

UniSterownik część 1 – Dekoder kodu RC-5

Seria projektów **UniSterownik** przeznaczona jest głównie dla tych, którzy do tej pory nie odważyli się na kontakt z mikroprocesorami. W artykule zaproponowany jest interesujący sposób przełamania lodów. Mianowicie nie trzeba pisać kodu – programu dla mikroprocesora. Można ściągnąć z Elportalu gotową, przykładową wersję kodu. Ale lepiej będzie samodzielnie wygenerować „własny” kod na komputerze PC – Autor udostępnił stosowny program. W pierwszej części artykułu szeroko opisany został sposób wpisywania gotowego kodu do procesora.

Dlaczego warto dowiedzieć się, jak zaprogramować mikroprocesor i jak pracują proste układy mikroprocesorowe?

Otóż mikroprocesory są układami niezwykle uniwersalnymi, które poprzez zmianę programu mogą zamieniać dany układ w różne urządzenia. Mikrokontrolery pełnią różnorodne funkcje, zależnie od oprogramowania i możliwości platformy – układu, w którym pracują. Przykładem prostej, uniwersalnej platformy sprzętowej jest prezentowany tu „UniSterownik”, sterowany „malutkim” mikrokontrolerem ATTiny13. Dzięki przygotowanym, gotowym programom, każdy Czytelnik może bez trudu przekształcić UniSterownik w układ pełniący funkcje efektów świetlnych, regulatora temperatury, zdalnego sterowania, wskaźnika napięcia czy „inteligentnego” włącznika i to bez użycia lutownicy!

Idea UniSterownika polega na tym, że na początek montujemy płytkę z układem, a potem będziemy zmieniać jego funkcje i przeznaczenie wyłącznie poprzez zmianę programu zawartego w mikrokontrolerze. Istotnym uzupełnieniem będą elementy wykonawcze, podłączane do wyjść płytki: diody LED, silniki, przekaźniki itp. Nasz układ będzie się zmieniał w coraz to inne urządzenie bez potrzeby przerabiania płytki. Co ciekawe, programy dla mikrokontrolera będą generowane przez program *UniSterownik RC5*, uruchomiony na komputerze PC, a użytkownik będzie mógł w prosty sposób dostosować program do swoich upodobań. Wygenerowanym kodem (plik .HEX) będzie trzeba zaprogramować mikrokontroler ATTiny13. Po zamontowaniu zaprogramowanego mikrokontrolera w układzie, UniSterownik będzie pełnił nowe funkcje zgodnie z nowym programem.

Jestem przekonany, że seria artykułów o zastosowaniach UniSterownika zachęci Czytelników do poszerzania swojej wiedzy na temat mikrokontrolerów i sposobów ich stosowania. Mogę zapewnić, że po pokonaniu pierwszych problemów związanych z

zaprogramowaniem mikroprocesora, wielu początkujących elektroników „połknie bakcyła” i będzie konstruować układy zawierające mikroprocesory. Na początek trzeba nauczyć się, jak wpisać gotowy program do mikrokontrolera. A to nie jest trudne, jak pokazuje dalsza część artykułu.

Opis układu

Schemat UniSterownika pokazany jest na **rysunku 1**. Układem steruje mikrokontroler ATTiny13. Oczekuje on na sygnał ze scalonego odbiornika podczerwieni **US_IR**. Nie musimy się zagłębiać w szczegóły. W układzie docelowym mogą być zastosowane różne scalone odbiorniki podczerwieni na 36kHz, np. TFMS5360, SFH5110-36, TSOP1736 lub TSOP31236. Szczególnie polecam układ TSOP31236, ponieważ może on pracować w szerokim zakresie napięć zasilania od 2,5V do 5V. Procesor po odebraniu z wyprowadzenia **PB4** prawidłowego sygnału RC5, dekoduje go i zgodnie z otrzymanym rozkazem steruje wyjściami. Niemniej wejście **PB4**, połączone ze złączem **Z_WE**, jest wejściem uniwersalnym i zależnie od konfiguracji programowej przeznaczone jest do odbioru dowolnych sygnałów cyfrowych lub analogowych.

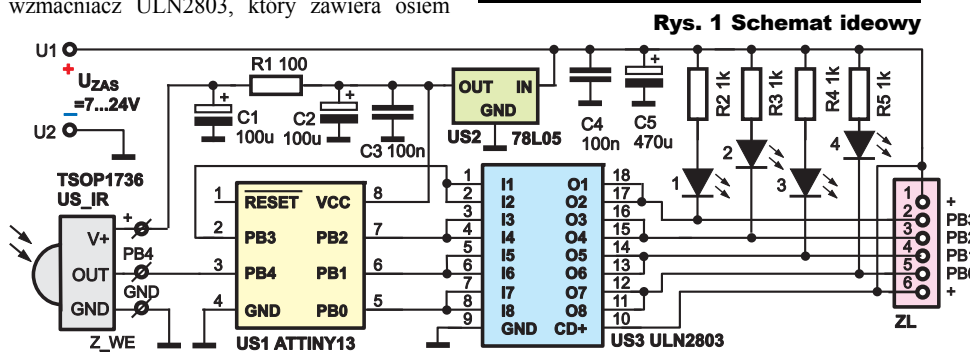
O działaniu układu decyduje program wpisany do mikrokontrolera. Wyjściami sterują wyprowadzenia **PB0**, **PB1**, **PB2** i **PB3**. Sygnały z tych wyprowadzeń są wzmacniane poprzez wzmacniacz **ULN2803**, który zawiera osiem

tranzystorów w układzie z otwartym kolektorem o wydajności 0,5A i napięciu do 50V. Przy połączeniu dwóch takich wyjść sterowany prąd może wynosić do 1A. Wyjścia wyprowadzone są na złącze **ZL**, do którego podłączamy sterowane elementy wykonawcze, czyli małe silniczki, żaróweczki, przekaźniki, itp. Podczas pisania lub analizy programu sterującego należy uwzględnić, że bufor w **ULN2803** są negatorami. Ze względu na stabilizator **78L05** napięcie **U_{ZAS}** nie powinno przekraczać 30V. Diody LED 1, 2, 3, 4 mają zadanie wizualizacji stanu wyjść i są bardzo przydatne podczas pisania i testowania oprogramowania.

Do sterowania układem UniSterownika **konieczny jest pilot zdalnego sterowania, pracujący w standardzie RC5**. Kod RC5 został opracowany przez firmę Philips. Duża część pilotów od starszego sprzętu Philipsa powinna prawidłowo współpracować z UniSterownikiem. Inna możliwość to zastosowanie pilota uniwersalnego, który pośród opisów obsługiwanego sprzętu będzie miał wymienione urządzenia firmy Philips i RC5. Standard RC5 jest wykorzystywany także przez inne firmy produkujące sprzęt RTV, dlatego zdobycie takiego pilota nie powinno być problemem.

Montaż i uruchomienie

Układ UniSterownika można zmontować i uruchomić na różne sposoby. Najlepiej



Rys. 1 Schemat ideowy

zmontować go na płycie drukowanej. Specjalnie opracowaną do tego płytkę widać na **rysunku 2**. Można zakupić kit AVT ze wszystkimi elementami. Ponieważ płytka jest jednostronna, łatwo ją też wykonać samodzielnie, np. metodą termotransferową – pliki projektu umieszczone są w Elportalu wśród materiałów dodatkowych do tego numeru. Jako złącza wejściowe Z_WE i wyjściowe ZL najlepiej użyć dociętych żeńskich gniazd GOLDPIN. Pod procesor ATTiny13 i układ ULN2803 należy włutować podstawki. Złącze Z_WE ma rozstawione wyprowadzenia tak, że w przypadku montażu odbiornika IR, można wprost w gniazdo włożyć jeden z czterech typów odbiorników podczerwieni (TFMS5360, TSOP1736, TSOP31236 lub SFH5110-36). Odbiornik SFH5110-36 będzie odwrócony, dlatego trzeba go odpowiednio wygiąć. Widok zmontowanej płytki UniSterownika pokazany jest na **fotografii 1**.

Fotografia 2 to widok UniSterownika zmontowanego na płycie stykowej – sposób wykonania, przewidziany dla początkujących i niecierpliwych – układ można uruchomić w przeciągu kilkunastu minut. Do montażu użyłem zworek firmy Velleman z zestawu WJW70. Dla uproszczenia, na płycie stykowej pominąłem elementy R1 i C1, ale w razie złej pracy odbiornika trzeba je dołożyć. Na fotografii dodatkowymi liniami zaznaczyłem krótkie zworki oraz rozprowdzenie początków linii zasilania.

Po złożeniu płytki, ale jeszcze przed zamontowaniem US1, US3, należy sprawdzić wartość napięcia zasilania. Podłączamy do złącza U_ZAS stałe napięcie (biegunowość!) i mierzymy napięcie na wyjściu stabilizatora US2 – powinno wynosić $5V \pm 0,2V$. Następnie montujemy bufor ULN2803 i łączymy przewodem „na krótko” wyprowadzenie 8 (+5V) podstawki procesora kolejno z jej wyprowadzeniami 2, 7, 6, 5 – powinny się zaświecić odpowiednio diody LED1, 2, 3, 4. Jeżeli te elektryczne testy wypadły pomyślnie, możemy włożyć do podstawki mikrokontroler. Układ ATTiny13 będziemy wielokrotnie montować i wymontowywać z włutowanej podstawki, więc po pewnym czasie blaszki jej styków mogą się odgiąć. Dlatego proponuję, żeby do testów używać mikrokontrolera zamontowanego w zwykłej podstawce. Takie rozwiązanie

z dwiema podstawkami ułatwia przenoszenie procesora pomiędzy płytką docelową a programatorem i minimalizuje możliwość uszkodzenia mikrokontrolera.

Dekoder kodu RC-5

Pierwszym programem, który uruchomimy na platformie UniSterownika, jest program prostego dekodera kodu RC-5. Umożliwi on przetestowanie świeżo zmontowanego układu. Zadaniem tego programu jest zdekodowanie kodu RC-5 i wskazanie jego „składników”, czyli wartości transmitowanego adresu i numeru rozkazu.

Można wykorzystać gotowy program wsadowy RC5.hex, dostępny w Elportalu. Jednak zgodnie z założeniem tej serii projektów, programy dla mikrokontrolera generowane będą raczej przez program *UniSterownik RC5*, uruchomiony na komputerze PC. Program ten został napisany i skompilowany w środowisku Visual Basic. Nie trzeba niczego instalować. Wystarczy skopiować plik *UniSterownik RC5.exe* do komputera i jest on gotowy do pracy. Do poprawnej pracy może być potrzebne uzupełnienie systemu Windows pakietem **FRAMEWORK3.5** - wymagany jest on w starszych wersjach systemu Windows.

Obsługa programu jest prosta. Uruchamiamy w komputerze program *UniSterownik RC5* i otwieramy zakładkę *Dekoder kodu RC5*. Na **rysunku 3** widzimy obraz zakładki programu dekodera. W zakładce możemy wybrać układ pracy dekodera, a następnie wciskamy klawisz opisany tekstem *Generuj program HEX dla ATTiny13*. Program powiadomi nas o wygenerowaniu pliku HEX z danymi do zaprogramowania mikrokontrolera. Otrzymamy również informację o wymaganym ustawieniu fusebitów. Ogólnie aplikacje, które opracowałem do detekcji kodu RC5, powinny mieć zaprogramowane Fusebity na zegar wewnętrzny RC 9,6MHz. Szczegóły w dalszej części artykułu.

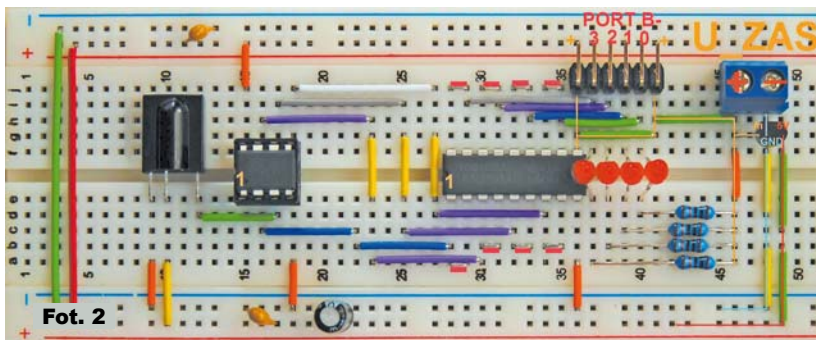
Teraz potrzebny będzie programator, podłączony do komputera PC. Wkładamy mikro-

kontroler do programatora i programujemy. Obszerne informacje o programatorze i jego obsłudze znajdziesz w dalszej części artykułu.

Po zaprogramowaniu mikrokontrolera montujemy go do podstawki UniSterownika i podłączamy zasilanie.

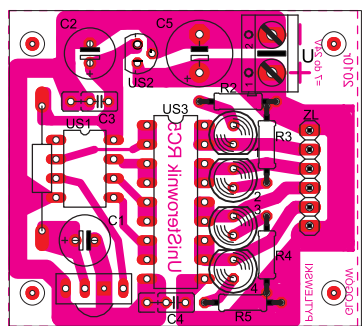
W oczekiwaniu na transmisję RC5, na diodach LED1, LED2, LED3 i LED4 wyświetlany jest prosty efekt świetlny, sygnalizujący gotowość układu do pracy. Po odebraniu prawidłowego kodu RC5, wszystkie diody LED1...LED4 krótko migną, po czym dłuższymi błysnięciami kolejno zostaną zaprezentowane wartości kodu. W sumie jeden kod RC-5 to dwie liczby dwucyfrowe. Najpierw dioda LED 1 błysnie kilka razy, następnie diody LED2...LED4. Jeżeli któraś dioda nie błysnie, oznacza to wartość 0. Znaczenie liczby błysnięć diod jest następujące: Liczba błysnięć LED1 odpowiada liczbie dziesiątek adresu. Liczba błysnięć LED2 odpowiada liczbie jednostek adresu, liczba błysnięć LED3 odpowiada liczbie dziesiątek rozkazu, a błysnięcia LED4 odpowiadają liczbie jednostek rozkazu. Na koniec po „wyświetleniu” danych z kodu RC5 wszystkie diody ponownie krótko migną. Odczytane wartości adresu i rozkazu danego przycisku pilota warto zanotować. Ułatwi to skonfigurowanie sterowania w innych programach, które będą opisywane dla UniSterownika.

A teraz parę słów wyjaśnienia, o co chodzi z tym rozkazem i adresem zawartym w kodzie RC5. Bity rozkazu zaznaczone na **rysunku 4**, jako „command bits” niosą informację, który przycisk na pilocie został naciśnięty. Natomiast bity adresu opisane jako „system bits” oznaczają, dla jakiego urządzenia przeznaczone są dane. Dzięki temu jednym pilotem możemy sterować wiele urządzeń i jeżeli użyjemy pilota do sterowania tunerem satelitarnym, to nie przełączy nam się program w telewizorze. Na przykład w pilocie uniwersalnym są dostępne przyciski do przełączania-wyboru sterowanych urządzeń TV1,



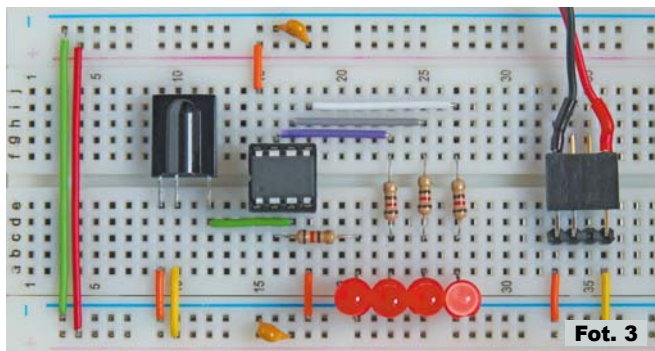
Fot. 2

Rys. 2



Fot. 1

Rys. 3



Fot. 3

SAT, VCR itd. Poprzez wybór urządzenia zmieniamy adres zawarty w emitowanym kodzie RC5 i w ten sposób możemy wybierać układ UniSterownika, który zareaguje na nasze polecenie. Adresy i rozkazy emitowane przy różnych ustawieniach pilota najlepiej przetestować układem dekodera kodu RC5.

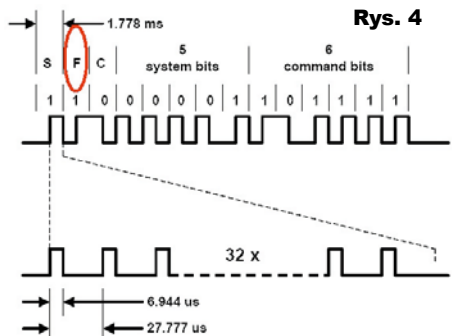
I jeszcze szczegół dla bardziej zaawansowanych: dekodery uwzględniają wartość stanu bitu funkcji, który był oznaczany dawniej jako drugi bit startu. Na rysunku 4 widoczna jest ramka transmisji kodowanej w standardzie RC5 wraz z zakreślonym na czerwono bitem F. Bit ten w nowszych pilotach Philipsa używany jest do sygnalizacji i sterowania poleceniami menu. Uznałem, że zgodnie z rozszerzonym zestawem komend RC5 potraktuję ten bit jako dodatkowy – 7 bit rozkazu. Dlatego w prezentowanych zastosowaniach jest uwzględniona jego **zanegowana** wartość, to znaczy, jeżeli wartość tego bitu wynosi 0, to wartość rozkazu zwiększana jest o 64.

Dekoder można też zmontować i uruchomić bez kostki ULN2803. Na **fotografii 3** przedstawiam widok najprostszej wersji dekodera, zmontowanej na płytce stykowej. Płytkę jest złożona zgodnie ze schematem z zakładki *Dekoder kodu RC5* w programie *UniSterownik RC5*. Schemat widoczny jest po wybraniu opcji *Dekoder zmontowany na płytce stykowej*.

Programator dla początkujących

Przy pierwszym kontakcie z mikrokontrolerem ogromnym problemem może się wydawać konieczność zaprogramowania układu. Szybko pragnę uspokoić, że **nie chodzi w tym przypadku o pisanie programu, bo ten uzyskujemy gotowy**, tylko o wgranie-wpisanie gotowego programu do pamięci procesora. Proponuję, żeby początkujący w dziedzinie programowania mikrokontrolerów postanowili teraz, że nie poddadzą się przy pierwszym niepowodzeniu i konsekwentnie będą próbowali choć raz zaprogramować mikrokontroler. Sam dobrze pamiętam, że początki bywają trudne.

Dla zachęty, przygotowałem opis, jak zbudować i uruchomić prosty programator mikro-



Rys. 4

kontrolerów

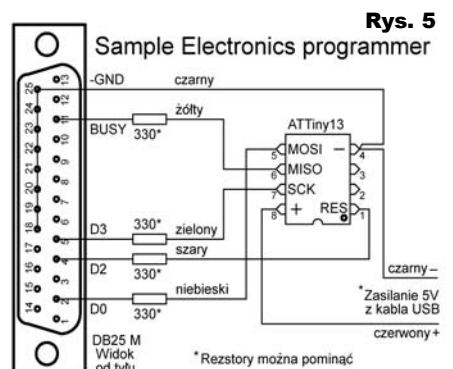
AVR w wersji Sample Electronics. **Programator ten przy odrobinie wprawy można zmontować i uruchomić w przeciągu kilkunastu minut.** W prezentowanym przykładzie jest on przystosowany do programowania mikrokontrolera ATTiny13, ale łatwo można ten układ przerobić do programowania innych mikrokontrolerów AVR. Wystarczy tylko właściwie przylutować odpowiednią podstawkę. Programator Sample Electronics jest podłączany do równoległego portu drukarkowego w komputerze PC. Port ten określany jest też często jako złącze drukarkowe, port LPT, Parallel Port lub oznaczany jest symbolem drukarki. Dlatego przed rozpoczęciem prac sprawdźmy, czy mamy dostęp do komputera ze złączem LPT i zainstalowanym systemem Windows. Jeżeli nasz komputer nie ma złącza LPT, to trzeba się postarać o dostęp do takiego komputera. Złącza LPT były montowane w starszych modelach komputerów, które dzisiaj mają małą wartość. Może się okazać, że taki komputer stoi niepotrzebny w piwnicy u naszego wujka, znajomego lub sąsiada – warto popytać.

Dane potrzebne do zaprogramowania mikrokontrolera AVR są zawarte w pliku z rozszerzeniem HEX. Plik z danymi HEX musimy przesłać do wnętrza mikrokontrolera. W tym celu wykorzystujemy programator, czyli układ elektryczny, pracujący jako interfejs przekazujący dane z komputera PC do mikrokontrolera. Ostatnim elementem potrzebnym do zaprogramowania mikrokontrolera jest program uruchomiony na PC, który można określić jako program sterujący procesem/przebiegiem programowania. Reasumując, pliki z danymi HEX przesyłamy do mikrokontrolera programatorem. Programator jest sterowany z PC za pomocą programu sterującego przesyłaniem danych.

Montaż układu elektrycznego – brzmi to poważnie, ale cała praca polega na polutowaniu kilku kabelków ze złączkami, więc następny krok to zebranie części i zmontowanie programatora. Na **fotografii 4** widać elementy potrzebne do budowy programatora. Potrzebujemy złącza DB25M (męskie z wystającymi szpileczkami), kilku przewodów i podstawki DIP8. Złącze DB25M i kabelki można pozyskać z starego kabla drukarkowego. Kabelki powinny być solidne – nie zbyt cienkie. Cienkie przewodziki będą się urywać, powodując czasochłonne poszukiwanie przyczyny awarii programatora. Dla zabezpieczenia portu LPT stosujemy rezystory o wartości od 300 do 600Ω. Układ można też zmontować bez rezystorów, ale trzeba go polutować bardzo starannie i uważać podczas użytkowania, żeby nie zrobić zwarcia. Do osłonięcia lutowanych połączeń warto zastosować cienką koszulkę termokurczliwą.

Montaż programatora

Montaż układu elektrycznego – brzmi to poważnie, ale cała praca polega na polutowaniu

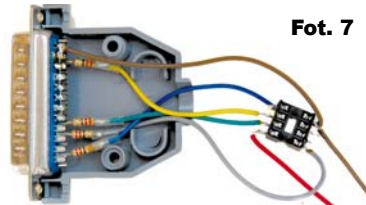


Rys. 5

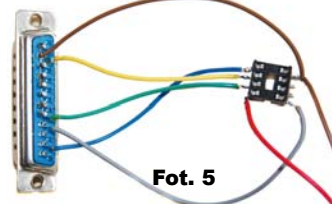
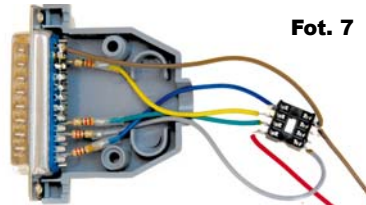
Sample Electronics programmer

* Rezystory można pominąć

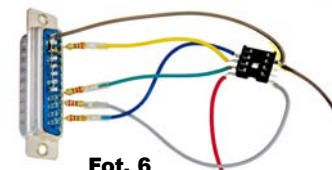
Fot. 4



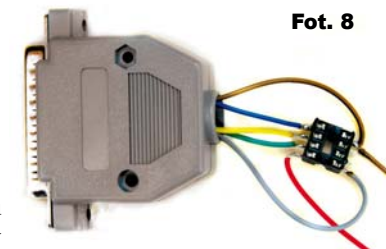
Fot. 7



Fot. 5



Fot. 6



Fot. 8

niu kilku kabelków ze złączkami, więc następny krok to zebranie części i zmontowanie programatora. Na **fotografii 4** widać elementy potrzebne do budowy programatora. Potrzebujemy złącza DB25M (męskie z wystającymi szpileczkami), kilku przewodów i podstawki DIP8. Złącze DB25M i kabelki można pozyskać z starego kabla drukarkowego. Kabelki powinny być solidne – nie zbyt cienkie. Cienkie przewodziki będą się urywać, powodując czasochłonne poszukiwanie przyczyny awarii programatora. Dla zabezpieczenia portu LPT stosujemy rezystory o wartości od 300 do 600Ω. Układ można też zmontować bez rezystorów, ale trzeba go polutować bardzo starannie i uważać podczas użytkowania, żeby nie zrobić zwarcia. Do osłonięcia lutowanych połączeń warto zastosować cienką koszulkę termokurczliwą.

Programator montujemy zgodnie ze schematem z **rysunku 5**. Rozpoczynając lutowanie, zwróćmy uwagę na wycięcie z boku podstawki oznaczające pin 1. Wycięcie to jest zaznaczone na dołączonym schemacie układu programatora. Dzięki temu wycięciu wiemy, w jaki sposób ma być ustawiony mikrokontroler podczas montowania go do podstawki. Zasilanie do programowanego mikrokontrolera podłączymy z portu USB. W tym celu musimy użyć jakiś stary kabel od przedłużacza USB lub np. od myszki, huba, itp. Kabel USB ucinamy tak, żeby złącze A pozostało z długim kawałkiem przewodu. Złącze A to jest to złącze, które wtykamy do portu USB w komputerze. Następnie odcinamy ekran oraz izolujemy przewody mające kolor biały i zielony. Pozostałe dwa przewody czarny (minus) i czerwony (plus) lutujemy do podstawki programatora zgodnie ze schematem. Na **fotografiach 5 i 6** przedstawione są dwa modele zmontowanych programatorów.

Przewody użyte do łączenia złącza DB25 z podstawką mikrokontrolera mogą być dłuższe od tych zaprezentowanych na fotografii, ale nie powinny być zbyt długie. Przewody dłuższe niż kilkadziesiąt centymetrów mogą

być źródłem zakłóceń utrudniających prawidłowe działanie programatora. Na całość zakładamy obudowę złącza (**fotografia 7**) i programator jest gotowy do użytku. Widok na **fotografii 8**.

Uruchomienie i obsługa programu sterującego programowaniem

Do obsługi naszego programatora wykorzystamy oprogramowanie ze środowiska BASCOM-AVR. Jest to przyjazny i intuicyjny program, a jego obsługa jest bardzo prosta. Jeżeli jeszcze nie mamy zainstalowanego programu BASCOM-AVR, to można za darmo pobrać wersję demo tego programu ze strony www.mcselec.com:

www.mcselec.com/index.php?option=com_dowcom&task=cat_view&gid=99&Itemid=54. Prezentowany tu opis przygotowałem na wersji demo 1.11.9.8, ale nowsze wersje nie powinny się niczym różnić w sposobie obsługi programatora Sample Electronics. Instalacja programu przebiega standardowo i wystarczy tylko potwierdzać kolejne pytania instalatora. Warto mieć świadomość, że oprogramowanie BASCOM-AVR to bardzo ciekawe środowisko IDE, umożliwiające pisanie, symulacje i kompilacje programów napisanych w języku BASIC, lecz obecnie interesuje nas tylko opcja programatora i sposób wpisywania danych do układu mikrokontrolera.

Po uruchomieniu programu BASCOM-AVR, wybieramy z paska narzędzi *Options* i następnie *Programmer*. Otworzy nam się okno konfiguracji programatora, widok na **rysunku 6**. W polu wyboru *Programmer* wybieramy *Sample Electronics programmer*. Sprawdzamy również, czy zaznaczone jest pole wyboru opisane jako *AutoVerify*. Opcja ta spowoduje, że po każdym programowaniu będzie sprawdzane czy mikrokontroler ma prawidłowo zapisane dane. Inną istotną opcją jest wybór właściwego adresu portu drukarkowego *LPT-adress*, do którego będzie podłączany programator. Na początek pozostawiamy wartość domyślną 378, ale w razie problemów z pierwszym uruchomieniem programatora może będzie trzeba wybrać inny adres. Po ustawieniu opcji klikamy na OK.

Aby uruchomić okno obsługi programatora, musimy otworzyć dowolny projekt. W tym celu wybieramy z paska narzędzi *File*

następnie *Open* i w otwartym okienku wybieramy ścieżkę do pliku np. *BASCOM AVR-SAMPLES-CHIPS-attiny13* lub otwieramy dowolny inny plik z rozszerzeniem *BAS*. W oknie edytora wyświetli nam się wybrany listing i już możemy uruchomić okno programatora, czyli program, który steruje przepisywaniem danych z dysku komputera PC do mikrokontrolera. Okno programatora uruchomimy, wybierając na pasku narzędzi *Program*, następnie *Send to Chip* i *Manual Program*. Ponieważ nie mamy jeszcze podłączonego programatora, podczas uruchamiania okna wystąpią komunikaty ostrzegawcze. Komunikaty te potwierdzamy, klikając OK.

Proponuję umieścić mikrokontroler w dodatkowej, zwykłej podstawie i dopiero taką „kanapkę” wkładamy w podstawkę programatora – mniej zniszczy się procesor oraz podstawki w programatorze i układzie docelowym. Tuż przed montażem ATtiny13 do programatora i podłączeniem do PC, warto dotknąć metalowej części obudowy komputera PC w celu rozładowania potencjału elektrostatycznego. Wkładając mikrokontroler do podstawki programatora, zwracam uwagę, aby był on właściwie ustawiony, czyli wycięcie lub kropka oznaczające wyprowadzenie 1 układu pokrywały się z wycięciem oznaczającym 1 na podstawie programatora. Niewłaściwe zamontowanie mikrokontrolera może uszkodzić układ lub programator, dlatego nie należy się spieszyć i wszystko dokładnie sprawdzić. Następnie podłączamy programator do złącza LPT i dopiero na koniec wtykamy kabel zasilania układu do portu USB.

Teraz możemy sprawdzić, czy nasz programator prawidłowo działa. W pasku narzędzi okna programatora wybieramy *Chip*, następnie *Identify*. Jeżeli nie pojawią się żadne komunikaty o błędach i w oknie wyboru mikrokontrolera automatycznie zostanie wybrany ATtiny13, to znaczy, że wszystko jest w porządku i możemy rozpocząć programowanie układu. W razie wystąpienia komunikatów o błędzie trzeba najpierw usunąć problem, korzystając z porad umieszczonych na końcu tego artykułu.

W pamięci mikrokontrolerów AVR są trzy typy wewnętrznej pamięci. Upraszczając nieco, programowana pamięć ma trzy części. Są to pamięć FLASH, pamięć EEPROM i tak zwane Fusebity.

Do pamięci FLASH wpisujemy właściwy program wykonywany przez mikrokontroler. W celu wpisania danych do pamięci FLASH w oknie programatora wykorzystujemy zakładkę *FlashROM*. Do pamięci FLASH programujemy dane z plików BIN lub HEX.

Pamięć EEPROM jest specjalną pamięcią dla ważnych danych. Czasami wymagane jest wpisanie danych do EEPROM podczas programowania mikrokontrolera. Dane takie są umieszczane w plikach z rozszerzeniem *EEP*. W celu wpisania danych do pamięci EEPROM korzystamy z zakładki *EEPROM*.

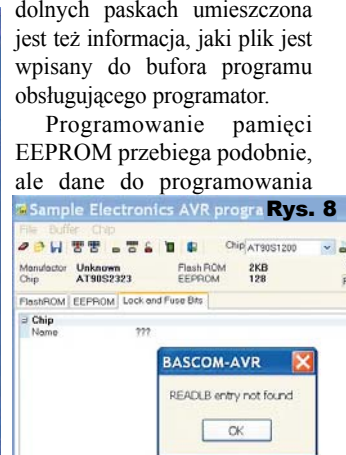
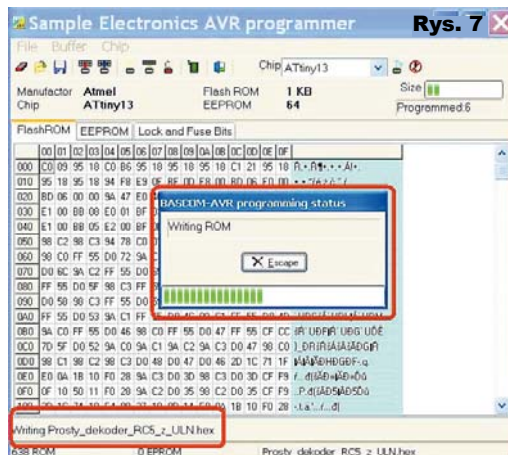
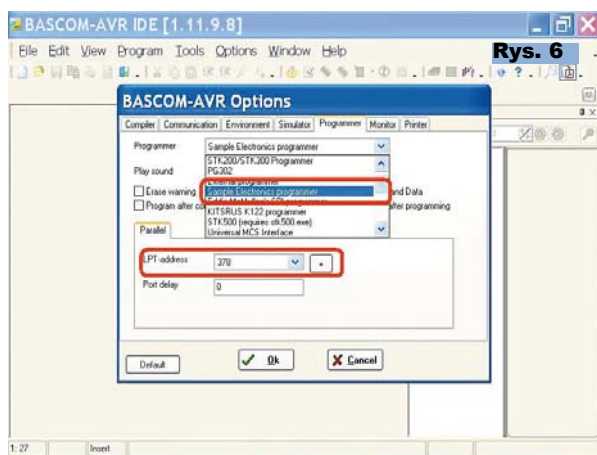
W trzecim obszarze pamięci mikrokontrolera, umieszczone są tzw. Fusebity, które przełączają i konfigurują pewne układy wewnątrz mikrokontrolera. Do programowania fusebitów wybieramy zakładkę opisaną jako *Lock and Fuse Bits*. Ważna może być kolejność wpisywania danych do mikrokontrolera. Najpierw programujemy FLASH, a następnie, jeśli jest taka potrzeba programujemy kolejno EEPROM i Fusebity.

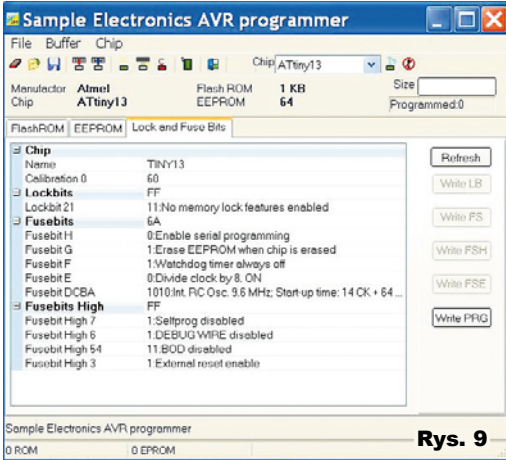
Po prawidłowym zidentyfikowaniu mikrokontrolera przez funkcję *Identify* możemy przystąpić do pierwszej fazy programowania, czyli wpisania programu do pamięci FLASH. Czynność tę można wykonywać wiele razy – około 10000 razy bez negatywnego wpływu na układ i pamięć procesora. Daje to możliwość wypróbowania bardzo wielu ciekawych aplikacji na jednym mikrokontrolerze ATtiny13.

Przed zaprogramowaniem pamięci FLASH, trzeba wskazać programowi, jakie dane chcemy wpisać do mikrokontrolera. W tym celu wybieramy w pasku narzędzi *Buffer*, następnie *Load from file*. Pojawi się okno wyboru pliku. Tu ważna uwaga! W oknie tym musimy zaznaczyć, jaki typ pliku zamierzamy wpisać do bufora programu. Do wyboru mamy ładowanie plików z rozszerzeniem BIN, HEX lub EEP. Pliki generowane przez program UniSterownik mają rozszerzenie HEX, dlatego aby je załadować, konieczne wybieramy typ HEX. Inaczej pliki nie pokażą się w oknie wyboru. Po wskazaniu pliku jest on ładowany do bufora *FlashROM*. Teraz wreszcie możemy zaprogramować mikrokontroler. Wybieramy na pasku okna programatora *Chip* i następnie *Autoprogram*. W oknie programatora pojawi się pasek przebiegu programowania (**rysunek 7**), a na dolnym pasku z lewej strony pojawi się komunikat o wyniku procesu wpisywania danych do pamięci. Na

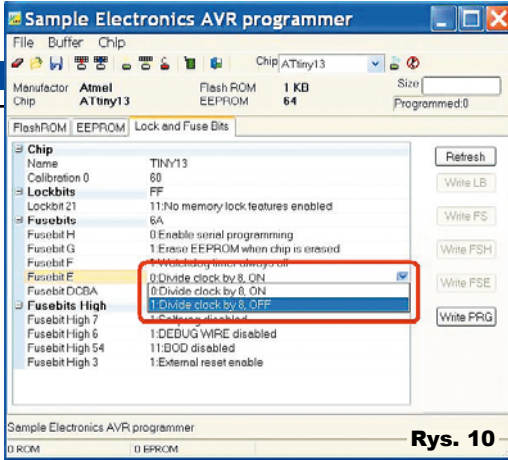
dolnych paskach umieszczona jest też informacja, jaki plik jest wpisany do bufora obsługującego programator.

Programowanie pamięci EEPROM przebiega podobnie, ale dane do programowania

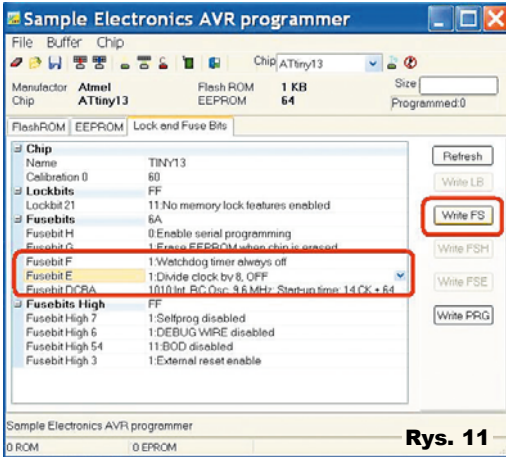




Rys. 9



Rys. 10

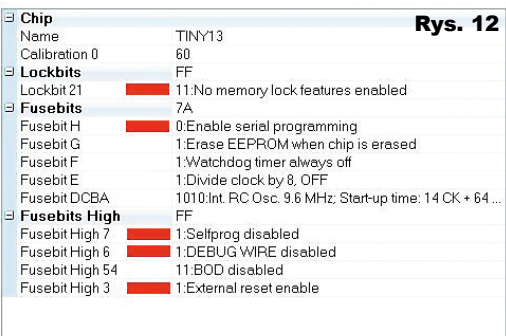


Rys. 11

EEPROM są zazwyczaj zawarte w plikach z rozszerzeniem EEP i oczywiście korzystamy z zakładki EEPROM.

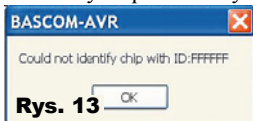
Ostatnia faza programowania to programowanie fusebitów. Jak wcześniej wspomniałem Fusebity są to specjalne „przełączniki” znajdujące się w środku mikrokontrolera, powodujące odpowiednie ustawienie jego wewnętrznych układów. Aby programy działały zgodnie z zamierzeniem programisty, Fusebity muszą być odpowiednio ustawione. Dla właściwej pracy programów generowanych dla platformy UniSterownika, Fusebity powinny ustawić taktowanie procesora na 9,6MHz oraz wyłączyć dzielenie sygnału zegarowego przez 8. Ustawienie innych fusebitów jest w tym przypadku mniej istotne. Nastawy fusebitów nie zmieniają się podczas programowania pamięci FLASH, dlatego nie musimy ciągle ich programować. Dla wszystkich aplikacji/zastosowań UniSterownika wystarczy raz ustawić Fusebity i procesor będzie prawidłowo wykonywał wszystkie proponowane programy.

Aby sprawdzić stan lub przeprogramować Fusebity, w oknie programatora wybieramy zakładkę *Lock and Fuse Bits*. Podczas otwierania zakładki podglądu stanu fusebitów, programator odczytuje ich aktualne ustawienia



Rys. 12

z mikrokontrolera. Jeżeli odczyt fusebitów zakończy się komunikatem jak na **rysunku 8**, to musimy sprawdzić podłączenie programatora według porad umieszczonych na końcu tego artykułu. Na **rysunku 9** przedstawiony jest prawidłowy odczyt z fabrycznymi ustawieniami fusebitów ATtiny13. Żeby ustawić pracę mikrokontrolera odpowiednio dla UniSterownika, trzeba zmienić Fusebit E, wyłączając dzielenie zegara systemowego przez 8, czyli wybieramy opcję *Divide clock by 8 OFF*. Zmiana opcji w tym programie polega na wybraniu nastawy z rozwijanego menu, w którym opisane są możliwe ustawienia (**rysunek 10**). Nie musimy od razu wszystkiego rozumieć, ale dla właściwej pracy układu trzeba opcje odpowiednio ustawić. **Bardzo ważne jest, żeby fusebitów nie zmieniać „jak popadnie”, ponieważ niektóre z nich przełączają funkcje mikrokontrolera w ten sposób, że po ich przestawieniu niemożliwe jest ponowne zaprogramowanie układu bez użycia specjalnego programatora.** Dlatego podczas zmiany ustawień fusebitów trzeba zachować rozwagę i Fusebity ustawić tak, jak zaleca konstruktor układu! Po każdej zmianie ustawienia danej opcji, po prawej stronie okna programu zostanie podświetlony odpowiedni przycisk, który wprowadza zmiany do pamięci mikrokontrolera (**rysunek 11**). W ATtiny13 są trzy grupy Fusebitów. Są to *Lockbits*, *Fusebits* i *Fusebits High*. Proponuję po wprowadzeniu jednej lub kilku zmian w danej grupie zmienione dane od razu zapisywać do pamięci. Jeżeli wprowadzimy zmiany w różnych grupach naraz, to nie wszystkie one zostaną zapamiętane. Dlatego zmiany najlepiej wprowadzać grupami. Po wciśnięciu przycisku *Refresh* zostanie odczytany aktualny stan fusebitów z mikrokontrolera, dzięki czemu możemy sprawdzić, czy wszystkie ustawienia są takie, jak zamierzaliśmy. Jeżeli coś niewłaściwie poprzestawiliśmy w oknie zmian fusebitów i jeszcze dane te nie zostały wprowadzone przyciskiem *Write..* to możemy się z tego wycofać, wciśkając *Refresh* lub wychodzimy z zakładki Fusebitów. Prawidłowe ustawienie fusebitów dla aplikacji UniSterownika przedstawione jest na **rysunku 12**, parametr *Calibration* może się różnić w poszczególnych układach. Pozostałe opcje powinny mieć ustawienia jak na rysunku. Na rysunku tym zaznaczyłem czerwonymi polami bity i opcje, których nie



Rys. 13



Rys. 14

Projekty AVT

powinniśmy zmieniać bez odpowiedniej wiedzy. Niektóre z tych opcji mogą zablokować pracę mikrokontrolera.

Jeżeli wszystkie opisane wcześniej procedury przebiegną bez błędów, to programowanie mikrokontrolera jest zakończone i można go odłączyć. Odłączenie programatora od komputera wykonujemy w odwrotnej kolejności jak podłączanie, czyli odłączamy kabel zasilania z USB, następnie wyciągamy złącze z portu LPT i wyciągamy mikrokontroler z podstawki programatora.

Jeżeli nie zdecydujesz się, Drogi Czytelniku, na budowę opisanego tu programatora, to pozostaje jeszcze możliwość zakupu. Istnieje mnóstwo typów programatorów. Ja przy opracowywaniu projektów używam obecnie programatora USBASP, ale pierwsze próby z programowaniem AVR-ów robiłem właśnie programatorem Sample Electronics. Po pewnym czasie problemem okazało się to, że w nowszych komputerach nie ma już portu drukarkowego LPT. Wiele informacji na temat programatorów do AVR oraz o sposobie ich wykonania i stosowania można znaleźć w Internecie. Wystarczy wpisać w wyszukiwarce USBASP, **Sample Electronics programmer** czy po prostu **programator AVR**. Dostępne w ofertach programatory można podłączać do komputera PC poprzez złącze USB, LPT, RS lub inne, ale współczesne komputery coraz rzadziej mają wbudowane złącza RS czy LPT. Dlatego najlepiej wybrać programator dołączający do PC poprzez złącze USB.

Dostępne na rynku programatory USB, nadające się do zaprogramowania procesorów AVR, nie są drogie i można je nabyć w wielu sklepach internetowych. Powinniśmy wybrać taki programator, na temat którego łatwo znaleźć w Internecie informacje, jak go uruchomić i jak skonfigurować oprogramowanie do jego pracy. Warto zwrócić uwagę na programatory z oferty AVT [przyp. red.: w następnym numerze EdW zaplanowany jest projekt prostego programatora USB – UniProg]. Taki jednorazowy wydatek umożliwi nam w przyszłości montowanie i uruchamianie wielu bardzo ciekawych urządzeń wykorzystujących procesory AVR.

Problemy z programatorem

Z doświadczenia wiem, że podczas pierwszych prób z nowym typem mikrokontrolera **ZAWSZE** występują jakieś trudności. Przyczyną problemów jest najczęściej niewłaściwa konfiguracja programu ładującego kod do procesora oraz tak banalne sprawy, jak brak zasilania na programowanym układzie, niewłaściwe połączenia czy też złe zamontowanie układu. Odwrotne włożenie procesora ATtiny13 w podstawkę na pewno spowoduje jego bardzo silne rozgrzanie oraz może spowodować uszkodzenie struktury układu scalonego (zależnie od wydajności prądowej zasilacza). Kilka razy zdarzyło mi się popełnić

ten błąd, ale na szczęście, układ ATTiny13 po ostygnięciu i właściwym zamontowaniu działał prawidłowo.

A oto ostrzeżenia o błędach, które napotkałem podczas korzystania z Sample Electronics programmer:

Rysunek 13 – Komunikat informujący o braku możliwości identyfikacji mikrokontrolera. ID o wartości FFFFFFFF zazwyczaj jest spowodowane brakiem mikrokontrolera w podstawie, a ID o wartości 000000 może oznaczać brak zasilania.

Rysunek 14 – Mało dla nas istotny komunikat, który jest wyświetlany w momencie otwarcia okna obsługi programatora. Informuje o tym, że nie jest dostępny plik wynikowy programu otwartego w oknie edytora środowiska BASCOM-AVR. Jeżeli skompilujemy program otwarty w oknie edytora, to plik wynikowy będzie automatycznie przepisywany do bufora programatora i komunikat nie będzie wyświetlany.

Rysunek 15 – Komunikat informujący o niezgodności typów mikrokontrolerów wybranych w programie i ustawieniach. My i tak musimy wybrać w oknie programatora typ mikrokontrolera, który będziemy chcieli programować, czyli ATTiny13. Ustawienie typu mikrokontrolera można dokonać ręcznie poprzez wybór w oknie programatora opcji z listy *Chip* lub automatycznie poprzez wybór polecenia *Identify*. Jeżeli mikrokontroler będzie w podstawie programatora, to po odczytaniu jego numeru ID zostanie automatycznie wybrany jego typ.

Rysunek 16 – Komunikat, który może wystąpić po wybraniu zakładki *Lock and Fuse Bits*, oznacza błąd podczas próby odczytania stanu fuse-bitów. Sygnalizuje brak możliwości odczytania danych z mikrokontrolera. Przy takim komunikacie postępujemy według porad poniżej.

Rysunek 17 – Komunikat ten może wystąpić po programowaniu układu lub podczas porównania danych z mikrokontrolera z danymi z bufora. Komunikat oznacza, że od wskazanego adresu bufora dane w mikrokontrolerze są niezgodne. Jeżeli adres, od którego występuje niezgodność, zmienia się, to znaczy, że w programatorze jest jakiś niepewny styk lub występują zakłócenia. Przy częstym występowaniu tego błędu po wywołaniu *Autoprogram* można spróbować inną metodę. Skasować zawartość mikrokontrolera, wybierając *Chip* i następnie *Erase*. Upewniamy się, czy układ został faktycznie wyzerowany, odczytując jego zawartość do bufora, wybierając *Chip* i następnie

Write buffer into chip, po czym weryfikujemy zgodność danych funkcją *Verify*.

W trakcie poszukiwań przyczyn złej pracy programatora wykonujemy test jego pracy przez odczyt Fuse-bitów – wybieramy zakładkę *Lock and Fuse Bits*. Jeżeli programator odczyta ich ustawienia bez błędów (rysunek 9), to możemy próbować zapisać dane do pamięci FLASH. Zapis tych danych powinien zostać potwierdzony komunikatem *Verified Ok*.

Jeżeli wystąpią problemy z uruchomieniem programatora **przy pierwszym podłączeniu** do komputera, to jeszcze raz sprawdzimy, czy układ jest zlutowany zgodnie ze schematem. Następnie sprawdzamy, czy port LPT właściwie pracuje. W tym celu uruchamiamy specjalne polecenia testujące pracę wyprowadzeń portu. W oknie programatora na pasku narzędzi wybieramy *File*, następnie *Test – Sample Programmer – D0*. Na wyprowadzeniu 5 podstawki mikrokontrolera powinno pojawić się napięcie około 5V. Po wybraniu *D2* napięcie 5V pojawi się na 1 wyprowadzeniu podstawki mikrokontrolera, a na pinie 5 zniknie. Po wybraniu *D3* napięcie 5V pojawi się na 7 wyprowadzeniu. Po wybraniu *Reset*, na wyprowadzeniach 1, 5 i 7 podstawki napięcie powinno wynosić około 0V. Pomiar napięcia na wyprowadzeniu 6 powinien być bliski 5V, ponieważ jest to linia wejściowa portu LPT, podciągnięta rezystorem do 5V. Po wybraniu *Busy* pojawi się komunikat *Busy pin reads 5V*. Następnie zwieramy do minusa linię *Busy*, czyli pin 6 z pinem 4 podstawki i po wybraniu *Busy* powinien pojawić się komunikat *Busy pin reads 0V*. Pomiar wykonujemy bez mikrokontrolera. Mierzmy również napięcie zasilania na wyprowadzeniach 4(-) i 8(+) podstawki, które po włożeniu wtyczki zasilania do portu USB powinno wynosić około 5V. Jeżeli mamy takie odczyty napięć, to port LPT działa prawidłowo, a przyczyn błędów trzeba szukać w konfiguracji programu obsługującego programator. Sprawdźmy ustawiony typ procesora itp., postępując zgodnie z dalszym opisem.

Jeżeli nie mamy takich napięć i odczytów sprawdzimy, czy programator jest wpięty do właściwego portu LPT – niektóre PC-ty miały dwa lub więcej portów LPT. Spróbujmy również zmienić konfigurację portu na inny adres w opcjach. Zmiany adresu portu LPT dokonujemy w oknie konfiguracji programatora (rysunek 6). Dostępne adresy możemy ustalić, korzystając z menu dżera urządzeń.

W Internecie znalazłem też informację, że przyczyną złej pracy programatora może być niewłaściwa konfiguracja portu LPT w BIOS-ie

komputera PC. Według tej informacji, port równoległy (Parallel Port) powinien być ustawiony w BIOS-ie na tryb *ECP*, w trybie *SPP* programator może nie działać.

Jeżeli wystąpią problemy podczas pracy programatora, który działał już na naszym komputerze, to najpierw sprawdzamy:

Czy programator jest wpięty do właściwego portu LPT? Czy jest w podstawie mikrokontroler i czy jest zamontowany we właściwy sposób? Może się też zdarzyć, że wyprowadzenia mikrokontrolera lub podstawki wygięły się i się nie stykają. Sprawdźmy, czy jest podane napięcie zasilania 5V z kabla zasilającego z portu USB? Czy jakieś połączenie lub kabelek nie urwały się? Warto usunąć źródła zakłóceń z okolicy programatora. Mogą to być np. telefon GSM, interfejs Bluetooth lub WIFI itp.

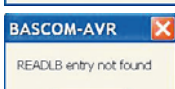
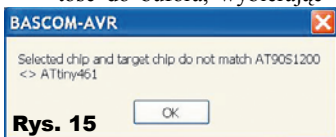
Jeżeli działa funkcja *Identify* i nie możemy zaprogramować układu, to należy sprawdzić, czy dołączone jest zasilanie i szukać przyczyn problemu w obsłudze programu. Sprawdzamy, czy wybrany jest właściwy typ mikrokontrolera w oknie programatora i czy na opisie u góry okna programatora jest nagłówek *Sample Electronics AVR programmer*. Kasujemy pamięć mikrokontrolera, wybierając na pasku *Chip* i *Erase*. Następnie sprawdzamy, czy właściwy plik programu HEX jest wpisany do okna bufora *FlashROM*. Ponownie wywołujemy funkcję *Autoprogram*. Warto od razu zaopatrzyć się w zapasowy mikrokontroler ATTiny13. W razie braku możliwości uruchomienia komunikacji z procesorem zamieńmy układ na zapasowy. W ten sposób możemy wykluczyć uszkodzenie mikrokontrolera jako przyczynę złej pracy programatora. Jeżeli dalej programator nie działa, to wykonajmy opisany wyżej test napięć na porcie LPT.

W przypadku wystąpienia trudności trzeba na spokojnie przeanalizować sytuację i wszystko posprawdzać, ponieważ na

99,9% problem powstał przez nasze niedopatrzenie lub brak wiedzy. W razie kłopotów z uruchomieniem aplikacji na UniSterowniku, proszę pytać. Myślę, że na wiele pytań uda się udzielić odpowiedzi przez Internet.

Praktyka pokazuje, że po pokonaniu pierwszych problemów z konfiguracjami i podłączeniami oraz po zaznajomieniu się z prostymi opcjami oprogramowania, wpisywanie programów do struktury mikroprocesora stanie się łatwą czynnością. **Ta umiejętność otworzy przed nami drzwi do świata wielkich możliwości mikroprocesorów.**

Wiesław Pytlewski
elewp@wp.pl



Wykaz elementów

US1.....	ATTINY13
US2.....	78L05
US3.....	ULN2803
LED1 do 4..	dowolne diody LED
US_IR-np.	TSOP31236, TFMS5360, TSOP1736 lub SFH5110-36
R1.....	100Ω
R2,R3,R4,R51kΩ
C1,C2.....	100µF/25V
C3,C4.....	100nF
C5.....	470µF/30V
U_ZAS.....	Złącze ARK2
Z_WE, ZL....	goldpin 2,54mm
Podstawki DIP8 i DIP18

Programator

Złącze DB25	Męskie
4 szt. rezystorów	330Ω
3 szt. zwykłych podstawek	DIP8

Komplet podzespołów z płytka jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2992.