System automatyki inteligentnego budynku, część 4

Ostatnią część cyklu poświęcamy ogólnemu przedstawieniu oprogramowania systemowego. Ze względu na ogromną objętość opisu, w drukowanej wersji EP publikujemy wyłącznie jego najistotniejsze fragmenty, a resztę wraz programami źródłowymi publikujemy na CD–EP3/2006B.

Realizacja oprogramowania systemu

Oprogramowanie systemu zostało napisane przy użyciu języka C i kompilatora KEIL. Język wysokiego poziomu został wybrany z powodu znacznego skomplikowania oprogramowania urządzeń oraz ze względu na powszechną jego znajomość wśród programistów. Stworzenie zbioru funkcji odwołujących się do sprzętu umożliwia zatem realizację pomostu pomiędzy programistą nie znającym struktury procesora (ani jego rozkazów asemblera), a sprzętem realizującym zadania systemu automatyki.

Ze względu na specyfikację budowanego systemu, oprogramowanie można podzielić na trzy współpracujące fragmenty. Częścią niezbędną dla działania całości jest wspólne dla wszystkich współpracujących komponentów oprogramowanie realizujące transmisję danych przy zachowaniu określonego protokołu. Kolejnym elementem jest zestaw programów służących do sterowania pracą urządzeń podrzędnych, natomiast najbardziej rozbudowaną częścią oprogramowania jest zestaw funkcji oraz prosty system operacyjny służący do budowy programów uruchamianych na sterowniku głównym i nadzorujących całą strukturą układu automatyki.

Oprogramowanie komunikacyjne

Po rozpoczęciu prac nad oprogramowaniem komunikacyjnym postanowiono skorzystać z gotowych rozwiązań standardowych protokołów wykorzystywanych w systemach automatyki. Założono, że wykorzystany będzie protokół MODBUS jako jeden z najprostszych, nie wymagający dodatkowych rozwiązań sprzętowych. Jednak próba przeniesienia algorytmu działania na mikrokontrolery wchodzące w skład systemu skończyła się niepowodzeniem. Jeszcze przed napisaniem kodu przeprowadzono wstępne obliczenia dotyczące głównie koniecznej do wykorzystania pojemności pamięci i okazało się, że zasoby oferowane przez najsłabsze zastosowane mikrokontrolery (układy AT89C2051) są niewystarczające do realizacji protokołu w pełni kompatybilnego z MODBUS, nie mówiąc już o zasobach niezbędnych do działania podstawowych funkcji urządzeń (poza komunikacją).

Możliwym wyjściem z sytuacji byłoby opracowanie protokołu zgodnego z MODBUS, ale tylko jednokierunkowo, np. poprzez ograniczenie maksymalnego rozmiaru ramki, eliminację niektórych trybów pracy. Postanowiono jednak pójść inną drogą i stworzyć własny, bardzo prosty protokół komunikacyjny, zupełnie jednak wystarczający dla potrzeb budowanego systemu, a równocześnie posiadający kilka unikalnych funkcji korzystnych przy komunikacji pomiędzy modułami. W ten sposób powstał protokół nazwany protokołem HMP (od inicjałów autorów).

Cechy podstawowe protokołu HMP

Opracowany protokół powstał z myślą nie tylko o budowanym systemie automatyki, ale także o innych urządzeniach (systemach) wymagających komunikacji. Z tego powodu główne procedury systemu zostały opracowane w sposób jak najbardziej uniwersalny, przejrzysty i łatwy do adaptacji do konkretnych zastosowań. Chociaż stosowane rozwiązania, takie jak: praca w podprogramie obsługi przerwania układu transmisji szeregowej i wykorzystanie specyficznych nazw rejestrów sterujących odpowiadają standardowemu mikrokontrolerowi 8051, to przystosowanie napisanych w języku C procedur do współpracy z innymi mikrokontrolerami kompatybilnymi z '51, czy też stanowiącymi zupełnie inną architekturę, nie stanowi większego problemu. Podobnie wprowadzenie do procedur obsługi rozkazów sterowania zewnętrznymi interfejsami transmisyjnymi (np. transceiverami RS–485 jak w opisywanym systemie) jest również bardzo proste.

Podczas opracowywania protokołu założono, że musi być on jak najprostszy algorytmicznie w celu ograniczenia rozmiaru potrzebnego kodu oraz koniecznej do zarezerwowania pamięci danych. Równocześnie podjęto decyzję, że tylko frag-

Protokół HMP charakteryzuje się zatem następującymi cechami:

- stała, ośmiobajtowa ramka transmisyjna,
- możliwość współpracy na jednej linii transmisyjnej 255 urządzeń, standardowo w konfiguracji Master–Slave,
- zabezpieczenie przed przekłamaniem transmisji,
- możliwość prowadzenia transmisji kodowanej,
- dowolna prędkość transmisji, zależna wyłącznie od ustalonych timeout'ów jak i od mocy obliczeniowej stosowanego mikrokontrolera, w prezentowanym systemie została wybrana prędkość wynosząca 57600 b/s,
- dowolny standard asynchronicznej transmisji szeregowej (minimum 8N1), a także możliwość adaptacji do innych interfejsów komunikacyjnych.

ment protokołu będzie niezmienny, wykorzystywany w każdym urządzeniu – chodzi tutaj o funkcje odpowiedzialne za samą transmisję jak i obróbkę odebranych danych i przygotowywanie ramek do wysyłki. Przyjęto zaś możliwość różnienia się między poszczególnymi urządzeniami części oprogramowania odpowiedzialnej za interpretację przesyłanych danych, z równoczesnym zastrzeżeniem kilku sytuacji, na które urządzenia systemu muszą reagować w ten sam sposób.

Jako zalety przedstawianego protokołu można wyróżnić:
- niewielka objętość kodu – dla mikrokontrolera 8051 główna część protokołu (bez interpretacji danych) zajmuje około 400 bajtów,
- duża możliwa liczba współpracujących urządzeń na jednej magistrali transmisyjnej, możliwa do rozszerzenia przez wykorzystanie dodatkowych bitów ramki dostępnych dla programisty,
- możliwość wysyłania komunikatów rozgłoszeniowych (do wszystkich układów danego podsystemu),
- możliwość zdalnej zmiany adresu urządzenia jak i klucza kodującego,
- możliwość dostosowania do systemów pracujących w standardzie *peer–to–peer*,
- przekazywanie danych przy pomocy systemu rejestrów, nie wymagające określenia nadawcy ramki.

Niestety prostota oprogramowania protokołu jest też okupiona pewnymi wadami, które w przypadku znacznej uciążliwości muszą być eliminowane w sposób odpowiedni dla danego systemu:
- brak kontroli odbioru ramki przez adresata,
- brak możliwości wykrycia kolizji na magistrali transmisyjnej, w takiej sytuacji ramki nadających jednocześnie urządzeń zaginą,
- przekazywanie danych przy pomocy rejestrów wiąże się z koniecznością rezerwacji dodatkowych komórek pamięci, dostępnych niejako z zewnątrz urządzenia i przechowujących parametry jego pracy,
- w przypadku urządzenia Master brak możliwości nadania jednym ciągiem ramek do kilku odbiorców, gdyż po nadaniu każdej ramki wymagającej odpowiedzi konieczne jest zachowanie czasu niezbędnego na odpowiedź urządzenia,
- prosty algorytmicznie system kodowania i zabezpieczenia przed przekłamaniami, odbiegający skutecznością od obecnie powszechnych rozwiązań, nie wymagający jednak złożonych obliczeń,
- znacznie utrudniona możliwość stosowania różnych prędkości transmisji różnych podsystemów pracujących przy użyciu tego samego medium transmisyjnego, wiążąca się z dużą ilością występujących timeout'ów wywoływanych wskutek błędnie odebranych bajtów ramek.

Opierając się na zaletach tego protokołu oraz uwzględniając wpływ jego wad można stworzyć skuteczny system transmisji danych w układzie wielu współpracujących ze sobą urządzeń.

Działanie protokołu HMP i obsługa od strony programu użytkownika

Działanie protokołu HMP w znacznym stopniu jest niezależne od programu użytkownika, gdyż główny algorytm realizowany jest w podprogramie obsługi przerwania. Uwaga ta dotyczy zwłaszcza procesu odbioru ramek, gdyż nie wymaga on żadnej ingerencji programu głównego.

Po starcie systemu licznik bajtów odbieranych *HMP_LICZNIK_RX* jest zerowany, włączany jest system przerwań i układ przechodzi do nasłuchu magistrali szeregowej. Odebranie jakiegokolwiek bajtu przez port szeregowy powoduje jego zapisanie w buforze *HMP_BUFOR_ODB* na pozycji o indeksie 0. Równocześnie inkrementowany jest licznik bajtów odbieranych. Odbiór kolejnych bajtów na magistrali wiąże się z zapisem ich w buforze na kolejnych pozycjach.

Zabezpieczeniem przed zawieszeniem się funkcji odbiorczej wskutek np. przerwy czy zwarcia na linii transmisyjnej, odbiorem niepełnej ramki czy też nieprawidłowym jej uformowaniem przez urządzenie współpracujące jest licznik timeout *HMP_TIMEOUT*. Jest on ustawiany na wartość początkową zdefiniowaną jako stała pod etykietą *HMP_TIMEOUT_WARTOSC* po odebraniu każdego bajtu ramki. Wyzerowanie tego licznika przed nadejściem kolejnego bajtu spowoduje zresetowanie funkcji odbiorczej i przejście do oczekiwania na pierwszy bajt ramki.

Po odebraniu pełnej ramki danych (ośmiu bajtów) podprogram obsługi protokołu przechodzi do obróbki odebranych informacji.

W pierwszym kroku realizowana jest analiza informacji zawartej na dwóch najmłodszych bitach bajtu typu ramki. W zależności od kombinacji bitów K1 i K0 funkcja obsługi przechodzi (lub nie) do rozkodowania następnych siedmiu bajtów przesyłanej informacji. Operacja ta wykonywana jest wprost na buforze *HMP_BUFOR_ODB*, co powoduje zamazanie niepotrzebnej już oryginalnej zawartości ramki.

Po rozkodowaniu danych transmitowanych w ramce dokonywane jest sprawdzenie bajtu adresu odbiorcy. Dalsze realizowanie obróbki danych uzależnione jest od wartości zapisanej w tym bajcie ramki. Przejście do kolejnego kroku analizowania ramki następuje w dwóch i tylko dwóch przypadkach: dzieje się tak wtedy, gdy adres przesyłany w ramce jest równy bieżącemu adresowi urządzenia, które ramkę odebrało lub też w sytuacji, w której odebrany adres równy jest zero – oznacza to odebranie ramki rozgłoszeniowej, przeznaczonej do wykonania przez wszystkie urządzenia działające w danej sieci. Należy w tym miejscu również wspomnieć, że ramka rozgłoszeniowa nigdy nie podlega kodowaniu, gdyż w przypadku ustawienia różnych kluczy kodowych w różnych urządzeniach, tego typu transmisja przestałaby pełnić swoją funkcję, gdyż nie dotarłaby do wszystkich urządzeń. Natomiast kodowanie poprzez adres odbiorcy nie ma sensu, gdyż wykonanie operacji bitowej sumy modulo 2 z bajtem równym zero daje w rezultacie bajt wejściowy – więc kodowanie nie pełni swojej funkcji ochronnej przed niepowołanym odczytem.

Ostatnim krokiem w procedurze obróbki odebranej ramki jest obliczenie bajtu sumy kontrolnej. Realizacja tego obliczenia daje w wyniku bajt kontrolny, który pozostaje porównać z bajtem sumy przesłanym w ramce. Zgodność wartości obliczonej lokalnie i przesłanej z urządzenia współpracującego decyduje o przekazaniu odebranych informacji do programu głównego.

Przekazanie odebranych danych do programu głównego odbywa się za pomocą tablicy *HMP_DANE_ODB* i zmiennej *HMP_TYP_ODB*. W tablicy zapisywane są następujące informacje (pod indeksami według podanej kolejności): 0 – bajt adresu, 1 – bajt numeru rejestru, 2, 3, 4, 5 – bajty danych (od najmłodszego). Zmienna *HMP_TYP_ODB* jest zapisywana natomiast zawartością odebranego bajtu typu ramki. Ponieważ bajt typu ramki nie może przyjąć wartości zerowej, to zawartość omawianego rejestru różna od zera informuje program główny danego urządzenia, że nadeszła nowa ramka, została ona prawidłowo zdekodowana i należy przystąpić do analizy transmitowanych nią danych. Oczywiście programista piszący program główny musi zadbać o wyzerowanie bajtu *HMP_TYP_ODB* po analizie danych z bufora, w przeciwnym razie nie będzie możliwe prawidłowe wykrycie nowych danych.

Sposób obróbki odebranych danych zależy wyłącznie od programu głównego i jest niezależny od działania procedur protokołu, określono jednak pewne funkcje, które muszą być przez wszystkie urządzenia systemu wykonywane w ten sam sposób. Ich rozróżnienie polega przede wszystkim na rozróżnianiu numerów rejestrów.

Pierwszą funkcją, która powinna być realizowana jednakowo we wszystkich współpracujących urządzeniach jest zapis danej do rejestru o numerze zero. Wykonanie takiej operacji spowoduje przywrócenie urządzeniu odbiorczemu jego adresu domyślnego, ustawianego po włączeniu zasilania. Dane transmitowane w ramce nie mają w tym wypadku znaczenia informacyjnego, jednak pełnią funkcję ochronną ze względu na konieczność zabezpieczenia przed przypadkowym przywróceniem adresu domyślnego. Wszystkie cztery bajty danych muszą zawierać powtórzony bajt numeru rejestru (skopiowana zawartość bajtu 4 ramki). Powiązany z tą funkcją jest jej odpowiednik dotyczący odczytu: odebranie ramki żądania odczytu danych z rejestru 0x00 spowoduje odesłanie przez to urządzenie do urządzenia Master adresu domyślnego, bez zmiany adresu obecnie obowiązującego. Obie powyższe funkcje mogą być przydatne w sytuacji przeprogramowania układu Master lub też po zaniku zasilania.

Kolejne funkcje polegają na obsłudze odczytu i zapisu rejestru o numerze 0x01. Zapis danych do tego rejestru spowoduje zachowanie pierwszego bajtu danych ramki jako nowego, obowiązującego od tej chwili adresu urządzenia. Dodatkowym zabezpieczeniem przed zmianą adresu na przypadkowy jest konieczność zapewnienia, aby wszystkie bajty danych ramki zawierały tą samą informację (czyli nowy adres). Odczyt danych z tego rejestru (0x01) spowoduje odesłanie do urządzenia Master ramki, która na wszystkich czterech pozycjach danych będzie przekazywać aktualnie obowiązujący adres urządzenia.

Ostatnimi funkcjami, których implementacja jest zalecana dla każdego komponentu systemu są funkcje odczytu i zapisu rejestru o numerze 0x02. Przy zapisie danych do tego rejestru układ odbierający ramkę powinien zmienić klucz kodowania na zawartość zapisaną w pierwszym bajcie danych (i powtórzoną dla bezpieczeństwa w pozostałych). Powtórzenie przekazywanej informacji we wszystkich bajtach danych także tutaj zabezpiecza układ przed przypadkową zmianą klucza. Odczyt informacji z omawianego rejestru spowoduje odesłanie w odpowiedzi ramki zawierającej na wszystkich pozycjach.

Odsyłanie odpowiedzi na wszystkie z omawianych rejestrów standardowych powinno się odbywać do układu nadrzędnego pod wybrany właśnie numer rejestru. Rozpoznanie z którego układu podrzędnego pochodzą informacje leży po stronie urządzenia Master.

Powyżej opisane specjalne znaczenia poszczególnych rejestrów stanowią oczywiście umowny sposób ich traktowania, jednak taki jaki jest wykorzystany w omawianym systemie automatyki. Ze względów praktycznych nie zaleca się implementowania tych funkcji w sterowniku Master, gdyż przypadkowa zmiana adresu czy klucza kodowego dla układu nadrzędnego mogłaby unieruchomić cały system.

W przypadku odebrania ramki z transmisją danych nie opisanych w żaden specjalny sposób, układ odbiorcy powinien przejść do analizy otrzymanej ramki. Odebranie ramki zapisu rejestru powinno zakończyć się przekazaniem bajtów danych do określonej komórki pamięci i przejściem do oczekiwania na kolejną ramkę. W tym przypadku nie jest wymagana żadna odpowiedź wysyłana do sterownika nadrzędnego, chociaż taka możliwość istnieje w zależności od wymagań programisty systemu.

Odbiór przez sterownik podrzędny ramki z rozkazem odczytu rejestru zobowiązuje go do utworzenia i odesłania ramki odpowiedzi. Adresem odbiorcy takiej ramki powinien być adres układu nadrzędnego, choć możliwe jest również wykorzystanie urządzenia podrzędnego jako swego rodzaju przekaźnika, mogącego przesłać informację do innego urządzenia podrzędnego. Jako dane należy oczywiście wysłać zawartość zaadresowanego rejestru. Najlepiej w taki sposób zorganizować oprogramowanie układu nadrzędnego, aby odsyłanie danych odbywało się do takiego samego numeru rejestru z jakiego zostało pobrane. Równoczesne przyjęcie unikalnych numerów rejestrów dla wszystkich układów podrzędnych umożliwia łatwe stworzenie ich kopii w pamięci sterownika nadrzędnego. Stworzony w ten sposób system komunikacyjny może się stać przezroczysty dla programu głównego, gdyż zapis lub odczyt konkretnej zmiennej będzie równoznaczny z odczytem parametru pracy urządzenia odległego. Tego typu rozwiązanie zastosowano w oprogramowaniu prezentowanego systemu automatyki.

Czas obróbki danych potrzebny na sformułowanie i odesłanie odpowiedzi zmusza układ Master na oczekiwanie na odpowiedź przed nadaniem kolejnej ramki, bądź to w celu uniknięcia przepełnienia bufora odbiorczego, bądź też zapobieżenia kolizji z odpowiedzią urządzenia podrzędnego w przypadku korzystania z magistrali przystosowanej do pracy simpleksowej.

Wysyłanie informacji za pomocą protokołu HMP jest równie proste jak jej odbiór. W pierwszym kroku należy zadbać o prawidłowe przygotowanie transmitowanych danych. Dokonuje się tego za pomocą tablicy *HMP_DANE_NAD*, w której należy umieścić informacje przeznaczone do wysyłki. Znaczenie poszczególnych pozycji tablicy jest identyczne jak dla *HMP_DANE_ODB*, więc pod indeksem zerowym należy umieścić adres odbiorcy, pod indeksem 1 numer rejestru, natomiast pod indeksami od 2 do 5 dane przeznaczone

do transmisji. Drugim ważnym elementem jest wpisanie do zmiennej *HMP_TYP_NAD* bajtu typu ramki, gdyż tylko na podstawie tej informacji będzie możliwe prawidłowe zakodowanie i wysłanie danych.

W celu przeprowadzenia transmisji ramki należy zadbać o to, aby w pętli głównej programu była wywoływana funkcja *HMP_nadaj()*. Jej wywołanie spowoduje zainicjowanie transmisji ramki na podstawie danych zawartych w omawianych powyżej zmiennych po spełnieniu pewnych warunków:

- układ odbiornika nie znajduje się w stanie odbioru ramki (zmienna *HMP_LICZNIK_RX* musi być równa zero),
- układ nadajnika nie nadaje wcześniejszej ramki (*HMP_LICZNIK_TX*=0),
- zawartość wpisana do zmiennej *HMP_TYP_NAD* jest różna od zera.

Po zainicjowaniu procesu transmisji ramki oprogramowanie protokołu samo zadba o prawidłową realizację transmisji. Zakończenie nadawania jednego z bajtów powoduje zgłoszenie przerwania układu transmisji szeregowej, co prowadzi do nadania kolejnego, aż do kompletnej emisji całej ramki. Zakończenie nadawania może być wykryte poprzez odczyt zmiennej *HMP_TYP_NAD*, która jest zerowana po transmisji ostatniego bajtu ramki.

Możliwości niestandardowego wykorzystania protokołu

Jak widać z opisu protokołu HMP zamieszczonego w poprzednich podpunktach, posiada on duże możliwości konfiguracyjne dostępne dla programisty. Przede wszystkim opracowany standard nie określa jednoznacznie sposobu interpretacji przesyłanych danych (cztery bajty informacji) ani też niewykorzystanych bitów bajtu typu ramki (sześć bitów). W praktyce umożliwia to stworzenie na bazie oprogramowania HMP dowolnego protokołu komunikacyjnego, mogącego adresować więcej urządzeń, obsługiwać więcej rejestrów, lub posiadającego inne niestandardowe funkcje jakie w danej chwili mogą być potrzebne.

Ilość danych transmitowanych jednocześnie w jednej ramce (4 bajty) jest zupełnie wystarczająca do zastosowań w systemach z mikrokontrolerami. Nic więc nie stoi na przeszkodzie, aby jeden z bajtów wykorzystać jako nagłówek czy słowo konfiguracyjne własnego standardu komunikacyjnego. Skorzystanie z oprogramowania protokołu HMP uwalnia w ten sposób programistę ze sprawdzania poprawności danych i adresowania, a przede wszystkim pozwala na pozbycie się uciążliwości związanych z obsługą sprzętu, co jest nie bez znaczenia dla osób nie mających obeznania z daną rodziną mikrokontrolerów.

Podsumowując można stwierdzić, że protokół HMP jest wart stosowania w prostych systemach opartych o mało wydajne mikrokontrolery, na tyle jednak skomplikowanych, że wymagających ustalonego standardu przesyłania informacji. Protokół ten również świetnie sprawdzi się w układach bardziej rozbudowanych, gdzie nie ma konieczności stosowania protokołów powszechnie stosowanych w automatyce, a także w sieciach, gdzie wymagana jest poufność przesyłanych danych, zapewniona przynajmniej na podstawowym poziomie (np. poprzez zastosowane tutaj kodowanie XOR).

Możliwości rozbudowy systemu

Prezentowany system posiada znaczne możliwości rozbudowy, mogące przyjmować różnego rodzaju kierunki. Możliwości rozbudowy sprzętu prezentowanego systemu są bardzo duże. Oprogramowanie komunikacyjne pozwala na bezkolizyjną obsługę w bezpośredni sposób 254 urządzeń (plus sterownik nadrzędny), natomiast możliwość realizacji na drodze programowej obsługi elementów protokołu HMP dostępnych dla użytkownika pozwala powiększyć tą liczbę nawet 64–krotnie. W praktyce obsługa tak dużej liczby urządzeń (ponad 16 tysięcy niezależnych modułów podrzędnych) może się stać jednak utrudniona ze względu na ograniczenia kanału komunikacyjnego. Niebagatelne znaczenie miałby również sposób dostarczenia napięcia zasilania do tak dużej liczby urządzeń.

Całkowicie niezależny sposób działania poszczególnych układów podrzędnych pozwala także na stworzenie układów kaskadowych, w której główny sterownik nadrzędny będzie sprawował nadzór nad określoną liczbą sterowników podrzędnych, natomiast te sterowniki będą kontrolować pracę sterowników położonych jeszcze niżej w hierarchii, obsługiwanych przez niezależną linię transmisyjną. Rozwiązanie takie pozwala na stworzenie sieci sterowania wieloma obiektami (mieszkaniami, procesami technologicznymi), obsługiwanymi niezależnie od siebie. Struktura taka daje również znaczną poprawę niezawodności całego systemu.

Na podobnej zasadzie jak rozbudowa liczby sterowników podrzędnych, możliwe staje się również zrealizowanie rozbudowy rodzaju i liczby kanałów komunikacyjnych. Poprzez sterowniki stanowiące pomost pomiędzy różnymi mediami transmisyjnymi, w łatwy sposób można zastosować w prezentowanym systemie zarówno innego rodzaju przewodowe kanały transmisji, jak i komunikację realizowaną niemal w dowolny sposób.

Jako przykład można podać tutaj komunikację za pomocą fal radiowych, komunikację z wykorzystaniem światła (podczerwień, widzialne światło laserowe), czy też zastosowanie modułów sprzęgających system z lokalną siecią komputerową, internetem, czy też standardową siecią telefoniczną.

Możliwości prezentowanego systemu automatyki mogą zostać znacznie rozbudowane poprzez opracowanie odpowiedniego oprogramowania przeznaczonego dla komputera PC. Oprogramowanie takie mogłoby pełnić różnego rodzaju funkcje zależne od potrzeb użytkownika. Jako przykłady można podać:

- możliwość wizualizacji stanów pracy systemu i zmiennych systemowych,
- możliwość zmiany parametrów pracy systemu poprzez oprogramowanie,
- możliwość opracowania wizualnego języka programowania, w którym program dla sterownika nadrzędnego budowano by z bloków funkcyjnych pełniących różne zadania.

Wykorzystując komputer PC do stałego nadzoru nad pracą systemu istnieje również możliwość eliminacji z układy sterownika nadrzędnego i przerzucenia jego zadań na komputer. Efektem takiego rozwiązania byłby znaczny wzrost możliwości systemu, gdyż w porównaniu do najszybszych mikrokontrolerów, możliwości współczesnych komputerów PC są praktycznie nieograniczone.

Paweł Hadam