Biuletyn WAT
Vol. LIX, Nr 3, 2010

Wykorzystanie protokołu UDP na potrzeby aplikacji do konfigurowania procesu paletyzacji kartonów dla robotów przemysłowych firmy Kawasaki

WOJCIECH KACZMAREK, MARCIN MISIEJUK[1]

Wojskowa Akademia Techniczna, Wydział Mechatroniki,
Instytut Systemów Mechatronicznych, 00-908 Warszawa, ul. S. Kaliskiego 2
[1]ASTOR, 00-739 Warszawa, ul. Stępińska 22/30

Streszczenie. W artykule przedstawiono sposób wykorzystania protokołu UDP (ang. *User Datagram Protocol*) na potrzeby aplikacji do konfiguracji procesu paletyzacji kartonów dla robotów przemysłowych firmy Kawasaki. Zaproponowane rozwiązanie, w połączeniu z opracowaną w ramach pracy dyplomowej w Instytucie Systemów Mechatronicznych Wojskowej Akademii Technicznej aplikacją, umożliwia łatwe oraz intuicyjne programowanie robotów firmy Kawasaki bez konieczności znajomości języka ich programowania. Główną zaletą takiego rozwiązania jest udostępnienie operatorowi prostego w obsłudze środowiska, skracającego znacznie czas przekonfigurowania robota w przypadku zmiany parametrów paletyzacji (wielkość palety, rozmiary kartonów, liczba warstw itd.).
Słowa kluczowe: protokoły transmisji, User Datagram Protocol, roboty przemysłowe, proces paletyzacji
Symbole UKD: 681.51

1. Wstęp

Lawinowo rozwijająca się technika komputerowa, zwłaszcza w dziedzinie automatyki i robotyki, wskazuje na celowość poszukiwania nowych rozwiązań dla zastosowań przemysłowych. Rozwiązania takie powinny zwiększać możliwości przeprogramowywania robotów przemysłowych, przy jednoczesnym upraszczaniu ich obsługi. Głównym celem takich poszukiwań powinno być zwiększenie wydajności i dochodowości linii produkcyjnych. Osiągnięcie tego celu jest możliwe poprzez wdrażanie do produkcji aplikacji dedykowanych konkretnym procesom technolo-

gicznym (np.: spawanie łukowe, paletyzacja, cięcie laserem), które pozwalają przede wszystkim na skrócenie czasu szkoleń operatorów oraz, dzięki swoim specyficznym zastosowaniom, uproszczenie samych środowisk programistycznych. Celowość takiego rozumowania jest wyraźnie widoczna w dobie rosnącej konkurencji, która niesie za sobą konieczność oferowania krótkich serii produkowanych artykułów (np.: zmiana kształtu butelek czy opakowań, etykiet, drobne modyfikacje wyglądu i funkcjonalności, dodawanie nowych modułów do istniejącego produktu), a tym samym częste modyfikacje uruchomionego cyklu produkcyjnego [2, 4, 5].

Obserwuje się, iż czołowe na rynku światowym firmy produkujące roboty przemysłowe (m.in.: ABB, FANUC, Mitsubishi) zdają się podążać w tym kierunku. Większość z nich proponuje obok bardzo rozbudowanych i posiadających ogromne możliwości środowisk do programowania robotów w trybach off-line (np.: ABB — środowisko RobotStudio, FANUC — środowisko RoboGuide, Mitsubishi — środowisko Cosimir), aplikacje wspierające konkretne procesy technologiczne (np.: firma ABB ma w swojej ofercie m.in. aplikacje: Arc Welding, Spot Welding, Die Casting, Assembly). Ciągle jednak można znaleźć procesy technologiczne, dla których takie oprogramowanie nie jest oferowane, a które są bardzo popularne na zautomatyzowanych liniach produkcyjnych. Tworzenie takich aplikacji można podzielić na trzy etapy:

— opracowanie właściwego środowiska z odpowiednim interfejsem użytkownika (może być zrealizowane w dowolnym języku programowania),
— opracowanie nowego lub wykorzystanie istniejącego protokołu transmisji, umożliwiającego przesłanie niezbędnych do poprawnego funkcjonowania robota parametrów do jego kontrolera i przygotowanie aplikacji umożliwiającej przepływ danych w tym protokole,
— opracowanie elastycznej aplikacji w języku programowania danego robota i zaimplementowanie jej w jego kontrolerze.

2. Proces paletyzacji na zautomatyzowanych liniach produkcyjnych

Paletyzacja jest procesem polegającym na ustawianiu na palecie w ustalonej konfiguracji przedmiotów, które pobierane są z bufora linii produkcyjnej. Paleta po zapełnieniu jest transportowana do magazynu, a na jej miejscu ustawiana jest kolejna, pusta paleta, po czym cały cykl się powtarza. Rolę paletyzatora może pełnić maszyna (najczęściej robot przemysłowy) lub człowiek, a sama paletyzacja może przebiegać na wiele sposobów [5].

Najprostsza jest paletyzacja ręczna, bez użycia urządzeń wspomagających. Jest to jednak sposób mało efektywny, główne ze względu na monotonię procesu oraz niejednokrotnie znaczną masę paletyzowanych detali. W wyniku tych niekorzystnych czynników, taki sposób paletyzacji występuje zazwyczaj w mało rozwiniętych przedsiębiorstwach małej produkcji.

W rozwijających się małych i średnich przedsiębiorstwach do paletyzacji wdrażane są roboty przemysłowe, najczęściej portalowe lub przegubowe. Dzięki tego typu rozwiązaniom, paletyzacja przebiega w sposób ciągły, ekonomiczny i bezpieczny. Wykorzystanie elastycznych robotów przemysłowych daje możliwość zmiany specyfikacji paletyzowanych detali (w pewnym zakresie może zmienić się ich masa oraz wymiary geometryczne). Jest to dużym atutem dla firm o zmieniającej się okresowo produkcji.

Przy doborze urządzeń paletyzujących należy zwrócić uwagę nie tylko na wielkość produkcji w danym przedsiębiorstwie, ale również na ilość dostępnej na hali produkcyjnej powierzchni. Jeżeli dostępna powierzchnia jest duża, wtedy można ten czynnik pominąć, jednak w przypadku gdy jest stosunkowo mała, należy przeanalizować wszystkie możliwości i wybrać najkorzystniejszą. Największy obszar zajmują roboty portalowe, z uwagi na wymaganą przez ich urządzenia peryferyjne dodatkową przestrzeń [4]. W przypadku znacznych ograniczeń powierzchniowych, rozwiązaniem jest zaprojektowanie urządzeń specjalistycznych lub użycie robota albo zespołu robotów przegubowych. Wybór pomiędzy powyższymi wariantami powinien być dokonany w oparciu o wiele innych czynników (koszty, funkcjonalność, obsługiwalność itp.).

Obecnie, z uwagi na częstą produkcję krótkich serii, bardzo ważną cechą urządzeń paletyzujących stał się czas niezbędny na przystosowanie ich do nowej produkcji. W przypadku robotów przemysłowych czas ten jest minimalizowany poprzez zastosowanie środowisk do programowania robotów w trybie off-line (np.: firma Kawasaki udostępnia środowisko PC-Roset). Jak wcześniej wspomniano, w ostatnim czasie, w odpowiedzi na zwiększającą się popularność robotów przegubowych wśród małych i średnich przedsiębiorstw, powstają aplikacje dedykowane dla najbardziej popularnych procesów przemysłowych. Dzięki tego typu aplikacjom oprogramowanie robota zajmuje stosunkowo mało czasu i nie wymaga specjalistycznej wiedzy na temat programowania robotów przemysłowych.

Dynamiczny rozwój robotyki w ostatnim czasie wpływa na tendencje do tworzenia uproszczonych aplikacji wspomagających obsługę procesów wykonywanych za pomocą robotów przemysłowych. Kontrolery robotów najnowszej generacji wyposażone zostały w ethernetowe moduły sieciowe, a obsługa komunikacji w programie robota wspomagana jest poprzez procedury, do których programista może się w łatwy sposób odwoływać.

3. Analiza problemu komunikowania się komputera PC z robotem firmy Kawasaki

Komunikacja pomiędzy robotem przemysłowym a komputerem PC może się odbywać przy wykorzystaniu magistrali szeregowej RS-232 lub sieci Ethernet. Standard RS-232 jest często wykorzystywany do komunikacji robota z urządze-

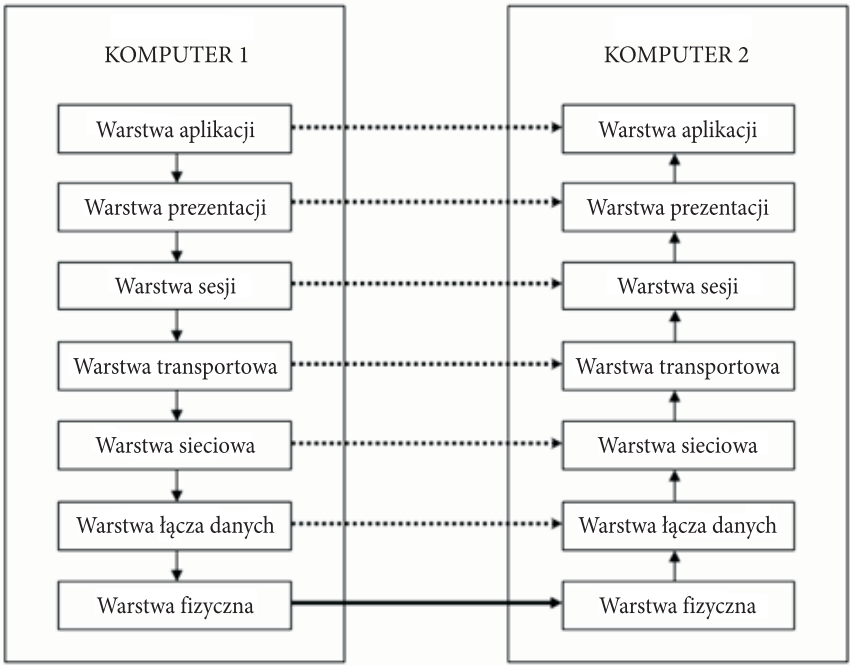
niami peryferyjnymi (np. kontrolerem spawarki). Z uwagi na niski transfer oraz ograniczenie do 15 metrów długości magistrali, odchodzi się od tworzenia sieci przemysłowych opartych o standard RS. Należy tutaj dodać, że obecnie odchodzi się również od implementowania portu szeregowego w komputerach osobistych. Z uwagi na powyższe ograniczenia do komunikacji pomiędzy komputerem PC a robotem przemysłowym w proponowanym rozwiązaniu zdecydowano się na technologię Ethernet. Jest to technologia wykorzystywana głównie przy tworzeniu lokalnych sieci komputerowych i przemysłowych. Opisuje ona specyfikację przewodów oraz sygnały, które są nimi przesyłane [9].

3.1. Specyfikacja technologii Ethernet

Magistrala sieci Ethernet może być bardzo rozbudowana. W jej skład mogą wchodzić przewody UTP/STP (*Unshielded Twisted Pair*/*Shielded Twisted Pair*) zakończone złączem RJ45, przewody koncentryczne, światłowody, koncentratory (huby) i inne urządzenia pracujące w warstwie fizycznej modelu OSI (*Open System Interconnection*). Model ten opisuje strukturę komunikacji sieciowej i jest traktowany jako model odniesienia dla protokołów sieciowych. Został opracowany przez międzynarodową organizację standaryzującą (ISO). Podstawowym założeniem modelu jest podział systemu na siedem warstw współpracujących ze sobą w ściśle określony sposób. Model OSI przedstawiony został na rysunku 1. Przerywanymi strzałkami przedstawiono komunikację pomiędzy równorzędnymi warstwami będącą wynikiem działania warstw niższych. Strzałkami ciągłymi zaznaczony jest rzeczywisty przepływ informacji.

Opis poszczególnych warstw modelu OSI:

— warstwa aplikacji — składa się z zestawu usług sieciowych dostępnych użytkownikom i aplikacjom,
— warstwa prezentacji — odpowiada za transformacje danych (szyfrowanie, kompresja),
— warstwa sesji — kontroluje interakcje pomiędzy węzłami sieci,
— warstwa transportowa — gwarantuje odpowiedni poziom niezawodności transferu; warstwa ta obejmuje mechanizmy i narzędzia nawiązywania połączenia i buforowania, numerowania oraz porządkowania pakietów,
— warstwa sieciowa — odpowiada za transfer plików pomiędzy węzłami sieci o dowolnej topologii; protokoły tej warstwy nie gwarantują doręczenia pakietów,
— warstwa łącza danych — odpowiada za transfer ramek danych pomiędzy dwoma węzłami sieci o topologii standardowej (np. magistrali) albo pomiędzy dwoma sąsiadującymi węzłami sieci w topologii dowolnej; w tej warstwie wprowadzane jest pojęcie adresu węzła fizycznego — MAC adresu,



Rys. 1. Model warstwowy OSI

— warstwa fizyczna — odpowiada za transfer bitów w kanale fizycznym (np. skrętce, kablu koncentrycznym lub światłowodzie); tutaj definiowane są charakterystyki otoczenia fizycznego oraz parametry sygnałów elektrycznych.

Rysunek 1 przedstawia sytuację, w której komputer 1 (klient) wysyła dane do komputera 2 (serwera). Dane te nie są wysyłane bezpośrednio z warstwy aplikacji klienta do warstwy aplikacji serwera. Muszą one przejść przez wszystkie warstwy modelu OSI. W każdej warstwie dodawane są dodatkowe informacje (np. długość pakietu, suma kontrolna CRC). Po stronie serwera dodatkowe informacje są wycinane z ramki. W rezultacie do warstwy aplikacji po stronie odbiorczej powinny dotrzeć dane w niezmienionej formie w stosunku do danych wysłanych z warstwy aplikacji klienta. Budowa ramki Ethernet w wersji 2 przedstawiona została w tabeli 1.

Tabela 1

Budowa ramki Ethernet w wersji 2

Preambuła	SFD	Adres MAC odbiorcy	Adres MAC nadawcy	Typ ramki	Dane	FCS
7 bajtów	1 bajt	6 bajtów	6 bajtów	2 bajty	46-1500 bajtów	4 bajty

Opis poszczególnych pól ramki Ethernet:
— preambuła — składa się z 7 bajtów naprzemiennych jedynek i zer [10101010101010101010101010101010101010101010101010101010],
— SFD — (ang. *Start Frame Delimiter*), czyli znacznik początkowy ramki [10101011],
— adres MAC odbiorcy/nadawcy — zawiera charakterystyczny numer powiązany z kartą sieciową, który jest niepowtarzalny i przypisany do danego urządzenia,
— typ ramki — jeżeli ma wartość mniejszą niż 1500 — oznacza długość danych, jeżeli ma większą długość— oznacza typ pakietu,
— dane — zawiera dane wysłane z warstwy aplikacji; jeżeli ciąg danych jest krótszy niż 46 bajtów— uzupełniany jest zerami,
— FCS — (ang. *Frame Check Sequence*), zawiera sumę kontrolną (CRC).

3.2. Protokoły sieciowe

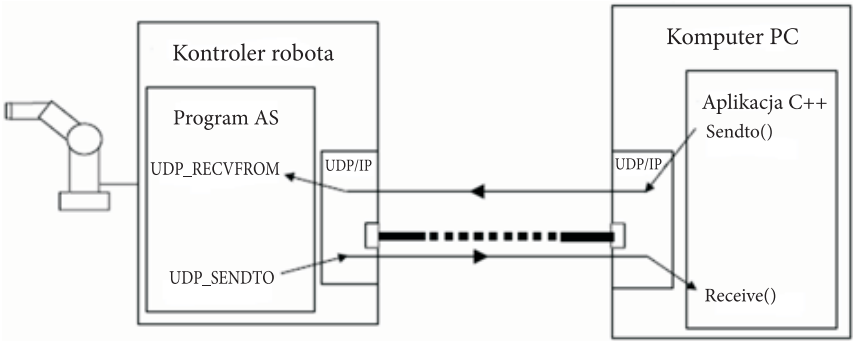
Do obsługi aplikacji sieciowych używane są najczęściej protokoły TCP (ang. *Transmission Control Protocol*) oraz UDP (ang. *User Datagram Protocol*). Protokoły te zaimplementowane są w warstwie transportowej modelu OSI. Umożliwiają one w połączeniu z protokołem IP (ang. *Internet Protocol*) komunikację pomiędzy dwoma urządzeniami sieciowymi. Protokół IP zaimplementowany jest w warstwie sieciowej i realizuje transmisję pakietów pomiędzy dowolnymi węzłami sieci. Przy czym IP nie ustanawia na potrzeby transmisji danych wirtualnego połączenia. Informacje są przesyłane w pakietach, a dane przesyłane są bez potwierdzenia doręczenia po stronie odbiorczej. Pakiet IP zawiera adres nadawcy i odbiorcy, identyfikator protokołu, „czas życia" pakietu (TTL) oraz sumę kontrolną służącą sprawdzeniu spójności pakietu. W przypadku wykrycia braku spójności pakietu po stronie odbiorcy jest on odrzucany, jednak nie jest wysyłane żądanie powtórzenia transmisji. Wszystkie mechanizmy zapobiegające utracie spójności danych muszą być zaimplementowane w protokołach warstw wyższych.

Do komunikacji w technologii Ethernet pomiędzy robotem przemysłowym Kawasaki a komputerem PC mogą być użyte protokoły UDP lub TCP. Do obsługi komunikacji pomiędzy rozbudowanymi aplikacjami wygodniej jest użyć protokołu TCP z uwagi na to, że ma on zaimplementowane mechanizmy powtarzania transmisji (w przypadku błędu) oraz kontroli spójności danych. Jest jednak kilkakrotnie wolniejszy od protokołu UDP z uwagi na konieczność nawiązywania połączenia logicznego pomiędzy komunikującymi się węzłami sieci. Cecha ta jest niezmiernie ważna w procesach przemysłowych, zwłaszcza tych, w których jednym z głównych czynników optymalizacyjnych jest czas. Dlatego, do obsługi mało rozbudowanych aplikacji, od których wymaga się dużej szybkości przepływu danych, oraz takich, które nie wymagają stałego połączenia logicznego pomiędzy dwoma węzłami sieci,

zalecane jest używanie protokołu UDP. Z uwagi na te uwarunkowania, w proponowanym rozwiązaniu zdecydowano się na wykorzystanie protokołu UDP.

Analiza komunikacji z wykorzystaniem protokołu UDP

Protokół UDP nie ustanawia połączenia logicznego pomiędzy węzłami sieci na potrzeby transmisji. Dane są nadawane do wszystkich węzłów sieci, jednak odbiera je ten węzeł, którego adres jest zgodny z adresem odbiorcy znajdującym się w pakiecie. Na rysunku 2 przedstawiono ogólny schemat komunikacji komputera PC z robotem przemysłowym firmy Kawasaki przy użyciu protokołu UDP [1].



Rys. 2. Ogólny schemat komunikacji komputera PC z robotem Kawasaki przy użyciu protokołu UDP

W celu odebrania danych przychodzących z komputera PC do kontrolera robota, opracowano stosowane programy komputerowe i zaimplementowano je na obu urządzeniach. Po stronie robota zaimplementowano aplikację serwera (hosta), natomiast po stronie komputera PC aplikację klienta.

Wykorzystująca protokół UDP aplikacja serwera napisana została w języku AS (język programowania robotów Kawasaki). Oparta jest ona o procedurę UDP_RECVFROM, dzięki której istnieje możliwość odebrania danych przychodzących z innych węzłów sieci [6, 7].

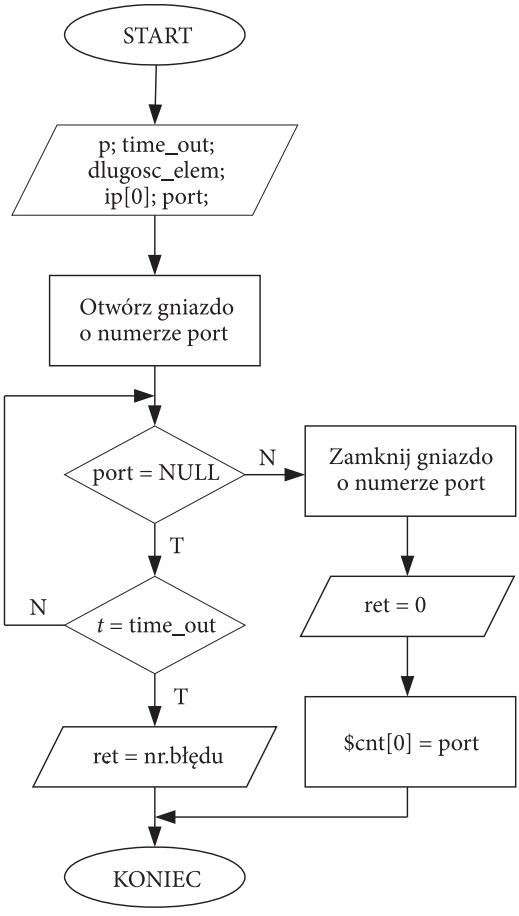
UDP_RECVFROM ret, port, $cnt[0], p, time_out, ip[0], dlugosc_elem, gdzie:

- ret — zmienna przechowująca numer błędu transmisji (0 jeżeli nie ma błędu),
- port — numer portu,
- $cnt[0] — deklaracja tablicy, do której zapisane zostaną dane (0 oznacza, iż elementy tablicy będą numerowane od 0 wzwyż),
- p — ilość elementów tablicy,
- time_out — czas [s], po którym zamykane jest gniazdo sieciowe w przypadku gdy nie dotrze do niego pakiet,

— ip[0] — tablica zawierająca adres IP komputera PC,
— dlugosc_elem — liczba znaków w pojedynczym elemencie tablicy.

Dane wysłane z aplikacji klienta w wyniku działania procedury UDP_RECVFROM są zapisywane do tablicy zmiennych ciągu znaków o nazwie $cnt[0]. Zawartość portu jest cyklicznie zapisywana do kolejnych elementów tablicy o długości scharakteryzowanej zmienną „dlugosc_elem". Uproszczony algorytm procedury UDP_RECVFROM przedstawiony został na rysunku 3.

Aby przesłać dane z komputera PC do robota w oparciu o protokół UDP, opracowano aplikację klienta. Aplikacja ta jest odpowiedzialna za wysłanie odpowiednio przygotowanych danych (parametrów robota niezbędnych do realizacji procesu paletyzacji) na określony port robota o zadanym adresie IP. W związku



Rys. 3. Algorytm procedury UDP_RECVFROM

z tym, iż parametry robota udostępniane są przez aplikację napisaną w środowisku Borland C++ Builder 6, również w tym przypadku posłużono się właśnie tym środowiskiem [3, 8].

Pierwszym etapem tworzenia aplikacji komunikacyjnej było wczytanie biblioteki gniazd. Wykorzystano do tego celu funkcję WSAStartup():

```
WSADATA wsda;
WSAStartup(MAKEWORD(2,2), &wsda);
```

Powyższy fragment kodu spowoduje wywołanie biblioteki Winsock 2.2. Przy czym parametr „wsda" jest wskaźnikiem struktury WSDATA, w której zawarte są informacje o wczytanej bibliotece.

Następnym etapem było utworzenie gniazda systemowego. Do tego celu wykorzystano funkcję socket():

```
SOCKET s;
s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

gdzie:

- AF_INET — parametr określający zastosowanie gniazda do obsługi protokołów z rodziny protokołów internetowych,
- SOCK_DGRAM — wskazanie na bezpołączeniowy transfer danych (gniazdo datagramowe),
- IPPROTO_UDP — wyznaczenie protokół UDP.

Przed operacją wysłania danych wypełniono strukturę adresową, co sprowadziło się do wpisania do struktury SOCKADDR_IN informacji o hoście:

```
SOCKADDR_IN addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(port);
addr.sin_addr.s_addr = inet_addr(szAddress);
```

gdzie:

- addr — wskaźnik do struktury adresowej SOCKADDR_IN, w której zostaną zapisane parametry hosta (rodzaj gniazda, port, adres IP odbiorcy),
- AF_INET — parametr określający zastosowanie gniazda do obsługi protokołów z rodziny protokołów internetowych,
- port — docelowy numer portu,
- szAddress — adres IP odbiorcy.

Do wykonania operacji wysłania zmiennej znakowej „bufor" posłużono się instrukcją sendto():

```
iMessageLen = strlen(bufor);
sendto(s, bufor, iMessageLen, 0, (struct sockaddr *) &addr, sizeof(addr)).
```

Opis parametrów funkcji sendto():
- s — wywołanie uprzednio utworzonego gniazda,
- bufor — zmienna przechowująca przesyłane dane,
- iMessageLen — długość komunikatu,
- 0 — znacznik określający metodę wysłania (0 oznacza brak znaczników),
- (struct sockaddr *) &addr — struktura adresu odbiorcy,
- sizeof(addr) — rozmiar struktury zawierającej adres odbiorcy.

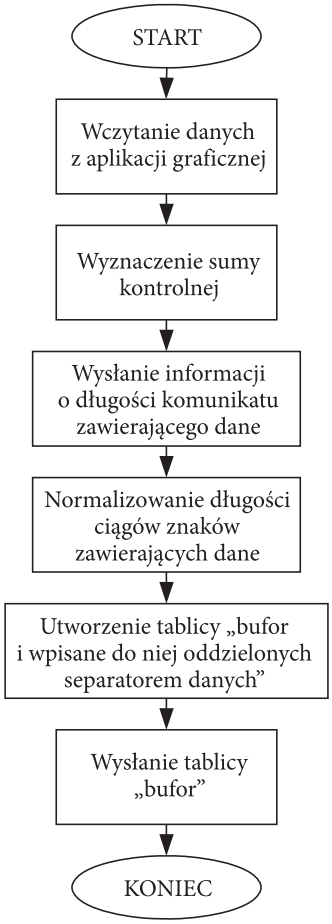
Po zakończeniu transmisji użyto instrukcji zamknięcia gniazda systemowego oraz biblioteki WinSock:

```
closesocket(s);
WSACleanup().
```

Omówione powyżej zasadnicze funkcje wykorzystywane w opracowanych aplikacjach umożliwiają komunikację pomiędzy komputerem PC a kontrolerem robota przemysłowego Kawasaki.

3.3. Aplikacja klienta

Przestawione powyżej zasady funkcjonowania oprogramowania zostały zrealizowane m.in. w oparciu o przedstawiony na rysunku 4 schemat algorytmu aplikacji klienta. Do głównych funkcji opracowanej aplikacji należy zaliczyć:
- pobranie danych z aplikacji graficznej dedykowanej procesowi paletyzacji,
- pobranie dodatkowych danych, wpisywanych przez użytkownika (adres IP komputera, numer portu, prędkość robota [%]),
- wyznaczenie sumy kontrolnej,
- rzutowanie danych ze zmiennych rzeczywistych na zmienne ciągu znaków,
- przetworzenie danych zawartych w zmiennych ciągu znaków polegające na znormalizowaniu ich długości poprzez dopisanie odpowiedniej liczby zer z lewej strony każdego z ciągów,
- zapisanie do tablicy „bufor" zmiennych w odpowiedniej kolejności,
- wysłanie informacji o ilości elementów tablicy „bufor" na zadany port robota o zadanym adresie IP,
- wysłanie tablicy „bufor" na zadany port robota o zadanym adresie IP.



Rys. 4. Algorytm aplikacji klienta

3.4. Aplikacja serwera

Aplikacja serwera (rys. 5) została opracowana w języku AS (język programowania robotów Kawasaki), a do jej głównych funkcji należy zaliczyć:

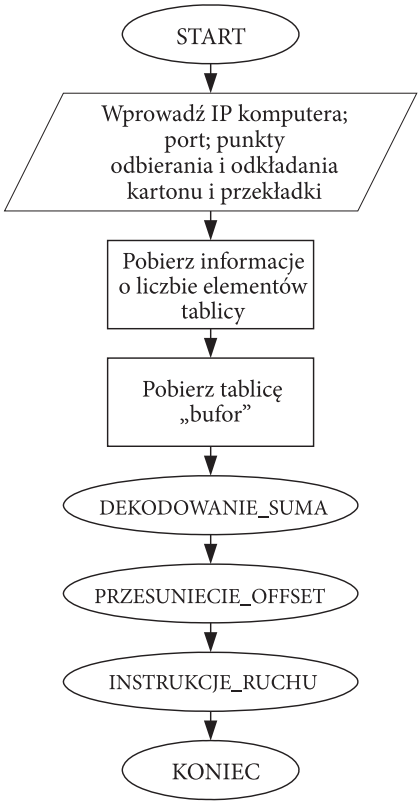
- pobranie numeru portu oraz adresu IP komputera PC, które są podawane przez użytkownika,
- odebranie informacji o ilości elementów tablicy znakowej,
- odebranie tablicy znakowej wysłanej przez klienta,
- dekodowanie elementów tablicy znakowej,
- rzutowanie zmiennych znakowych na zmienne rzeczywiste,
- sprawdzenie sumy kontrolnej,
- wyznaczenie punktu offsetowego oraz przesunięcie punktu odniesienia położenia kartonu na palecie do środka kartonu,

— wykonanie instrukcji ruchów na podstawie informacji otrzymanej od klienta.

W wyniku dekodowania (funkcja DEKODOWANIE_SUMA — rys. 5) wszystkie dane zawarte w tablicy $cnt zostają wyodrębnione i zamienione ze zmiennych tekstowych na wartości rzeczywiste, a następnie zsumowane w celu sprawdzenia sumy kontrolnej.

W wyniku działania funkcji PRZESUNIECIE_OFFSET (rys. 5) zostaje przesunięty punkt chwytania przedmiotu oraz wyznaczone są pozycje pośrednie efektora robota. Przesunięcie punktu chwytania kartonu wymagane jest z uwagi na to, że współrzędne kartonów generowane przez aplikację graficzną są pozycjami względem lewego górnego rogu kartonu. Przesunięcie to dotyczy wyłącznie kartonów obróconych o 90 stopni.

W wyniku działania funkcji INSTRUKCJE_RUCHU (rys. 5) wyznaczane są kolejne pozycje punktu odłożenia kartonu (po) i przekładki (po_pr) oraz generowana jest trajektoria ruchu robota na podstawie wyznaczonych punktów. Wyznaczane jest



Rys. 5. Uproszczony algorytm aplikacji serwera

również położenie punktu offsetowego. Wszystkie pozycje wyznaczane są w każdej iteracji, a algorytm został opracowany z uwzględnieniem optymalizacji trajektorii oraz bezpiecznego przejścia pomiędzy poszczególnymi punktami węzłowymi.

Oprogramowanie zostało przetestowane na robocie Kawasaki w firmie AB Industry w Ożarowie Mazowieckim. Test polegał na skonfigurowaniu kartonów na palecie za pomocą aplikacji graficznej i wysłaniu danych wygenerowanych na tej podstawie do kontrolera robota (rys. 6). Testy oprogramowania potwierdziły założenie, iż przeprogramowanie robota w przypadku zmiany parametrów paletyzacji jest dużo szybsze i łatwiejsze oraz że nie jest do tego działania konieczna znajomość programowania robota Kawasaki w języku AS.

Rys. 6. Testowanie opracowanych aplikacji

4. Podsumowanie

W artykule przedstawiono wybrane zagadnienia związane z problematyką wykorzystania protokołu UDP (ang. *User Datagram Protocol*) na potrzeby aplikacji do konfiguracji procesu paletyzacji kartonów dla robotów przemysłowych firmy Kawasaki. Tworzenie tego typu oprogramowania podzielono na trzy odrębne etapy:

— opracowanie właściwego środowiska z odpowiednim interfejsem użytkownika (może być zrealizowane w dowolnym języku programowania),
— opracowanie lub wykorzystanie istniejącego protokołu transmisji, umożliwiającego przesłanie niezbędnych do poprawnego funkcjonowania robota parametrów do jego kontrolera i przygotowanie aplikacji umożliwiającej przepływ danych w tym protokole,
— opracowanie elastycznej aplikacji w języku programowania danego robota i zaimplementowanie jej w jego kontrolerze;

skupiając się w szczególności, ze względu na obszerność zagadnienia, na etapie drugim (pozostałe etapy zostaną przedstawione w odrębnych artykułach).

Zaproponowane rozwiązanie umożliwia łatwe oraz intuicyjne przeprogramowanie robotów firmy Kawasaki wykonujących proces paletyzacji bez konieczności znajomości języka ich programowania. Główną zaletą takiego rozwiązania jest udostępnienie operatorowi prostego w obsłudze środowiska, skracającego znacznie czas przekonfigurowania robota w przypadku zmiany parametrów paletyzacji (wielkość palety, rozmiary kartonów, liczba warstw itd.). Należy tutaj zaznaczyć, iż wyposażenie zrobotyzowanej komory produkcyjnej do paletyzacji w dodatkowy panel operatorski pozwoli na programowanie robotów bez konieczności używania do tego celu panelu nauczania, co pozwoli na skrócenie czasu szkolenia załogi oraz zabezpieczenie robota przed niechcianymi zmianami konfiguracyjnymi w przypadku, kiedy stanowisko nie jest obsługiwane przez wykwalifikowanego inżyniera. Z drugiej strony, biorąc pod uwagę rosnącą na zautomatyzowanych liniach produkcyjnych liczbę robotów przemysłowych oraz wzrost kształconych inżynierów, należałoby zastanowić się nad implementacją tego typu oprogramowania w kontrolerze robota i zobrazowaniem aplikacji dedykowanych konkretnym procesom technologicznym na dotykowym ekranie panelu nauczania, co już dzisiaj próbują wdrażać światowi potentaci robotyki przemysłowej (m.in.: firma ABB). Rozwiązanie takie przyspieszy czas programowania robotów oraz wyeliminuje dodatkowy panel operatorski, co z całą pewnością obniży koszty całego stanowiska.

Opracowana w środowisku Borland C++ Builder 6 aplikacja komunikacyjna klienta podczas przeprowadzania testów została połączona z opracowaną wcześniej aplikacją graficzną. Dzięki temu nie ma potrzeby uruchamiania dwóch odrębnych programów na komputerze PC. Aplikacja serwera, czyli oprogramowanie kontrolera robota, została opracowana w języku AS, który jest podstawowym narzędziem programowania robotów Kawasaki.

Dodatkowo dla celów testowych opracowana została aplikacja serwera, umożliwiająca odebranie danych wygenerowanych za pomocą aplikacji graficznej dedykowanej procesowi paletyzacji na komputerze PC. Oprogramowanie to, w połączeniu z środowiskiem PC-Roset do programowania robotów Kawasaki w trybie off-line może posłużyć do testowania utworzonego oprogramowania oraz komunikacji w sieci LAN.

Oprogramowanie zostało przetestowane na robocie Kawasaki w firmie AB Industry. Test polegał na skonfigurowaniu kartonów w aplikacji graficznej, a następnie przesłaniu wygenerowanych na tej podstawie danych do kontrolera robota. Sprawdzona została zgodność danych po stronie nadawczej i odbiorczej oraz zgodność trajektorii ruchów robota z trajektorią wynikającą z konfiguracji kartonów w aplikacji graficznej. Test potwierdził prawidłowe działanie opracowanego oprogramowania.

Tworzenie nowych i rozwijanie już istniejących środowisk wspierających programowanie robotów w trybie off-line jest jednym z najprężniej rozwijających się kierunków związanych z rozwojem robotyki. Przy czym nie należy tutaj zapominać o tworzeniu i rozwijaniu specjalistycznych, dodawanych do oprogramo-

wania głównego pakietów oprogramowania, których głównym zadaniem będzie udostępnianie prostych w obsłudze interfejsów zrozumiałych dla operatorów konkretnych stanowisk produkcyjnych, a jednocześnie wprowadzających pewien poziom elastyczności, co w dzisiejszych czasach przy często zmieniającym się produkowanym asortymencie jest niezmiernie ważne.

Dziś nikogo nie trzeba przekonywać, że komputerowe wspomaganie projektowania, wytwarzania i eksploatacji zrobotyzowanych systemów produkcyjnych wpływa znacząco na:

- skrócenie czasu programowania robotów przemysłowych,
- ułatwienie tworzenia oprogramowania robotów przemysłowych,
- umożliwienie programowania robotów przez osoby nieposiadające specjalistycznej wiedzy w tym zakresie,
- zapewnienie optymalnej trajektorii ruchu robotów przemysłowych,
- zwiększenie bezpieczeństwa podczas programowania robotów.

Artykuł wpłynął do redakcji 28.07.2009 r. Zweryfikowaną wersję po recenzji otrzymano w listopadzie 2009 r.

LITERATURA

[1] *AS Language Reference Manual*, Kawasaki Heavy Industries LTD, 2007.
[2] P. Dutkiewicz, W. Wróblewski, *Modelowanie i sterowanie robotów*, PWN, Warszawa, 2003.
[3] M. Flenov, *C++ Elementarz hakera*, Helion, 2005.
[4] J. Honczarenko, *Roboty przemysłowe. Elementy i zastosowanie*, WNT, Warszawa, 1992.
[5] W. Kaczmarek, *Elementy robotyki przemysłowej*, WAT, Warszawa, 2008.
[6] *PC-Roset Ligot — Pierwsze Kroki*, ASTOR Sp. z. o.o.
[7] *Programowanie w języku AS*, ASTOR Sp. z. o.o.
[8] A. Stasiewicz, *C++ Builder. Całkiem inny świat*, Helion, 2005.
[9] *TCP/IP Communication Manual*, Kawasaki Heavy Industries LTD, 2007.
[10] K. Tomaszewski, *Roboty przemysłowe*, WNT, Warszawa, 1993.

W. KACZMAREK, M. MISIEJUK

Using UDP protocol for configuration of palletization process of packets for Kawasaki industrial robots

Abstract. In this paper, the way of using the User Datagram Protocol for industry applications was presented. The authors touched a very important problem of creating applications for Kawasaki industry robots that give the possibility of the hardware configuration of palletization processes. Moreover, in the article, the authors presented how the information from PC is sent to a robot computer.

Keywords: transmission protocols, User Datagram Protocol, industry robots, palletization process
Universal Decimal Classification: 681.51