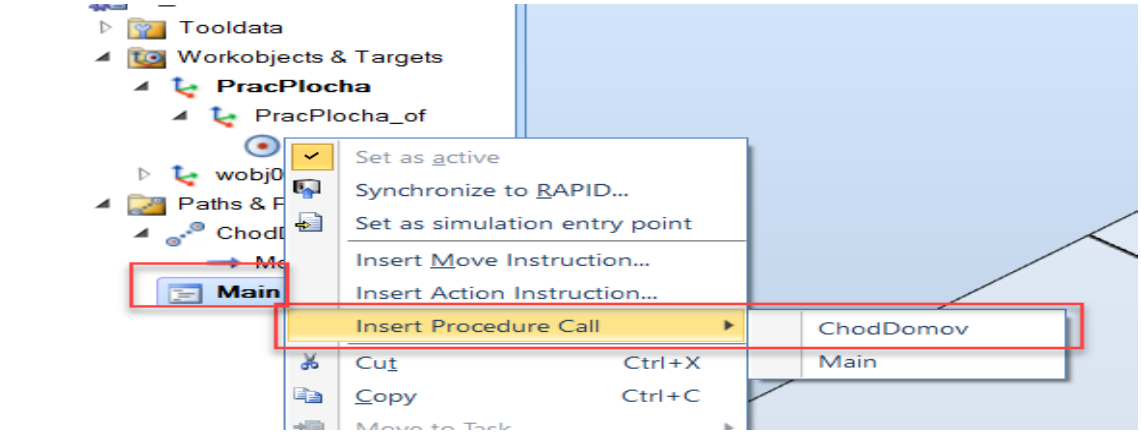
15. Vytvoríme *Path* z *Targetu*. Pravý klik na *Target -> Add to new Path*. Premenujeme náš *Path* na *ChodDomov*.
16. Zmeníme pohybový typ inštrukcie v našej ceste na *Joint*.



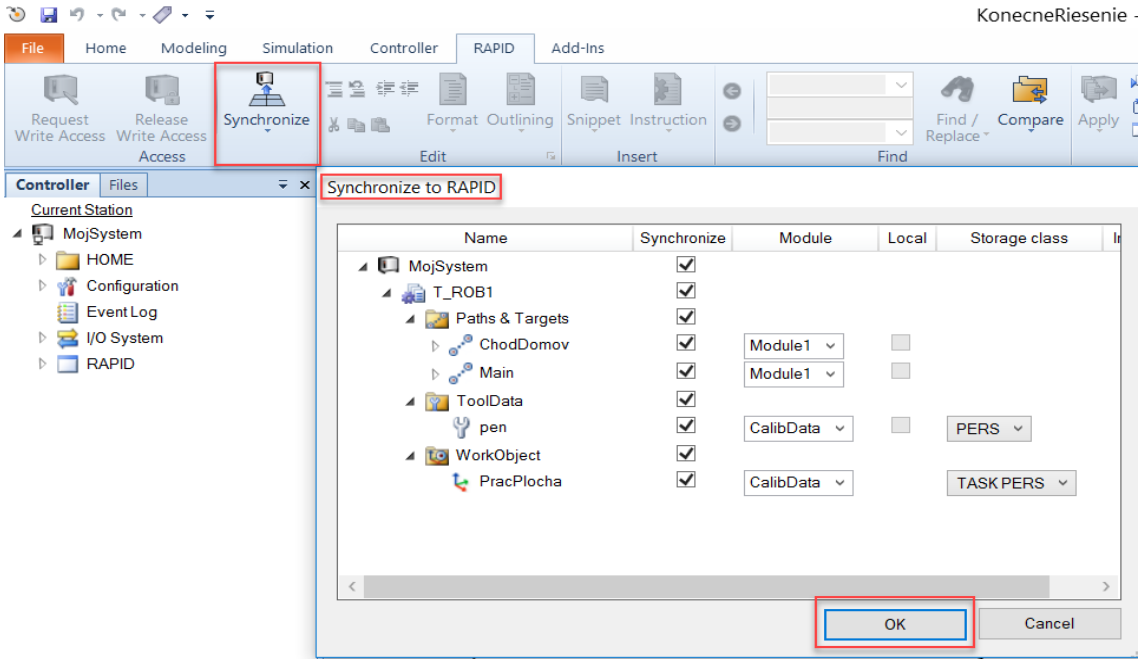
17. V záložke *Home* zvolíme *Path -> Empty Path*. Vytvorený *Path* pomenujeme *Main*



18. Pravý klik na náš *Path Main -> Insert Procedure Call -> ChodDomov.*



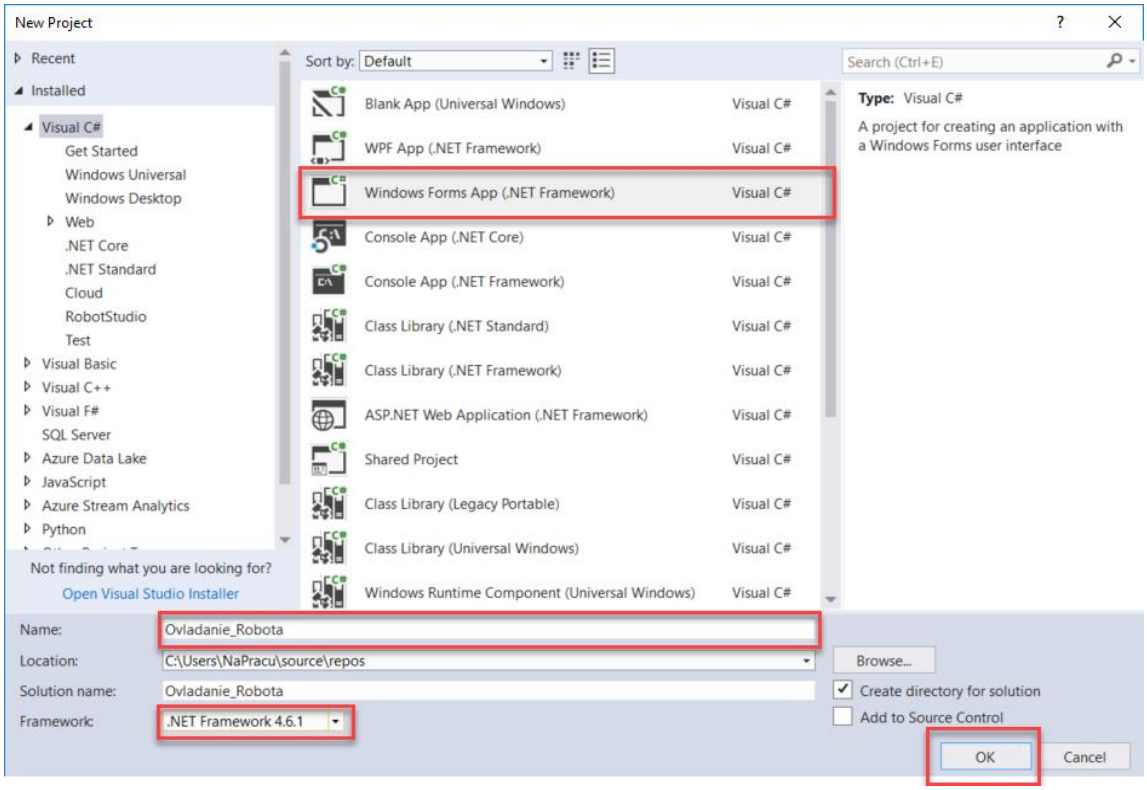
19. V záložke RAPID klikneme na *Synchronize -> Synchronize to RAPID ->Ok.*



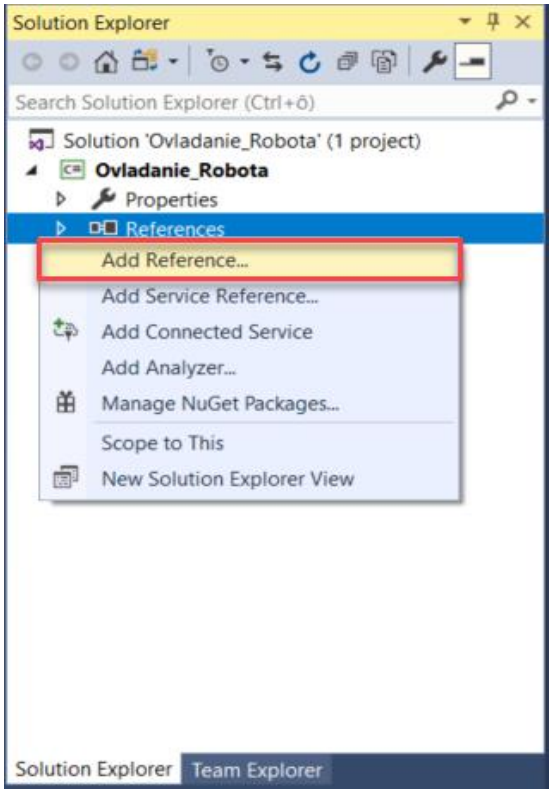
20. Napíšeme zdrojový kód (príklad uvedený v predošlej prílohe)

10.5 Postup tvorby C# aplikácie na strane Visual Studio

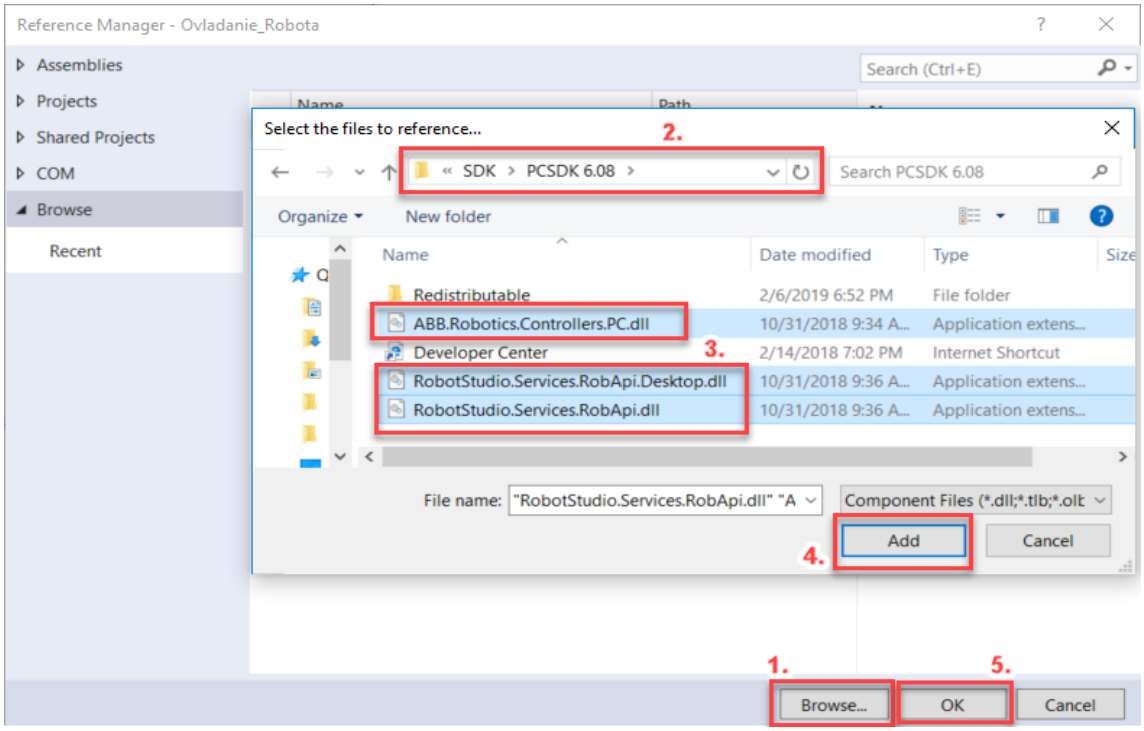
1. Spustíme *Visual Studio* a vytvoríme nový projekt typu *Windows Forms App* pomocou zakliknutia *File -> New -> Project* v hornom *Toolbare*. Skontrolujeme, či máme nastavenú najnovšiu verziu *.NET* a zvolíme názov. Stlačíme *OK*.



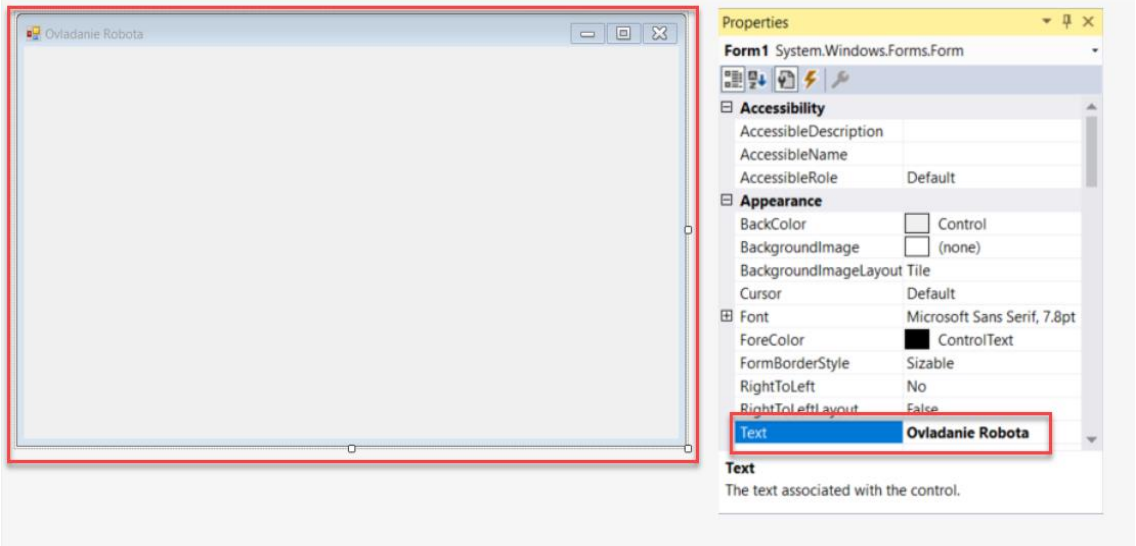
2. Pridáme potrebné knižnice do referencii pomocou *References -> Add Reference*.



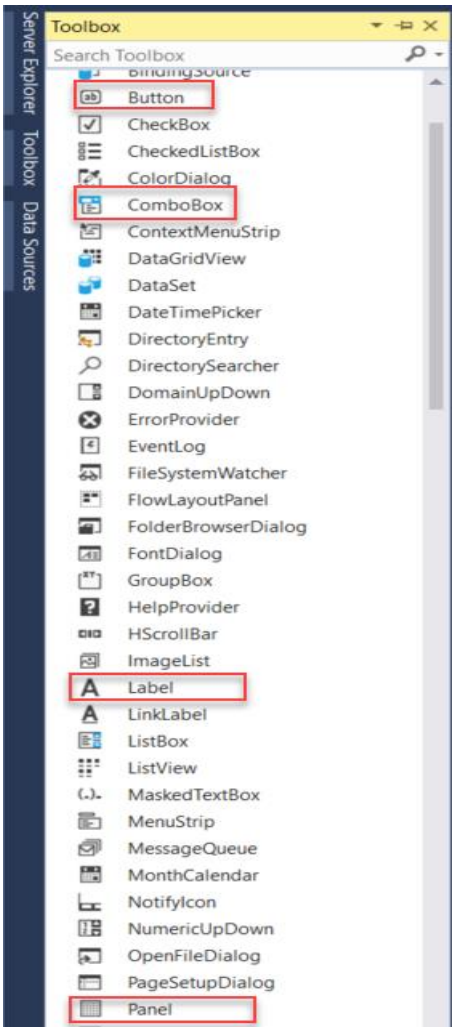
3. Pomocou tlačidla *Browse* nájdeme priečinok, kde je nainštalované rozšírenie *PCSDK*. *Označíme knižnice -> Add -> OK*.

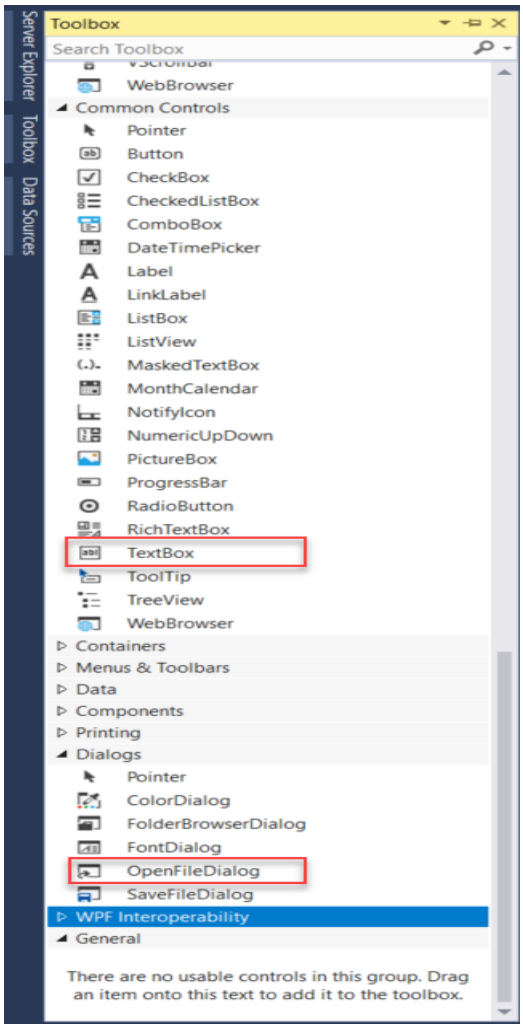


4. Zmeníme názov formy aplikácie. Dvojklik na *Form1 -> Appearance -> Text*.

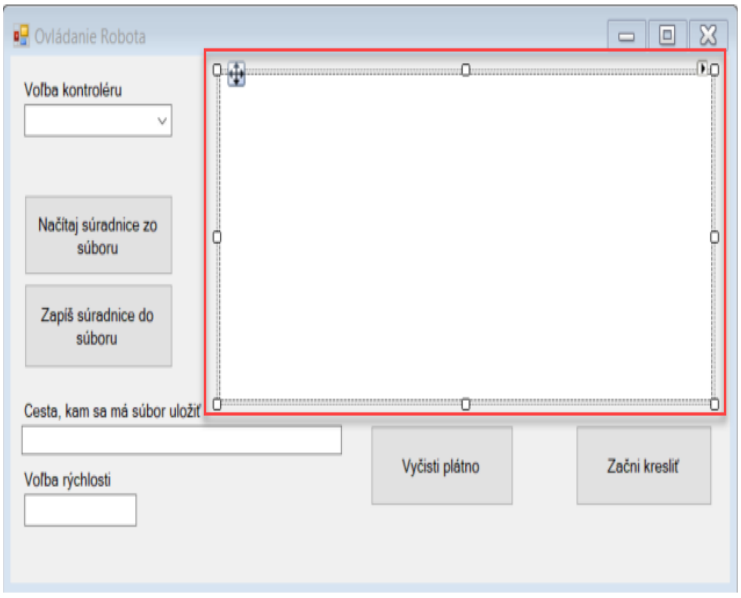


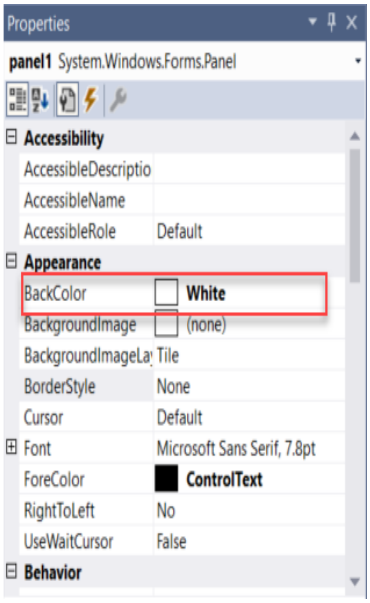
5. Otvoríme *Toolbox* a vyberieme 4x *Button*, 1x *ComboBox*, 3x *Label* a 1x *Panel* a 2x *TextBox*. Taktiež pridáme openFileDialog na výber súboru.



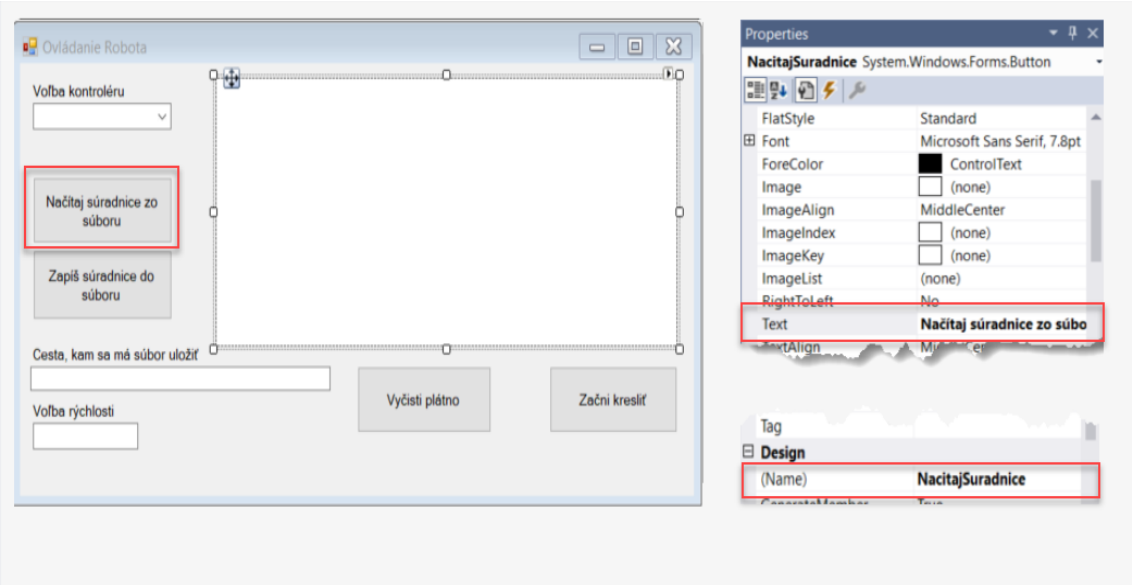


6. Zmeníme *Text* a *Name* *Buttonov*, *Labelov* a *Panelu* v *Properties*. Vykonáme pre všetky naše nástroje.





7. Zmeníme farbu pozadia plátna na bielu v *Properties*.



8.Napíšeme zdrojový kód.