Wstęp

Minęły bezpowrotnie czasy, kiedy konstruowanie układów elektroniki z wykorzystaniem mikrokontrolera było albo bardzo trudne, albo wręcz niemożliwe, ze względu na znikomy dostęp do tego typu urządzeń. Nie każdy mógł zdobyć mikrokontroler z rodziny np. ’51, a ten, komu to się udało, zaliczany był do grona prawdziwych szczęściarzy. Dzisiaj praktycznie w każdym sklepie elektronicznym można kupić tego typu podzespoły z oferty szerokiego grona producentów. Układy te są dzisiaj ogólnodostępne i bardzo tanie, a jednocześnie ogromnie zwiększyły się ich możliwości.

Poza mikrokontrolerami, przystępne cenowo stały się również inne podzespoły – takie jak: wyświetlacze LCD, cyfrowe czy analogowe czujniki temperatury, przetworniki C/A i A/C, bez których trudno wyobrazić sobie konstruowanie bardziej rozbudowanych i profesjonalnych urządzeń.

Należy dodać, że wybór narzędzi służących do programowania mikrokontrolerów jest dużo większy niż dawniej.

Ponieważ pojemność pamięci Flash przeznaczonej dla programu, oferowana przez dzisiejsze mikrokontrolery jest bardzo duża przy znikomej cenie, wykorzystanie jej całej jest trudne (nawet w przypadku zaawansowanych programów i narzędzi oferujących niski stopień optymalizacji kodu). Wobec tego nie trzeba tak bardzo zwracać uwagi na to, by kod wynikowy programu zajmował mało pamięci, a zatem nie ma przymusu, by mikrokontrolery programować w asemblerze.

Każdy system tworzony z wykorzystaniem elektroniki zwyczajowo musi mieć jakieś dane wejściowe i wyjściowe, z których te drugie zazwyczaj są w jakimś stopniu zależne od tych pierwszych. Danymi czy sygnałami wejściowymi mogą być: sygnały cyfrowe pochodzące z innych układów elektronicznych, sygnały otrzymywane z wszelkich przycisków i przełączników, sygnały analogowe lub cyfrowe z czujników różnych wielkości fizycznych. I tutaj pojawiają się pewne ograniczenia.

Pierwszym z nich jest wydajność lub moc obliczeniowa naszego mikrokontrolera, zwłaszcza gdy jest to typowy 8-bitowiec. Mamy oczywiście możliwość zakupu układów takich jak procesory sygnałowe, które potrafią dokonywać obróbki danych w bardzo krótkim czasie – praktycznie w czasie rzeczywistym. W tej książce będziemy jednak zajmować się tylko i wyłącznie 8-bitowymi mikrokontrolerami AVR.

Druga sprawa to prezentacja danych. Dostępny asortyment jest bogaty i czasami w zupełności wystarczający. Mamy przecież popularne wyświetlacze LED, alfanumeryczne czy graficzne wyświetlacze LCD, które w porównaniu z latami ubiegłymi, są coraz tańsze i bardziej dostępne. Oprócz tego każdy może pokusić się o własną prezentację danych wyjściowych, np. na monitorze komputerowym. Czasem wystarczą zwykłe diody LED czy sygnały dźwiękowe generowane przez buzzer piezoelektryczne.

To jednak nie wszystko. Ważną sprawą są opcje związane ze sterowaniem czy programowaniem naszego urządzenia i mamy tutaj na myśli informacje, które będą istotne podczas pracy mikrokontrolera. Wyobraźmy sobie zwykły termoregulator. O ile informacja o wartości temperatury będzie odczytywana przez mikrokontroler praktycznie bezpośrednio z czujnika temperatury (pomijamy sytuację wykorzystania np. zewnętrznego przetwornika A/C), o tyle progi temperatury, przy których nastąpi jakaś reakcja na nie – np. zwarcie styków przekaźnika czy wysterowanie tranzystora, zapewne przekazywana będzie do termoregulatora z zewnątrz.

W najprostszym przypadku, informacje te można zapisać „na sztywno" w kodzie programu naszego mikrokontrolera, jednak chyba każdy zgodzi się, że jest to rozwiązanie mało uniwersalne, zwłaszcza w przypadku, gdy chcemy np. sprzedawać stworzony przez nas termoregulator. Nawet w jednostkowym przypadku to proste rozwiązanie jest mało komfortowe, bo niestety każda zmiana wartości progowych temperatury będzie się wiązać z przeprogramowywaniem mikrokontrolera po kompilacji zmodyfikowanego programu.

Podobne jak w przypadku podzespołów służących wizualizacji czy przekazywaniu informacji, również tutaj mamy w czym wybierać. Nasz termoregulator możemy wyposażyć przecież w najprostszą klawiaturę matrycową, ba, jeśli tylko ktoś będzie miał takie życzenie, nawet klawiaturę komputerową, różnego typu przełączniki albo przyciski.

Trzecią kwestią jest coraz częstsza potrzeba zapisywania zbieranych danych, archiwizowania informacji (takich jak choćby przebieg pracy naszego urządzenia), by w każdym praktycznie momencie można było sprawdzić, co działo się w interesującym nas przedziale czasu. Musimy tu liczyć się z koniecznością istnienia w naszym układzie zegara czasu rzeczywistego. Mamy tutaj do wyboru rozmaite dostępne rozwiązania: pamięci EEPROM (zewnętrzne lub wewnętrzne – wbudowane w mikrokontroler), pamięci Flash, karty SD/MMC oraz CF.

Zanim zamkniemy ten rozdział, należy zadać sobie pytanie – czy to wszystko?

Odpowiedź jest jednoznaczna – NIE!

Mamy przecież jeszcze komputer, dostępny dzisiaj praktycznie w każdym domu i w wielu innych miejscach! Nawet jeśli domowy komputer ma już swoje przeznaczenie, to przy obecnych możliwościach poradzi sobie z jeszcze jedną, maleńką aplikacją. Jedyne, co musimy zrobić, to zapewnić komunikację pomiędzy tworzoną przez nas „elektroniką" a tymże komputerem i można to zrobić w najtańszy z możliwych sposobów – poprzez interfejs szeregowy RS232C.

Jeśli Twój komputer, Czytelniku, nie jest w niego wyposażony (czemu coraz mniej można się dzisiaj dziwić), to się nie załamuj – w ofercie prawie każdego sklepu komputerowego kupisz kartę rozszerzającą Twój komputer nawet o 4 porty albo konwerter USB<->RS232C. Pierwsze rozwiązanie zapewnia większą kompatybilność sprzętową, co ma czasem istotne znaczenie.

Komputer, jako bardzo rozbudowane urządzenie, poszerzy nam funkcjonalność naszych układów. Będzie to miało również bezpośrednie przełożenie na koszt ich budowy. Jeśli mamy już komputer i nie trzeba go specjalnie kupować, to dlaczego go nie wykorzystać? W najprostszym wypadku zaoszczędzimy choćby na samych

przełącznikach, które możemy zastąpić przyciskami umieszczonymi na formatce naszego programu pracującego na komputerze. Przy założeniu, że będziemy pracować z wykorzystaniem darmowego środowiska programistycznego, jedyny koszt to czas, który poświęcimy na napisanie programu.

Zanim przejdziemy dalej, podzielimy tworzone przez nas systemy na dwie grupy, w zależności od funkcji, jaką będzie pełnił komputer.

Pierwsza grupa to systemy wspomagane przez komputer z ograniczeniem funkcjonalności aplikacji komputerowej jako narzędzia służącego jedynie programowaniu (zadawaniu nastaw czy też pewnych, wymaganych parametrów) naszych układów elektronicznych.

W tym przypadku rola komputera ograniczy się do jednorazowego przekazania pewnych informacji np. granicznych wartości dla naszego termoregulatora. Po zaprogramowaniu termoregulator rozpocznie samodzielną pracę.

Druga grupa to systemy, w których komputer zostanie wykorzystany zarówno jako narzędzie do programowania – podobnie jak poprzednio oraz jako narzędzie do wizualizacji, analizy czy rejestracji danych.

Jeśli komputer zostanie przewidziany na stałe jako nieodłączny element tworzonych przez nas narzędzi, to dlaczego nie mielibyśmy obarczyć go możliwie jak największą liczbą zadań, z którymi poradzi sobie przecież znakomicie?

Tego typu zestawienia są bardzo korzystne, a czasem wręcz nieuniknione.

Tak więc – zaczynamy!

Prezentowane przykłady programów dla mikrokontrolerów AVR napisano z wykorzystaniem pakietu Bascom-AVR w wersji 1.11.9.3. Testową wersję tego narzędzia można pobrać z działu Download ze strony producenta *http://www.mcselec.com.*

Prezentowane przykłady programów dla komputera napisano z komercyjnej i bezpłatnej wersji Visual Basic Express 2008, którą pobrać można bezpośrednio ze strony *http://www.microsoft.com/express/vb/Default.aspx.*

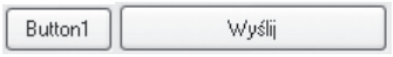
Przed przystąpieniem do programowania, zaleca się również instalację Microsoft NET.Framework 1.1 + Service Pack1, który może być konieczny do poprawnej pracy dołączonego programu testowego.

W trakcie omawiania programów dotyczących VB Express trudno opisywać kolejno wszystkie czynności, które składają się na powstanie konkretnego programu. Autor zakłada, że Czytelnik ma wystarczającą wiedzę z zakresu użytkowania Bascoma AVR oraz środowiska Visual Basic Express 2008.

Aby przedstawiane tutaj listingi mogły w pełni współgrać z tworzonymi przez nas projektami, wszystkie nazwy użytych komponentów muszą być poprawnie napisane.

By uniknąć wszelkich rozbieżności, zwłaszcza na początku przed nowo zakładanym projektem, będzie przedstawiona tabelka komponentów użytych w projekcie wraz z opisem rodzaju komponentu (*Button*, *CobmoBox* itp.). W tabelce będą także podane zmiany, które wystąpiły w komponentach.

Przykład: na formatce pojawi się przycisk sterujący *Button*. Jak wiadomo, zaraz po umiejscowieniu go na formatce przyjmuje on swoje standardowe ustawienia, takie jak nazwa czy opis. Jeśli chcemy, by tworzony program, będący pewnego rodzaju „interfejsem użytkownika", stał się bardziej czytelny, wszystkim komponentom nadamy nowe nazwy i nowy opis, aby potencjalny użytkownik wiedział, jakie działanie kryje się pod danym komponentem, np. *Button1* zmieni swoją nazwę na *Send*, a jako opis będzie *Wyślij*.



Typ komponentu	Właściwości	Opis zmiany
Button	*Name*	*Send*
	Text	*Wyślij*

Przyjęcie takiej konwencji pozwoli zaoszczędzić sporo czasu, który byłby potrzebny na wyszukiwanie potencjalnych błędów, wynikających z różnic nazewnictwa. W docelowych przykładach większość zmian znajdzie się już w strukturze samego programu. Jedyną czynnością Czytelnika będzie rozmieszczenie elementów wg własnego zapotrzebowania. Pomocne mogą okazać się dołączone do projektów „zrzuty" obrazów formatek. Alternatywą dla leniuszków będzie po prostu otwarcie gotowego projektu.

Wszystkie projekty prezentowane w książce można pobrać ze strony: *www.btc.pl/pliki/rsavr.zip*.