Łukasz Sobaszek

Wykorzystanie środowiska Visual Studio w procesie dydaktycznym nauki programowania robotów przemysłowych

JEL: I23 DOI: 10.24136/atest.2019.251
Data zgłoszenia: 28.01.2020 Data akceptacji: 10.02.2020

W artykule przedstawiono możliwości wykorzystania popularnego środowiska programistycznego w celu opracowywania aplikacji użytkownych, wspomagających naukę programowania i obsługi współcześnie stosowanych robotów przemysłowych. Na wstępie scharakteryzowano podstawowe zagadnienia z zakresu wykorzystania robotów, metod ich programowania, a także współczesnych narzędzi stosowanych w tym obszarze. Ponadto omówiono możliwości wykorzystania środowiska Visual Studio w obszarze wsparcia procesu dydaktycznego nauki programowania robotów przemysłowych. W pracy zaprezentowano autorską aplikację komputerową, a także przedstawiono jej przykładowe zastosowanie.

Słowa kluczowe: robotyzacja, programowanie robotów, Visual Studio.

Wstęp

W obszarze zautomatyzowanego wytwarzania oraz wspierania procesów produkcyjnych wykorzystanie robotów znajduje coraz szersze zastosowanie. Ich implementacja pozwala na zwiększanie wydajności i elastyczności produkcji, wzrost bezpieczeństwa, osiągnięcie wysokiej jakość produkcji, podwyższenie niezawodności, a także zmniejszenie kosztów produkcji [1, 3].

Zakres wykorzystania robotów podczas realizacji procesów produkcyjnych jest bardzo szeroki. Znajdują one zastosowanie głównie na stanowiskach uciążliwych i niebezpiecznych, a także tam, gdzie brakuje wykwalifikowanych pracowników – w procesach spawania, malowania, czy klejenia [2]. Coraz częściej roboty implementowane są do realizacji innych prac, mając na celu głównie wspomaganie pracy ludzi. Przykładem mogą tu być autonomiczne roboty będące elementami bezzałogowych systemów transportowych w działach logistyki, czy nowoczesny i rozwojowy trend wykorzystania robotów przemysłowych jako maszyn realizujących zasadniczą obróbkę mechaniczną [3, 4].

Należy zatem stwierdzić, iż roboty przemysłowe znajdują coraz szersze zastosowanie podczas realizacji procesów produkcyjnych. W konsekwencji istotne jest, aby przyszli inżynierowie posiadali wiedzę z zakresu programowania oraz obsługi tego typu urządzeń. Programowanie robotów wymaga jednak niejednokrotnie posiadania specjalistycznej wiedzy dotyczącej konkretnych modeli robotów. Przykładem może być tutaj chociażby odmienność składni wykorzystywanych języków programowania. Istnieje zatem potrzeba opracowywania rozwiązań, które wspomagać będą proces nauki oraz obsługi robotów przemysłowych. W tym obszarze zastosowanie znaleźć może odpowiednie wykorzystanie popularnych środowisk programistycznych wykorzystujących języki programowania wysokiego poziomu. W niniejszej publikacji zaprezentowano wykorzystanie pakietu Visual Studio w procesie opracowywania aplikacji wspierających proces dydaktyczny nauki programowania robotów przemysłowych.

1.Programowanie robotów przemysłowych

1.1.Podział metod programowania

W celu wykorzystania robota przemysłowego do wspomagania różnorodnych procesów, niezbędne jest jego uprzednie zaprogramowanie. Mimo, iż każda z firm produkujących roboty proponuje własne rozwiązania w zakresie ich programowania, bardzo często sposób wprowadzania poleceń oraz składnia języka są do siebie zbliżone [5]. Wynika to z faktu, iż języki programowania robotów zostały opracowane w oparciu o języki programowania popularne w informatyce. Zatem zarówno strukturę jak i składnię bardzo łatwo można porównać ze znanymi w obszarze programowania komputerów rozwiązaniami. Do elementów wspólnych należy zaliczyć chociażby instrukcje warunkowe, czy możliwość deklaracji zmiennych. Dodatkowymi elementami w zakresie programowania robotów jest wykorzystanie specjalistycznych poleceń mających na celu chociażby wykonywanie ruchów manipulatora, czy komunikowanie się z portami wejścia/wyjścia [5, 6].

W obszarze programowania robotów przemysłowych wykorzystuje się różnorodne metody. Wyróżnia się programowanie [5]:

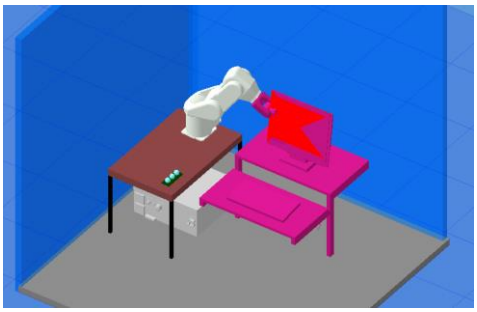
a) Online – wymagające obecności robota, stąd też tego typu metody nazywa się inaczej metodami nietekstowymi lub konwersacji. Nauka pracy robota może odbywać się wówczas poprzez programowanie ręczne z wykorzystaniem Teach Pendant'a (ruchem „od punktu do punktu" lub w trybie ciągłym), a także poprzez programowanie przez nauczanie (manipulowanie robotem przy wyłączonych hamulcach lub z wykorzystaniem czujników zamieszczonych na ramieniu robota).
b) Offline – realizowane bez robota, do którego głównych zalet należy zaliczyć brak konieczności zatrzymywania produkcji podczas modyfikacji programów. Robot programowany jest wówczas z wykorzystaniem odpowiedniego narzędzia w postaci programu komputerowego. Tworzenie i testowanie aplikacji odbywa się wówczas wyłącznie w środowisku wirtualnym.
c) Hybrydowe – łączące w sobie elementy zarówno metod online jak i offline.

Zatem operator lub integrator posiada do dyspozycji szereg metod pozwalających w odpowiedni sposób zapisywać ciąg poleceń robota, które następnie przetwarzane są przez kontroler w celu realizacji zakładanych czynności.

1.2.Narzędzia programowania robotów przemysłowych

Stosowanie narzędzi programowania offline w postaci programów komputerowych jest popularnym i aktualnym trendem. Rozwiązania tego typu wykorzystywane są głównie do programowania robotów w sposób zdalny, ale także do symulacji ich pracy w zdefiniowanym środowisku. Większość z popularnych narzędzi jest opracowywane przez firmy produkujące roboty przemysłowe. Do najpopularniejszych należy zaliczyć [7, 8]: RobotStudio (ABB), KUKA.Sim Pro (KUKA), MotoSIM EG-VRC (Yaskawa), RoboGuide (FANUC), Epson RC+ (Epson) czy K-ROSET (Kawasaki). Większość z wymienionych środowisk charakteryzuje się podobną budową oraz zbliżonymi możliwościami – dostępna jest w nich szeroka gama modeli robotów, chwytaków oraz elementów składowych

środowiska pracy (m. in. barier, linii transportowych, obrabiarek, elementów roboczych). Możliwe jest także korzystanie z wirtualnego kontrolera robota oraz jego zdalne programowanie. Programy tego typu pozwalają także na przeprowadzenie wielu analiz – wykrywanie kolizji, szacowanie czasu pracy, dokonywanie pomiarów odległości i wiele innych (rys. 1).



Rys. 1. Przykład wykrywania kolizji w wirtualnym środowisku programowania robotów [8]

1.3. Ograniczenia proponowanych rozwiązań

Proponowane przez producentów narzędzia programowania robotów posiadają wiele zalet. Należy jednak także pamiętać o ich ograniczeniach. Są one szczególnie widoczne, gdy rozpatruje się zastosowanie tego typu oprogramowania w procesie dydaktycznym nauki obsługi i programowania robotów przemysłowych [5, 7].

Mimo, iż wykorzystanie narzędzi wspierających programowanie robotów daje użytkownikowi wiele możliwość, złożoność tego typu rozwiązań może okazać się ograniczeniem podczas procesu dydaktycznego. Sama nauka obsługi środowiska symulacyjnego jest już trudnym i wymagającym czasu zadaniem, co w konsekwencji wydłuża zdobywanie umiejętności programistycznych. Ponadto znaczna ilość dostępnych poleceń obsługi i programowania robota powoduje trudność podczas zdobywania nowej wiedzy. Rozpoczęcie nauki od zaawansowanego i złożonego środowiska jest odstępstwem od głównej zasady kształcenia, jaką jest stopniowanie trudności. Praktyka wskazuje również, iż sama funkcjonalność proponowanych narzędzi jest w znacznym stopniu ograniczona (m. in. poprzez występujące błędy oprogramowania), co w konsekwencji zaburza proces sprawnego programowania robota.

Kolejne ograniczenie stanowi problem dostępności oprogramowania. Co prawda na rynku istnieją rozwiązania, które oferowane są bezpłatnie lub w wersji edukacyjnej, jednak większość z programów wymaga zakupu odpowiednich licencji. W konsekwencji wiąże się to z dodatkowymi kosztami użytkowania robota. Zakupując robota odpowiedniej firmy użytkownik otrzymuje co prawda dedykowane mu oprogramowanie, jednak zazwyczaj posiada ono licencję jednostanowiskową.

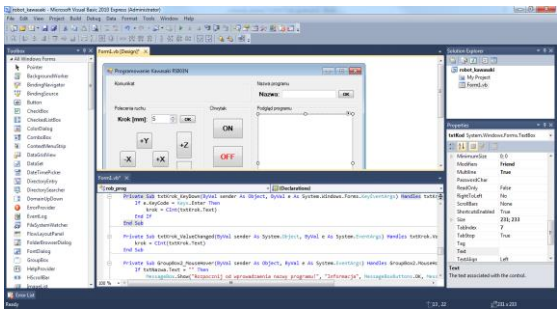
Z punktu widzenia procesu dydaktycznego, istotne ograniczenie tego typu rozwiązań stanowić może sam fakt wirtualnego kontaktu z robotem. Co prawda programowanie offline daje użytkownikowi szereg możliwości wizualizacji, jednak zdobywanie umiejętności poprzez testowanie własnych rozwiązań na rzeczywistym obiekcie przynosi w procesie dydaktycznym zdecydowanie większe korzyści.

W związku z powyższym, zasadnym jest budowanie aplikacji użytkowych wspomagających naukę programowania robotów. Opracowywanie własnych rozwiązań pozwala bowiem dostosować stopień zaawansowania aplikacji do wiedzy uczestników procesu dydaktycznego, a także daje pełną dowolność w aspekcie projektowania funkcjonalności oprogramowania. Zastosowanie znaleźć może tu chociażby popularne środowisko programistyczne Visual Studio, którego idea wykorzystania w tym procesie została przedstawiona w kolejnym rozdziale.

2.Środowisko programistyczne Visual Studio

2.1.Ogólna charakterystyka środowiska

Visual Studio to wszechstronne, zintegrowane środowisko programistyczne firmy Microsoft, przeznaczone do tworzenia zróżnicowanych aplikacji. Umożliwia ono kompleksową realizację projektu (w ramach jednego zagadnienia istnieje możliwość realizowania kilku odrębnych zadań). Środowisko to pozwala na wykonanie zintegrowanego oprogramowania, będącego kombinacją aplikacji okienkowych, internetowych, wykonywanych z linii poleceń czy web serwisów [9]. Niewątpliwą zaletą środowiska jest sam sposób budowy aplikacji (uprzednie przygotowanie projektu graficznego, a następnie kodowanie), intuicyjność, a także wbudowany interpreter, który wspomaga użytkownika podczas opracowywania kodów źródłowych (rys. 2).



Rys. 2. Interfejs użytkownika w środowisku Visual Studio .

Cechy charakterystyczne Visual Studio to [9]:

a) jeden zintegrowany interfejs użytkownika (IDE) dla kilku języków i różnych typów projektów,
b) połączenie z przeglądarką internetową,
c) wbudowane narzędzia do debugowania,
d) dostosowany do oczekiwań użytkownika interfejs.

Przedstawione powyżej zalety środowiska Visual Studio powodują, iż jest ono doskonałym rozwiązaniem podczas opracowywania aplikacji wspomagających proces programowania robotów przemysłowych. Szeroka gama dostępnych języków programowania powoduje, iż tego typu rozwiązania mogą być opracowywane z wykorzystaniem dowolnej składni.

2.2. Możliwości wykorzystania środowiska

Odpowiedni dobór elementów środowiska Visual Studio, a także integracja języków programowania (komputera oraz robota) pozwala opracowywać w pełni kompleksowe rozwiązania dedykowane wsparciu nauki programowania robotów przemysłowych. Schemat ideowy opracowywania tego typu rozwiązań został przedstawiony na rys. 3. Główną cechą proponowanego rozwiązania jest wykorzystanie faktu, iż języki programowania robotów wywodzą się w dużej mierze z języków programowania komputerów, przez co w łatwy sposób można je ze sobą łączyć.

Pierwszy etap proponowanego rozwiązania obejmuje klasyczne wykorzystanie środowiska Visual Studio. Użytkownik opracowuje wygląd graficzny aplikacji za pomocą kontrolek dostępnych w bibliotece środowiska. Zastosowanie takich elementów jak przyciski, suwaki, pola tekstowe, przyciski typu radio, itp. pozwala na opracowanie wyglądu programu w postaci panelu operatorskiego. Szerokie możliwość budowy interfejsu dają użytkownikowi dużą dowolność. Kolejny etap prac polega na wprowadzaniu kodu źródłowego, a zatem zasadniczym programowaniu aplikacji. Należy jednak nadmienić, iż wprowadzane polecenia mają za zadanie głównie obsługę graficznej części aplikacji – m.in. obsługę zdarzeń (kliknięcie, najechanie myszy), podświetlanie odpowiednich przycisków, sprawdzanie odpowiednich warunków (wartości w poszczególnych polach), dodawania zawartości do pól tekstowych. Część funkcjonalna apli-

kacji opracowywana jest bowiem dopiero w chwili zaimplementowania poleceń języka programowania pracy robota. Wykorzystanie odpowiednich elementów języka programowania komputera (jak chociażby możliwość łączenia znaków) umożliwia integrację poleceń sterujących robotem z opracowaną uprzednio warstwą graficzną aplikacji. Szeroka gama poleceń sterujących robotem również daje duże możliwości w zakresie budowy aplikacji. W konsekwencji opracowana zostaje w pełni funkcjonalna aplikacja.

Istotnym elementem wykorzystania środowiska jest konieczność zastosowania odpowiednich poleceń umożliwiających eksport oraz import opracowanego kodu sterującego robotem do kontrolera robota. Visual Studio, poprzez zastosowanie wbudowanych w środowisku okien dialogowych zapisu/odczytu pliku, w prosty sposób umożliwia realizację tego zadania. Zapisane pliki programu muszą zostać następnie przesłane do kontrolera. Kopiowanie opracowanego pliku programu może obywać się z wykorzystaniem terminala, który zazwyczaj jest aplikacją służącą do zdanej komunikacji z kontrolerem robota. Wymaga on jednak połączenia za pomocą sieci LAN lub WLAN. Terminal umożliwia także uruchomienie zaimportowanego kodu wygenerowanego za pomocą aplikacji. Drugą z opcji jest wykorzystanie portu USB, służącego do wymiany danych. Wówczas plik z zapisanym kodem programu robota w klasyczny sposób zostaje skopiowany z dowolnej pamięci zewnętrznej do pamięci kontrolera [8]. Zastosowanie znajduje tutaj dodatkowo Teach Pendant robota, służący do obsługi systemu operacyjnego robota.

Ostatnim elementem procesu jest weryfikacja poprawności kodu programu poprzez jego fizyczne uruchomienie oraz obserwację zgodności pracy robota.

W kolejnym rozdziale przedstawiono implementację proponowanego procesu w postaci autorskiej aplikacji wspomagającej naukę programowania robota przemysłowego Kawasaki RS003N, która została opracowana w środowisku Visual Studio.

3. Przykład aplikacji wspomagającej programowanie robota przemysłowego

3.1. Stanowisko laboratoryjne

Prace nad proponowanym oprogramowaniem zostały przeprowadzone na stanowisku laboratoryjnym wyposażonym w robota przemysłowego Kawasaki RS003N (rys. 4). Przedstawione stanowisko znajduje zastosowanie podczas zajęć dydaktycznych realizowanych w ramach kierunku Robotyzacja Procesów Wytwórczych.

Robot Kawasaki RS003N jest sześcioosiowym robotem przegubowym o sześciu stopniach swobody. Maksymalne obciążenie robota wynosi 3 kg, zaś jego maksymalna prędkość, to 6000 mm/s. Jednostkami zasilającymi są bezszczotkowe serwonapędy, a napięcie zasilające wynosi 200–240 V [10]. Za sterowanie ramieniem robota odpowiedzialny jest samowspierający kontroler, który jednocześnie odpowiada za możliwość komunikacji z robotem poprzez odpowiednie protokoły.

Rys. 4. Stanowisko laboratoryjne do programowania oraz symulacji pracy robota Kawasaki RS003N.

Ruch robota można programować za pomocą ręcznego programatora (Teach Pendant'a), przy pomocy dedykowanego oprogramowania K-ROSET oraz poleceń języka AS. Teach Pendant ma charakter panelu operatorskiego, który umożliwia sterowanie oraz programowanie robota (za pomocą zapamiętywania pozycji lub poleceń języka AS). K-ROSET jest zaawansowanym oprogramowaniem, które pozwala symulować trajektorię ruchu robota, analizować czasy cykli, wyszukiwać potencjalne kolizje oraz analizować pozycje umieszczenia robota na stanowisku. AS natomiast jest językiem programowania robotów stworzonym przez firmę Kawasaki, zawierającym zbiór poleceń ruchu, obsługi sygnałów wejścia/wyjścia, sterowania chwytakiem, itp.

3.2. Charakterystyka języka AS

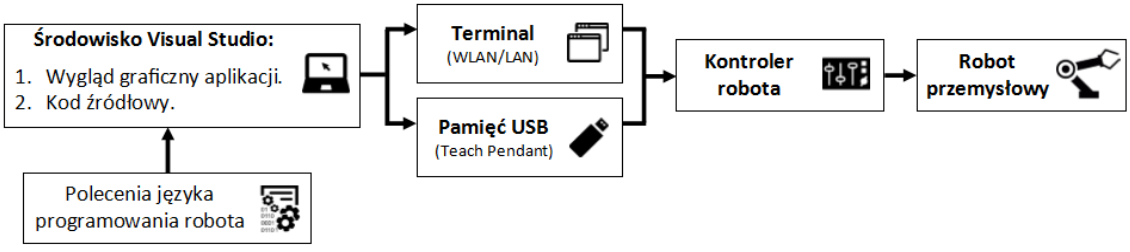
Programowanie robota za pomocą poleceń języka AS przypomina klasyczne programowanie wysokopoziomowe. Wykorzystując szereg poleceń (rozpoznawanych przez kontroler) programista określa kolejne zachowania robota. Podobnie jak w większości języków programowania dane mają postać: stałych, zmiennych lokalnych lub globalnych. Standardowe są również typy danych: dane numeryczne, ciągi znaków alfanumerycznych w kodzie ASCII oraz wartości logiczne. Instrukcje języka AS obejmują polecenia sterujące wykonaniem programu, definiowania pozycji robota, ruchowe, sterujące prędkością i dokładnością osiągania pozycji docelowej oraz narzędziem [11].

Język AS umożliwia wiele opcji wykonywania ruchów robota. Jeżeli podczas programowania wykorzystuje się bazę punktów (współrzędnych punktów zapisanych w pamięci kontrolera), wówczas możliwe są następujące rodzaje interpolacji:

a) liniowa (LMOVE),
b) kołowa (C1MOVE),
c) liniowa w przestrzeni konfiguracyjnej manipulatora (JMOVE).

W przypadku, gdy etap definiowania punktów zostaje pominięty, zastosowanie znajdują następujące polecenia ruchu:

a) DRIVE – obrót zdefiniowanej osi o zadany kąt z określoną prędkością,



Rys. 3. Wykorzystanie środowiska Visual Studio w procesie programowania robotów przemysłowych.

b) DRAW – przesuw robota zgodnie z zadanymi przemieszczeniami w osiach X, Y i Z, a także kątami obrotu względem tych osi.

Kluczowe są także instrukcje sterujące prędkością (SPEED) i dokładnością osiągania pozycji (ACCURACY). Sterowanie chwytakiem robota odbywa się zaś za pomocą instrukcji uruchomienia (OPEN lub CLAMP) i wyłączenia (CLOSE lub CLAMP) narzędzia. Do sterowania przebiegiem programu wykorzystuje się standardowe instrukcje wyboru (IF, THEN, ELSE), pętle (FOR, DO, WHILE) i instrukcję skoku (GOTO) [11]. Na rys. 5 został przedstawiony przykład prostego programu polegającego na ruchu i pobraniu elementu za pomocą chwytaka.

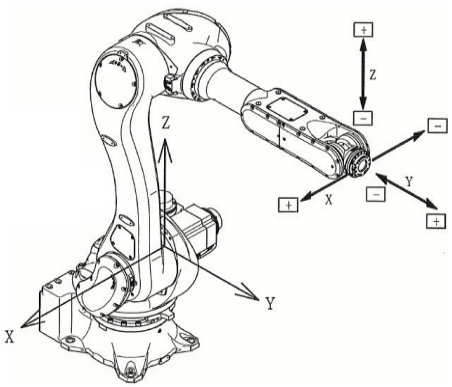
```
.PROGRAM paleta3x3()

TOOL tool1
SPEED 40

JAPPRO #start, 100
LMOVE #start
CLAMP 1
LAPPRO #start, 100
JAPPRO #p01, 50
LMOVE #p01
CLAMP -1
```

Rys. 5. Fragment kodu napisanego w języku AS

Przejrzystość składni języka AS sprawia, iż może znaleźć on zastosowanie podczas opracowania aplikacji wspomagającej naukę programowania robotów Kawasaki. Głównym elementem tego typu rozwiązania jest odpowiednia integracja poleceń wykorzystywanego języka programowania komputerów z językiem AS.

3.3. Opracowana aplikacja

Celem prezentowanych prac było opracowanie aplikacji umożliwiającej generowanie programów sterujących robotem przemysłowym Kawasaki RS003N. W tym celu wykorzystano omówione w poprzednim rozdziale środowisko Visual Studio, zaś językiem kodowania był Visual Basic. Założono, iż opracowywana aplikacja zostanie wykorzystana podczas nauki podstaw programowania robota i umożliwi sterowanie robotem zgodnie z układem współrzędnych osi narzędzia (tzw. układzie TOOL) (rys. 6). Wykorzystano zatem polecenia ruchu, które umożliwiają przemieszczanie się ramienia bez konieczności uprzedniego deklarowania współrzędnych punktów ruchu (polecenia DRAW). Dodatkowo przyjęto, iż program powinien umożliwiać złożone ruchy robota (obroty osi) oraz obsługę chwytaka podciśnieniowego. W procesie budowy aplikacji wykorzystano ideę opracowania programu przedstawioną w niniejszym artykule.

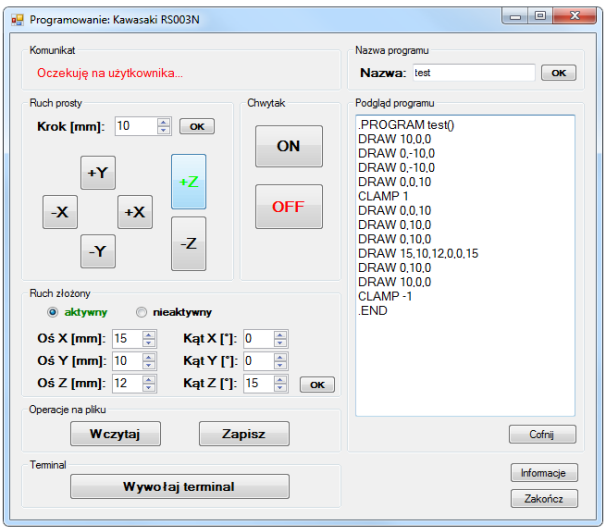


Rys. 6. Przyjęty układ współrzędnych programowania robota [10].

Zgodnie z przedstawionym w niniejszej pracy schematem postępowania ogół prac obejmował:

1. Opracowanie interfejsu aplikacji.
2. Programowanie aplikacji.
3. Implementację odpowiednich poleceń języka AS.

Wygląd graficzny proponowanej aplikacji został opracowany w takich sposób, aby programowanie robota odbywało się w intuicyjny i przystępny sposób. Poszczególne elementy funkcjonalne zostały pogrupowane za pomocą odpowiednich kontrolek, co zwiększa przejrzystość interfejsu (rys. 7). W górnej części okna umieszczono pole komunikatów, dzięki którym użytkownik na bieżąco informowany jest o konsekwencjach swojego działania. Takie rozwiązanie pozwala chociażby na skojarzenie składni generowanego kodu z zachowaniem robota. Poniżej okna komunikatów znajdują się zasadnicze przyciski aplikacji, umożliwiające generowanie poleceń ruchu – użytkownik ma do wyboru ruchy proste (w pojedynczych osiach), a także ruchy złożone (przemieszczenia złożone wraz z obrotami). Decyzję co do parametrów ruchu podejmuje sam użytkownik. Obok poleceń ruch zamieszczono przyciski pozwalające na obsługę chwytaka robota. Całość generowanego programu zostaje zestawiona w polu tekstowym o wielu liniach, które znajduje się w grupie „Podgląd programu". Dzięki wykorzystaniu odpowiednich okien dialogowych użytkownik informowany jest o poprawnej kolejności działań (pierwszą z czynności jest zdefiniowanie nazwy programu, a następnie możliwe jest generowanie programu robota). W razie popełnienia błędu użytkownik posiada możliwość cofnięcia ostatniego kroku programu.



Rys. 7. Interfejs graficzny opracowanej aplikacji.

W opracowanym kodzie źródłowym aplikacji dokonano implementacji poleceń języka AS. Była ona możliwa dzięki zastosowaniu odpowiednich poleceń języka Visual Basic (rys. 8).

```
Private Sub btnChwytakON_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnChw
    txtKod.Text = txtKod.Text & "CLAMP 1" & vbCrLf
    btnChwytakON.ForeColor = Color.Green
    btnChwytakOFF.ForeColor = Color.Black
    txtChwytakInfo.Visible = True
End Sub

Private Sub btnNazwaOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnNazwa
    txtKod.Text = txtKod.Text & ".PROGRAM " & txtNazwa.Text & "()" & vbCrLf

End Sub

Private Sub btnX_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnX.Click
    txtKod.Text = txtKod.Text & "DRAW " & krok & ",0,0" & vbCrLf
    lblInfo.Text = "Oś X: przemieszczenie do przodu o " & krok & " mm."
End Sub

Private Sub btn_X_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_X.Click
    txtKod.Text = txtKod.Text & "DRAW -" & krok & ",0,0" & vbCrLf
    lblInfo.Text = "Oś X: przemieszczenie do tyłu o " & krok & " mm."
End Sub
```

Rys. 8. Fragment kodu źródłowego aplikacji.

Dzięki możliwości łączenia znaków (operator „&") w polach tekstowych generowane są polecenia ruchu robota wraz z parametrem dystansu przejścia, który przechowywany jest w pamięci komputera jako zmienna „krok". Kluczowym elementem kodu jest także znak przejścia do nowej linii (operator „vbCrLf"), który umożliwia zachowanie poprawnej struktury programu w języku AS. W kodzie zamieszczono także szereg instrukcji warunkowych typu IF, które sprawdzają poprawność wprowadzanych danych, a także decydują o stanie przycisków (aktywny/nieaktywny).

W celu zapisu rezultatów działania programu w postaci pliku tekstowego z rozszerzeniem „*.PG" wykorzystano kontrolki zewnętrznych okien dialogowych (OpenFileDialog oraz SaveFileDialog). Umożliwiają one zarówno zapis, jak i odczyt uprzednio opracowanego kodu sterującego. Są one także pomoce podczas zapisu programu na zewnętrzny nośnik USB. W celu możliwość wczytania opracowanego programu bezpośrednio z pozycji komputera – w aplikacji zamieszczono przycisk wywołania terminala, który uruchamia aplikację KCwinTCP.

Rezultatem przeprowadzonych prac było opracowanie użytkowej aplikacji umożliwiającej przystępne programowanie robota Kawasaki RS003N w której wykorzystywane są podstawowe polecenia języka AS.

Prezentowane rozwiązanie może być odpowiednio modyfikowane i rozbudowywane o kolejne elementy – stopień zaawansowania aplikacji zależny jest bowiem od stanu wiedzy i umiejętności osób uczestniczących w procesie dydaktycznym. Opracowany program komputerowy stanowi ponadto doskonałą bazę do tworzenia szeroko pojętych aplikacji użytkowych, dedykowanych konkretnym operacjom technologicznym, których realizacja wspierana jest pracą robotów przemysłowych.

Podsumowanie

Roboty przemysłowe znajdują szerokie zastosowanie we współczesnych procesach produkcyjnych. Ich wykorzystanie nie ogranicza się jedynie do operacji pomocniczych, ale coraz częściej polega na realizacji zasadniczych operacji technologicznych. Istnieje zatem potrzeba, aby osoby znajdujące przyszłe zatrudnienie w firmach produkcyjnych posiadałby podstawową wiedzę z zakresu programowania współczesnych robotów przemysłowych. Wykorzystanie podczas nauki tej umiejętności środowiska Visual Studio możne znacznie usprawnić ten proces. Pozwala bowiem ono na opracowywanie aplikacji o ściśle zdefiniowanych możliwościach i funkcjonalności, które dedykowane są konkretnym modelom robotów. Nauka programowania staje się wówczas prostsza i bardziej przystępna. Zaprezentowana w niniejszej publikacji aplikacja jest przykładem tego typu rozwiązania. Umożliwia ona bowiem naukę podstawowych poleceń ruchu oraz generowanie kodu sterującego robotem Kawasaki. Możliwości opracowywania tego typu oprogramowania w środowisku Visual Studio są jednak zdecydowanie większe. Należy zatem podejmować pracę nad budową zaawansowanych aplikacji użytkowych – nie tylko wspomagających naukę programowania, ale także znajdujących zastosowanie jako narzędzia wspomagające pracę robotów w ściśle sprecyzowanych zastosowaniach – jak chociażby procesy spawania czy malowania.

Bibliografia:

1. Hajduk M., Koukolová L., Trends in Industrial and Service Robot Application, Applied Mechanics and Materials, 791, pp. 161–165, doi:10.4028/www.scientific.net/AMM.791.161
2. Hajduk M., Jenčik P., Jezný J., Vargovčik L., Trends in Industrial Robotics Development, Applied Mechanics and Materials, 282, pp. 1–6, doi:10.4028/www.scientific.net/AMM.282.1
3. Sobaszek Ł., Gola A., Świć A., Kierunki rozwoju robotyki w aspekcie projektowania współczesnych systemów produkcyjnych, Innowacje w zarządzaniu i inżynierii produkcji, t. 1, [red:] Knosala Ryszard – Opole: Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, 2017, s. 460–471.
4. Pedersen M. R., Nalpantidis L., Andersen R. S., Schou C., Bøgh S., Krüger V., Madsen O., Robot skills for manufacturing: From concept to industrial deployment, Robotics and Computer-Integrated Manufacturing, Vol. 37, 2016, pp. 282–291, https://doi.org/10.1016/j.rcim.2015.04.002.
5. Kaczmarek W., Panasiuk J., Języki programowania a programowanie robotów, Napędy i sterowanie, Nr 3, 2019, str. 72–78.
6. Honczarenko J.: Roboty Przemysłowe. Budowa i zastosowanie, Wydawnictwa Naukowo-Techniczne, Warszawa, 2004.
7. Kumičáková A. Rengevič A., Automation of Manufacturing Technologies with Utilisation of Industrial Robots, Applied Computer Science, Vol. 11, No. 3, 2015, pp. 5–18.
8. Sobaszek Ł., Gola A.: Perspective and Methods of Human – Industrial Robots Cooperation. Applied Mechanics and Materials, 2015, vol. 791, pp. 178–183, https://doi.org/10.4028/www.scientific.net/AMM.791.178
9. Halvorson M.: Microsoft Visual Basic 2010 Step by Step, Microsoft Press, Washington, 2010.
10. Kawasaki Robot Materials, RS003NFE70 – Robot Specification, Kawasaki Heavy Industries, Ltd.
11. Kawasaki Robot Materials, Kawasaki Robot Controller E Series Operation Manual, Kawasaki Heavy Industries, Ltd.

Using the Visual Studio environment in the didactic process of industrial robots programming

The paper presents the possibilities of using the popular programming environment to develop applied applications that support teaching process of programming and operating modern industrial robots. First of all, the basic issues of the use of robots, methods of robot programming and tools used in this area were characterized. Moreover, the possibilities of using the Visual Studio environment to support the didactic process of industrial robot programming teaching were discussed. In addition, the paper presents the author's computer application and an example of its use.

Keywords: robotisation, robot programming, Visual Studio.

Autorzy:

dr inż. **Łukasz Sobaszek** – Politechnika Lubelska, Wydział Mechaniczny, Katedra Informatyzacji i Robotyzacji Produkcji, Zakład Robotyzacji i Organizacji Produkcji, l.sobaszek@pollub.pl