Inspiromat pro výuku a Tecomat: logika (nejenom) pro programátory (část 1)

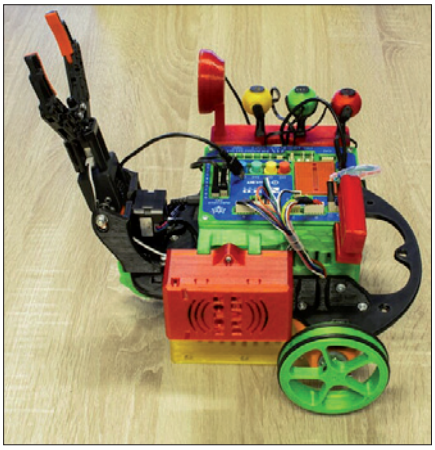
Seriál tímto dílem mění své téma a stává se kurzem aplikované logiky. Nejprve se bude zabývat kombinační logikou a později se zaměří na sekvenční logické systémy.

Úvodní poznámky

Výklad je řešen formou dvou souběžných „vláken“. První (tanec s Foxtrotem a robotem) je tvořen řešenými příklady logických úloh s programy pro PLC. Jsou použity jazyky ST a IL podle normy IEC 61131-3 a grafický jazyk CFC – všechny v prostředí vývojového systému Mosaic. Tento způsob výkladu je určen „netrpělivým“ čtenářům, kteří rádi řeší úlohy hned, bez teoretické přípravy – „učí se plavat skokem do hluboké vody“. Druhé „vlákno výkladu“ (trocha teorie nikoho nezabije) je tradičním kurzem aplikované logiky. Seznamuje se základními pojmy, s booleovskými operátory AND, OR, NOT a s dalšími logickými funkcemi, s pravidly Booleovy algebry a s metodikou algoritmizace kombinačních logických úloh. Je určen čtenářům, kteří se raději seznámí s potřebnou teorií dříve, než řeší úlohy z praxe – a těm, kteří se ocitnou v situaci: „když selžou všechny pokusy, je na čase přečíst příručku“. Metodika algoritmizace logických úloh je popsána obecně a lze ji využít k následnému programování, ale i k řešení elektroniky s pevnou logikou.

Proč se dnes zabývat logikou

Tématem této obsáhlé části seriálu je aplikovaná logika. Přívlastek „aplikovaná“ zde volíme pro odlišení od všeobecně používaného pojmu „matematická logika“, která je poměrně náročným oborem matematiky a je i jedním z nástrojů pro řešení systémů s umělou inteligencí. Učebnice matematické logiky nejsou „snadným čtením“. Naproti tomu je aplikovaná logika tradičním nástrojem pro navrhování logických systémů pro praxi. Její teorie sice vychází z matematické logiky, ale je přístupná většině řešitelů logických systémů. Bývá tradičním předmětem výuky na technicky zaměřených odborných školách – středních i vysokých. Literatura na toto téma byla nejvíce rozšířená v éře logických systémů na bázi pevné logiky. Nejprve to byly systémy využívající reléovou techniku (zhruba do 60. let 20. století), potom elektronické systémy s diskrétními součástkami (zhruba 60. až 70. léta). Konjunkturu zažívaly publikace o aplikované logice v éře integrovaných obvodů (zhruba od 70. let do konce 20. století). V éře programovatelných systémů (zhruba od 90. let do současnosti) zájem o aplikovanou logiku postupně slábne. Důvodem je skutečnost, že vývoji elektronických systémů (na bázi mikroelektronických součástek, zákaznicky programovatelných obvodů, mikrořadičů, mikropočítačů a řídicích systémů) se nyní věnuje jen nemnoho specializovaných firem a noví absolventi odborných škol nemají mnoho příležitostí, kde své znalosti uplatnit – i když v těch několika firmách by absolventy se znalostí aplikované logiky uvítali a rádi zaměstnali.



Obr. 1. Mobilní výukový robot Foxee se svou řídicí kostkou

V současné době se většina úloh logického typu řeší programem (počítačů, mikrořadičů a programovatelných automatů). Výuka se proto zaměřuje především na programování. Náplň výuky je ale ovlivněna všeobecně rozšířeným omylem, že k výuce programování stačí zvládnout některý z programovacích jazyků a že znalost logiky již není potřebná. To je i důvodem, proč učebnice aplikované logiky téměř vymizely z našeho knižního trhu. Naštěstí se výuka logiky zachovala na většině našich odborných škol. Kapitoly o logice a o navrhování logických systémů jsou i nadále součástí současných učebnic automatizace [10], [11], vydaných péčí Českomoravské společnosti pro automatizaci. Metodika je zde ale tradičně orientována na navrhování logických systémů s pevnou logikou, převážně s použitím integrovaných obvodů malé a střední hustoty integrace.

Ve výuce v oboru informatiky je aplikovaná logika (ve většině případů) naprosto ignorována – a to navzdory skutečnosti, že mnoho úloh řešených programovatelnými systémy je logického typu, ačkoliv to na první pohled není vždy zřejmé. Ke zvládnutí profese programátora nestačí naučit se některý ze zvolených jazyků, jeho základní příkazy a potřebnou syntaxi. Důležitější je znalost tvorby algoritmů. Ty rozhodují o kvalitě a úspěšnosti programového řešení. Právě algoritmizace je podstatou tvořivé programátorské práce. Samotný programovací jazyk je jen nástrojem, kterým lze vytvořený algoritmus přepsat (zakódovat) do „řeči programovatelného systému“, který je „nosičem programu“ a program vykonává. Je to v podstatě rutinní práce, která nevyžaduje zvláštní kvalifikaci. Skutečným programátorem je ten, který umí tvořit efektivní algoritmy pro řešení zadaných úloh.

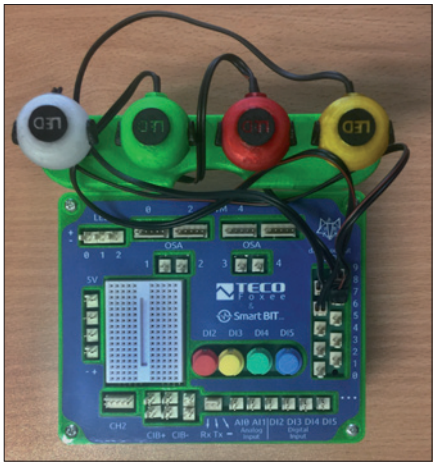
Podobně to platí při výuce cizího jazyka. Nestačí znalost gramatiky a základní slovní zásoby. Je nutné seznámit se s reáliemi prostředí, kde se tím jazykem mluví, rozumět mluvenému, umět bezchybně hovořit a naučit se v jazyce myslet.

Analýza jako prevence problémů

Pro programování je nejdůležitější fáze analýzy problému, provedená před začátkem programování. V ní se rozhoduje o koncepci řešení, o použití vhodných algoritmů a o vhodném postupu práce na programu. Řeší se rozdělení úlohy (dekompozice) na dílčí problémy, komunikační rozhraní mezi podúlohami, jejich rozdělení mezi členy řešitelského týmu a způsob komunikace mezi nimi. Důležité je předem stanovit postup ladění a metodiku ověřování správnosti výsledného řešení. Význam má i provedení a rozvržení úkolů na vytvoření dokumentace. Ta je povinnou součástí každého produktu, tedy i vytvořeného programu. Její kvalitní zpracování je důležité i pro jeho dodavatele. Přehledně a komplexně zpracovaná příručka uživatele seznamuje se všemi funkcemi programu a dovoluje jejich efektivní využívání – přispívá ke spokojenosti uživatele a ke kladnému hodnocení produktu. Současně omezuje riziko neoprávněných reklamací a stížností zaviněných neznalostí uživatele a jeho chybnými představami o chování programu. Důležitá je i kompletní a detailně provedená dokumentace programu – popis použitých algoritmů i řešení samotného programu a jeho průběžné komentování. Má význam pro efektivní provádění budoucích změn programu, které je nutné vždy předpokládat. Popis poslouží dosavadním tvůrcům programu (již po krátké době zapomenou detaily řešení, zejména „chytré finty“), ale především jiným programátorům (třeba i z jiné firmy), kteří budou dodatečné změny provádět.

Flexibilita a přehlednost programu

Důležitou vlastností programu je jeho flexibilita a přehlednost. Vždy je nutné počítat s tím, že se program bude měnit. Bude třeba průběžně odstraňovat chyby programátora vzniklé při řešení programu i chyby způsobené nepřesným zadáním nebo nevyjasněnými požadavky zadavatele. Představy zadavatele a uživatele se postupně vyvíjejí, a vznikají tak nové požadavky na změny v chování a na nové funkce programu. Měnit se bude i „společenské prostředí“, v němž bude systém provozován, vyvíjet se budou i „vzory“ a konkurenční systémy, s nimiž bude vytvořený systém srovnáván. Bude se vyvíjet i technika a s nimi rovněž požadavky na systém, např. na komunikaci, na operátorské rozhraní a způsob obsluhy a na nové funkce. Při volbě koncepce systému se rozhoduje i o řešitelnosti funkcí, které v době jeho zadání nebyly známy. Je třeba respektovat základní pravidlo, že „programový systém, který se průběžně neudržuje a nerozvíjí, postupně umírá“.



Obr. 2. Kostka Foxee s barevnými signálkami připravená pro řešení příkladů

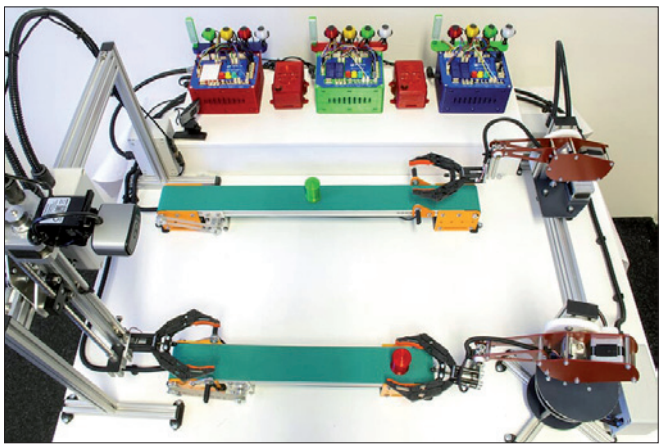
Finanční rozměr analýzy

Kvalita řešení, spolehlivost programu a doba jeho řešení ovlivňují nejenom výslednou cenu, ale i pověst a pozici dodavatele na trhu – a tím i šanci na další zakázky. To vše lze vyčíslit v penězích. Přesto mnohdy bývá fáze analýzy opomíjena, nebo dokonce vynechávána. Při analýze se nic viditelného neděje – členové týmu spolu živě diskutují (spolu i se zadavateli), studují, mlčky hledí z okna nebo do zdi. Tvořivou atmosféru lze obtížně měřit a hodnotit. Netrpělivý šéf by raději viděl horečnou aktivitu, prsty hbitě běhající po klávesnici, obrazovky plnící se řádky s příkazy programu nebo textem. Trpělivost se ale vyplatí. Po kvalitní analýze spěje řešení k závěru rychleji a přímočaře, bez zbytečné improvizace, omylů a následných reklamací.

Je pravděpodobné, že programátor, který bezprostředně po zadání problému usedne ke klávesnici, začne plnit obrazovku a paměť počítače příkazy programu, bude mít později problémy, zejména v závěru práce, při ladění, předávání programu uživateli i při rutinním provozování programu – možná se jeho dílo stane „nekonečným příběhem“ a „noční můrou“. Není to ale pravidlem. Zkušený programátor zvládne vytvořit nepříliš komplikovaný program téměř bez přípravy – obzvláště když se zadaná úloha podobá jeho dosud vyřešeným problémům. Situaci usnadní, jestliže může použít „konfekční řešení“, tedy standardizovaný postup nebo osvědčený vzor řešení („prefabrikát programu“), který jen podle potřeby upraví, popř. jen parametrizuje. Postup usnadní možnost použít knihovnu ověřených a osvědčených funkčních bloků. Pak se programování promění v sestavování schématu. Tím ovšem není popřen význam fáze analýzy problému – jen je zde využívána dříve provedená analýza, jejíž výsledky jsou zobecněny pro třídu podobných úloh. Podobný postup často využívají dodavatelské firmy, které řeší problematiku jednoho oboru, např. řízení technického vybavení budov a jejich energetiky, řízení pracovních strojů a jejich pomocných mechanismů, řízení chemických a potravinářských technologií apod.

Pro koho je seriál určen

Protože významnou část problémů řešených programovatelnými systémy (troufáme si tvrdit, že většinu) tvoří úlohy logického typu, je účelné zvládnout metodiku jejich efektivní algoritmizace. To je cílem předkládané části seriálu. Je určen především pro výuku na středních odborných školách zaměřených na obory automatizace, programování a informačních technologií (IT). Věříme, že bude přínosný i pro studenty technických univerzit, ale i pro výuku dospělých – pro revitalizaci („upgrade“) jejich kvalifikace nebo pro potřeby rekvalifikace. Znalost logiky je potřebná i pro odborníky z jiných profesí, především pro konstruktéry, technology, energetiky a projektanty. Je ale užitečná i pro zadavatele a investory automatizovaných systémů. Znalost logiky jim poskytne nadhled, získají představu o možnostech programu, o složitosti a ceně řešení. Důležitá je především pro komunikaci zadavatelů s tvůrci programu, s analytiky a programátory. Přesné zadání úlohy je podmínkou jejího kvalitního a bezchybného řešení. Usnadní průběh a zkrátí dobu řešení a omezí výskyt chyb a reklamací zaviněných nepřesným zadáním.



Obr. 3. Skupina řídicích kostek Foxee pro distribuované řízení složitějšího mechanismu

Tanec s Foxtrotem a robotem

Výklad v této části je veden formou řešených příkladů a komentářům k nim. Předpokládá dostupnost kompaktního řídicího systému, pracovně pojmenovaného jako kostka Foxee (*obr. 1*). V něm je zabudován programovatelný automat Tecomat Foxtrot ve své vestavné verzi CP 1972 (*obr. 2*). Programuje se tedy ve vývojovém systému Mosaic, stejně jako ostatní systémy Tecomat. Při nedostupnosti kostky lze využít jinou učební pomůcku se systémem Tecomat, např. některou sestavu EDUtec, výukový kufřík nebo jakoukoliv jinou laboratorní sestavu používající systém Tecomat (nejspíše Tecomat Foxtrot). Při nedostupnosti fyzického PLC je možné úlohy programovat ve virtuálním PLC v prostředí Mosaic.

Doporučená literatura

Cílem této části seriálu je výuka aplikované logiky a její využití při programování. Nezabývá se ale výukou samotného programování a popisem programovacích jazyků. Příklady jsou uváděny v jazycích ST (*Structured Text*, strukturovaný text) a LD (*Ladder Diagram*, příčkový diagram, jazyk kontaktních schémat), popsaných mezinárodní normou IEC EN 61131-3 (dále jen „normou“). Používán je rovněž grafický jazyk CFC (*Continuous Function Chart*, grafický jazyk, který je „příjemnější“ obdobou jazyka blokových schémat FBD). Je oblíben a v praxi často používán. Rovněž využívá aparát této normy. Umožňuje snadnější programování (které je spíše „kreslením schémat“) a přehlednějších programů. Tyto tři jazyky se v praxi používají nejčastěji a současně jsou vhodné i pro výuku. Ze stejného důvodu se zde nepoužívají zbývající jazyky IL (*Instruction List*, jazyk souboru instrukcí) a FBD (*Function Block Diagram*, jazyk funkčních bloků). Popis používaných jazyků ani zásad normy tu není uváděn a je očekává-

na jejich (alespoň minimální) znalost. Předpokládáme, že programy řešených příkladů jsou natolik názorné a srozumitelné, že z nich lze pochopit podstatu vykládané problematiky. K samostatnému řešení obdobných úloh stačí postup „nápodoby“. Pro zájemce o hlubší seznámení s programováním PLC podle normy IEC EN 61131-3 uvádíme seznam doporučené literatury – z praktických důvodů tak činíme hned na začátku seriálu.

Literatura:

[1] ŠMEJKAL, Ladislav a Josef ČERNÝ. *Esperanto programátorů PLC: programování podle normy IEC/EN 61131-3: speciální vydání časopisu Automa* [online]. Děčín: Automa – časopis pro automatizační techniku, 2017 [cit. 2018-09-06]. Dostupné z: http://tecoacademy.cz/wp-content/uploads/2017/04/Esperanto-final.pdf

[2] ČSN EN 61131-3 ED. 2 (187050). *Programovatelné řídicí jednotky – Část 3: Programovací jazyky*. Vydání druhé. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2013.

[3] KOHOUT, Luděk. *Norma pro řídicí systémy IEC 61131* [online]. neuvedeno: neuvedeno, 2007 [cit. 2018-09-06]. Dostupné z: http://www.edumat.cz/texty/zaciname_IEC61131.pdf

[4] –. *Programování PLC podle normy IEC 61131-3 v prostředí Mosaic* [online]. Jedenácté vydání. neuvedeno: neuvedeno, 2009 [cit. 2018-09-06]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00321_01_mosaic_progiec_cz

[5] –. *Začínáme v prostředí Mosaic* [online]. Vydání sedmé. Kolín: TECO, 2008 [cit. 2018-09-06]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00320_01_mosaic_progstart_cz

[6] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. *PLC a automatizace*. Praha: BEN – technická literatura, 1999. ISBN 80-860-5658-9.

[7] ŠMEJKAL, Ladislav. *PLC a automatizace*. Praha: BEN – technická literatura, 2005. ISBN 80-730-0087-3.

[8] MARTINÁSKOVÁ, Marie a Ladislav ŠMEJKAL. *Řízení programovatelnými automaty*. Praha: České vysoké učení technické, 1998, 2000, 2003. ISBN 80-010-1766-4.

[9] –. *Knihovny pro programování PLC Tecomat podle IEC 61131-3* [online]. Osmé vydání. Kolín: TECO, 2006 [cit. 2018-09-06]. Dostupné z: https://docplayer.cz/13318771--Knihovny-pro-programovani-plc-tecomat--podle-iec-61-131-3.html

[10] Kol. *Automatizace a automatizační technika: systémové pojetí automatizace – Automatické řízení*. Praha: Computer Press, 2000. ISBN 80-7226-247-5.

[11] Kol. *Automatizace a automatizační technika: systémové pojetí automatizace*. Brno: Computer Press, 2012. ISBN 978-80-251-3628-7.

[12] Neuvedeno. *Automatizace pro SŠ* [online]. neuvedeno: neuvedeno, 2018. Dostupné z: neuvedeno. Licence na reditel@betlemska.cz.

[13] ČERNÝ, Jiří. *Foxee v 1.0 – základní dokumentace*. Kolín – Hradec Králové: Teco – SmartBIT, 2018.

Ing. Ladislav Šmejkal, CSc., Teco, a. s. a externí redaktor Automa, Ing. Josef Kovář a Ing. Zuzana Prokopová, učitelé automatizace na SPŠ Zlín

TECO Advanced Automation
Tecomat Foxtrot
Platforma pro automatizaci a komunikaci strojů, procesů, budov a dopravy
TECO Academy
www.tecomat.cz IEC-61131 | IoT | Smart House | Smart City | Industry 4.0 www.tecoacademy.cz

RICAIP Testbed Day završil přípravnou fázi projektu

Na konci srpna 2018 se u příležitosti završení první přípravné fáze projektu RICAIP konala v prostorách Testbedu pro Průmysl 4.0 v budově CIIRC ČVUT slavnostní akce RICAIP Testbed Day.

Cílem bylo více než stovce hostů z řad českých i zahraničních firem i veřejné správy představit principy, na kterých vznikne RICAIP – Výzkumné a inovační centrum pro pokročilou průmyslovou výrobu (*Research and Innovation Centre on Advanced Industrial Production*). RICAIP společně realizují čtyři akademičtí partneři – dva české výzkumné instituty, tedy CIIRC ČVUT a CEITEC VUT, a dvě přední německé výzkumné organizace, DFKI (Německé výzkumné centrum pro umělou inteligenci) a ZeMA (Centrum pro mechatroniku a automatizaci).

RICAIP virtuálně propojí testbedy v České republice a v Německu a položí tak základy česko-německého výzkumného centra v oblasti pokročilé distribuované výroby. V uplynulých dvanácti měsících pracovali partneři projektu na převedení této vize do konkrétních kroků a dokumentů tak, aby mohli během tohoto podzimu předložit žádost o podporu na realizaci centra RICAIP a získat prostředky z evropských i národních zdrojů v rámci výzvy Horizon 2020 Teaming II. Záměrem je rozvinout RICAIP do evropské výzkumné infrastruktury, jedné z prvních svého druhu v Evropě.

Rok odborníci pracovali na převedení této vize do konkrétních kroků a dokumentů tak, aby bylo možné na podzim předložit žádost o podporu na realizaci RICAIP a získat prostředky z evropských i národních zdrojů. RICAIP ale není jen myšlenka, RICAIP poskytuje konkrétní řešení pro průmysl. O tom se mohli účastníci akce přesvědčit při prezentaci prvního výzkumného demonstrátoru. Seznámili se s novým způsobem organizace výroby spočívající v digitalizaci procesů a propojení výrobních zařízení v několika lokalitách. Na těchto principech jsou založeny budoucí nové automatizované distribuované výrobní sítě, které urychlí zavádění nových výrobních procesů, ušetří náklady na výrobu a usnadní vstup společnostem na trh a do hodnotových řetězců.

Distribuovaná výroba není mýtus – RICAIP představuje řadu postupných kroků vedoucích ke zcela novému organizování výroby s digitalizací procesů a propojením výrobních zařízení ve více lokalitách.

Více informací o projektu RICAIP je možné nalézt na: www.ricaip.eu.

Radim Adam