Krystian ŁYGAS, Wojciech DANILCZUK

TRANSLATOR G-CODU NA JĘZYK ROBOTA KAWASAKI

W artykule opracowano translator znormalizowanego języka zapisu poleceń dla urządzeń CNC (G-Code) do języka AS obsługiwanego przez roboty firmy Kawasaki. W tym celu powstał program komputerowy przekształcający komendy zapisane w standardzie G-codu na kod w języku AS. Program translacji został napisany w języku Python. Walidacja translatora polegała na testach z wykorzystaniem robota Kawasaki RS003N. Program grawerowania powierzchni został przekształcony na kod w języku AS. Proces grawerowania został zasymulowany i zrealizowany z wykorzystaniem robota.

WSTĘP

W coraz większej liczbie aplikacji przemysłowych wykorzystuje się roboty we współpracy z obrabiarkami CNC. Często wykorzystywane są one do operacji przenoszenia detali z/do obrabiarek, wykonywania operacji technologicznych takich jak spawanie, klejenie czy gratowanie [5]. Prowadzi to do sytuacji w których osoby projektujące bądź serwisujące tego typu aplikację muszą posiadać znajomość zarówno kodu obsługiwanego przez obrabiarki CNC oraz język robota dedykowany danej aplikacji. W przypadku maszyn CNC nie stanowi to problemu, znaczna część maszyn obsługiwana jest przez G-code opisany w normie ISO 6983 [6]. Maszyny te są stosowane na rynku od wielu lat, znajomość poleceń języka G-code jest rozpowszechniona zarówno wśród serwisantów, technologów i konstruktorów. Języki programowania robotów są determinowane przez producentów. Nie podlegają one żadnej normie i różnią się w zależności od marki robota. Stąd pojawił się pomysł autorów na opracowanie translatora języka G-code na język robota. Pozwoli to ułatwić współpracę z robotami osobom znających standardy języka G-code. Autorzy uważają, że pozwoli to skrócić czas programowania bądź modyfikacji aplikacji w których robot współpracuje z maszynami CNC bądź sam wykonuje proces technologiczny.

Opracowany przez autorów translator tłumaczy komendy zapisane w języku G-code na język AS stosowany w robotach Kawasaki. Sam translator został przygotowany w języku Python [4]. Omawiane w artykule cechy robotów przemysłowych dotyczą się robotów szeregowych.

1. JĘZYKI PROGRAMOWANIA ROBOTÓW PRZEMYSŁOWYCH

Języki programowania robotów różnią się w zależności od producenta manipulatorów. Każdy posiada swoja własna składnie oraz strukturę. Dla przykłady roboty firmy FANUC programowane są w języku KAREL, w robotach KUKA stosuje się język KRL natomiast Kawasaki obsługuje system oraz język AS. Niezależnie od producenta i języku wszystkie roboty obsługują szereg typowych poleceń. W każdym w ww. języków mamy możliwość definicji zmiennych - kontrolery robotów mogą operować na zmiennych różnego typu. Najczęściej są to liczby typu integer, ciągi znaków ASCI (char), macierze i zmienne boolowskie. Kontrolery robotów są też w stanie wykonywać operacje matematyczne, zarówno te proste jak dodawanie i odejmowanie, ale również bardziej zaawansowane jak obliczanie funkcji sinus, czy generowanie liczb losowych. Kolejną cechą języków programowania robotów jest obsługa podstawowych struktur logicznych. Warunki logiczne oraz pętle są bardzo przydatne przy definiowaniu algorytmu sterowania całym programem robota, nie tylko jego komendami ruchu ale również obsługą wejść / wyjść i komunikacją po typowych dla przemysłu standardach (ETHERNET, PROFINET, PROFIBUS, etc.). Ważną grupą komend są definicje układów współrzędnych robota. W zależności od manipulatora bądź narzędzia jakie zamontowane ma na głowicy robot w różny sposób definiuje się centralny punkt narzędziowy (an. Tool central point, TCP, Utool) oraz układy współrzędnych związane z bazą ruchu (ang. Base, Uframe). Kolejną grupę komend stanowią definicje związane z prędkością oraz dokładnością ruchu. Ostatnią rodziną poleceń, najbardziej kojarzącą się z programowaniem robotów, są komendy ruchu. Roboty szeregowe wykonują ruchy z interpolacją przegubową (ang. Joint movement) gdzie każda z osi robota jest poruszana osobno, ruchy z interpolacja liniową (ang. Linear movement) gdzie układ współrzędnych związanych z narzędziem/ manipulatorem robota porusza się w kartezjańskim układzie współrzędnych oraz ruch z interpolacja kołową (ang. Circular movement) stosowany do wykonywania ruchów po okręgu.

2. JĘZYK PROGRAMOWANIA MASZYN CNC (G-CODE)

Struktura oraz polecenia języka programowania maszyn sterowanych numerycznie - G-code są znormalizowane. Opisuje je norma ISO 6983 [6] oraz DIN 66025 [7], [8]. Są one również powszechnie opisane w literaturze np. [1], [2]. Komendy języka G-code to przede wszystkim komendy ruchu maszyny (ruch przestawczy G0, ruch roboczy G1, ruch po okręgu G2 G3). Następną część poleceń dotyczy operacji technologicznych wykonywanych przez obrabiarkę (nawiercanie G81, toczenie gwintów G33 etc.). Dodatkowo maszyny sterowanie numerycznie obsługują polecenia związane z kompensacją narzędzia G43, pozycjonowaniu absolutnym lub przyrostowym G90 G91, wymiarowanie w calach i milimetrach G20 G21. Opisane wyżej funkcje typu G wymagają wprowadzenia parametrów min. położenia w osi XYZ, wybór prędkości wrzeciona S oraz posuwu F. Maszyny CNC obsługują również funkcje typu M. Dotyczą one przede wszystkim poleceń pomocniczych takich jak włączenie chłodziwa M07, włączenie obrotów wrzeciona M05 czy wybór kierunku obrotów wrzeciona M03, M04.

3. PODOBIEŃSTWA JĘZYKA G-CODE ORAZ JĘZYKA AS

W języku G-cod początek programu zawiera instrukcje odnoszące się do sposobu interpretacji poleceń wydawanych przez programistę. Przykładami mogą być komendy: G90- pozycjonowanie

bezwzględne, G91- pozycjonowanie przyrostowe, G53- wybranie układu współrzędnych maszynowych, G54-G59- wybranie układu współrzędnych przedmiotu, G20- wymiarowanie w celach, G21- wymiarowanie w milimetrach, G94- programowanie prędkości posuwu w [mm/min], G95- programowanie prędkości posuwu w [mm/obr] itp. Wywołanie odpowiednich komend powoduje późniejszy sposób interpretacji poleceń ruchu. Również definiuje się parametry technologiczne obróbki: F- prędkość posuwu, S – prędkość wrzeciona itp. W języku AS [3] nie jest stosowany powyższy schemat, lecz pewnym podobieństwem mogą charakteryzować się procedury wyboru układu współrzędnych. W robotach Kawasaki możliwy jest wybór układu współrzędnych związanego z narzędziem robota bądź globalnego układu współrzędnych. Wśród funkcji definiujących układ współrzędnych robota wyróżnić można: BASE- definiuje wartości przekształcenia podstawy, które określają relację ustawienia pomiędzy współrzędnymi globalnymi, a współrzędnymi wyjściowymi podstawy, TOOL – definiuje wartości przekształcenia narzędzia, które określają relację ustawienia pomiędzy współrzędnymi narzędzia, a współrzędnymi wyjściowymi narzędzia, SETHOME- konfiguruje pozycję domową SET2HOME- konfiguruje drugą pozycję domową. Również na początku programu definiowane są parametry ruchu: SPEED- określa prędkość ruchu, ALWAYS- odniesienie do całego programu, ACCURACY- zdefiniowanie dokładności ruchu.

Podstawowymi komendami języka G-code służącymi do poruszania głowicy wrzeciona w interpolacji liniowej są G0, G1. Różnią się głównie prędkościami ruchu. Komenda G0 oznacza ruch przestawczy, służy do pozycjonowania narzędzia przy dużych prędkościach oraz zerowych siłach skrawania. W przypadku obróbki stosuje się komendę G1 która służy do przemieszczania narzędzia z określonymi technologicznie parametrami ruchu. Zaś parametrem określającym ruch głowicy jest np. F2000 o jednostce zależnej od komend G94/G95.

```
G0 X0 Y0 Z0
G1 Z-10
G1 X50
G1 Y50
G1 X0
```

Rys. 1. *Przykład komend ruchu języka G-code*

Przykładowy kod języka przedstawia (Rys. 1), gdzie narzędzie przemieszcza się do współrzędnych (0,0,0) w ruchu przestawczym, następnie obniża o wartość 10 mm w kierunku osi Z, następnie wykonuje ruchu roboczy o 50 mm w kierunku osi X, analogicznie w kierunku Y oraz znów w osi X do współrzędnej (0,50,-10).

Język AS nie wyróżnia ruchu przestawczego od roboczego. Możliwe jest w nim przesuwanie narzędzia w interpolacji przegubowej (JMOVE) oraz liniowej (kartezjańskiej) (LMOVE). Prędkość ruchu deklaruje się podobnie jak w przypadku języka G-code, z tą różnicą, że komendę zadająca prędkość umieszcza się przed linią z rozkazem wykonania ruchu. Zmieniając wartość prędkości i przyspieszenia zadanego robota można uzyskać ruch przestawczy i roboczy.

a)
```
SPEED 10 ALWAYS
ACCURACY 1 ALWAYS
JMOVE #pick
LMOVE ref+place
LMOVE #pick
```

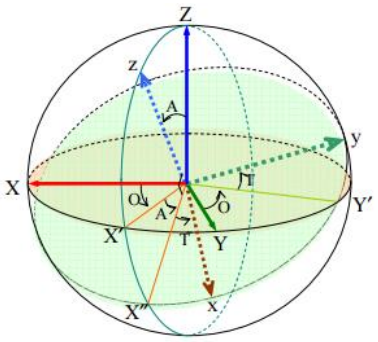
b)
```
LMOVE TRANS(0,0,-10,0,0,0)
LMOVE TRANS(0,0,-6.35,0,0,0)
LMOVE TRANS(0,0,-6.35,0,0,0)
LMOVE TRANS(0,0,-6.35,0,0,0)
```

c)
```
JMOVE #pick
JMOVE #place
```

d)
```
LMOVE odniesienie
LMOVE odniesienie +TRANS(0,0,-10,0,0,0)
LMOVE odniesienie + TRANS(0,0,-6.35,0,0,0)
LMOVE odniesienie + TRANS(0,0,-6.35,0,0,0)
LMOVE odniesienie + TRANS(0,0,-6.35,0,0,0)
```

Rys. 2. *Przykłady komend ruchu języka AS; a) interpolacja przegubowa i liniowa z zdefiniowanymi punktami ruchu; b) interpolacja liniowa, ruch do punktu; c) interpolacja przegubowa do zdefiniowanego punktu; d) interpolacja liniowa do punktu ze zmiennym offsetem*

W przypadku robotów przemysłowych współrzędne orientacji określane są przy pomocy kątów Eulera (Rys.3), zaś pozy przy pomocy współrzędnych kartezjańskich [3]. Wśród których wyróżnia się współrzędne pozy (X,Y,Z) oraz współrzędne orientacji (O,A,T) (Orientation, Azimuth, Tool), co zostało przedstawione. Rys.2 przedstawia kilka możliwości zadania ruchu przy pomocy zapisu języka AS, gdzie podpunkt: a) w interpolacji przegubowej oraz liniowe ze zdefiniowanymi punktami ruchu zapisanymi w pamięci kontrolera robota, b) interpolację liniową, gdzie współrzędne końcowe ruchu określa funkcja *TRANS*, c) interpolację przegubową ze zdefiniowanymi punktami w pamięci kontrolera, d) interpolację liniową do punktu „odniesienie" z translacją zapisaną przy pomocy funkcji *TRANS*.



Rys. 3 *Kąty Eulera [źródło: 3]*

Wszystkie kąty Eulera wykorzystywane są dla manipulatorów o sześciu stopniach swobody (6-dof, ang. *degree of freedom*). W przypadku manipulatorów do paletyzacji które posiadają cztery stopnie swobody, następuje redukcja kątów Eulera do jednego.

Ruch po łuku/okręgu w języku G-code zależy od notacji zapisu, jak również płaszczyzny, po której wykonany będzie ruch. Istnieje możliwość definicji punktu początkowego, promienia łuku oraz punktu końcowego, jak również punktu początkowego, punktu końcowego z współrzędną odnoszącą się do środka obrotu (Rys. 4.a). Szczegółowo procedura jest opisana w normie [6].

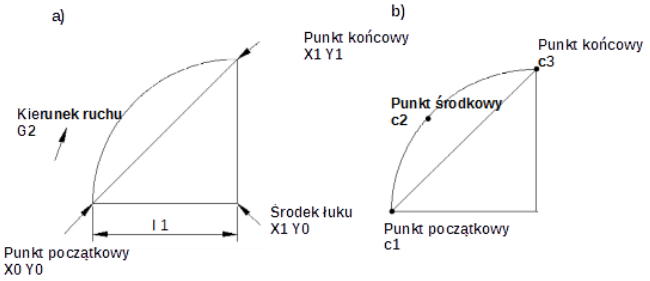
a)
```
G0 X0 Y0
G2 X1 Y1 I1 F10
```

b)
```
JMOVE c1
C1MOVE c2
C2MOVE c3
```

Rys. 4. *Kod programu służący do ruchu po łuku*

W przypadku języka AS definiuje się punkt początkowy ruchu, poprzez wykonanie ruchu do niego c1 (Rys. 5.b), punktu końcowego do łuku c3 (Rys. 5.b) oraz punktu znajdującego się na łuku c2 (Rys. 5.b). Wykorzystywane są do tego komendy C1MOVE (punkt w

środku łuku, oraz C2MOVE (końcowy punkt ruchu), co zostało pokazane na (Rys. 4.b).

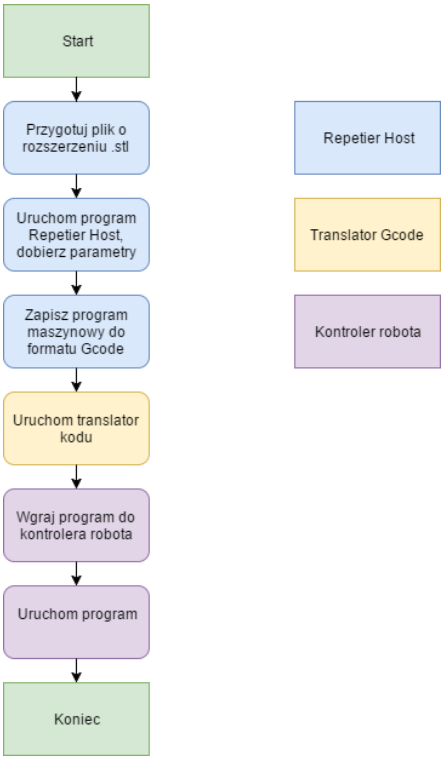


Rys. 5. *Procedura ruchu po łuku; a)G-code, b)AS*

Roboty przemysłowe swoją konstrukcją są przeznaczone do wykonywania operacji o wysokiej dynamice. Prędkości ruchu są większe niż w obrabiarkach numerycznych. W przypadku robota Kawasaki RS003N, maksymalna możliwa do uzyskania prędkość wynosi 6000 mm/s [5].

4. PROGRAM W PYTHONIE

Założeniem pracy było wykonanie programu służącego do translacji G-code do języka AS. Dokonano tego przy pomocy języka programowania Python, z zaimportowanymi bibliotekami *math* (zawierającą funkcje oraz stałe matematyczne) oraz *sympy* (zawierającą funkcje operacji matematycznych, symbolicznych) potrzebnymi do wykonania programu. Pełny proces przygotowania oraz walidacji systemu przedstawia (Rys. 6). Operację wykonane przez translator oznaczono kolorem według legendy (Translator Gcode).



Rys. 6*. Algorytm tworzenia G-code, translacji na język AS oraz walidacji na robocie.*

Tłumaczony G-code został wygenerowany na podstawie pliku .stl w programie Repetier Host (Rys. 8). Utworzony plik, zapisany w formacie *.gcode,* jest wejściem do programu translacyjnego. Wyjściem translatora jest kod zapisany w języku robota Kawasaki.

Program translatora w pierwszym kroku wczytuje do zmiennej *openfile* zawartość pliku o rozszerzeniu *.gcode*. Następnie przy pomocy pętli *for* sprawdza zawartość każdej z linii programu, znajduje komendy języka G-code, następnie znajduje wartości parametrów i przypisuje je do odpowiednich zmiennych. Uzyskane informacje używa do dodania kolejnych linii do zmiennej o nazwie *writeline,* której zawartość pod koniec programu zostaje zapisana do pliku o rozszerzeniu .as (format programu obsługiwany przez kontroler robota). Zawartość dodanej linii uzależniona jest od uzyskanych wcześniej wartości zmiennych oraz od rodzaju użytej komendy. Przykładowy kod dla operacji G0 został przedstawiony na rys. 7.

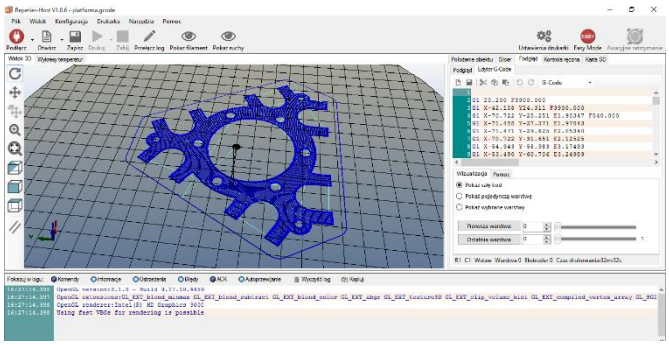
```
for line in openfile:
    if line[0:3]=='G0 ':
        Xpoprzednie=X
        Ypoprzednie=Y
        Zpoprzednie=Z
        numberx=line.find("X")
        numbery=line.find("Y")
        numberz=line.find("Z")
        if numberx==-1:
            X=Xpoprzednie
        else:
            X=line[numberx+1:numbery-1]
        if numbery==-1:
            Y=Ypoprzednie
        else:
            Y=line[numbery+1:numberz-1]
        if numberz==-1:
            Z=Zpoprzednie
        else:
            Z=line[numberz+1:len(line)-1]
        print X,Y,Z
        Z=-float(Z)
        writefile2.writelines("LMOVE odniesienie + TRANS(%s,%s,%s,0,0,0)\n" %(X,Y,Z))
        writefile.writelines(line)
        Z=-float(Z)
```

Rys. 7. *Fragment programu translatora*

Funkcja G1 została wykonana w analogiczny sposób. Z tą różnicą, że została zmieniona komenda odpowiedzialna za dokładność pozycjonowania narzędzia, oraz prędkość ruchu.

Kolejną zaimplementowaną funkcją były komendy G2, G3, które służą w zapisie G-code do przemieszczania narzędzia po łuku/okręgu. Dość duże różnice pomiędzy podobnymi funkcjami języków G-code oraz As, spowodowały, że należało opracować algorytm translacji komend. Ze względu na rozbudowaną zawartość, przedstawiono jego opis słowny. Analogicznie jak dla operacji funkcji G0 oraz G1 zostały odnalezione komendy określające wykonywany ruch: G2 oraz G3, oraz zostały przypisane do zmiennych parametry ruchu. Różnice pomiędzy obydwoma językami wymusiły napisanie algorytmu służącego do wyznaczenia współrzędnej środka łuku c2. Dokonano tego poprzez wyznaczenie równania punktu środkowego łuku oraz punktu znajdującego się pomiędzy współrzędnymi c1 oraz c3. Następnie przy użyciu tych punktów wyznaczono równanie funkcji liniowej przecinającej zarysowywany łuk. Następnie znaleziono część wspólną łuku oraz prostej, z czego wyznaczono współrzędną c2. Analogicznie jak w przypadku komendy G0 zapisano odpowiednie komendy ruchu do zmiennej, która na końcu programu została zapisana do pliku zgodnym z kontrolerem robota.

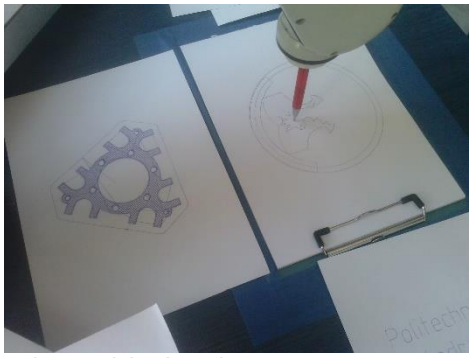
Przy pomocy omawianego algorytmu translatora uzyskano program służący do tłumaczenia języka G-code do języka AS zgodnego z kontrolerami robotów Kawasaki.



***Rys. 8**. Podgląd trajektorii G-codu uzyskanego w aplikacji Repetier Host*

5. PROCES WALIDACJI

Proces walidacji polegał na wygenerowaniu w aplikacji Repetier Host G-code. Wygenerowany program został poddany procesowi translacji. Następnie powstały plik w rozszerzeniu .as został wgrany do sterownika robota i uruchomiony. W celu obserwacji pracy robota manipulator wyposażono w narzędzie wykreślające trajektorię ruchu.



***Rys. 9.** Uzyskana trajektoria ruchu*

Zadaniem manipulatora było odwzorowanie ruchu zapisanego w postaci G-codu (Rys. 8). Uzyskano poprawną pracę translatora co jest widoczne na rys. 9.

PODSUMOWANIE

Założeniem projektu było przystosowanie robota szeregowego Kawasaki RS-003N do wykonywania operacji zgodnych z zapisem języka G-code. Opracowano program w języku Python który tłumaczył język G-code na język AS. Przeprowadzenie szeregu testów potwierdziło poprawną prace translatora. Uzyskano możliwość programowania robota w języku G-code który jest powszechniej znany niż język programowania robotów Kawasaki.

BIBLIOGRAFIA

1. M. J. Peterson, CNC Programming: Basics & Tutorial Textbook. Scotts Valley, Calif.: Createspace, 2008.
2. W. Grzesik i P. Nieslony, Programowanie obrabiarek CNC, 3. Auflage. Warszawa: Wydawnictwo Naukowe Pwn, 2017.
3. Kawasaki Heavy Industries,LTD. Kawasaki Robot Controller E Series, Installation and Connection Manual (February 2016: 13th Edition)
4. The official home of the Python Programming Language. 2017, http://www.python.org/
5. The official home of the Kawasaki Robotics, 2017 http://robotics.kawasaki.com
6. ISO 6983 Automation systems and integration, Numerical control of machines, Program format and definitions of address words
7. DIN 66025-1 Numerical control of machines, format; general requirements
8. DIN 66025-2 Industrial automation; numerical control of machines; format, preparatory and miscellaneous functions

G-code translator for Kawasaki Robots

The article presents a standardized language translator, write commands for the CNC machines (G-code) to the language supported by the Kawasaki robots (AS language). For this purpose, a computer program converting G-code commands into AS code was developed. The translation program was written in Python. The translator's validation consisted of tests using the Kawasaki RS003N robot. The surface engraving program has been converted to AS code. The engraving process was simulated and executed using a robot.

Autorzy:
mgr inż. Krystian Łygas – Politechnika Lubelska, Wydział Mechaniczny, Katedra Automatyzacji, k.lygas@pollub.pl,
mgr inż. Wojciech Danilczuk– Politechnika Lubelska, Wydział Mechaniczny, Katedra Automatyzacji,
wojciech.danilczuk@pollub.edu.pl.