Multidiszciplináris tudományok, 10. kötet. (2020) 4 sz. pp. 444-451 **https://doi.org/10.35925/j.multi.2020.4.50**

Ipari tesztberendezések PLC programozását támogató informatikai alkalmazás fejlesztése

Kovács Alex Pál
műszaki szolgáltató, Miskolci Egyetem, Felsőoktatási és Ipari Együttműködési Központ
3515 Miskolc, Miskolc-Egyetemváros, e-mail: iitkalex@uni-miskolc.hu

Szabó Martin
egyetemi tanársegéd, Miskolci Egyetem, Informatika Intézet, Alkalmazott Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: iitszabo@uni-miskolc.hu

Absztrakt

A megnövekedő piaci igények, minőségi előírások és az egyre szigorúbb autóipari szabványok miatt az alkatrészgyártók, termékeik tesztelésére egyre nagyobb hangsúlyt fektetnek, és szükségszerűen egyre több tesztberendezést használnak. Az új PLC vezérelt tesztgépek programozásához egyedi tudás szükséges, amelyet legtöbb esetben új munkaerő felvételével vagy a cégnél elhelyezkedő munkavállalók képzésével elégítenek ki. A cikkben bemutatott alkalmazás célja a Starters E-Components Generators Automotive Hungary Kft. tesztelési folyamatinak szoftveres támogatása, ahol az elkészített szoftver, a PLC eszközök sajátosságaitól függetlenül, azok programozási módszerét általánosítva, valamennyi tesztberendezés programozásához közös platformot kínál. Az eddig használt idő alapú tesztelési folyamatok mellett, az alkalmazás támogatja az esemény alapú folyamatok létrehozását is. A cikk Kovács Alex Pál „Esemény vezérelt tesztelési folyamat informatikai támogatásának megvalósítása" című szakdolgozatának a kivonata.

***Kulcsszavak**: ipari tesztelési folyamat, programozható logikai vezérlő, PLC*

Abstract

Due to growing market demands, quality standards, and more stringent automotive standards, component manufacturers are placing increasing emphasis on extensive testing of their products and therefore use an increasing variety of test equipment. Programming new PLC-controlled test machines requires specialized knowledge, which is most often a serious challenge for test engineers. In this article we have presented a software application supporting the existing product testing processes of Starters E-Components Generators Automotive Hungary Kft. In addition to the time-based testing processes used so far, the new software application also supports the programming of event-based processes. The article is an excerpt from Alex Pal Kovacs's BSc diploma work entitled "Esemény vezérelt tesztelési folyamat informatikai támogatásának megvalósítása".

***Keywords**: product testing process, programmable logical controller, PLC*

1. Bevezetés

A cikkben bemutatott célszoftvert a Starter E-Components Generators Automotive Kft. termékfejlesztési- és tesztelési folyamatainak támogatására készítettük el. A vállalat számos saját terméket gyárt, amelyeket a sorozatgyártását előtt szabványos tesztnek kell alávetni annak érdekében, hogy megbizonyosodhassanak a gyártmány magas minőségéről. A tesztelés során a termékeket

különböző mesterséges terheléseknek teszik ki, amelyeket PLC berendezésekkel, speciális kamrákban végeznek el. A széles termékskála miatt több különböző tesztgépekre van szükség ahhoz, hogy a termékek összes releváns, szabványokban is előírt tulajdonságát megvizsgálhassák. A tesztkamrákban vagy beépített, vagy saját fejlesztésű PLC vezérléseket használnak. A termékek tesztelési módszereit nemzetközi szabványok írják elő, valamint a megrendelők egyedi próbákat is kérhetnek a magas minőség biztosítása miatt. A tesztlabor folyamatos fejlesztés alatt áll, folyamatosa új gépek kerülnek üzembe, a meglévők vezérlő programjait is folyamatosan fejlesztik. Az újabb és újabb eszközök eltérő programozási nyelvvel, paraméterezési lehetőségekkel rendelkezhetnek, így az egyes tesztek programozásához a fejlesztőmérnököknek szüksége van az adott gép nyelvének és lehetőségeinek mély ismeretére.

Az általunk fejlesztett alkalmazás képes ezen sajátosságok elfedésére, a programozási módszer egységesítésére, oly módon, hogy a felhasználónak ne legyen szüksége az adott eszköz mélyebb ismeretére. További követelmény, hogy a szoftver legyen egyszerűen kezelhető, átlátható, így csökkentve a betanulással járó közvetlen és közvetett kockázatokat és költségeket. Futtathatónak kell lennie mind asztali számítógépen, mind Android operációs rendszert futtató mobil platformon (jelen esetben ipari tableten) úgy, hogy a két elkészített alkalmazás közötti különbözőségek ne befolyásolják a felhasználót a tesztek elkészítésének hatékonyságában. Utóbbi platformra a hordozhatósága miatt van szükség, mivel az internetes kapcsolat kiépítése nem megoldott a számítógépek és a PLC eszközök között. Az ipari tabletről viszont kábel segítségével a helyszínen feltölthetők az előállított tesztszekvenciák. A fájl formátumának alkalmazhatónak kell lennie a vállalatnál található valamennyi tesztelő berendezésen. A növekvő vevői igények miatt az ipari partner szükségesnek érezte az eddig használt idő alapú tesztelés mellett az esemény alapú tesztelés bevezetését is, így az alkalmazásnak az ilyen típusú tesztek létrehozását is támogatnia kell.

2. Programozható logikai vezérlők rövid bemutatása

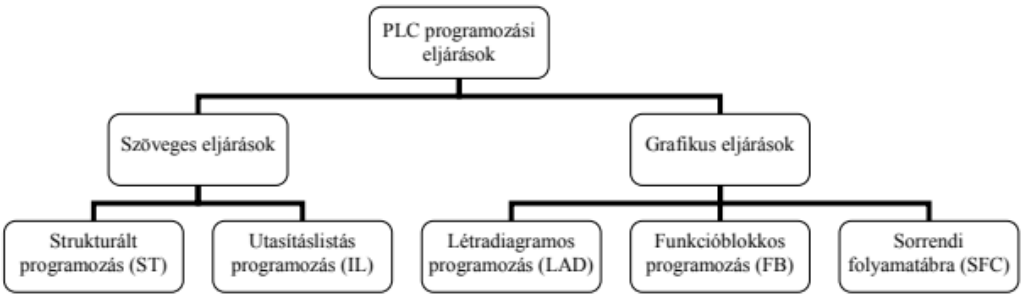
A PLC (Programmable Logical Controller) azaz a programozható logikai vezérlő egy olyan mikroprocesszorral ellátott, villamos energiával működtetett szabályzó és irányító berendezés, amely nem tekinthető számítógépnek. Eredetileg az autóiparban található régi relés vezérlők leváltására alkották meg, azonban napjainkban szinte kivétel nélkül minden iparágban megtalálható [1]. A PLC-k története, egy, a General Motors által 1968-ban kiírt pályázat folyamán kezdődött, melynek két pályázója a MODICON és az ALLEN-BRADLEY voltak.

Az 1969-es évben megjelent Richard Morley és Odo J. Struger által tervezett Modicon 084 elnevezésű berendezés. A konstrukció az autóiparban került először felhasználásra az 1970-es évek elején, amely később több változáson ment keresztül, így legfontosabb jellemzői csak az 1980-as évek közepére alakultak ki.

PLC-k alkalmazásának nagy előnye, hogyha egy termék gyártási technológiája változik, akkor ne legyen szükség új huzalozott vezérlő kialakítására, hanem elegendő a vezérlő átprogramozása az új gyártási technológia igényeinek megfelelően. A PLC-k széles körű alkalmazását az alábbi előnyök indokolják még [1]:

- mivel a hardver univerzális, gazdaságos megoldást nyújt,
- gyors prototípusfejlesztést tesz lehetővé,
- a programtervezés, a szimuláció, a validáció és a telepítés egyszerűen elvégezhető,
- mozgó alkatrészt nem tartalmaznak, így hosszú élettartamúak,
- ipari kivitelben készülnek,
- diagnosztikai feladatok ellátásra is alkalmasak,

- programozásuk számítógéppel, gyakran grafikus módszerekkel támogatott,
- kommunikációs felületük révén hierarchikus/osztott folyamatirányításra is alkalmasak.



1. ábra. *PLC programozási eljárások osztályozása. [2]*

A PLC-k fejlesztése folyamán a gyártók számos különböző programozási nyelvet fejlesztettek ki. Napjainkban a különböző PLC programozási nyelvekből kifolyólag az eszközök között természetes kompatibilitási problémák merülnek fel. Ennek következtében megnőtt az igény mind a gyártók, mind a felhasználók részéről a nemzetközi szabványban rögzített programnyelvek kifejlesztésére. Az IEC 61131-3:2003 vagy MSZ-EN 61131-3-2003 szabvány határozza meg a PLC eszközök kötelező programozási nyelveit [1]. A szabvány főbb jellemzőit összegyűjtöttük az 1. táblázatban.

A sokszor az egyes feladatok megoldásához több különböző gép egyidejű használatára van szükség. A programozási eszközök hardver specifikusak. Majdnem minden PLC forgalmazó a saját nézőpontja alapján valósítja meg az egyedi IEC 61131 szabvány implementációját [3][4].

Az IEC 1131-3 számú nemzetközi szabvány az egész világra egységesíteni kívánja a programnyelveket és ezek jelöléseit. A szabvány nem új programozási nyelveket ír le, hanem a meglévő nyelveket egyesíti közös tulajdonságaik alapján [5].

1. táblázat. *IEC 61131-3:2003 szabvány [6][1].*

Leírás	**Angol rövidítés**	**Német rövidítés**	**Megjelenítés**	**Megjegyzés**
Utasításlista	IL	AWL	karakteres	Egyszerű sorszámozott utasításokat, címeket, konstansokokat tartalmaz
Létradiagram	LD	KOP	grafikus	Olyan, mint egy elektromos kapcsolási rajz, amely 90°-kal el van forgatva
Funkcióblokk diagram	FBD	FBS	grafikus	Logikai szimbólumokat tartalmaz, különösen a Boole-algebrai feladatok megoldásához jó
Strukturált szöveg	ST	ST	karakteres	Hasonló, mint a magas szintű programozási nyelvek (C, Pascal, Basic)
Sorrendi folyamatábra	SFC	AS	grafikus	Egyfajta összetett folyamatábra (Grafcet [8])

A programvezérlés egy lépésenként haladó vezérlés. Az egyikről a következőre lépés egy feltétel teljesülésekor következik be. Ezt lefutó vezérlésként is nevezik, amely további két csoportra bontható.

- Az idő alapú vezérlés egy olyan programvezérlés, ahol a léptetési feltétel kizárólag időfüggő.
- Az esemény alapú vezérlés során a léptetési feltétel csak a vezérelt berendezés folyamatainak jeleitől függ. A vezérlőprogram lépésekre bontható. A vezérelt folyamat (a vezérlőprogram) lépésekre bontható, az egyik lépésről a következő lépésre történő továbbhaladás feltételekhez – a léptetési feltételekhez – kötött [7].

A PLC alapú irányító rendszerek az 1970-es évek elején rohamosan kezdek elterjedni, viszont azóta sem alakult ki széles körben elfogadott formális specifikáció. A Grafcet [8] az egyik legelterjedtebb, szabványosított leírási módszer, viszont a használata korlátozott. Ez alapján alakult ki a SFC (Sequantial Function Chart) programozási nyelv, amely szintaktikailag hasonló, de szemantikailag eltér. A CIF 3 [9] egy olyan tudományos eszköz, amely bár leginkább a hibrid rendszerek modellezését célozta meg, szintén tartalmaz programkód generátort. A koncepció működik, viszont az alkalmazott szöveges modellező nyelv rendkívül összetett, így a használata külön képzést igényel.

A [10] publikációban a szerzők bemutatnak egy egyedi, formális (statikus vizsgálattal vagy verifikációs technikákkal ellenőrizhető) specifikációs módszert PLC programokhoz PLCspecif néven. Továbbá bemutatnak egy olyan módszert, amely alkalmas a PLC-n futtatható kódok előállítására. Céljuk egy olyan módszer kidolgozása, amely megfelel a PLC komponensek leírására, egyszerűen használható és telepíthető. Az alkalmazásuk kimenete strukturált szöveg, saját szintaktikai és szemantikai követelményekkel, konfigurálható, könnyen olvasható és értelmezhető, továbbá betartja az ide vonatkozó fejlesztési konvenciók és szabványok előírásait. A [11] cikk egy automatikus PLC programozási módszert mutat be, amely a tervezés alatt álló rendszer kívánt viselkedésének modellezéséből indul ki és a kimeneten egy LAD (létradiagram) nyelven megírt teljes értékű program jelenik meg. Az idő mérésére állapotgépet használnak. A modell formális, ugyanakkor olvasható, validálható és megfelel a PLC programokkal szemben támasztott biztonsági követelményeknek is. A [12] tanulmány egy olyan keretrendszert mutat be, amely UML modellekből állít elő IEC 61131-3 szabványnak megfelelő programkódot.

3. Kifejlesztett alkalmazás bemutatása

A kifejlesztett szoftver a partner cégnél történő termékfejlesztési- és tesztelési folyamat szerves részét képezi. A fejlesztés fő célja egy olyan segédeszköz létrehozása, ami a különböző, speciális PLC nyelvek megtanulása nélkül támogatja a tesztmérnököket a PLC programok megírásában és a megírt programok kezelésében. Az eszköz kimenetének olyan formátumot és adatstruktúrát kell kiválasztani, amelyet sajátosságaitól függetlenül (programozási nyelv, gyártó, stb.) minden tesztberendezés közvetlenül támogat.

A probléma megoldásához az alábbi követelményeket állítottuk fel. Keresni kell egy olyan általános leíró formátumot (meta-nyelvet), amivel a PLC nyelvek közötti különbségeket el lehet fedni és a tesztekben a folyamatra és a funkcióra (a vezérlési logikára) lehet koncentrálni a körülményes szintaktika betanulása helyett. Ezt követően készíteni kell egy egyszerű alkalmazást, amivel az egyes gépek eltérő nyelvei egységes módon kezelhetők, új szekvenciák hozhatók létre és a meglévők szerkeszthetők. Fel kell készíteni az alkalmazást a PLC programban alkalmazott ciklikus végrehajtási szekvenciák kezelésére. Ha a vezérlés ezt nem támogatja, akkor compilereknél is gyakran alkalmazott „*loop-unrolling*” technikát kell alkalmazni a ciklusok kezeléséhez. A kialakított adatstruktúrának biztosítania kell az idő és az esemény alapú teszt szekvenciák kezelését is, mivel az ipari partner mind a kettő vezérlési típust használja a tesztelési folyamatai során. Az utasítások végrehajtása egyszerre függhet az eltelt időtől és előre meghatározott esemény vagy események bekövetkezésétől is.

A fejlesztés során kiemelt figyelmet fordítottunk arra, hogy a lehetőségekhez mérten olyan szoftveres technológiákat alkalmazzunk, amelyek egyszerűen átültethetők Android környezetbe is.

3.1. Tesztberendezések sajátosságainak leírása

A szoftver fejlesztésének első lépése az eszközök sajátosságait tartalmazó fájl formátumának meghatározása. Mivel a partnercég tesztberendezései számos különböző utasításkészlettel rendelkeznek, így egy olyan univerzális adatstruktúra kialakítására, megtervezésére van szükség, amely a PLC vezérléstől függetlenül, egységesen definiálja az egyes eszközök eltérő lehetőségeit. Az átadott minták alapján, valamint a megfogalmazott követelménynek legjobban megfelelve, a JavaScript Object Notation (JSON) szöveges adatstruktúra használta mellett döntöttünk. A formátum előnyei közé tartozik, hogy egyszerűen létrehozható, rugalmas és felépítése átlátható. A 2. ábrán bemutatjuk az alkalmazás tesztelése során használt IPX teszt állomás JSON formátumnak megfelelő leírását.

```
{
    "machine": "IPX Testbench",
    "commands": [
        {
            "id": "-1",
            "description": "Repeat test n times",
            "parameters": ["waiting time with time dimension (e.g. 1h)"],
            "value": "[%d]"
        },{
            "id": "0",
            "description": "Pause for a given time",
            "parameters": ["waiting time (e.g. 2000 s)"],
            "value": "[%d];[%d];[%d];[%d]"
        },{
            "id": "1",
            "description": "Motor operation",
            "parameters": ["waiting time (e.g. 2000 s)",
                    "RPM [1/min]",
                    "Current [A]",
                    "Flow Rate [1/min]"],
            "value": "[%d];[%d];[%d];[%d]"
        },{
```

2. ábra. *IPX teszt állomás leírása JSON formátumban*

A megtervezett adatstruktúrával minden tesztberendezés egységesen definiálható. Egy eszközhöz tartozó fájl a gép nevét és az utasítás listát tartalmazza. Egy utasítás elemei sorrendben a következők:

1. teszt egyedi azonosítója,
2. teszt leírása,
3. teszteléshez szükséges paraméterek listája és
4. a paraméterekhez tartozó értékek listája.

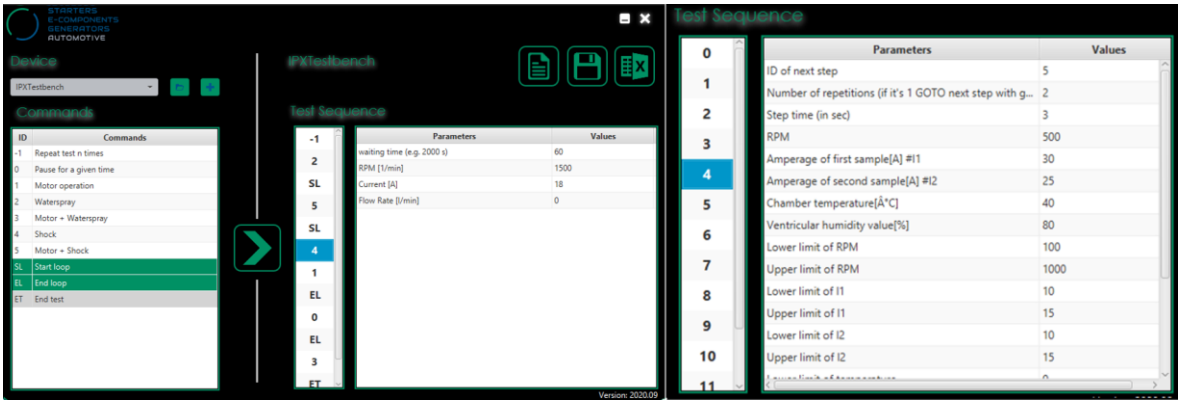
Az első utasítás egyedi azonosító száma minden esetben „-1” és az adott teszt szekvencia ismétlési számát határozza meg. Minden további utasítás esetén a tesztek leírása, a paraméterek és a hozzájuk tartozó értékek leírása és száma eszközönként eltérhet. Például az élettartam vizsgálat mindössze egyetlen utasítást foglal magába, viszont a paraméterek száma meghaladhatja a húszat is.

Az alkalmazás tartalmaz előre definiált sablonokat, ami segíti az új eszközök konfigurálását. A kiválasztott sablon kiegészíthető üres parancs blokkokkal, a paraméretek és a hozzájuk tartozó értékek testre szabhatók. Így nincs szükség a JSON fájlok közvetlen létrehozására vagy szerkesztésére, arra az alkalmazáson belül is van lehetőség.

3.2. Utasítás lista felépítése

Az utasítások felépítését szintaktikai követelmények rögzítik, viszont a paraméterek számára nem vonatkozik korlátozás. A 3. ábra bal oldalán lévő tesztberendezéshez hat különböző utasítás tartozik eltérő paraméterlistával. A „-1” azonosító számú utasítás (R) használata miatt a teljes szekvencia *n*

alkalommal fut le. Emellett viszont a példában elkészített teszt folyamat belső ciklusokat is tartalmaz, amelyeket szintén kezelni kell. Ehhez definiáltuk a *Start Loop* (SL) és *End Loop* (EL) kiegészítő utasításokat, továbbá létrehoztunk egy *End Test* (ET) utasítást, amely egy teljes szekvencia végét jelzi. A felsorolt kiegészítő parancsok automatikusan jönnek létre az egyes eszközök betöltésekor. Az SL utasítás paraméter értéke a ciklus ismétlések száma. A végrehajtás az SL_1 és EL_1 parancsok közötti utasításokat *i* alkalommal megismételi. Ha ezek között újabb SL_2 található, a végrehajtás belép, a blokkot végre hajtja *j* alkalommal, majd folytatja az EL_2 után következő parancs beütemezésével. A ciklusok és a beágyazások számára nem vonatkozik korlátozás.



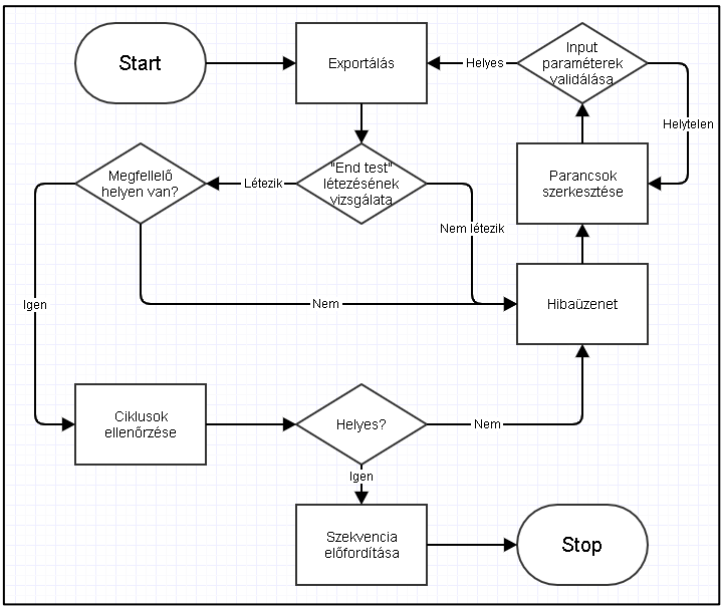
3. ábra. *Példa egy teszt szekvenciához tartozó parancslista felépítésére*

A 3. ábra jobb oldalán bemutatott élettartam teszt egyetlen utasításból áll (a teljes tesztszekvencia ismétlésére vonatkozó parancs kivételével), viszont a példában 20 darab paraméter tartozik hozzá. Az utasítás első paraméterének értéke határozza meg a következő lépés azonosító számát, így ebben az esetben nincs szükség a ciklikus végrehajtás speciális kezelésére. A lépések azonosító száma a létrehozáskor automatikusan növekszik.

3.3. Teszt szekvencia exportálása

Az elkészített teszt szekvencia kizárólag akkor exportálható, ha megfelel az adott szintaktikai követelményeknek. Az alkalmazás, valós idejű hibakezeléssel akadályozza meg a hibás tesztek létrehozását. A paraméterekhez tartozó értékek formátumának (szöveg vagy szám) és az eszközök leírásában található formátumnak meg kell egyeznie. Továbbá a megadott értékek nem lehet üresek (NULL) és intervallum megadása esetén a határértékeket nem léphetik át. Az SL és EL utasítások csak párban használhatók és minden tesztet ET utasítással kell lezárni. Ha a felhasználó elhagyja az ET utasítást, azt a program automatikusan hozzáfűzi. Az exportálási folyamat működését a 4. ábra szemlélteti.

Az előfordított szekvencia CSV formában jön létre, mivel ez a formátum minden PLC vezérlés számára értelmezhető. A beütemezett utasítások, beleértve a beágyazott ciklusok között létrehozott parancsok is, soronként jelennek meg a fájlban. Tehát például a 3. ábra bal oldalán bemutatott teszt ciklus eredménye, is utasítások szekvenciája lesz a ciklusszámnak megfelelően, többször ismételve.



4.ábra. *Teszt szekvencia exportálása*

4. Összefoglalás

A cikk egy olyan alkalmazást bemutat be, ami leegyszerűsíti a különböző szintaktikájú vagy eltérő paraméterezésű programozási nyelveket használó PLC tesztberendezések programozását, valamint támogatja a különböző tesztszekvenciák felhasználóbarát megtervezését. A feladat megoldására egy általános paraméter és utasításleíró metanyelvet alkalmaztunk, ami az eszközök programozási nyelveinek különbözőségeit egységesíti, majd a konkrét gépek utasításait és paramétereit *CSV* formátumú előfordított fájlban tárolja. Az egyes berendezések képesek a CSV-t saját specifikus PLC programozási környezetükkel értelmezni.

Az elkészült alkalmazást sikeresen integráltuk az ipari partnerünk termékfejlesztési- és tesztelési folyamatába. A partner által kapott visszajelzések alapján a kifejlesztett kísérleti szoftver jelentősen leegyszerűsíti a tesztszekvenciák elkészítését, gyakorlati használatát.

5. Köszönetnyilvánítás

A kutató munka az Európai Unió és a magyar állam támogatásával, az Európai Regionális Fejlesztési Alap társfinanszírozásával, a GINOP-2.3.4-15-2016-00004 projekt keretében valósult meg, a felsőoktatás és az ipar együttműködésének elősegítése céljából.

Irodalom

[1] Ajtonyi, I.: *PLC és SCADA-HMI rendszerek I.,* 1. kötet. PLC programozás az IEC 61131-3 szabvány szerint. Miskolc, AUT-INFO Kiadó, 2007.
[2] Maczik, M. A.: *PLC ismeretek és példatár*, 2013.
[3] Gasser, T., Xiao, D.: *Virtual Machine and Code Generator for PLC-Systems,* 2013.
[4] Johansson, S., Öhman, M.: *Prototype implementation of the PLC standard IEC 1131-3,* Department of Automatic Control, Lund Institute of Technology, 1995.

[5] Maslar, M.: *PLC standard programming languages: IEC 1131-3,* In Conference Record of 1996 Annual Pulp and Paper Industry Technical Conference. IEEE, 1996, pp. 26-31
[6] Ferenczi, I.: *PLC programozási alapismeretek*, 2017.
[7] Szabó, T.: *Gépészeti automatizálás*, Budapest, Eudutus Főiskola, 2011.
[8] *IEC 60848 GRAFCET specification language for sequential function charts*, IEC Std., 2013.
[9] van Beek, D., et al.: *CIF 3: Model-based engineering of supervisory controllers*, in Tools and Algorithms for the Construction and Analysis of Systems, ser. LNCS. Springer, 2014, vol. 8413, pp. 575–580. **https://doi.org/10.1007/978-3-642-54862-8_48**
[10] Darvas, D., Viñuela, E. B., Majzik, I.: *PLC code generation based on a formal specification language,* In 2016 IEEE 14th International Conference on Industrial Informatics (INDIN) pp. 389-396. IEEE, 2016. **https://doi.org/10.1109/INDIN.2016.7819191**
[11] Sacha, K.: *Automatic code generation for PLC controllers*, In International Conference on Computer Safety, Reliability, and Security. Springer, Berlin, Heidelberg, 2005, pp. 303-316. **https://doi.org/10.1007/11563228_23**
[12] B. Vogel-heuser, B., Ieee, M., Witsch, D., Katzke, U.: *Automatic Code Generation from a UML model to JEC 61131-3 and system configuration tools*, 2005, pp. 1034–1039.