Rozwiązania programowe w sterowaniu zaawansowanymi aplikacjami wizyjnymi

Jacek Dunaj
Przemysłowy Instytut Automatyki i Pomiarów PIAP, Al. Jerozolimskie 202, 02-486 Warszawa

Streszczenie: W artykule zawarto informacje o rozwiązaniach softwarowych zastosowanych w sterowaniu zaawansowanymi aplikacjami wizyjnymi dla przemysłu samochodowego, jakie zrealizowano w Przemysłowym Instytucie Automatyki i Pomiarów PIAP w Warszawie. Opisano kryteria wyboru oprogramowania wizyjnego, wymagany format obsługiwany przez programy wizyjne, sposób uruchamiania ich w trybie pracy automatycznej oraz definiowania sposobu interpretacji ich wyników i formatowania raportu końcowego. Przedstawiono dwie metody indywidualnego konfigurowania kontroli wizyjnych złożonych wyrobów, w zależności od zastosowanego układu sterowania. Omówiono opis współpracy aplikacji sterującej wykonywaniem kontroli wizyjnych a robotem przemysłowym zastosowanym do przemieszczania kamery w jednej z przedstawionych aplikacji.

Słowa kluczowe: kontrola wizyjna, NeuroCheck, sterownik PLC, aplikacja komputera PC, robot przemysłowy, baza danych, sterownik ODBC

1. Wprowadzenie

Systemy wizyjne na masową skalę w przemyśle zaczęto wprowadzać pod koniec XX w. wraz ze wzrostem mocy obliczeniowych procesorów. Jednym z obszarów ich zastosowania jest kontrola wyrobów po montażu końcowym. Obejmuje ona sprawdzenie obecności wybranych elementów, w tym ich położenia względem siebie, ocenę wymiarów (polegającą na wykonaniu pomiarów wizyjnych i sprawdzeniu, czy mieszczą się w granicach tolerancji), ocenę kształtu, stanu powierzchni, identyfikację napisów i porównanie ich z wzorcami, etc. Przemysłowy Instytut Automatyki i Pomiarów PIAP był jednym z pierwszych integratorów automatyki w Polsce, który wprowadził do swojej oferty aplikacje przemysłowe wyposażone w systemy wizyjne. Potrzeby potencjalnych klientów dotyczyły najczęściej kontroli złożonych wyrobów końcowych. Taka kontrola miała obejmować szereg elementów montowanych na podobnych wyrobach. Mogły się one różnić od siebie skompletowaniem, miejscem montażu poszczególnych części, a także specyficznymi cechami, jak napisy umieszczane na ich korpusach i tabliczkach znamionowych. Jednym z powtarzających się wymagań było zapewnienie, że aplikacja wizyjna po niewielkich modyfikacjach będzie mogła sprawdzać wyroby, które dopiero w przyszłości będą wprowadzone do produkcji.

Autor korespondujący:
Jacek Dunaj, jdunaj@piap.pl

Artykuł recenzowany
nadesłany 28.04.2015 r., przyjęty do druku 10.08.2015 r.

2. Wybór oprogramowania wizyjnego

Bez względu na platformę sprzętową aplikacje wizyjne są zazwyczaj tworzone na komputerze PC za pomocą odpowiedniego oprogramowania. W przypadku kamer inteligentnych aplikacja jest wykonywana przez procesor kamery. Wówczas oprogramowanie komputera PC do tworzenia aplikacji wizyjnej zapewnia także pełną obsługę kamery – uruchamianie, zatrzymywanie i debugowanie programu, podgląd stanu wejść i sterowanie wyjściami kamery, ustawianie zmiennych systemowych (np. nastawy zegara i kalendarza w kamerze), adresację i parametryzację portów transmisyjnych, etc. W zależności od wyposażenia kamery, czynności te są wykonywane za pośrednictwem standardowych interfejsów komunikacyjnych RS-232, Ethernet, USB, IEEE 1394.

Innym rozwiązaniem jest realizacja aplikacji przez komputer PC pod kontrolą oprogramowania wizyjnego. W takim przypadku komputer PC musi być wyposażony w karty akwizycji obrazu do obsługi kamer lub kamery są sprzężone z komputerem za pomocą standardowego interfejsu zapewniającego transmisję obrazów w czasie rzeczywistym.

Oba rozwiązania mają swoje wady i zalety. Drugi wariant jest bardziej uniwersalny ze względu na dodatkowe możliwości sterowania innymi urządzeniami automatyki i wykorzystaniem oprogramowania biurowego (bazy danych, arkusze kalkulacyjne) i był częściej wybierany do złożonych zastosowań komercyjnych.

Podstawowym działaniem programów kontroli wizyjnej jest analiza obrazu obiektu. Obraz może być ładowany do pamięci komputera bezpośrednio z kamery, ale jego źródłem może być także bitmapa odczytana z pliku .bmp. W pierwszym przypadku oprogramowanie wizyjne musi współdziałać z kartą akwizycji obrazu lub znać format danych wymienianych z kamerą za pomocą standardowego intrerfejsu. Z kolei możliwość wykorzystania obrazu odczytanego z pliku .bmp jest cenną cechą pozwalającą na szybsze tworzenie i modyfikowanie programu

bez fizycznej obecności kamery. Bardzo istotne jest także, w jaki sposób można sterować oprogramowaniem wizyjnym realizującym aplikacje. Oprogramowanie wizyjne musi zapewnić możliwość automatycznego uruchamiania różnych programów aplikacyjnych bez bezpośredniego udziału operatora. Znane są firmy, które specjalizują się w opracowywaniu uniwersalnego oprogramowania, mają doświadczenie i oferują swoje sprawdzone produkty na rynku komercyjnym. Zadaniem inżynierów PIAP jest efektywne wykorzystanie funkcji wybranego oprogramowania, opracowanie schematu końcowej aplikacji możliwej do zastosowania w różnych stanowiskach kontroli wizyjnej.

3. Oprogramowanie wizyjne NeuroCheck

Do tworzenia, edycji, a następnie realizacji programów kontroli wizyjnych kilku wdrożonych aplikacji przemysłowych PIAP wybrano pakiet oprogramowania NeuroCheck niemieckiej firmy NeuroCheck GmbH, pracujący pod kontrolą systemu operacyjnego Windows. W miarę wprowadzania kolejnych wersji systemu Windows (95, 98, NT, 2000, XP, 7) pakiet ten przystosowywano do współpracy z nimi. Swoje oprogramowanie firma NeuroCheck GmbH udostępnia bezpłatnie. W pełnej wersji pakiet ten można zainstalować na dowolnym komputerze PC, a jego aktualne możliwości funkcjonalne zależą od klucza sprzętowego (ang. *dongle*) umieszczonego w wolnym porcie USB lub LPT. W najprostszej wersji demo (bez klucza) zablokowane są wszystkie elementy sprzętowe związane z obsługą kamer i kart We/Wy. W tej wersji aplikacja wizyjna może być tworzona tylko na podstawie bitmapy odczytanej z pliku .bmp i nie można jej uruchamiać w trybie automatycznym, ale tylko przy pomocy myszy i klawiatury komputera. Tym niemniej wersja demo ma większość funkcji edycyjnych programu i spełnia istotną rolę edukacyjną. Przystosowanie gotowej aplikacji do współpracy z kamerą wymaga tylko zmiany źródła obrazu, a więc modyfikacji pojedynczego parametru odpowiedniej instrukcji.

Program kontroli wizyjnej (ang. *check routine*) realizowany przez pakiet NeuroCheck składa się co najmniej z jednego modułu kontroli (ang. *check*). Pojedynczy moduł kontroli, będący rodzajem podprogramu, zawiera szereg makroinstrukcji (ang. *check functions*), które realizują transfer obrazu z kamery do pamięci komputera, filtrują go, wyznaczają fragmenty obrazu do analizy, przeprowadzają analizę i mogą obsługiwać sygnały dwustanowe – jeśli komputer wyposażono w kartę We/Wy obsługiwaną przez oprogramowanie NeuroCheck. Zainicjowanie programu kontroli wizyjnej zawsze wiąże się z uruchomieniem pierwszego modułu, a wykonanie kolejnych zależy od ustawionych warunków przejścia w module poprzedzającym (np. moduł nr 2 będzie wykonany, jeśli kontrola wizyjna w module nr 1 przyniosła wynik negatywny). Programy kontroli wizyjnej są zapisywane w plikach o rozszerzeniu .CHR, i dodatkowo, podczas określania ich właściwości przy pomocy NeuroCheck-a, mogą być oznaczane numerami. Obie te cechy zamiennie można wykorzystywać do ich wybierania w trybie pracy automatycznej.

Wśród dostępnych okien dialogowych oprogramowania NeuroCheck są okna opisane jako Remote control (rys. 1). Pozwalają one określić, w jaki sposób NeuroCheck będzie sterowany w trybie pracy automatycznej, a także w jaki sposób program ten będzie przekazywał wyniki aplikacji wizyjnych.

Sterowanie programem NeuroCheck w trybie pracy automatycznej można realizować na dwa sposoby:
- z zewnętrznego źródła, np. sterownika PLC (opcja „Program processes input signals") za pomocą sygnałów dwustanowych, transmisji za pośrednictwem portu szeregowego lub karty sieciowej,
- przez nadrzędną aplikację pracującą współbieżnie na komputerze wizyjnym (opcja „Program is remote-controlled by master").

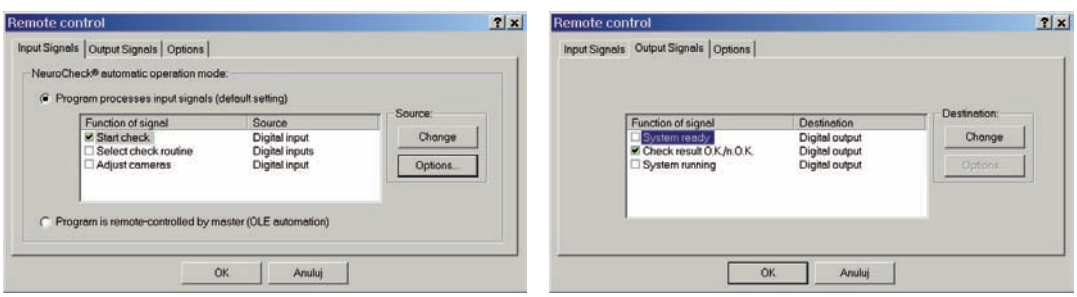
W realizowanych aplikacjach wykorzystano oba sposoby.

3.1. Sterowanie programem NeuroCheck za pomocą sygnałów dwustanowych

Sterowanie za pomocą sygnałów dwustanowych jest dostępne, gdy komputer wizyjny jest wyposażony w kartę We/Wy obsługiwaną przez NeuroCheck (np. kartę DT2820 firmy Data Translation, OPTOIO-PCI16 firmy Wasco). Wymaga to uprzedniego zadeklarowania (rys. 1) następujących parametrów:
- nazwy katalogu w pamięci masowej komputera wizyjnego zawierającego pliki .CHR z aplikacjami wizyjnymi, każdej aplikacji podczas jej tworzenia i edycji należy przypisać unikatowy numer służący do jej wyboru,
- określenia którymi dwustanowymi sygnałami wejściowymi karty We/Wy komputera współpracujący sterownik PLC będzie mógł wybierać programy wizyjne do wykonania (sygnały „select check routine") oraz którym sygnałem inicjuje wykonanie wybranego programu (sygnał „start check"),
- określenia którymi dwustanowymi sygnałami wyjściowymi karty We/Wy komputera program NeuroCheck będzie sygnalizował swoje załączenie i ustawienie w tryb pracy automatycznej (sygnał „system running"), potwierdzenie wykonania programu kontroli wizyjnej (sygnał „system ready") i jego wynik (sygnały Check result OK/Not OK).

Sterowanie realizowane jest w postaci opisanej sekwencji działań. Sterownik PLC odczytuje stan sygnału system running karty We/Wy informujący o uruchomieniu programu NeuroCheck i ustawieniu go w tryb pracy automatycznej. Jeśli uzyska potwierdzenie, to wysterowuje sygnały wskazujące numer programu wizyjnego do wykonania (sygnały select check routine), a następnie sygnałem start check inicjuje jego wykonanie. Program NeuroCheck wykonuje wybrany program kontroli i w zależności od wyniku wysterowuje jeden z dwóch sygnałów Check result OK lub Check result Not OK, następnie sygnałem system ready potwierdza zrealizowanie programu i ważność wyniku.

Sterowanie za pomocą sygnałów dwustanowych ma istotne ograniczenie – wynik aplikacji wizyjnej ma postać dwuwartościową i nie ma prostego sposobu przekazania do współpracującego sterownika wartości wykonanego pomiaru wizyjnego lub treści odczytanego napisu. Ten rodzaj sterowania dobrze sprawdza się w aplikacjach wymagających sprawdzenia kompletności montażu, ale bez konieczności wnikania we właściwości kontrolowanych elementów.



Rys. 1. Okna dialogowe programu NeuroCheck do konfigurowania sterowania i przekazywania wyników w trybie pracy automatycznej

Fig. 1. NeuroCheck's dialog boxes to configure the vision control and presentation of results in the automatic mode

3.2. Sterowanie programem NeuroCheck przez nadrzędną aplikcję komputera wizyjnego

Podczas pracy w trybie automatycznym programem NeuroCheck można sterować z poziomu aplikacji pracującej współbieżnie na komputerze wizyjnym. Jej działanie opiera się na wykorzystaniu oprogramowania NeuroCheck jako serwera aplikacji OLE (ang. Object Linking and Embedding). Technika OLE [13] polega na wymianie obiektów między aplikacjami działającymi w środowisku MS Windows. Typowym przykładem jest wklejanie obiektów utworzonych za pomocą arkuszy kalkulacyjnych do dokumentów edytorów tekstów. W automatyce pojęcie OLE uległo rozszerzeniu i obejmuje dostęp do określonych funkcji jednej aplikacji (serwera) przez inną aplikację (klienta). W opisywanym przypadku aplikacja nadrzędna (klient) steruje wybieraniem programów wizyjnych przez NeuroCheck (serwer), inicjuje ich wykonanie i odczytuje wyniki. Aplikacja może też realizować wszystkie zadania związane ze sterowaniem stanowiskiem wizyjnym – współpracować z transporterami, robotem, stołem obrotowym, czytnikiem kodu kreskowego, wykonywać obsługę baz danych, generować i drukować raport końcowy etc.

W sterowaniu w trybie OLE aplikacja nadrzędna (klient) inicjuje połączenie z NeuroCheck (serwer), następnie sprawdza klucz licencyjny oprogramowania. Sprawdzenie ma na celu kontrolę uprawnień (praca w trybie OLE jest możliwa tylko dla klucza full-version lub run-time-version) oraz potwierdzenie, czy NeuroCheck został skonfigurowany jako serwer OLE (opcja „Program is remote-controlled by master" (rys. 1). Następnie aplikacja wykorzystując udostępnione przez serwer NeuroCheck metody, może wskazać plik o rozszerzeniu .CHR zawierający aplikacyjny program wizyjny i zainicjować jego wykonanie. Przed wykonaniem tego programu możliwe jest zdalne parametryzowanie niektórych jego makroinstrukcji (ang. check functions), a po jego wykonaniu – odczyt wyników generowanych przez inne makroinstrukcje. Aplikacja nadrzędna ma więc dostęp nie tylko do globalnego wyniku wykonania programu wizyjnego, ale także do wyników wybranych makroinstrukcji. Przedstawiony mechanizm sterowania programem NeuroCheck w trybie OLE opisano w [8]. Aplikację nadrzędną można utworzyć w języku C/C++, Visual Basic lub Borland Delphi, ale ze względu na efektywność kodu zalecany jest język C/C++. Dostarczana jest też biblioteka Ncheck.tlb zawierającą informacje o typach danych, funkcjach i klasach udostępnianych przez serwer NeuroCheck.

4. Koncepcja kontroli wizyjnej złożonych wyrobów

Wspomniano uprzednio, że zapytania ofertowe dotyczące budowy stanowisk kontroli wizyjnej dotyczyły najczęściej sprawdzania wielu elementów, które w różnych konfiguracjach mogły być montowane w podobnych wyrobach. Naturalnym rozwiązaniem byłoby opracowanie tylu aplikacji wizyjnych, ile jest sprawdzanych wyrobów, każda do kontroli innego wyrobu. Jednakże w tym rozwiązaniu pojawia się pewien problem – zmiana właściwości wybranego elementu wymusza modyfikację wszystkich aplikacji, które sprawdzają ten element. Ponadto te same fragmenty oprogramowania będą powtarzać się w wielu programach wizyjnych, co utrudnia wprowadzanie ewentualnych poprawek. Dlatego przyjęto koncepcję modułowości programów kontroli wizyjnej polegającą na tym, że pojedyncza aplikacja NeuroCheck sprawdza tylko jeden wybrany element wyrobu i niezależnie od pozostałych określa, czy kontrolowany detal spełnia zadane kryteria oceny. O tym, które programy mają być wykonywane decyduje nie sztywna sekwencja zapamiętana w komputerze wizyjnym, ale sekwencja narzucona przez współpracujący sterownik PLC lub nadrzędną aplikację komputera wizyjnego. W drugim przypadku można też ustalać kolejność wykonywania poszczególnych programów wizyjnych.

5. Realizacja kontroli wizyjnej za pomocą sterownika PLC

W przypadku sterowania kontrolą wizyjną przez współpracujący sterownik PLC przyjmuje się założenie, że na stanowisku można maksymalnie sprawdzać m różnych wyrobów, a liczba wszystkich elementów (detali) podlegających kontroli wynosi n, przy czym nie każdy element występuje na danym wyrobie. Należy zatem uruchomić n różnych programów wizyjnych, każdy do sprawdzania pojedynczego elementu i opracować schemat sterowania wyborem tych programów do kontroli całego wyrobu.

W tym celu w pamięci sterownika PLC zadelarowano dwuwymiarową tablicę

```
bool Wyroby[m][n];
```

uporządkowaną według schematu:

	element 1:	element 2:	…….	element *n*:
wyrób 1:	*true/false*		…….	*true/false*
wyrób 2:	*true/false*		…….	*true/false*
			…….	
wyrób *m*:	*true/false*	*true/false*	…….	*true/false*

Jeśli element tablicy `Wyroby[i][j]` ma wartość *true*, to oznacza, że podczas sprawdzania *i*-tego wyrobu ma zostać wykonanana kontrola wizyjna *j*-tego elementu (detalu), w przeciwnym razie (*false*) kontrola tego elementu będzie pominięta.

Kolejna, tym razem jednowymiarowa tablica `Elementy[n]` zawiera dane niezbędne do wykonania kontroli wizyjnej poszczególnych elementów (detali). Każdy element tej tablicy przechowuje informację opisaną strukturą:

```
struct element {
short          NumerProgramu;
char           *Opis
short          SterowanieOswietlaczami;
}
```

gdzie:

`NumerProgramu` – kombinacja wyjściowych sygnałów dwustanowych służących do wyboru programu NeuroCheck sprawdzającego wybrany element,

`*Opis` – wskaźnik (ang. *pointer*) do tekstu komentarza wyświetlanego na monitorze podłączonym do sterowanika podczas kontroli wizyjnej wybranego elementu,

`SterowanieOswietlaczami` – kombinacja wyjściowych sygnałów dwustanowych służących do wysterowania poszczególnych oświetlaczy przed realizacją kontroli wizyjnej wybranego elementu.

Kontrola wizyjna wyrobu rozpoczyna się od odczytania z tablicy `Wyroby[][]` informacji, które elementy mają być sprawdzane, a następnie na kolejnym wykonywaniu programów sprawdzających wybrane detale. Wyniki kontroli poszczególnych elementów są zapamiętywane w pamięci sterownika, a po ich zakończeniu służą do określenia globalnego wyniku kontroli całego wyrobu.

Pierwszą złożoną aplikacją zrealizowaną przez Przemysłowy Instytut Automatyki i Pomiarów PIAP i pracującą według przedstawionego schematu było stanowisko do kontroli wizyjnej metalowych stelaży siedzeń samochodowych wykonane dla zakładów „Faurecia" w Grójcu [1, 2]. Było to stanowisko złożone z 12 nieruchomych kamer analogowych z oświetlaczami pierścieniowymi zamocowanymi wewnątrz światłoszczelnego kiosku. Po umieszczeniu stelaża siedzenia w specjalnym uchwycie operatorzy stanowiska wskazywali typ siedzenia (wyrobu) za pomocą jednego z 20 przycisków wyboru, a następnie inicjowali przemieszczenie uchwytu do środka i wykonanie kontroli wizyjnej. Zastosowanym sterownikiem PLC był sterownik PS HC20 firmy

Festo, o architekturze komputera PC pracujący pod kontrolą systemu operacyjnego MS DOS. Jego aplikacja, napisana w języku C i częściowo asemblera procesora 86, sterowała wykonywaniem kontroli wizyjnych siedzeń oraz działaniem samego stanowiska, tzn. obsługą transportera, przycisków, lampek sygnalizacyjnych i drukarki tamponowej. Aplikację wyposażono w interfejs technologa, który pozwalał na modyfikację i zapamiętywanie w pamięci masowej zawartości tablic `Wyroby[m][n]` i `Elementy[n]`, a w trakcie wykonywania kontroli wizyjnej – wyświetlał informację o wynikach oceny poszczególnych detali. Ze sterownikiem współpracował komputer przemysłowy PC firmy Advantech, pracujący pod kontrolą systemu operacyjnego Windows z osadzonym oprogramowaniem NeuroCheck. Komputer ten wyposażono w trzy karty akwizycji obrazu DT3155 PCI, każda do obsługi czterech kamer i pojedynczą kartę We/Wy dwustanowych DT2820 (obie produkcji Data Translation).

6. Realizacja kontroli wizyjnej za pomocą nadrzędnej aplikacji komputera PC

Kontrola wizyjna wykonywana dla zakładów Faurecia Automotive Polska w Grójcu odpowiadała na pytanie, czy dany element zamontowano czy przeoczono jego montaż. Programy wizyjne nie wnikały ani we właściwości elementu, ani w dużej mierze w prawidłowość montażu. Przyjętym i uzasadnionym założeniem było to, że we wszystkich rodzajach stelaży dany detal (jeśli występował) mocowano zawsze w tym samym miejscu ramy, a więc do sprawdzenia jego obecności niezależnie od wybranego stelaża można było wykorzystać tę samą kamerę.

Większe problemy wystąpiły podczas realizacji aplikacji wizyjnej dla przemysłu maszynowego, która miała sprawdzać skrzynie biegów produkowane w zakładach Eaton w Tczewie. W tym przypadku kontrolowano nie tylko obecność różnych elementów, ale także ich wymiary i zgodność napisów grawerowanych na poszczególnych częściach sprawdzanych skrzyń. Od początku było wiadomo, że każda odmiana skrzyni biegów różni się od pozostałych nie tylko liczbą sprawdzanych detali, ale także kolejnością wykonywania elementarnych kontroli. Nawet sprawdzanie identycznych elementów, jak napisy na tabliczkach znamionowych, były różnie realizowane w zależności od odmiany skrzyni, nie tylko ze względu na różnice w ich lokalizacji, ale także ze względu na to, że inne elementy skrzyni mogły je zasłaniać lub powodować dodatkowe refleksy od zewnętrznych źródeł światła. Sprawdzanym elementem mogła być obecność tabliczki znamionowej, zgodność niektórych napisów na tabliczce i na odlewie korpusu skrzyni, pomiar średnicy pokrywy wałka, obecność śruby z otworem etc. W niektórych przypadkach programy kontroli wizyjnej dawały prostą odpowiedź „jest detal" lub „brak detalu", w innych zwracały odczytany napis, jeszcze w innych – wartość liczbową pomiaru wizyjnego. Jeśli program wizyjny identyfikował napis, to nie wystarczało wyświetlenie informacji o tym, co odczytano. Pełna informacja oznaczała ocenę prawdopodobieństwa poprawnej identyfikacji i porównanie z tym, co faktycznie powinno zostać nabite na elementach skrzyni biegów (np. numer wałka napędowego).

Firma Eaton zastrzegła sobie, aby aplikacja sterująca stanowiskiem była w pełni uniwersalna, tzn. aby bez wprowadzania do niej dodatkowych modyfikacji możliwa była kontrola wizyjna skrzyń, które dopiero wejdą do produkcji, a parametryzowanie tej kontroli mogło odbywać się we własnym zakresie. Dodatkowym żądaniem było pełne archiwizowanie wyników kontroli wraz ze zdjęciami sprawdzanych elementów.

Różnorodność odmian skrzyń biegów i rozmieszczenie sprawdzanych elementów sprawiło, że projektując stanowisko zrezygnowano z wielokamerowego systemu (stosowanego w aplikacji dla Faurecii Grójec), na rzecz pojedynczej kamery zamocowanej na ramieniu robota. Ponieważ mimo użycia manipulatora część elementów skrzyń nadal pozostawała poza zasięgiem kamery, dodatkowo zastosowano czteropozycyjny stół obrotowy, na którym osadzano sprawdzaną skrzynię. Zasadniczej zmianie musiała ulec koncepcja sterowania aplikami wizyjnymi, ponieważ wynik kontroli większości elementów nie miał już postaci tylko dwuwartościowej Check OK/ Check Not OK. Konieczne było opracowanie sposobu definiowania algorytmu wyznaczającego rezultat kontroli pojedynczego elementu generowany na podstawie wyników działania kilku programów wizyjnych. W efekcie opracowano schemat aplikacji, który z pewnymi modyfikacjami zastosowano też w stanowiskach pomiarowych korpusów świec zapłonowych wykonanych dla firmy Iskra Kielce, fabryki poszyć siedzeń samochodowych Lear w Tychach (tego projektu ostatecznie nie sfinalizowano, ale z powodów nie dotyczących systemu wizyjnego) oraz stanowiskach wystawowych promujących PIAP.

Aby aplikacja nadrzędna funkcjonowała niezależnie od typu wyrobu (skrzyni biegów), a jednocześnie, aby samo stanowisko było łatwo adaptowalne do kontroli nowych skrzyń bez potrzeby modyfikacji samej aplikacji zdecydowano, że wszelkie informacje niezbędne do przeprowadzenia pełnej kontroli skrzyni będą zawarte w bazie danych oraz w dodatkowym pliku tekstowym opisującym sposób interpretacji wyników i postać raportu końcowego. Narzucono także ograniczenia dotyczące formatu programów kontroli wizyjnej w zależności od właściwości elementu, który miały sprawdzać.

6.1. Formaty programów kontroli wizyjnej – założenia

Zasady stosowania NeuroCheck jako serwera aplikacji OLE zakładają udostępnianie szeregu informacji, które pozwalają m.in. na odczytanie rezultatów wykonania niektórych makroinstrukcji. W tym celu aplikacja klienta NeuroCheck musi mieć informację o położeniu instrukcji w programie oraz znać format informacji, którą zamierza odczytać. Dla ujednolicenia odwołań do poszczególnych programów kontroli wizyjnej z poziomu aplikacji OLE przyjęto podział na kilka kategorii programów oraz narzucono pewne ograniczenia co do ich formatu.

6.1.1. Programy kontroli obecności detalu

Kategoria obejmująca programy wizyjne sprawdzające obecność lub prawidłowość zamocowania pojedynczego detalu. Programy te zwracają jedynie wartość logiczną Check OK/Check Not OK, określającą czy zidentyfikowano obecność detalu lub czy zidentyfikowano jego prawidłowe położenie. Z punktu widzenia aplikacji nadrzędnej nie jest istotne, ile modułów kontroli (ang. *check*) zawiera program kontroli wizyjnej (ang. *check routine*), jak są wykonywane i jakie instrukcje zawierają. Aplikacja nadrzędna do dalszych rozważań bierze pod uwagę tylko globalny wynik generowany przez program, nie odczytuje rezultatu wykonania pojedynczego modułu kontroli, ani rezultatów wykonania pojedynczych makroinstrukcji (ang. *check function*).

6.1.2. Programy identyfikacji napisów

Kategoria obejmuje programy wizyjnej identyfikacji napisów. Programy te zwracają trzy różne informacje: liczbę znaków zidentyfikowanych, znaki zidentyfikowane oraz logiczny wynik identyfikacji Check OK/Check Not OK. Ostatnia wielkość określa, czy prawdopodobieństwo poprawnej identyfikacji znaków jest poniżej lub powyżej stałej wartości ustalonej w programie wizyjnym. Można przyjąć, że cyfrę „8" odczytano poprawnie, jeśli ustalono, że prawdopodobieństwo poprawnej identyfikacji wynosi 90%. Jeśli próg ten wynosi tylko 20%, to za cyfrę „8" program wizyjny może przyjąć zarówno cyfrę „8", jak i „3". Każda z wartości zwracanych przez programy tej kategorii musi podlegać weryfikacji przez aplikację nadrzędną. Przyjęto zatem następujące założenia dotyczące formatu tej kategorii programów:

1. Program kontroli wizyjnej może być złożony z dowolnej liczby modułów kontroli.

2. Szczegółowe informacje dotyczące liczby odczytanych znaków oraz same zidentyfikowane znaki uzyskuje się odczytując wyniki wykonania makroinstrukcji zawartych w ostatnim module kontroli programu wizyjnego.
3. Liczbę zidentyfikowanych znaków obligatoryjnie podaje pierwsza makroinstrukcja Count Rols zawarta w ostatnim module kontroli programu wizyjnego. Jeśli jest to niezbędne, wewnątrz tego modułu mogą występować inne makroinstrukcje Count Rols, ale ich rezultaty nie uwagę są istotne dla aplikacji nadrzędnej. Wynik Count Rols jest niezbędny dla ustalenia zakresu pętli programowej odczytującej zidentyfikowane znaki. W makroinstrukcji Count Rols trzeba zablokować opcję wystawiania końcowej oceny wykonania programu kontroli wizyjnej (tzn. by nie mogła wystawiać informacji Check OK/ Check Not OK, tym samym przedwcześnie kończyć wykonywanie całego programu kontroli wizyjnej).
4. Jako ostatnia makroinstrukcja wewnątrz ostatniego modułu kontroli musi wystąpić makroinstrukcja Evaluate Classes. Jej zadaniem jest przekazanie do nadrzędnej aplikacji OLE informacji o zidentyfikowanych klasach oraz prawdopodobieństwie ich wystąpienia. Obligatoryjnie przyjmuje się, że nazwa klasy (ciąg znakowy odpowiadający nazwie klasy) składa się tylko z jednego znaku, który odpowiada identyfikowanemu znakowi (np. klasa o nazwie „8" odpowiada znakowi 8). Makroinstrukcja ta ma odblokowaną opcję wystawiania oceny wykonania programu kontroli wizyjnej. Jeśli jest to niezbędne, wewnątrz tego modułu kontroli mogą wcześniej występować inne instrukcje Evaluate Classes, ale rezultatów ich wykonania aplikacja nadrzędna nie bierze pod uwagę.

Aplikacja nadrzędna ten rodzaj kontroli wizyjnej obsługuje w następujący sposób:
— z bazy danych odczytuje, ile znaków powinno być zidentyfikowanych,
— określa, ile modułów kontroli zawiera program kontroli wizyjnej, aby wyznaczyć indeks ostatniego modułu,
— określa, ile makroinstrukcji zawiera ostatni moduł w programie kontroli wizyjnej, aby wyznaczyć indeks ostatniej makroinstrukcji (z założenia jest to makroinstrukcja Evaluate Classes),
— w ostatnim module kontroli odszukuje pierwszą w kolejności makroinstrukcję Count Rols, a następnie jako rezultat jej wykonania odczytuje, ile znaków faktycznie zidentyfikowano. Jeśli wartość ta nie zgadza się z odczytem z bazy danych, rezultat działania programu kontroli wizyjnej będzie Check Not OK, ale to nie zakończy obsługi programu,
— odczytując rezultat wykonania ostatniej makroinstrukcji Evaluate Classes ustala, jaki ciąg znaków faktycznie został odczytany (na podstawie zidentyfikowanych klas),
— odczytując wynik logiczny wykonania całego programu kontroli wizyjnej (Check OK/ Check Not OK) aplikacja nadrzędna ustala, czy identyfikacja znaków jest powyżej, czy poniżej ustalonego progu pewności.

Rezultatem wykonania programu identyfikacji napisu jest zatem informacja:
— ile znaków zidentyfikowano (wynik makroinstrukcji Count Rols),
— jakie znaki zidentyfikowano (wynik makroinstrukcji Evaluate Classes),
— jaki jest wynik logiczny wykonania całego programu kontroli wizyjnej (Check OK/ Check Not OK),
— jaka jest ewentualna przyczyna rezultatu Check Not OK.

W dalszej części aplikacja nadrzędna określa, czy zidentyfikowane znaki są tymi, jakie powinny być nabite na detalu (np. przez porównanie z napisem wzorcowym z bazy danych lub z napisem odczytanym czytnikiem kodu kreskowego).

6.1.3. Programy wykonujące pomiary wizyjne

Kategoria obejmuje programy wykonujące pomiary wizyjne, np. pomiary średnic wałków. Zwracają one do aplikacji nadrzędnej dwie różne informacje: wartość zmierzonej wielkości (liczba zmiennoprzecinkowa) oraz logiczny rezultat wykonania programu Check OK/ Check Not OK. W drugim przypadku wartość Check OK jest zwracana, jeśli program wizyjny ocenił, że zmierzona wartość mieści się w granicach tolerancji określonych wewnątrz tego programu (nie ma potrzeby, aby aplikacja nadrzędna znała granice tolerancji). Przyjęto następujące założenia dotyczące formatu tej kategorii programów:
1. Pojedynczy program kontroli wizyjnej wykonuje pomiar jednej wielkości (np. średnicy wałka w jednym miejscu).
2. Program kontroli wizyjnej może być złożony z dowolnej liczby modułów kontroli.
3. Szczegółowe informacje dotyczące wartości pomiaru uzyskuje się odczytując wyniki wykonania makroinstrukcji zawartych w ostatnim module kontroli programu wizyjnego.
4. Ostatnią makroinstrukcją wewnątrz ostatniego modułu kontroli obligatoryjnie musi być makroinstrukcja Check Allowances. Przekazuje ona do aplikacji nadrzędnej informację o wyniku pomiaru.
5. Jeśli jest to niezbędne, wewnątrz ostatniego modułu kontroli wcześniej mogą występować inne instrukcje Check Allowances, ale ich rezultaty aplikacja nadrzędna nie bierze pod uwagę.

Aplikacja nadrzędna ten rodzaj kontroli wizyjnej obsługuje w następujący sposób:
— określa ile modułów kontroli zawiera program wizyjny, tak aby wyznaczyć indeks ostatniego modułu,
— określa ile makroinstrukcji zawiera ostatni moduł kontroli, tak aby wyznaczyć indeks ostatniej makroinstrukcji (z założenia jest to makroinstrukcja Check Allowances),
— odczytując rezultat wykonania ostatniej makroinstrukcji Check Allowances, w ostatnim module kontroli określa wynik pomiaru,
— odczytując wynik logiczny wykonania całego programu wizyjnego (Check OK/ Check Not OK) aplikacja nadrzędna ustala, czy pomiar mieści się w przedziale tolerancji (na podstawie rezultatu Check OK lub Check Not OK wystawianego przez Check Allowances.

Rezultatem wykonania programu jest wynik pomiaru wizyjnego i informacja, czy wynik mieści się w przedziale tolerancji.

Przynależność do każdej z wymienionych kategorii programów wizyjnych jednoznacznie narzuca sposób obsługi programu przez aplikację nadrzędną.

6.2. Atrybuty programów wizyjnych

Każdy program kontroli wizyjnej musi mieć jednoznacznie określone atrybuty pozwalające aplikacji nadrzędnej zlokalizować ten program w pamięci masowej, wskazać go do wykonania przez oprogramowanie NeuroCheck, wybrać sposób odczytu wyników oraz przygotować stanowisko do jego wykonania. Atrybutami tymi są:

1. Nazwa programu wizyjnego. Atrybut określa lokalizację pliku, tzn. urządzenie, katalog i nazwę pliku zawierającego program kontroli wizyjnej realizowany przez pakiet NeuroCheck.

2. Opis programu wizyjnego. Atrybut określa komunikat wyświetlany przez aplikację nadrzędną podczas realizacji programu. Ważną funkcję atrybut ten pełni przy archiwizowaniu wyników działania programu. Firma Eaton zastrzegła, że archiwizowane mają być nie tylko wyniki kontroli wizyjnych, ale i zdjęcia sprawdzanych elementów. W tym przypadku opis programu jest wykorzystywany przez aplikację nadrzędną do generowania nazwy pliku zawierającego zdjęcie sprawdzanego elementu.

3. Typ programu wizyjnego. Atrybut określa przynależność do jednej z kategorii programów wizyjnych. Przyjęto, że wartość 0 oznacza kategorię programów kontroli obecności detalu, 1 – kategorię programów identyfikacji napisów, 2 – kategorię programów wykonujących pomiary wizyjne.

4. Liczba znaków. Atrybut stosowany tylko do kategorii programów identyfikacji napisów. Jedną z informacji zwracanych

przez tę kategorię programów jest liczba zidentyfikowanych znaków. Jeśli program wizyjny ocenił, że prawdopodobieństwo poprawnej identyfikacji jest powyżej ustalonego progu (wynik Check OK), to wcale nie oznacza, że odczytany napis jest poprawny. Drugim kryterium oceny poprawności odczytu jest weryfikacja liczby zidentyfikowanych znaków z liczbą znaków, jakie faktycznie powinny być wygrawerowane na elemencie. Właśnie tę liczbę znaków określa ten atrybut.

5. Czas ekspozycji kamery. W aplikacji do kontroli wizyjnej skrzyń biegów zastosowano wysokiej klasy cyfrową, monochromatyczną kamerę wizyjną A101p firmy Basler o rozdzielczości 1300 px × 1030 px. Jest ona wyposażona w obiektyw, w którym sterowanie nastawami ostrości i przysłony można realizować tylko ręcznie, a więc w sposób niedostępny w automatycznym trybie pracy stanowiska. Ze względu na różne rozmiary sprawdzanych elementów różne są odległości z których są one „fotografowane" przez kamerę. Dlatego jedyną metodą na uzyskanie poprawnego obrazu jest sterowanie czasem ekspozycji kamery i źródłami światła. Czas ekspozycji kamery można ustawiać w zakresie od 140 µs do 131 070 µs, a programowanie odbywa się za pomocą kanału RS-232 komputera wizyjnego. Atrybut ten określa czas ekspozycji kamery stosowany podczas realizacji programu kontroli wizyjnej.

6. Oświetlacze. Aplikacja nadrzędna stanowiska do kontroli wizyjnej skrzyń biegów umożliwia sterowanie czterema różnymi oświetlaczami. Dwa oświetlacze pierścieniowe zamontowano współosiowo z obiektywem kamery, a jeden oświetlacz typu *backlight* zamocowano na podstawie stołu obrotowego (czwarty oświetlacz jest rezerwowy). Atrybut ten określa, jakie oświetlacze mają być zapalane podczas realizacji programu kontroli wizyjnej.

7. Położenie stołu obrotowego. Każda skrzynia biegów w czasie kontroli wizyjnych jest umieszczona w specjalnym uchwycie osadzonym w stałym położeniu względem blatu stołu obrotowego. Ze względu na dostęp kamerą do poszczególnych elementów każdej skrzyni, blat stołu można ustawiać w czterech stabilnych pozycjach względem jego podstawy. Atrybut określa, w jakim położeniu ma być ustawiony blat stołu względem jego podstawy podczas realizacji programu kontroli wizyjnej.

Aplikacja nadrzędna sterująca kontrolą wizyjną nie potrzebuje żadnej informacji o tym, co faktycznie dany program wizyjny sprawdza (opis programu jest informacją dla operatora). Istotne jest, jaki program wizyjny ma zostać wykonany przez pakiet NeuroCheck (atrybut Nazwa programu), jak wysterować poszczególne elementy stanowiska podczas jego wykonywania (atrybuty Czas ekspozycji kamery, Oświetlacze, Położenie stołu obrotowego) oraz jak odczytać i zinterpretować rezultaty wykonania programu (atrybuty Typ programu i Liczba znaków).

6.3. Oznaczenia programów wizyjnych

Aplikację nadrzędną i niezbędną bazę danych zaprojektowano tak, że w ramach sprawdzania pojedynczego wyrobu (skrzyni biegów) możliwe jest wykonanie do 52 różnych programów wizyjnych. W przypadku stanowiska wykonanego dla firmy Eaton faktyczna liczba programów wymaganych do sprawdzenia przeciętnej skrzyni nie przekraczała 30. Istotne było określenie, które programy wizyjne i w jakiej kolejności mają być wykonywane w przypadku sprawdzania danego wyrobu (skrzyni biegów). Każdy z 52 potencjalnych programów wizyjnych ma przyporządkowane 7 atrybutów, a więc potencjalnie w bazie danych potrzeba $7 \times 52 = 364$ pól z atrybutami. Przyjęto, że każdy program kontroli wizyjnej jest oznaczony pojedynczą, różną dla każdego programu, literą alfabetu łacińskiego A–Z lub a–z. Korzystając z tego przyporządkowania można określić ciąg literowy o długości nieprzekraczającej 52 znaków, w którym kolejność liter jednoznacznie opisuje kolejność wybierania poszczególnych programów kontroli wizyjnej. Informację o kolejności wykonywania programów oraz ich atrybuty aplikacja nadrzędna odczytuje z bazy danych.

6.4. Baza danych z informacjami dla kontroli wizyjnych

Bazę danych z informacjami dla kontroli wizyjnych utworzono w formacie Access 2000 i zapisano w pliku: DaneDlaKontroliWizyjnych.mdb.

Z punktu widzenia aplikacji nadrzędnej miejsce zapisania pliku z bazą danych (tj. urządzenie i katalog) są dowolne, ponieważ aplikacja odwołuje się do niej przez mechanizm systemu Windows określany jako ODBC (ang. *Open Database Connectivity*). Sama baza zawiera następujące tabele: DaneProb1, DaneProb2.

Podział na dwie tabele wynika z faktu, że program Access 2000 nakłada ograniczenia dotyczące liczby pól (kolumn) zawartych w pojedynczej tabeli – maksymalnie tabela może zawierać ok. 240 pól (tabela DaneProb1 zawiera 208 pól, tabela DaneProb2 – 183 pola).

6.4.1. Struktura tabeli DaneProb1

Tabela DaneProb1 zawiera 208 pól (kolumn), które można podzielić na następujące grupy:

- pole identyfikujące wyrób – skrzynię biegów (1 kolumna). Jest ono kluczem podstawowym tabeli DaneProb1, na jego podstawie aplikacja wybiera rekord z informacjami dla kontroli wizyjnych określonego wyrobu,
- pole identyfikujące kolejność wykonywania kontroli wizyjnych (1 kolumna). Jest to pole zawierające ciąg znakowy określający kolejność wykonywania poszczególnych programów kontroli wizyjnej (parametry tych programów są zawarte w dalszych kolumnach obu tabel DaneProb1 i DaneProb2). Kolejność wykonywania programów będzie określona przez porządek liter A–Z, a–z. Ciąg znaków BzDAa wymusza wykonanie kolejno pięciu programów wizyjnych o nazwach zawartych w kolumnach: „B Nazwa programu", „z Nazwa Programu", „D Nazwa Programu", „A Nazwa Programu", „a Nazwa Programu". Pole to nie ma wartości domyślnej i nie może zawierać innych znaków niż małe i duże litery alfabetu łacińskiego.
- pole identyfikujące program robotowy (1 kolumna). Zawartość tego pola określa położenie pliku w pamięci masowej robota (urządzenie, katalog i nazwę pliku) zawierającego podprogram robota sterujący manipulatorem podczas kontroli wizyjnej wybranego wyrobu (skrzyni biegów). Aplikacja nadrzędna przesyła kanałem RS-232 do sterownika robota odczytane z bazy położenie pliku. Na podstawie tej informacji program główny robota wybiera, a następnie uruchamia właściwy moduł programowy sterujący manipulatorem podczas wykonywania kontroli wizyjnej.
- pole identyfikujące plik konfiguracyjny raportu (1 kolumna). Zawartość tego pola identyfikuje plik konfiguracyjny raportu (urządzenie, katalog i nazwę pliku konfiguracyjnego). Informacja zawarta w pliku konfiguracyjnym raportu jest rodzajem programu umożliwiającego wykonanie globalnej oceny kontroli wizyjnej określonego wyrobu (skrzyni biegów) oraz formatowanie raportu końcowego.
- pola identyfikujące format wydruku raportu (12 kolumn). Pola te zawierają informacje dotyczące formatowania wydruku raportu końcowego z wynikami kontroli wizyjnej. Informacjami tymi są: rodzaj użytej czcionki, jej rozmiar, odstępy między wierszami, położenie i rozmiar ramek oraz okna wydruku w stosunku do wymiarów kartki papieru, na której aplikacja wykonuje wydruk raportu końcowego. Wymiary kartki papieru i orientacja wydruku są ustalane globalnie jako ustawienia domyślnej drukarki zainstalowanej w systemie operacyjnym Windows.
- pola identyfikujące napisy wzorcowe (10 kolumn). Pola te zawierają 10 różnych tekstów wzorcowych charakterystycz-

nych dla wyrobu (skrzyni biegów). Każdy tekst wzorcowy określa stały napis, jaki zawsze powinien występować na wybranym elemencie określonej odmiany skrzyni biegów, niezależnie od jej egzemplarza. Takim napisem jest np. numer identyfikujący wałek napędowy. Jednym z zadań aplikacji wizyjnej jest identyfikacja napisu i porównanie go z tekstem wzorcowym odczytanym z bazy danych.

– pola identyfikujące kolejne 26 kontroli wizyjnych (26 × 7 = 182 kolumny). Pola te zawierają informację dla wykonania 26 różnych kontroli wizyjnych oznaczonych dużymi literami od A do Z (po 7 pól dla każdej kontroli):

Program kontroli wizyjnej oznaczony literą „A”:	**Program kontroli wizyjnej oznaczony literą „B”:**	**.....**	**Program kontroli wizyjnej oznaczony literą „Z”:**
A Nazwa programu	B Nazwa programu	.....	Z Nazwa programu
A Opis kontroli wizyjnej	B Opis kontroli wizyjnej	.....	Z Opis kontroli wizyjnej
A Typ programu	B Typ programu	.....	Z Typ programu
A Liczba znakow	B Liczba znakow	.....	Z Liczba znakow
A Czas ekspozycji kamery	B Czas ekspozycji kamery	.....	Z Czas ekspozycji kamery
A Oswietlacze	B Oswietlacze	.....	Z Oswietlacze
A Polozenie stolu obrotowego	B Polozenie stolu obrotowego	.....	Z Polozenie stolu obrotowego

Dla określonego wyrobu (typu skrzyni biegów) nie każda z 26 grup pól musi zawierać informacje, obligatoryjnie niezbędne jest wypełnienie tylko tych kolumn, które odpowiadają kontrolom wizyjnym zadeklarowanym w polu identyfikującym kolejność wykonywania kontroli wizyjnych

6.4.2. Struktura tabeli DaneProb2

Tabela DaneProb2 jest uzupełnieniem tabeli DaneProb1 i zawiera 183 pola (kolumny), które można podzielić na następujące grupy:

– pole identyfikujące wyrób, czyli skrzynię biegów (1 kolumna). Jest ono kluczem podstawowym tabeli DaneProb2, na podstawie jego zawartości aplikacja wybiera rekord z informacjami dla kontroli wizyjnych określonego wyrobu (skrzyni biegów),

– pola identyfikujące kolejne 26 kontroli wizyjnych (26 × 7 = 182 kolumny) Pola te zawierają informację dla wykonania 26 różnych kontroli wizyjnych oznaczonych małymi literami od a do z (po 7 pól dla każdej kontroli):

Program kontroli wizyjnej oznaczony literą „a”:	Program kontroli wizyjnej oznaczony literą „b”:	.....	Program kontroli wizyjnej oznaczony literą „z”:
a Nazwa programu	b Nazwa programu	.....	z Nazwa programu
a Opis kontroli wizyjnej	b Opis kontroli wizyjnej	.....	z Opis kontroli wizyjnej
a Typ programu	b Typ programu	.....	z Typ programu
a Liczba znakow	b Liczba znakow	.....	z Liczba znakow
a Czas ekspozycji kamery	b Czas ekspozycji kamery	.....	z Czas ekspozycji kamery
a Oswietlacze	b Oswietlacze	.....	z Oswietlacze
a Polozenie stolu obrotowego	b Polozenie stolu obrotowego	.....	z Polozenie stolu obrotowego

Znaczenie tych pól, charakter i ograniczenia wprowadzania danych są identyczne jak w przypadku tabeli DaneProb1.

Aplikacja nadrzędna może tylko odczytywać informacje z bazy danych dla kontroli wizyjnych. Tak więc do wprowadzania, edycji i usuwania informacji z tej bazy niezbędne jest użycie programu Access. Dla ułatwienia wprowadzania i przeglądania informacji do/z obu tabel dodatkowo wykonano dwa formularze (rys. 2): Formularz1 dla tabeli DanePróbl, Formularz2 dla tabeli DanePróbl2.

6.4.3. Weryfikacja danych odczytanych z bazy przed wykonaniem kontroli wizyjnej

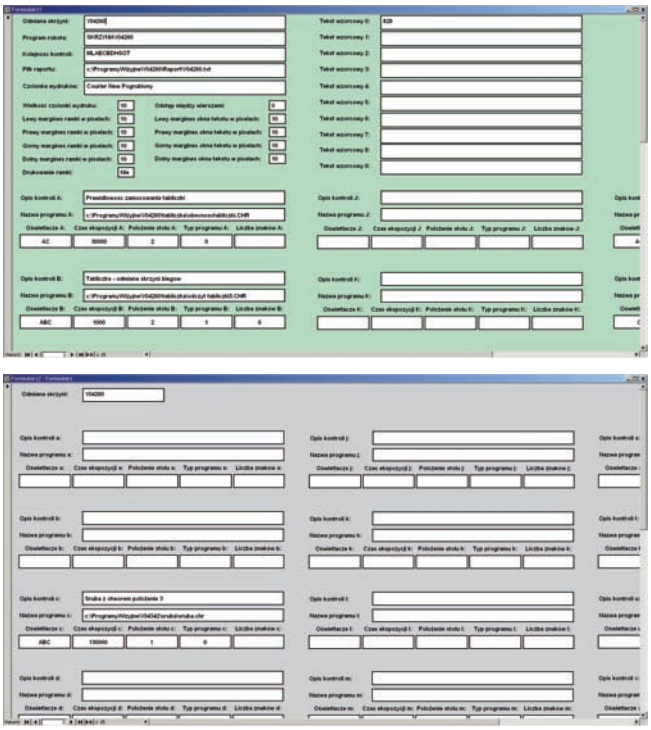
Podczas definiowania niektórych pól obu tabel bazy danych DaneDlaKontroliWizyjnych.mdb ustawiono dodatkowe filtry kontrolujące poprawność wprowadzanych danych. Po wpisaniu przez operatora identyfikatora wyrobu (skrzyni biegów) aplikacja nadrzędna stanowiska przeszukuje tabele DanePróbl i DanePróbl2 w poszukiwaniu rekordów odpowiadających wprowadzonemu identyfikatorowi, odczytuje z nich informację, a następnie poddaje ją dodatkowej weryfikacji. Sprawdzane są m.in. obecność w pamięci masowej komputera wizyjnego wszystkich programów NeuroCheck wymienionych w polach identyfikujących kolejność wykonywania kontroli wizyjnych i atrybuty tych programów (szczegółowy opis zawiera [3]). Jednak nie wszystkie informacje odczytane z bazy danych mogą zostać zweryfikowane przed rozpoczęciem wykonywania kontroli wizyjnej. Aplikacja nadrzędna nie może sprawdzić fizycznej obecności programu aplikacyjnego robota w jego pamięci masowej, zweryfikowania danych formatowania wydruku raportu końcowego przed utworzeniem tego raportu oraz skontrolowania poprawności formatów programów kontroli wizyjnej.

Ponieważ lista nieprawidłowości może być długa, szczególnie w trakcie opracowywania kontroli wizyjnej nowego wyrobu, toteż w trakcie weryfikacji tworzony jest dodatkowy plik tekstowy Diagnostic.txt. Do tego pliku sukcesywnie dopisywane są komunikaty o kolejnych błędach wykrytych podczas weryfikacji informacji odczytanej z bazy danych oraz pliku konfiguracyjnego raportu. W przypadku wykrycia błędu operator może przeglądać zawartość tego pliku, a więc odczytać opis znalezionych błędów (rys. 3).

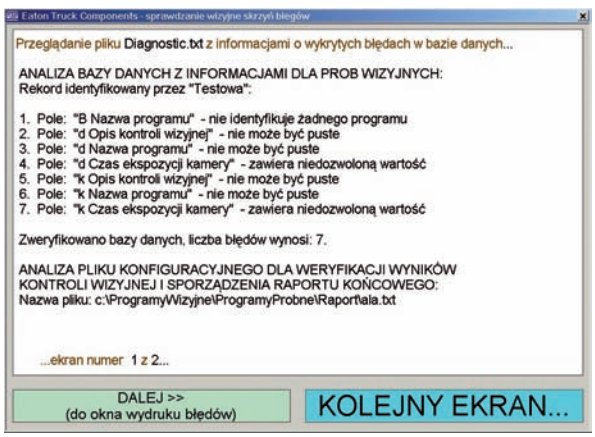
Plik Diagnostic.txt jest zapisywany w tym samym katalogu, co aplikacja nadrzędna. Jego zawartość dotyczy tylko ostatnio wykonanej weryfikacji (informacje nie są dopisywane do pliku).

6.5. Raport końcowy przeprowadzonej kontroliwizyjnej wyrobu

Liczba programów wizyjnych realizowanych w ramach sprawdzania danego wyrobu (skrzyni biegów) nie przekłada się na liczbę zadań kontrolnych, ponieważ każde zadanie może wymagać wykonania od jednego do kilku różnych programów NeuroCheck, np. ocena obecności pojedynczej śruby z otworem wymaga wykonania trzech różnych programów, a końcowa ocena jest sumą logiczną rezultatów każdego z nich. W przypadku stanowiska dla firmy Eaton każda odmiana skrzyni biegów, z punktu widzenia technicz-



Rys. 2. Widok dwóch formularzy do wprowadzania i edycji informacji do bazy danych dla kontroli wizyjnych
Fig. 2.View of two forms for introduction and edit informationto database for vision controls



Rys. 3. Okno przeglądania błędów w bazie danych lub/i w pliku konfiguracyjnym raportu
Fig. 3: Window to view the errors in the database and / or in the configuration file of the report

nej realizacji kontroli i oceny jej wyników, różni się od pozostałych. Dlatego projektując schemat aplikacji przyjęto rozwiązanie, w którym każdemu wyrobowi (każdej odmianie skrzyni biegów) przyporządkowany jest inny algorytm globalnej oceny wszystkich wykonanych kontroli i tworzenia raportu końcowego. Definicja tego algorytmu, zawarta w dodatkowym pliku tekstowym, jest rodzajem programu realizowanym przez aplikację nadrzędną po zakończeniu kontroli wizyjnej. Informacje o położeniu i nazwie tego pliku tekstowego zawiera pole Plik konfiguracyjny raportu tabeli DaneProb1 bazy danych.

Aplikacja komputera wizyjnego umożliwia wykonanie do 52 różnych programów kontroli wizyjnej oznaczonych literami A–Z i a–z. W tabeli DanePrób1 zdefiniowano też 10 pól, w których do każdego wyrobu można przyporządkować 10 różnych tekstów wzorcowych, oznaczonych cyframi 0–9. Podczas wprowadzania danych wyrobu przed rozpoczęciem kontroli wizyjnej, oprócz jego identyfikatora (odmiany skrzyni biegów), operator może wprowadzić ciąg znaków określających numer montażowy i numer seryjny wyrobu. Teksty te mogą być porównywane z tekstami odczytanymi przez system wizyjny z różnych części kontrolowanego wyrobu.

Każdy pojedynczy wiersz pliku konfiguracyjnego raportu odpowiada pojedynczemu wierszowi wstawianemu do raportu końcowego. Dodatkowo każdemu wierszowi z pliku konfiguracyjnego aplikacja przyporządkowuje pojedynczą zmienną logiczną. O tym, czy zmiennej tej zostanie przyporządkowana wartość OK czy Not OK decyduje instrukcja, której mnemonik rozpoczyna wiersz. Globalny rezultat kontroli wizyjnej jest wyznaczany jako iloczyn logiczny zmiennych przyporządkowanych do każdej linii pliku konfiguracyjnego. Przyjmuje się ponadto, że jeśli dana instrukcja pliku konfiguracyjnego nie generuje wyniku logicznego (np. instrukcje nakazujące wstawienie do raportu informacji o dacie i czasie kontroli wizyjnej) to za wynik logiczny tej instrukcji przyjmuje się wartość OK.

Dodatkowo wprowadzono kilka ograniczeń na format tego pliku:
- każdy wiersz pliku konfiguracyjnego musi rozpoczynać się mnemonikiem instrukcji. Nie można używać tzw. pustych znaków (spacje, znaki tabulacji) przed samą instrukcją oraz wewnątrz niej (np. do oddzielenia poszczególnych parametrów),
- po ostatnim parametrze instrukcji można umieścić dodatkowy tekst jako komentarz, ale tekst ten musi rozpoczynać się i kończyć znakiem pojedynczego apostrofu '. Wewnątrz tego tekstu można stosować znaki spacji,
- parametry instrukcji mogą odpowiadać tylko tym programom kontroli wizyjnej, które wymieniono w kolumnie Kolejnosc wykonywania kontroli wizyjnych tabeli DanePrób1,
- plik konfiguracyjny nie może zawierać więcej niż 120 wierszy.

Format pliku konfiguracyjnego jest rygorystycznie sprawdzany po wykonaniu weryfikacji informacji odczytanej z bazy danych DaneDlaKontroliWizyjnych.mdb, a przed rozpoczęciem kontroli wizyjnych, a komentarze o wykrytych nieprawidłowościach dopisywane do pliku Diagnostic.txt.

Poniżej opisano instrukcje, które można umieszczać w pliku konfiguracyjnym raportu.

6.5.1. Informacja o odmianie skrzyni biegów

Format instrukcji: &#

Wstawienie do raportu końcowego informacji o odmianie skrzyni biegów.

6.5.2. Informacja o numerze montażowym skrzyni biegów

Format instrukcji: &@

Wstawienie do raportu końcowego informacji o numerze montażowym (Assembly No) skrzyni biegów.

6.5.3. Informacja o numerze seryjnym skrzyni biegów

Format instrukcji: &$

Wstawienie do raportu końcowego informacji o numerze seryjnym (Trans No) skrzyni biegów.

6.5.4. Informacja o tekście wzorcowym odczytanym z bazy danych

Format instrukcji: &X gdzie *X* oznacza jedną z cyfr 0–9.

Wstawienie do raportu końcowego informacji o tekście wzorcowym oznaczonym wybraną cyfrą.

6.5.5. Informacja o dacie wykonania kontroli wizyjnej

Format instrukcji: &D

Wstawienie do raportu końcowego informacji o dacie wykonania kontroli wizyjnej w formacie „dzień-miesiąc-rok".

6.5.6. Informacja o czasie wykonania kontroli wizyjnej

Format instrukcji: &C

Wstawienie do raportu końcowego informacji o czasie wykonania kontroli wizyjnej w formacie „godzina:minuta:sekunda".

6.5.7. Informacja o katalogu zawierającym zdjęcia sprawdzanych elementów

Format instrukcji: &I

Wstawienie do raportu końcowego informacji o katalogu, w którym zapisano zdjęcia kontrolowanych elementów wykonane podczas realizacji programów wizyjnych.

6.5.8. Informacja o liczbie programów kontroli wizyjnej

Format instrukcji: &L

Wstawienie do raportu końcowego informacji o liczbie programów kontroli wizyjnej realizowanych podczas sprawdzania wyrobu.

6.5.9. Wstawienie do raportu końcowego komentarza tekstowego

Format instrukcji: `'....'`

Wstawienie komentarza zdefiniowanego między znakami apostrofów. Użycie komentarza można zrealizować w wierszu zawierającym inną instrukcję – bezpośrednio po parametrach tej instrukcji. Tekst ten zostanie wstawiony do wiersza raportu przed informacją generowaną przez samą instrukcję, np.:

```
&D'Data kontroli: _ _'
&T'_ Godzina kontroli: '
```

do raportu końcowego zostanie wstawiona następująca informacja (dodatkowe spacje umożliwiają formatowanie tekstu):

```
Data kontroli: _ _ _29-KWI-2015
_ Godzina kontroli: 12:22:31
```

Dodatkowe znaki, umieszczone po apostrofie zamykającym dodatkowy tekst, ale przed znakiem nowej linii (0×0A) lub powrotu karetki (0×0D) wyznaczającym w pliku tekstowym koniec wiersza, są przez aplikację odrzucane i mogą być traktowane jako komentarz umieszczany w pliku konfiguracyjnym raportu.

6.5.10. Informacja o wyniku wykonania pojedynczego programu kontroli wizyjnej

Format instrukcji: *-X,* gdzie *X* może być literą A–Z lub a–z.

Wstawienie do raportu końcowego informacji o wyniku wykonania pojedynczego programu kontroli wizyjnej wymienionego jako parametr instrukcji. Parametr *X* określa dowolny program A–Z lub a–z. Informacja wpisywana do raportu końcowego zależy od kategorii programu wizyjnego, którego dotyczy instrukcja:

1. Jeśli jest to program z kategorii programów wizyjnych sprawdzających obecność lub prawidłowość zamocowania pojedynczego detalu, to do raportu końcowego może zostać wstawiony jeden z dwóch komunikatów:
   - gdy program wizyjny jako wynik logiczny zwrócił wartość Check OK:
     `Wynik: OK`
   - gdy program wizyjny jako wynik logiczny zwrócił wartość Check Not OK:
     `Wynik: Not OK`
2. Jeśli jest to program z kategorii programów wizyjnych identyfikujących napisy, to do raportu końcowego może zostać wstawiony jeden z pięciu komunikatów. W każdym z nich przyjęto oznaczenia: *X* – liczba znaków zidentyfikowanych przez program, *Y* – liczba znaków, które program miał zidentyfikować (informacja z bazy danych), *'xxx'* – znaki zidentyfikowane przez program;
   - jeśli liczby *X* i *Y* są jednakowe i program wizyjny jako wynik logiczny zwrócił wartość Check OK (potwierdzenie, że prawdopodobieństwo poprawnej identyfikacji jest powyżej progu ustalonego w tym programie):
     X `na` *Y* `znaków: 'xxx' Wynik: OK`
   - jeśli liczby *X* i *Y* są różne i program wizyjny jako wynik logiczny zwrócił wartość Check OK (potwierdzenie, że prawdopodobieństwo poprawnej identyfikacji jest powyżej progu ustalonego w programie):
     X `na` *Y* `znaków: 'xxx' Błąd: Różna liczba znaków.`
   - jeśli liczby *X* i *Y* są jednakowe i program wizyjny jako wynik logiczny zwrócił wartość Check Not OK (prawdopodobieństwo poprawnej identyfikacji znaku(ów) jest poniżej progu ustalonego w programie):
     X `na` *Y* `znaków: 'xxx' Błąd: Identyfikacja.`
   - jeśli liczby *X* i *Y* są różne i program wizyjny jako wynik logiczny zwrócił wartość Check Not OK (prawdopodobieństwo poprawnej identyfikacji znaku(ów) jest poniżej progu ustalonego w programie):
     X `na` *Y* `znaków: 'xxx' Błąd: Identyfikacja, różna liczba znaków.`
   - jeśli aplikacja nadrzędna na podstawie analizy odwołania do programu wizyjnego stwierdziła, że jego format nie odpowiada formatowi programu z kategorii programów wykonujących identyfikację napisów:
     X `na` *Y* `znaków: 'xxx' Błąd: Format programu.`
3. Jeśli jest to program z kategorii obejmującej programy wykonujące pomiary wizyjne, to do raportu końcowego może zostać wstawiony jeden z czerech komunikatów. W każdym z nich przyjęto oznaczenia:
   x.xxx – liczba rzeczywista oznaczająca wynik pomiaru.
   - jeśli program wizyjny jako wynik logiczny zwrócił wartość Check OK to oznacza, że wartość *x.xxx* pomiaru mieści się w granicach tolerancji określonych w programie:
     x.xxx `Wynik: OK`
   - jeśli program wizyjny jako wynik logiczny zwrócił wartość Check Not OK to oznacza, że wartość *x.xxx* pomiaru jest różna od 0.0, ale nie mieści się w granicach tolerancji określonych w programie:
     x.xxx `Błąd: Poza granicami tolerancji.`
   - jeśli program wizyjny jako wynik logiczny zwrócił wartość Check Not OK lub wynik pomiaru wynosi 0.0 to oznacza, że program wizyjny nie mógł zidentyfikować obiektu (fizyczny brak obiektu, problem z właściwym pozycjonowaniem skrzyni/kamery, problem z oświetleniem itp.):
     x.xxx `Błąd: Brak obiektu.`
   - jeśli aplikacja nadrzędna po analizie odwołania do programu wizyjnego stwierdziła, że jego format nie odpowiada formatowi programu należącego do kategorii programów wykonujących pomiary wizyjne:
     x.xxx `Błąd: Format programu.`

6.5.11. Informacja o iloczynie logicznym rezultatów wykonania kilku programów kontroli wizyjnej

Format instrukcji: **X... Y,* gdzie *X... Y* mogą być literami A–Z lub a–z.

Wstawienie do raportu końcowego informacji o iloczynie logicznym wyników wykonania programów wizyjnych wymienionych jako parametry tej instrukcji. Parametry *X*, ..., *Y* określają dowolne programy A–Z lub a–z. Instrukcja ta ma ograniczenia:

- może zawierać od 1 do 52 parametrów,
- ponieważ realizuje proste działanie logiczne, nie ma żadnych ograniczeń do typów programów wizyjnych wymienionych jako parametry,
- jeśli instrukcji przyporządkowano tylko jeden parametr, to wynikiem działania jest rezultat logiczny programu wizyjnego, który odpowiada temu parametrowi,
- lista parametrów może zawierać elementy powtarzające się.

Do raportu końcowego może zostać wstawiony jeden z dwóch komunikatów:

- jeśli wszytkie programy wizyjne wymienione jako paramery instrukcji zwróciły wartości Check OK:
  `Wynik: OK`
- jeśli co najmniej jeden program wizyjny z programów wymienionych jako paramery instrukcji zwrócił wartość Check Not OK:
  `Wynik: Not OK`

6.5.12. Informacja o sumie logicznej rezultatów wykonania kilku programów kontroli wizyjnej

Format instrukcji: $+X...Y$, gdzie $X...Y$ mogą być literami A–Z lub a–z.

Wstawienie do raportu końcowego informacji o sumie logicznej wyników wykonania programów wizyjnych wymienionych jako parametry tej instrukcji. Parametry X, ..., Y określają dowolne programy A–Z lub a–z. Instrukcja ta ma ograniczenia:
- może zawierać od 1 do 52 parametrów,
- ponieważ realizuje proste działanie logiczne, nie ma żadnych ograniczeń do typów programów wizyjnych wymienionych jako jej parametry,
- jeśli instrukcji przyporządkowano tylko jeden parametr, to wynikiem działania jest rezultat logiczny programu wizyjnego, który odpowiada temu parametrowi,
- lista parametrów może zawierać elementy powtarzające się.

Do raportu końcowego może zostać wstawiony jeden z dwóch komunikatów:
- jeśli co najmniej jeden program wizyjny z programów wymienionych jako paramery instrukcji zwrócił wartość Check OK:
  `Wynik: OK`
- jeśli wszytkie programy wizyjne wymienione jako paramery instrukcji zwróciły wartości Check Not OK:
  `Wynik: Not OK`

6.5.13. Informacja o wyniku porównania dwóch tekstów

Format instrukcji: $=XY$, gdzie XY mogą być literami A–Z i a–z, cyframi z zakresu 0–9 lub jednym ze znaków: #, @, $.

Wstawienie do raportu końcowego informacji o wyniku porównania dwóch napisów odczytanych przez dwa programy kontroli wizyjnej X i Y. Parametry X i Y określają dowolne programy A–Z i a–z. Dodatkowo do określenia wartości parametru mogą być stosowane znaki:
- cyfry z zakresu 0–9 – porównanie zidentyfikowanego napisu z jednym z dziesięciu tekstów wzorcowych,
- znak # – opisującym odmianę skrzyni biegów (SpecNo),
- znak @ – porównanie z tekstem opisującym numer montażowy skrzyni biegów (AssemblyNo),
- znaku $ – porównanie z tekstem opisującym numer seryjny skrzyni biegów (TransNo).

Do raportu końcowego zostanie wstawiona informacja o wyniku logicznym wykonania tej instrukcji (informacja OK lub Not OK). Instrukcja ta ma ograniczenia:
- jest to instrukcja dwuparametrowa (nie można porównywać trzech lub większej liczby napisów),
- każdy z programów wizyjnych wymieniony jako jej parametr musi być programem typu 1, czyli programem identyfikującym napis,
- oba parametry instrukcji mogą być jednakowe (instrukcja wykona porównanie napisu z samym sobą).

W opisie komunikatów zamieszczonych poniżej będzie używany termin „wynik logiczny związany z analizą napisu". Przyjmuje się, że dla tekstów wzorcowych i tekstów wprowadzanych z klawiatury komputera wizyjnego lub za pomocą czytnika kodu kreskowego (odmiana, numer montażowy, numer seryjny) wynik logiczny związany z analizą napisu ma zawsze wartość OK. W przypadku tekstu identyfikowanego programem wizyjnym, wynik logiczny związany z analizą napisu ma wartość OK, jeśli spełnione są dwa warunki:
- program wizyjny zidentyfikował napis z prawdopodobieństwem większym od progu ustalonego w programie (program wizyjny zwrócił logiczną wartość Check OK),
- liczba zidentyfikowanych znaków była równa liczbie znaków, które faktycznie powinny znajdować się na analizowanym elemencie skrzyni biegów (wartość odczytana z odpowiedniego pola bazy danych).

Do raportu końcowego może zostać wstawiony jeden z ośmiu komunikatów:
- jeśli wyniki logiczne związane z analizami napisów obu obiektów wymienionych jako parametry instrukcji mają wartość OK i napisy są jednakowe:
  `Brak różnic, Wynik OK`
- jeśli wyniki logiczne związane z analizami napisów obu obiektów wymienionych jako parametry instrukcji mają wartość OK i napisy są różne:
  `Są różnice, Wynik Not OK`
- jeśli wyniki logiczne związane z analizami napisów obu obiektów wymienionych jako parametry instrukcji mają wartość Not OK (napisy nie są już porównywane):
  `Napisy: Not OK, Wynik Not OK`
- jeśli wynik logiczny związany z analizą napisu obiektu wymienionego jako pierwszy parametr instrukcji ma wartość Not OK (napisy nie są już porównywane):
  `Napis 1 Not OK, Wynik Not OK`
- jeśli wynik logiczny związany z analizą napisu obiektu wymienionego jako drugi parametr instrukcji ma wartość Not OK (napisy nie są już porównywane):
  `Napis 2 Not OK, Wynik Not OK`
- jeśli oba porównywane napisy mają zerową długość:
  `Brak obu napisów, Wynik Not OK`
- jeśli napis odpowiadający obiektowi wymienionemu jako pierwszy parametr instrukcji ma zerową długość:
  `Brak napisu 1, Wynik Not OK`
- jeśli napis odpowiadający obiektowi wymienionemu jako drugi parametr instrukcji ma zerową długość:
  `Brak napisu 2, Wynik Not OK`

6.5.14. Informacja o globalnym rezultacie przeanalizowanych instrukcji kontroli wizyjnych

Format instrukcji: &W

Instrukcja ta wstawia do raportu końcowego informację o globalnym wyniku dotychczas przeanalizowanych instrukcji. Aby był to globalny wynik wszystkich instrukcji, musi być umieszczona na końcu pliku konfiguracyjnego raportu.

6.5.15. Przykład pliku konfiguracyjnego raportu

```
'WYNIK KONTROLI SKRZYNI BIEGÓW'
&D'  Data kontroli:  '
&C'  Godz. kontroli: '
' '
&#'- odmiana skrzyni biegów: '
&@'- numer montażowy:        '
&$'- numer seryjny:          '
' '
'KOMPLETNOŚĆ MONTAŻU ELEMENTÓW:'
'============================='
-A'- sprawdzenie obecności tabliczki: '
-G'- sprawdzenie zaczepu:             '
-N'- sprawdzenie gniazda:             '
+abc'- sprawdzenie obecności otworu:    '
' '
'POMIARY WIZYJNE:'
'================'
'- kąt dźwigni zmiany biegów:'
-L'   Pomiar = '
'- średnica wałka napędowego:'
-W'   Pomiar = '
' '
'ODCZYTY NAPISÓW ZE SKRZYNI BIEGÓW:'
'================================='
'- odczyt z tabliczki - Odmiana: '
-B'   '
'- odczyt z tabliczki - Customer No:'
-C'   '
'- odczyt z tabliczki - ModelNo: '
-D'   '
```

```
'- odczyt z tabliczki - Nr ser.:'
-E'   '
'- odczyt z odlewu    - Nr ser.'
-F'   '
'- odczyt numeru wałka:'
-S'   '
'- nr wałka z bazy danych:'
&0'   '
' '
'ANALIZA NAPISÓW ZE SKRZYNI BIEGÓW:'
'================================='
'- porównanie odmiany skrzyni'
'  wprowadzonej czytnikiem z odmianą'
'  odczytaną z tabliczki znamionowej:'
=#B'   '
' '
'- porównanie nr seryjnego skrzyni'
'  wprowadzonego czytnikiem z nr seryjnym'
'  odczytanym z tabliczki znamionowej:'
=$E'   '
' '
'- porównanie nr seryjnego skrzyni'
'  wprowadzonego czytnikiem z nr seryjnym'
'  odczytanym z odlewu korpusu:'
=$F'   '
' '
'- porównanie nr wałka odczytanym kamerą'
'  z nr wałka z bazy danych:'
=0S'   '
```

6.5.16. Przykład końcowej postaci raportu

```
WYNIK KONTROLI SKRZYNI BIEGÓW
  Data kontroli:  10-Lut-2015
  Godz. kontroli: 12:52:31

- odmiana skrzyni biegów: ZY06365
- numer montażowy:        546843
- numer seryjny:          TC261691

KOMPLETNOŚĆ MONTAŻU ELEMENTÓW:
==============================
- sprawdzenie obecności tabliczki: Wynik: OK
- sprawdzenie zaczepu:             Wynik: OK
- sprawdzenie gniazda:             Wynik: Not OK
- sprawdzenie obecności otworu:    Wynik: OK

POMIARY WIZYJNE:
================
- kąt dźwigni zmiany biegów:
   Pomiar = 22.49 Wynik: OK
- średnica wałka napędowego:
   Pomiar = 56.17 Błąd: Poza granicami tolerancji.

ODCZYTY NAPISÓW ZE SKRZYNI BIEGÓW:
==================================
- odczyt z tabliczki - Odmiana:'
   7 na 7 znaków: 'ZY06365' Wynik: OK '
- odczyt z tabliczki - Customer No:'
   10 na 10 znaków: '5010545195' Wynik: OK
- odczyt z tabliczki - ModelNo: '
   9 na 9 znaków: 'FS/6309AV' Wynik: OK
- odczyt z tabliczki - Nr ser.: '
   8 na 8 znaków: 'TC261691' Wynik: OK '
- odczyt z odlewu    - Nr ser. '
   8 na 8 znaków: 'TC261691' Wynik: OK
- odczyt numeru wałka: '
   7 na 7 znaków: '8880788' Wynik: OK
- nr wałka z bazy danych:'
   8880788

ANALIZA NAPISÓW ZE SKRZYNI BIEGÓW:
==================================
- porównanie odmiany skrzyni
  wprowadzonej czytnikiem z odmianą
  odczytaną z tabliczki znamionowej:
   Brak różnic, Wynik OK
- porównanie nr seryjnego skrzyni
  wprowadzonego czytnikiem z nr seryjnym
  odczytanym z tabliczki znamionowej:
   Brak różnic, Wynik OK
- porównanie nr seryjnego skrzyni
  wprowadzonego czytnikiem z nr seryjnym
  odczytanym z odlewu korpusu:
   Brak różnic, Wynik OK
- porównanie nr wałka odczytanym kamerą
  z nr wałka z bazy danych:
   Brak różnic, Wynik OK
```

6.6. Archiwizacja, przeglądanie i wydruk wyników kontroli wizyjnych

Wyniki kontroli wizyjnej wyrobu są archiwizowane, a zakres archiwizacji obejmuje:

— zapis wyników kontroli do bazy danych,
— zapis do plików graficznych w formacie .JPG zdjęć sprawdzanych elementów,
— zapis do pliku tekstowego raportu końcowego z przeprowadzonych prób wizyjnych.

Zapis wyników kontroli do bazy danych i do raportu końcowego jest realizowany po wykonaniu wszystkich programów wizyjnych przewidywanych dla danego wyrobu. Zapis zdjęć sprawdzanych elementów do plików graficznych ma miejsce po zakończeniu wykonywania pojedynczego programu wizyjnego. Pliki graficzne i raport końcowy są zapisywane w podkatalogach utworzonych w folderze, np.: d:\WynikiProb

Nazwa podkatalogu zostaje utworzona na podstawie odczytu kalendarza i zegara komputera wizyjnego, przy wykorzystaniu mechanizmu tworzenia długich nazw dostępnych w systemie Windows. Pełna nazwa podkatalogu ma następujący format: d:\WynikiProb*rrrr-mm-dd_hh-mm-ss* przy czym wszytkie fragmenty nazwy, które wyszczególniono kursywą są zmienną częścią tej nazwy i kolejno określają: rok (*rrrr*), miesiąc (*mm*), dzień (*dd*), godzinę (*hh*), minutę (*mm*) i sekundę (*ss*). Odczyt zegara systemowego, na podstawie którego generowana jest nazwa podkatalogu następuje w chwili rozpoczęcia kontroli wizyjnej wyrobu.

6.6.1. Rejestrowanie wyników kontroli wizyjnej do bazy danych

Bazę danych, w której zapisywane są wyniki kontroli wizyjnych sprawdzanych wyrobów utworzono w formacie Access 2000 i zapisano w pliku: WynikiKontroliWizyjnych.mdb

Analogicznie jak w przypadku bazy danych z informacjami dla kontroli wizyjnych dostęp do bazy wyników aplikacja uzyskuje korzystając ze sterowników ODBC. Sama baza zawiera pojedynczą tabelę: WynikiProb

W tej bazie rejestrowane są następujące dane:

— odmiana, numer montażowy i numer seryjny wyrobu (skrzyni biegów). Są to dane wprowadzone przez operatora stanowiska za pomocą czytnika kodu kreskowego lub klawiatury komputera przed rozpoczęciem kontroli wizyjnej,
— informacja o dokładnym czasie (data, godzina, minuta, sekunda) określającym moment, w którym rozpoczęto wykonywanie kontroli wizyjnej (gdy operator stanowiska potwierdził wyjście ze strefy chronionej). Czas ten jest odczytywany z zegara komputera wizyjnego,
— informacja o miejscu (urządzenie i katalog), gdzie zapisano plik raportu i pliki graficzne ze zdjęciami skontrolowanych elementów,
— globalny wynik kontroli wizyjnej wyrobu,
— wyniki wykonanych programów kontroli wizyjnych.

W bazie danych nie są rejestrowane wyniki kontroli wizyjnych poszczególnych elementów, ale logiczne rezultaty zrealizowanych programów wizyjnych. Wynika to z faktu, że wynik kontroli elementu może być efektem wykonania kilku programów i trudno byłoby wprowadzić jedną uniwersalną strukturę danych odpowiednią dla każdego wyrobu. Tabela WynikiProb zawiera 52 kolumny oznaczone literami A–Z i a–z:

„Wynik proby A", ... , „Wynik proby Z"
„Wynik proby a", ... , „Wynik proby z"

w których zapisywane są wyniki wykonania programów wskazanych w polach bazy danych DaneDlaKontroliWizyjnych.mdb:

„A Nazwa programu", ... , „Z Nazwa programu"
„a Nazwa programu", ... , „z Nazwa programu"

Wartość wpisywana do każdego z pól może przyjmować wartości:

- +1 – program wizyjny zwrócił wartość Check OK,
- 0 – program wizyjny nie został wykonany, ponieważ nie było go na liście programów do wykonania (jego symbolu nie wymieniono w polu identyfikującym kolejność wykonywania kontroli wizyjnych bazy danych DaneDlaKontroliWizyjnych.mdb)
- –1 – program wizyjny zwrócił wartość Check Not OK.

6.6.2. Rejestrowanie do plików graficznych zdjęć sprawdzanych elementów

Jednym z ciekawszych zadań postawionych przed wykonawcami stanowiska wizyjnego dla firmy Eaton była rejestracja zdjęć sprawdzanych elementów. Mechanizm wykonywania tej rejestracji wykorzystuje makroinstrukcję Transfer Image, która obligatoryjnie zawsze występuje w każdym programie kontroli wizyjnej realizowanym przez NeuroCheck. Aby umożliwić rejestrację do pliku .bmp należy w każdym programie wizyjnym odpowiednio skonfigurować wyjścia (output) tej makroinstrukcji, ustalając nazwę katalogu (np. d:\WynikiProb) oraz przedrostek nazwy pliku (np. Photo), w którym będą zapisywane obrazy. Rejestracja zdjęcia do pliku graficznego ma miejsce bezpośrednio po zrealizowaniu każdego pojedynczego programu wizyjnego i odbywa się w następujący sposób:

- pakiet oprogramowania NeuroCheck wykonując instrukcję Transfer Image programu wizyjnego rejestruje do pliku o nazwie d:\WynikiProb\Photo001.bmp zdjęcie elementu. Zarejestrowane zdjęcie nie odpowiada jednak pełnemu obrazowi uzyskanemu z kamery, lecz jego fragmentowi określonemu podczas parametryzacji instrukcji Transfer Image,
- aplikacja nadrzędna odczytuje utworzoną bitmapę, poddaje ją konwersji na format .jpg i zapisuje pod nazwą utworzoną na podstawie atrybutu programu wizyjnego o nazwie Opis programu. Po wykonaniu konwersji źródłowy plik .bmp zostaje usunięty.

Konwersja z formatu .bmp na format .jpg jest konieczna ze względu na duży rozmiar plików .bmp. Do konwersji zastosowano procedury biblioteczne [14], a prawa licencyjne do ich komercyjnego wykorzystania zakupiono od jednego z autorów książki.

6.6.3. Tworzonenie, przeglądanie i wydruk raportu końcowego

Po zakończeniu kontroli wizyjnej wyrobu, zgodnie z algorytmem zdefiniowanym w pliku konfiguracyjnym raportu, powstaje raport końcowy. Jest on zapisywany pod nazwą Raport.txt w tym samym podkatalogu, co pliki graficzne ze zdjęciami elementów. W trakcie tworzenia raportu końcowego wyznaczany jest globalny wynik kontroli całego wyrobu (OK lub Not OK).

Raport końcowy jest wyświetlany w jednym z okien dialogowych aplikacji (rys. 4). Na zielono podświetlane są zadania, które zakończyły się wynikiem OK, a na czerwono – wynikiem Not OK.

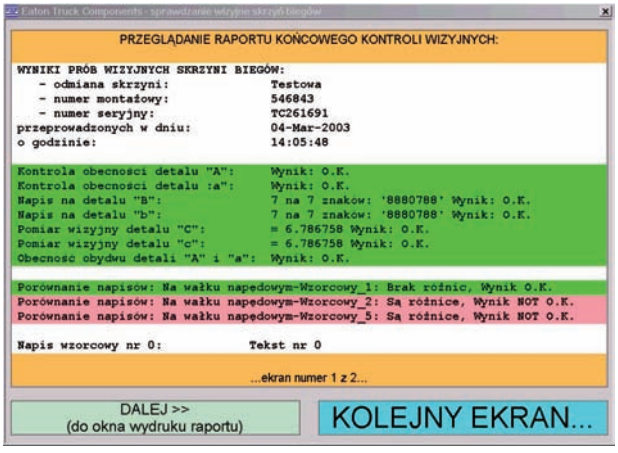
Raport jest plikim tekstowym, nie zawiera dodatkowych informacji o drukarce, wymiarach papieru i czcionce. Wydruk jest wykonywany na domyślnej drukarce systemu Windows z preferencjami wydruku ustawionymi dla tej drukarki. Samą formę wydruku tzn. czcionkę, odległości między wierszami, wielkości marginesów etc. można indywidualnie ustalać dla każdego wyrobu i są one pamiętane w bazie danych z informacjami dla kontroli wizyjnych. Aplikacja nadrzędna nie ma funkcji podglądu wydruku, m.in. ze względu na zbyt skomplikowaną obsługę tej funkcji w warunkach przemysłowych. W związku z tym weryfikacja preferencji wydruku oraz prawidłowego doboru czcionki, wymiarów ramki i okna następuje dopiero podczas tworzenia obrazu strony bezpośrednio przed skierowaniem go na drukarkę.

6.7. Sterowanie robotem przez komputer wizyjny

W przypadku stanowiska dla firmy Eaton różnorodność typów skrzyń biegów i rozmieszczenie sprawdzanych elementów stanowiły problem podczas projektowania uniwersalnego systemu wielokamerowego do kontroli wizyjnej wszystkich skrzyń. Dlatego najkorzystniejszym rozwiązaniem było zastosowanie robota przemysłowego z pojedynczą kamerą zamontowaną na jego manipulatorze. Narzuciło ono konieczność opracowania sposobu sterowania robotem z poziomu aplikacji nadrzędnej.

Jedną z rozważanych koncepcji sterowania było zadawanie współrzędnych kolejnych punktów pomiarowych za pomocą przesyłek z komputera wizyjnego do robota przez interfejs RS-232 lub sieć Ethernet. W uproszczeniu, program robota składałby się z pojedynczej pętli programowej, w której prowadzony byłby nasłuch kanału transmisyjnego. Po odbiorze przesyłki ze współrzędnymi punktu – manipulator realizowałby ruch do tego punktu. Zbiory ze współrzędnymi wszystkich punktów kontrolnych dla każdej skrzyni biegów byłyby przechowywane w pamięci komputera wizyjnego w postaci plików tekstowych, arkuszy kalkulacyjnych lub w bazie danych. W rozwiązaniu tym byłby tylko jeden, prosty program robota, możliwy do zastosowania w przypadku każdej skrzyni biegów. Definiowanie współrzędnych punktów polegałoby na ręcznym przemieszczaniu manipulatora robota, odczycie jego położenia, a następnie manualnym wprowadzeniu odczytanej informacji do pamięci komputera wizyjnego. Rozwiązanie to zostało jednak zarzucone ze względu na potencjalne trudności, jakich spodziewano się przy niezbędnych korekcjach położeń niektórych punktów.

Pomysł sterowania ruchem robota za pomocą przesyłek ze współrzędnymi punktów docelowych wykorzystano w PIAP przy realizacji innych aplikacji wizyjnych. W 2013 r. wykonano model linii technologicznej z robotem ABB IRb-140 dla Politechniki Poznańskiej [6], a w 2014 r. stanowisko demonstracyjne z robotem Kuka KR16 na Targi Automaticon 2014 [7]. W obu przypadkach realizowanym zadaniem była identyfikacja położenia elementu na palecie i podjęcie go chwytakiem robota. Rozpoznanie położenia i sterowanie robotem realizowały kamery inteligentne firmy Sick (stanowisko dla Politechniki) i firmy Cognex (stanowisko targowe).



Rys. 4. Okno przeglądania raportu utworzonego na podstawie pliku konfiguracyjnego

Fig. 4. Window to view the report based on the configuration file

W przypadku stanowiska dla firmy Eaton z robotem ABB IRb-1400, wybrano tradycyjne rozwiązanie. Aplikacja robota, w dalszej części określana jako program główny, zawiera instrukcje komunikacji łączem RS-232 z komputerem wizyjnym. Oprócz programu głównego, w pamięci komputera sterującego robotem znajduje się szereg plików, każdy z podprogramem obsługującym kontrolę wizyjną innej skrzyni biegów. W podprogramach tych zdefiniowano współrzędne punktów pomiarowych jako parametry odpowiednich instrukcji ruchu, a więc można je łatwo korygować przy pomocy funkcji uczenia, dostępnej w oprogramowaniu systemowym robota. Aplikacja komputera wizyjnego przesyła interfejsem RS-232 informację o ścieżce dostępu do pliku z podprogramem robota właściwym do obsługi kontroli wybranej skrzyni biegów. Informacja ta jest odczytywana z bazy danych. Program główny robota zawiera instrukcję pozwalającą wywołać dowolny podprogram zdefiniowany w pliku, do którego ścieżkę dostępu określa parametr tej instrukcji. Uruchomiony podprogram steruje ruchem manipulatora podczas kontroli wizyjnej wybranej skrzyni. Sekwencja przemieszczania manipulatora musi być zsynchronizowana z sekwencją wykonywania kolejnych programów wizyjnych. Po zakończeniu kontroli sterowanie robotem zostaje przekazane z podprogramu do jego programu głównego.

Podobne rozwiązanie, z podziałem na program główny robota i podprogramy do kontroli różnych wyrobów, zastosowano w stanowisku demonstracyjnym PIAP prezentowanym na kilku targach branżowych. Urządzeniem przemieszczającym kamerę był robot firmy Kuka, co z góry determinowało sposób wyboru podprogramu obsługi kontroli wizyjnej danego wyrobu. W przypadku robotów firmy Kuka zdalny wybór podprogramów wykonywany jest za pomocą sygnałów dwustanowych z komputera wizyjnego do robota. Jest to realizowane w następujący sposób: Sterownik robota ma w pamięci plik o nazwie:

c: \ KRC \ ROBOTER \ KRC \ R1 \ cell.scr

Jest to „szkielet" programu dostarczany przez firmę Kuka, zawierający m.in. instrukcje odczytu WE dwustanowych zadeklarowanych jako wejścia wyboru aplikacji robota. Program cell.src należy więc uzupełnić o instukcje wywołujące podprogramy obsługi kontroli wizyjnych w zależności od kombinacji sygnałów wejściowych. W przypadku robota Kuka w bazie danych z informacjami dla kontroli wizyjnych zamiast położenia i nazwy pliku z właściwym podprogramem robota należy umieścić kombinację sygnałów dwustanowych służących do wyboru takiego podprogramu.

6.7.1. Używane terminy

Aby szczegółowo wyjaśnić sposób sterowania robotem przez aplikację komputera podczas wykonywania kontroli wizyjnych złożonych wyrobów należy sprecyzować znaczenie kilku użytych terminów. Przedstawiony w tym i następnym podrozdziale opis dotyczy robota ABB użytego w aplikacji wykonanej dla firmy Eaton.

Bezpieczne położenie manipulatora. Oprogramowanie systemowe robota umożliwia zdefiniowanie tzw. strefy bezpiecznej. Strefa bezpieczna jest fragmentem przestrzeni roboczej manipulatora z wyłączeniem walca zdefiniowanego względem jego podstawy. W przypadku, gdy narzędzie robota wyjdzie poza przestrzeń tego walca zakłada się, że przebywa ono w strefie bezpiecznej. Informację o położeniu narzędzia w stosunku do strefy bezpiecznej oprogramowanie systemowe robota może wydawać sterując sygnałem na wybranym wyjściu dwustanowym (działanie niezależnie od aplikacji). W przypadku opisywanej aplikacji sygnałem tym jest dwustanowy sygnał SAFE_ROBOT_POSITION.

Robocze położenie manipulatora. Położenie manipulatora, w którym aplikacja nadrzędna komputera realizuje kontrolę wizyjną pojedynczego elementu lub ruch stołu obrotowego. Przebywanie w położeniu roboczym oprogramowanie robota potwierdza utrzymując wysoki stan („1") na wyjściu dwustanowym ROBOT_IN_POSITION. Obrót stołu można realizować, jeśli robocze położenie manipulatora jest jednocześnie położeniem bezpiecznym (sygnał SAFE_ROBOT_POSITION=1).

START_PROGRAMU_GLOWNEGO_ROBOTA. Sygnał dwustanowy z komputera do robota podłączony do jego wejścia systemowego „StartMain". Sygnał ten (po uprzednim zatrzymaniu programu) powoduje wystartowanie od pierwszej instrukcji programu głównego robota. Aby uruchomić ten program, aplikacja nadrzędna komputera wizyjnego ustawia na tym wyjściu „1" i utrzymuje ten stan aż do momentu uzyskania potwierdzenia (tzn. „1") na wejściu ROBOT_ PROGRAM_ RUNNING.

KONTYNUACJA_PROGRAMU_ROBOTA. Sygnał dwustanowy z komputera do robota podłączony do jego wejścia systemowego Start. W trakcie realizacji kontroli wizyjnych program robotowy może być zatrzymany przez stop awaryjny lub naruszenie kurtyny ochronnej. W pierwszym przypadku aplikacja nadrzędna zakończy kontrolę, w drugim przypadku będzie można kontytuować program robotowy od instrukcji, na której został zatrzymany. Tak więc, po uzyskaniu potwierdzenia przez operatora, np. przy pomocy odpowiedniego przycisku umieszczonym na panelu stanowiska aplikacja nadrzędna ustawia na tym wyjściu „1" i utrzymuje ten stan aż do momentu uzyskania potwierdzenia (tzn. „1") na wejściu ROBOT_ PROGRAM_ RUNNING.

STOP_PROGRAMU_ROBOTA. Sygnał dwustanowy z komputera wizyjnego do robota podłączony do jego wejścia systemowego „STOP PROGRAMU". Jest on wysterowywany („1") w dwóch przypadkach: po zakończeniu wykonywania wszystkich kontroli wizyjnych dla danego wyrobu i uzyskaniu potwierdzenia, że robot znajduje się w pozycji bezpiecznej, w trakcie wykonywania kontroli wizyjnej – po naruszeniu kurtyn ochronnych.

W obu przypadkach stan „1" jest utrzymywany do momentu uzyskania potwierdzenia (tzn. „0") na wejściu ROBOT_ PROGRAM_ RUNNING.

KOMPUTER_OK. Sygnał dwustanowy z komputera wizyjnego do robota, utrzymywany w stanie „1" podczas wykonywania kontroli wizyjnych. Jeśli aplikacja nadrzędna zmieni go na „0" to oznacza, że podprogram obsługi wybranego wyrobu powinien przemieścić manipulator robota do pozycji wyjściowej, ustawić wartość „0" sygnału ROBOT_APPLICATION_RUNNING i przekazać sterowanie do programu głównego robota. Nie jest tutaj konieczne wysterowywanie innych sygnałów dwustanowych, ponieważ dla aplikacji nadrzędnej potwierdzeniem jest wyłączenie sygnału ROBOT_APPLICATION_RUNNING. Przedwczesne przerwanie kontroli wizyjnej danego wyrobu może nastąpić w dwóch przypadkach:

— jeśli zostanie ona ręcznie przerwana przez operatora stanowiska (jest taka opcja w programie aplikacyjnym komputera wizyjnego),
— jeśli w trakcie automatycznego wykonywania kontroli wizyjnych zostanie wykryty błąd uniemożliwiający ich dalsze kontynuowanie (błąd zapisu do pliku, niewłaściwy format programu NeuroChecka, etc.).

START_MOVING. Sygnał dwustanowy z komputera wizyjnego do robota, którego wartość „1" ma wymusić na podprogramie obsługi wybranego wyrobu przemieszczenie manipulatora do kolejnej pozycji, w której będzie wykonywana kontrola wizyjna lub zmiana pozycji stołu obrotowego. Wartość „1" jest utrzymywana do momentu, aż program obsługi wybranego wyrobu ustawi „0" na wyjściu „ROBOT_IN_ POSITION"(jest to potwierdzenie przyjęcia rozkazu ruchu).

Na sygnał ten ma reagować tylko podprogram obsługi wybranego wyrobu, program główny robota ma go ignorować. W momencie zarejestrowania „1" na wejściu „START_MOVING" podprogram obsługi wybranego wyrobu powinien ustawić „0" na wyjściu „ROBOT_IN_POSITION", przemieścić manipulator do następnego punktu roboczego, a następnie wysterować („1") to wyjście. Przez punkt roboczy rozumie się położenie manipulatora, w którym ma być wykonywana kontrola wizyjna lub przemieszczenie stołu obrotowego. W tym rozumieniu punktem roboczym

nie są żadne punkty pośrednie, przez które przemieszcza się manipulator robota.

ROBOT_PROGRAM_RUNNING. Sygnał dwustanowy z robota do komputera oznaczający („1"), że został uruchomiony program aplikacyjny robota. Jest to sygnał systemowy robota, tzn. za jego sterowanie odpowiada nie program aplikacyjny robota, ale jego system operacyjny (sterowanie tym sygnałem wymusza odpowiednie skonfigurowanie sygnału „CycleOn" w komputerze robotowym). Sygnał ten jest gaszony („0") w momencie zatrzymania programu robotowego na skutek:
— załączenia stopu awaryjnego,
— zatrzymania programowego przez wysterowanie („1") sygnału STOP_ PROGRAMU_ ROBOTA.

Zatrzymanie wykonywania programu aplikacyjnego robota jest równoznaczne z zatrzymaniem przemieszczania manipulatora. Program aplikacyjny robota można wznowić od jego pierwszej instrukcji (patrz sygnał START_PROGRAMU_GLOWNEGO_ ROBOTA) lub od instrukcji, na której został zatrzymany (patrz sygnał KONTYNUACJA_PROGRAMU_ROBOTA).

ROBOT_APPLICATION_RUNNING. Sygnał dwustanowy z robota do komputera określający („1"), że program główny robota przekazał sterowanie do podprogramu obsługi wybranego wyrobu. Sygnał ten jest gaszony („0") w momencie zakończenia wykonywania programu obsługi wybranego wyrobu. Sterowanie tym sygnałem wykonuje program główny robota.

SAFE_ROBOT_POSITION. Sygnał dwustanowy z robota do komputera wizyjnego określający („1") przebywanie robota w bezpiecznej strefie, tzn. takiej, w której możliwy jest ruch obrotowy stołu. Strefę bezpieczną robota definiuje się podczas konfigurowania robota, a uaktywnia jedną z pierwszych instrukcji programu głównego robota (instrukcja ta może być tylko raz wykonana, inaczej system operacyjny robota zgłosi błąd). Sygnałem tym steruje zatem nie program aplikacyjny robota, ale jego oprogramowanie systemowe. W aplikacji dla firmy Eaton zakłada się, że bezpieczna strefa może (ale nie musi) obejmować także punkty zatrzymania manipulatora, w których wykonywana jest kontrola wizyjna, ale obligatoryjnie obejmuje te punkty, w których ma być wykonywane przemieszczanie stołu obrotowego. Zakłada się ponadto, że w momencie rozpoczynania kontroli wizyjnych manipulator robota musi znajdować się w bezpiecznej strefie, a po zakończeniu kontroli – zostać przemieszczony do tej strefy (punkt wyjściowy robota musi znajdować się w strefie bezpiecznej).

ROBOT_IN_POSITION. Sygnał dwustanowy z robota do komputera oznaczający („1"), że manipulator znajduje się w pozycji roboczej, tzn. pozycji, w której możliwe jest wykonanie kontroli wizyjnej lub zmiana położenia stołu obrotowego. Bezpośrednio po odczycie „1" na wejściu „START_MOVING" program obsługi wybranego wyrobu powinien zgasić ten sygnał („0"), przemieścić manipulator do następnej pozycji roboczej a następnie wysterować go („1").

EMERGENCY_STOP. Sygnał dwustanowy z robota (z wyjścia systemowego EmStop) do komputera oznaczający („1" – odwrócona logika) załączenie stopu awaryjnego robota (brak informacji, co spowodowało załączenie stopu awaryjnego).

KURTYNA_OCHRONNA. Sygnał dwustanowy z kurtyny ochronnej do komputera wizyjnego. Do momentu rozpoczęcia kontroli wizyjnej jest on ignorowany (na wyjściach z komputera zezwalających na działanie programu robota i obrót stołu są wtedy i tak utrzymywane stany blokujące te działania). W trakcie wykonywania kontroli wizyjnej powoduje zatrzymanie wykonywania programu robotowego i ruch stołu obrotowego.

KONTYNUACJA_PROGRAMU. Sygnał dwustanowy do komputera z dodatkowego przycisku zamontowanego na panelu operatorskim stanowiska. Sygnał z tego przycisku jest ignorowany do momentu aż na skutek naruszenia kurtyn ochronnych wykonywanie programu robota zostaje zatrzymane (patrz opis sygnału STOP_PROGRAMU_ROBOTA). Wówczas wznowienie programu od instrukcji, w której nastąpiło zatrzymanie może nastąpić dopiero po wciśnięciu przycisku KONTYNUACJA (patrz także opis sygnału KONTYNUACJA_PROGRAMU_ROBOTA).

6.7.2. Współpraca aplikacja nadrzędna komputera wizyjnego – robot przemysłowy

Stanem wyjściowym do rozpoczęcia wykonywania kontroli wizyjnej jest stan, w którym:
— w pamięci operacyjnej robota znajduje się program główny robota, ale nie jest on uruchomiony (sygnał systemowy ROBOT_PROGRAM_RUNNING = „0"),
— oprogramowanie systemowe robota sygnalizuje, że manipulator przebywa w bezpiecznym położeniu (sygnał systemowy SAFE_ ROBOT_POSITION = „1", warunek obligatoryjny),
— napędy manipulatora robota pozostają załączone.

Sterowanie robotem przez aplikację nadrzędną komputera wizyjnego przebiega w następującym porządku (przejście do następnego kroku algorytmu może nastąpić tylko wtedy, gdy spełnione są warunki określone w danym kroku):
1. Aplikacja nadrzędna komputera wizyjnego załącza („1") sygnał START_PROGRAMU_ GŁÓWNEGO_ROBOTA i utrzymuje ten stan aż na wejściu ROBOT_PROGRAM_RUNNING pojawi się „1" (oznacza to, że wykonywanie programu aplikacyjnego robota zostało wznowione). Samo oczekiwanie na potwierdzenie trwa nie dłużej niż 5 sekund, a przekroczenie tego czasu skutkuje wyświetleniem komunikatu o błędzie i przerwaniem realizacji kontroli wizyjnych.
2. Aplikacja nadrzędna łączem RS-232 przesyła do układu sterowania robota przesyłkę zawierającą nazwę podprogramu obsługi wybranego wyrobu. Po sprawdzeniu formatu przesyłki i zinterpretowaniu zawartej w niej informacji program główny robota tym samym torem przekazuje do komputera nadrzędnego komunikat potwierdzający jej przyjęcie lub zawierający kod ewentualnego błędu. Brak takiego potwierdzenia w ciągu 15 sekund albo odczytanie informacji o wystąpieniu błędu skutkuje wyświetleniem stosownego komunikatu i przerwaniem realizacji kontroli wizyjnych.
3. Aplikacja nadrzędna przechodzi w stan oczekiwania na potwierdzenie od układu sterowania robota uruchomienia podprogramu obsługi wybranego wyrobu (a ściślej na potwierdzenie przekazania sterowania z programu głównego robota do programu obsługi wybranego wyrobu). Potwierdzenie to polega na ustawieniu przez aplikację robota wartości „1" sygnału ROBOT_ APPLICATION_RUNNING i musi być zrealizowane w czasie nie dłuższym niż 5 sekund. Wartość „1" sygnału ROBOT_ APPLICATION_RUNNING oprogramowanie robota musi utrzymywać przez cały czas pracy podprogramu obsługi wybranego wyrobu, a następnie zgasić („0") ten sygnał.
4. Aplikacja nadrzędna komputera wizyjnego sprawdza pozycję wyjściową robota. Pozycję tę układ sterowania robota sygnalizuje poprzez utrzymywanie wysokiego („1") stanu sygnałów SAFE_ROBOT_POSITION i ROBOT_IN_POSITION (z punktu „widzenia" aplikacji nadrzędnej każda pozycja manipulatora, w której oba te sygnały są wysterowane, jest pozycją wyjściową). Ponieważ manipulator robota w momencie rozpoczynania kolejnej kontroli wizyjnej nie musi znajdować się w „rzeczywistej" pozycji wyjściowej, przyjmuje się, że pierwszą instrukcją podprogramu obsługi wybranego wyrobu jest przemieszczenie manipulatora do tego punktu i dopiero wtedy ustawienie „1" na wyjściu ROBOT_IN_POSITION. Musi to nastąpić w czasie nie dłuższym niż 15 sekund.

Od tego momentu aplikacja nadrzędna realizuje cykl czynności związanych z przemieszczaniem manipulatora do kolejnych punktów roboczych, w których wykonywane są kontrole wizyjne i zmiany położenia stołu obrotowego. Przebiega to w następującym porządku:
1. Aplikacja nadrzędna ustawia „1" na wyjściu START_MOVING i utrzymuje ten stan do momentu, w którym podprogram obsługi wybranego wyrobu ustawi wartość „0" na wyjściu dwu-

stanowym ROBOT_IN_POSITION. Oznacza to, że układ sterowania robota zaakceptował i realizuje polecenie przemieszczenia manipulatora do następnego punktu.

2. W momencie zakończenia przemieszczania manipulatora podprogram obsługi wybranego wyrobu ustawia „1" na wyjściu ROBOT_IN_POSITION. Jest to informacja dla aplikacji nadrzędnej o osiągnięciu następnego punktu, w którym możliwa jest realizacja kolejnej kontroli wizyjnej lub zmiana pozycji stołu obrotowego. Jeśli ruch odbywa się przez punkty pośrednie, to na wyjściu ROBOT_IN_POSITION podprogram obsługi wybranego wyrobu musi utrzymywać stan „0".

   Aplikacja nadrzędna komputera wizyjnego nie narzuca żadnych ograniczeń czasowych na zmianę punktu roboczego. W przypadku potencjalnego błędu w podprogramie obsługi wybranego wyrobu (np. brak instrukcji ustalającej wartość „1" sygnału ROBOT_IN_POSITION po zrealizowaniu instrukcji ruchu) może spowodować to konieczność ręcznego wyprowadzenia aplikacji nadrzędnej ze stanu oczekiwania na potwierdzenie dojścia do pozycji roboczej.

Z kilku powodów technicznych, które nie będą tutaj omawiane, aplikacja nadrzędna została tak zaprojektowana, że kolejne kontrole wizyjne i zmiany pozycji stołu obrotowego muszą być realizowane w różnych punktach zatrzymania manipulatora. Jeśli dwa (lub więcej) sąsiednie punkty robocze mają te same współrzędne i tę samą orientację narzędzia, to konieczne jest wycofanie manipulatora do punktu pośredniego i jego powtórne przemieszczenie do punktu roboczego przy wykonaniu pełnego cyklu sterowania sygnałami START_MOVING i ROBOT_IN_POSITION. Sam podprogram obsługi wybranego wyrobu nie odróżnia punktów roboczych, w których aplikacja nadrzędna realizuje albo kolejną kontrolę wizyjną albo przemieszcza stół obrotowy. Obie czynności mogą być wykonane tylko wtedy, gdy sygnał ROBOT_IN_POSITION = 1, ale dla ruchu stołu dodatkowo konieczne jest, aby wartość sygnału SAFE_ROBOT_POSITION także była równa „1".

Omówienia wymaga także sterowanie robotem po zakończeniu kontroli wizyjnych. Zależy ono od położenia ostatniego punktu roboczego, w którym aplikacja wykonywała kontrolę wizyjną:

— jeżeli punkt ten znajduje się poza strefą bezpieczną (tzn. sygnał SAFE_ROBOT_POSITION jest równy „0"), to aplikacja wymusza przemieszczenie manipulatora do punktu wewnątrz tej strefy w taki sam sposób, jak podczas jego przemieszczania związanego z wykonywaniem kontroli wizyjnych (podprogram obsługi wybranego wyrobu musi zatem zawierać odpowiednie instrukcje ruchu),
— jeżeli punkt ten znajduje się w strefie bezpiecznej (tzn. sygnał SAFE_ROBOT_POSITION jest równy „1"), to ostatni punkt roboczy jest traktowany przez aplikację nadrzędną komputera wizyjnego jako punkt spoczynkowy (wyjściowy), a zatem manipulator nie jest przemieszczany.

Ostatnią czynnością wykonywaną przez aplikację nadrzędną jest zatrzymanie programu aplikacyjnego robota. W tym celu ustawia ona „1" na wyjściu STOP_PROGRAMU_ROBOTA i utrzymuje ten stan do momentu, w którym oprogramowanie systemowe robota ustawi wartość „0" na wyjściu dwustanowym ROBOT_PROGRAM_RUNNING.

6.8. Budowa stanowiska i oprogramowanie narzędziowe

Pierwszą złożoną aplikacją kontroli wizyjnej zrealizowaną przez PIAP dla przemysłu maszynowego, sterowaną przez nadrzędny program komputera PC była praca wykonana dla Zakładów firmy „Eaton" z Tczewa. Wykorzystano w niej komputer przemysłowy firmy Advantech z systemem operacyjnym Windows, wyposażony dodatkowo w trzy specjalizowane karty:

— kartę akwizycji obrazu PCimage-SDIG firmy Matrix Vision GmbH,
— kartę 16-We/16-Wy dwustanowych OPTORE-PCI 16STANDARD firmy Wasco,
— kartę z 4 portami RS-232/RS-422/RS-485 firmy Advantech.

Wyposażenie stanowiska stanowił robot przemysłowy IRb-1400 firmy ABB, stół obrotowy TS002E firmy WEISS GmbH, kamera Basler A101p współpracująca z kartą akwizycji obrazu i czytnik kodu kreskowego Welch Allyn 3800 służący do wprowadzania odmiany skrzyni biegów, numeru montażowego i numeru seryjnego. Sterowanie stanowiskiem zrealizowano za pomocą sygnałów dwustanowych. Z czterech portów RS-232 stosowanej karty jeden był wolny (COM4), a pozostałe wykorzystano do obsługi czytnika kodu kreskowego (COM1), do programowania czasu ekspozycji kamery Basler (COM2) i do transmisji informacji o programie robota IRb-1400 (COM3). Podobną konfigurację sprzętową, z robotem firmy Kuka, ale bez stołu obrotowego wykorzystano do budowy stanowiska promującego PIAP na kilku targach branżowych. Bez robota i stołu obrotowego zbudowano dwa stanowiska pomiarowe dla firmy Iskra Zakłady Precyzyjne z Kielc, i stanowisko kontrolne dla zakładu Lear Corporation w Tychach (tego stanowiska ostatniecznie nie sfinalizowano).

Program sterujący stanowiska napisano w języku C/C++. W pierwszej wersji wykonanej dla firmy Eaton do jego uruchomienia wykorzystano środowisko Visual C++ v. 6.0. W kolejnych wersjach środowiskiem uruchomieniowym był Visual Studio .NET 2003. Tym oprogramowaniem posługiwano się tworząc pozostałe wymienione aplikacje.

7. Podsumowanie

W zapytaniach ofertowych, które PIAP otrzymuje od potencjalnych klientów, dotyczących możliwości zbudowania stanowisk kontroli wizyjnej często pojawia się żądanie, aby bez wykonywania dodatkowych i kosztownych modyfikacji była możliwość sprawdzania wyrobów, które w przyszłości dopiero wejdą do produkcji. W artykule przedstawiono dwa sposoby realizacji sterowania, które w mniejszym lub większym stopniu spełniają to żądanie. W obu przypadkach aplikacja sterująca wykonywaniem kontroli wizyjnej była tworzona w języku C lub C++, w obu wykorzystano pakiet NeuroCheck. Jednak osobie przystosowującej każde stanowisko do kontroli nowych wyrobów nie jest potrzebna ani znajomość języka C/C++ ani wiedza o środowisku Visual Studio, ponieważ wszystko, co potrzebne do prawidłowej pracy stanowiska, jest tworzone i konfigurowane poza samą aplikacją. Niezbędna wiedza obejmuje następujący zakres:

— znajomość środowiska NeuroCheck w zakresie tworzenie programów kontroli wizyjnych realizowanych przez ten pakiet,
— umiejętność tworzenia prostych programów robotowych zawierających instrukcje ruchu z punktu do punktu i obsługi We/Wy dwustanowych,
— ogólną znajomość baz danych w formacie Access 2000 w zakresie wpisywania i edycji informacji przy wykorzystaniu gotowych formularzy,
— wiedzę na temat tworzenia tzw. pliku konfiguracyjnego raportu na podstawie którego określany jest globalny wynik kontroli i generowany raport końcowy.

Oczywiście tam, gdzie nie zastosowano robota także wiedza na temat tworzenia jego programów aplikacyjnych jest zbędna. PIAP, poza szkoleniem z zakresu obsługi stanowiska, oferuje także kurs dotyczący tworzenia programów wizyjnych dla pakietu NeuroCheck oraz kurs tworzenia prostych aplikacji robotowych. W obu przypadkach szczegółowo omawiana jest budowa programów, które są wykorzystywane do kontroli wyrobów z bieżącej produkcji. W przypadku aplikacji wykonanej dla firmy Eaton jej przeszkoleni pracownicy byli w stanie opracować i wdrożyć oprogramowanie wizyjne i robotowe dla kontroli nowej skrzyni w przeciągu zaledwie dwóch dni.

Podziękowanie

Publikacja powstała jako wynik projektu ARIALE „Automatyzacja i robotyzacja dla nowej zreindustrializowanej Europy".

Projekt ten jest realizowany przy wsparciu finansowym Komisji Europejskiej w ramach programu „Uczenie się przez całe życie" („Lifelong Learning Programme"). Publikacja odzwierciedla jedynie stanowisko autora. Komisja Europejska nie ponosi odpowiedzialności za umieszczoną w niej zawartość merytoryczną ani za sposób wykorzystania zawartych w niej informacji

Bibliografia

1. Syryczyński A., Dunaj J., Szawłowski A., Wawerek Z., *Zintegrowany system wizyjnej kontroli wytwarzania*, „Pomiary Automatyka Robotyka", 12/2001.
2. Dunaj J., *Instrukcja obsługi stanowiska kontroli wizyjnej siedzeń samochodowych w Zakładach "Faurecia"*, materiały Przemysłowego Instytutu Automatyki i Pomiarów PIAP.
3. Dunaj J., *Opis działania i obsługa aplikacji „EATON" stanowiska do kontroli wizyjnej skrzyń biegów*, materiały Przemysłowego Instytutu Automatyki i Pomiarów PIAP.
4. Wawerek Z., *Machine vision, widzenie maszynowe albo…*, „Pomiary Automatyka Robotyka" 4/2008, 18–19.
5. Wawerek Z., Zabielski M., *Autonomiczne stanowisko kontroli wizyjnej detali*, „Pomiary Automatyka Robotyka", 4/2008, 26–28.
6. Dunaj J., Pachuta M., *Opis programu robota ABB IRb-140 i programu kamery SICK IVC-2D wykonanych na potrzeby Laboratorium Modelowania Elastycznych Linii Produkcyjnych Politechniki Poznańskiej*, materiały Przemysłowego Instytutu Automatyki i Pomiarów PIAP, 2013.
7. Dunaj J., *Opis programu demonstracyjnego robota Kuka KR16 na targi Automaticon 2014*, materiały Przemysłowego Instytutu Automatyki i Pomiarów PIAP, 2014.
8. *NeuroCheck Industrial Vision Systems – Programmer's Reference*, NeuroCheck GmbH, 1999.
9. *NeuroCheck Industrial Vision Systems – User-Manual*, NeuroCheck GmbH, 2002.
10. *NeuroCheck Machine Vision for Industry – Getting Started*, NeuroCheck GmbH, 1999.
11. *NeuroCheck Machine Vision for Industry – Training Course*, NeuroCheck GmbH, 1997
12. *NeuroCheck Machine Vision for Industry – User-Manual*, NeuroCheck DS GmbH, 1999
13. Bates J., Tompkins T., *Poznaj Visual C++*, Wydawnictwo MIKOM, 1999.
14. Leinecker R.C., Archer T., *Visual C++ 6 Vademecum profesjonalisty*, Wydawnictwo Helion, 2000.
15. Petzold Ch., *Programowanie Windows*, Wydawnictwo RM, Warszawa 1999.
16. *Microsoft Visual C++ 6.0 MFC Library Reference*, Microsoft Press 1998.
17. ABB Robotics, *User's Guide 3HAC 7793-1 For BaseWare OS 4.0.60*.
18. ABB Robotics, *Instrukcja obsługi IRC5 z panelem FlexPendant*.
19. KUKA Roboter GmbH, *KUKA System Software (KSS) – KR C2/KR C3 Configuration* (*External Automatic*), 2004.
20. KUKA Roboter GmbH, *KUKA System Software (KSS) – KR C2/KR C3 Expert Programming*, 2003.

Software Solutions in the Control Systems of Advanced Vision Applications

Abstract: The article contains information about the software solutions used in the control of advanced vision applications for the automotive industry realized in the Industrial Research Institute for Automation and Measurements PIAP. Author describes what criteria guided the selection of vision software, what must be the format of vision programs, how they are run in automatic mode, and how to define the interpretation of the results and a final report format. Two methods of configuring individual vision control of complex product depending on the control system used are presented. The article contains a description of the cooperation between the application controlled vision programs with industrial robot used to move the camera in one of the submitted application.

Keywords: vision control, NeuroCheck, PLC, PC application, industrial robot, database, ODBC drivers

mgr inż. Jacek Dunaj
jdunaj@piap.pl

W 1980 r. ukończył studia na Wydziale Elektrycznym Politechniki Warszawskiej, od 1985 r. jest zatrudniony w Przemysłowym Instytucie Automatyki i Pomiarów PIAP. Specjalizuje się w programowaniu mikroprocesorów, kontrolerów, sterowników i robotów przemysłowych, systemów wizyjnych a także komputerów PC programowanych w języku asemblera i C/C++ w środowisku różnych systemów operacyjnych. Współautor oprogramowania kilku urządzeń opracowanych w PIAP oraz wielu wdrożeń przemysłowych, w szczególności wymagających współpracy różnych urządzeń automatyki i wykorzystania oprogramowania biurowego (baz danych, arkuszy kalkulacyjnych).