WYDAWNICTWO

ATNEL

Język C

Pasja programowania mikrokontrolerów 8-bitowych

Mirosław Kardaś

Atnel 2012

Książka jest kontynuacją publikacji „Mikrokontrolery AVR. Język C. Podstawy programowania". Tym razem mamy do czynienia z szeregiem ciekawych projektów, ponieważ cała książka stanowi część warsztatową oraz praktykę programowania. Przeznaczona jest przede wszystkim dla pasjonatów mikrokontrolerów, hobbystów, amatorów, a także początkujących, którzy mieli okazję przeczytać pierwszą część. Praktyczne sposoby programowania są przekazane na przykładach tworzenia przeróżnych bibliotek programowych potrzebnych do obsługi wielu interesujących układów peryferyjnych. Są tu także aplikacje testowe, które przybliżają sposoby korzystania z omawianych bibliotek we własnych projektach. Całość oprogramowania tworzona jest przy wykorzystaniu zaawansowanych technik opierających się na: zdarzeniach (events), timerach programowych oraz programowaniu wielowątkowym.

Korekta: Krystyna Pawlikowska
Opracowanie graficzne: Karolina Kardaś

Wydawnictwo Atnel
70-777 Szczecin, ul. Jasna 15/33
tel. 91 463 56 83
fax 91 882 10 99
e-mail: biuro@atnel.pl
www.atnel.pl

Wydanie I

ISBN 978-83-931797-1-8

Książkę tę dedykuję Panu Ryszardowi Łozowskiemu.

Wspaniały człowiek, wybitny specjalista w dziedzinie telewizji analogowej, twórca urządzeń, które wyprzedzały jego epokę. Pasjonat elektroniki i mechaniki precyzyjnej, swoją pasją zarażał wszystkich w otoczeniu, podobnie jak spokojem i pogodą ducha. Kochał życie. Już nie dokończy swoich planów, a miał ich wiele, w tym udoskonalenie urządzeń medycznych służących do rehabilitacji pourazowej. Wcześniejsze ich wersje, skonstruowane przez niego od podstaw, do dzisiaj pracują w wielu sanatoriach na terenie Polski.

Miałem szczęście i zaszczyt spotkać Pana Ryszarda na swojej drodze. Każdy dzień, który z nim spędziłem, a było ich niestety tylko 200, zapamiętam do końca życia. Myślę, że nie będzie nadużyciem, jeśli powiem, że był moim przyjacielem. Od Niego bardzo dużo się nauczyłem i dzięki Niemu nabrałem pokory wobec codziennych kłopotów, z którymi on tak dobrze sobie radził. W pracy wyznawał dewizę, że nie ma rzeczy, których nie da się zrobić, nie narzekał, po prostu mierzył się z problemami i rozwiązywał je w sposób niespotykany w dzisiejszych czasach. Z tego powodu każdy, kto z nim współpracował, miał okazję uczyć się nie tylko elektroniki, ale także podejścia do życia.

Mirosław Kardaś

Spis treści

1. Podczerwień – praktyczne zastosowania

Na początku przyznam, że zagadnienia związane z zastosowaniem kodowania i dekodowania w podczerwieni są moim ulubionym tematem. Wynika to głównie z tego, że w bardzo wielu moich projektach w mniejszym lub większym zakresie wykorzystuję podczerwień. Temat nie jest nowy ani odkrywczy, jednak większość ludzi kojarzy go na pierwszym miejscu z typową obsługą popularnych pilotów podczerwieni do sterowania różnych urządzeń. Rzeczywiście tak jest, że technologia ta zyskała szerokie zastosowanie dzięki znanym urządzeniom, jak telewizory, później magnetowidy itp. W dzisiejszych czasach trudno wręcz spotkać popularne urządzenia przetwarzające obraz czy dźwięk bez zdalnego sterowania w podczerwieni. Kamery, aparaty fotograficzne, odtwarzacze DVD… Dlaczego piloty cieszą się taką popularnością? Czy tylko lenistwo leżało u podstaw stworzenia tej technologii? W początkowym okresie rozwoju (pierwsze piloty zdalnego sterowania wykorzystywały ultradźwięki zamiast podczerwieni) pilot zdalnego sterowania miał niewiele możliwości i klawiszy funkcyjnych. Jego zadaniem było rzeczywiście zdalne włączenie lub wyłączenie urządzenia, dzięki czemu użytkownik nie musiał wstawać z fotela. Większość pozostałych funkcji regulacyjnych trzeba było i tak wykonywać ręcznie, manipulując gałką lub klikając przyciski znajdujące się na urządzeniu. W miarę rozwoju technologii i w chwili wejścia w życie mikrokontrolerów sytuacja zmieniła się diametralnie. Teraz zdalnie sterowane urządzenie posiada tylko kilka przycisków wyprowadzonych na zewnątrz obudowy, które zawiadują podstawowymi funkcjami. Cała reszta możliwości konfiguracyjnych, regulacji itp. została przeniesiona do pilotów. Niejednokrotnie tani odtwarzacz DVD, gdy jego użytkownik zgubi albo zniszczy przypadkowo pilota, staje się wręcz bezużytecznym przedmiotem. Zatem śmiało można stwierdzić, że w dzisiejszych czasach pilot zdalnego sterowania nie jest już dodawany do urządzenia w celu zaspokojenia lenistwa użytkownika. Co jest głównym powodem takiego stanu rzeczy? Próbowałeś kiedyś się nad tym zastanowić? Jeśli chwilę pomyśleć, to oczywiste staje się, że ograniczenie liczby klawiszy sprzętowych umieszczonych w obudowie urządzenia zostało podyktowane:

- zwiększeniem ergonomii urządzeń i miniaturyzacją ich obudów. (Wyobrażasz sobie dzisiejsze płaskie telewizory z wąskim obramowaniem? Czy można wprowadzić tam klawiaturę do obsługi całości?);
- większymi kosztami związanymi ze stosowaniem trwałych klawiatur lub nawet pojedynczych klawiszy. Klawiaturę w postaci zewnętrznego pilota jest stosunkowo łatwo i niedrogo wymienić na nową. Nie potrzeba do tego nawet wzywać serwisu. Wystarczy kupić oryginalny pilot lub zamiennik. Natomiast uszkodzenie wbudowanej klawiatury bywa jednoznaczne z koniecznością oddania do serwisu całego urządzenia;

– uproszczeniem systemu transmisji danych oraz ich odbioru/dekodowania w podczerwieni;
– możliwością wprowadzania olbrzymiej liczby funkcji przy jednocześnie niskim koszcie produkcji pilota zdalnego sterowania.

Biorąc to pod uwagę, bardzo często jestem zdziwiony, dlaczego tak spora liczba osób, które używają mikrokontrolerów w swoich hobbystycznych lub komercyjnych projektach, tak rzadko korzysta z zalet omawianej technologii. Bardzo często nawet na etapie prototypowania swojego projektu, gdy jest on realizowany jeszcze na płytce stykowej bądź w zestawie uruchomieniowym, dostrzegam przedziwną dla mnie praktykę – zaopatrywania się w przeróżnej wysokości klawisze typu *microswitch*. Podobnie ich liczba na niektórych zestawach uruchomieniowych bardzo mnie dziwi. Owszem, obsługa sprzętowa klawiszy podłączanych bezpośrednio do mikrokontrolera to także podstawa. Jednak wiele osób nie zdaje sobie sprawy nie tylko z efektu drgań styków, który trzeba wyeliminować, ale także ze sposobów praktycznego i nieblokującego oprogramowania takich pojedynczych klawiszy lub klawiatur. Efekty tego można spotkać na przykład podczas prezentacji amatorskich projektów na różnych forach internetowych lub w postaci filmów na serwisie youtube.com. Prezentowane urządzenia, ich koncepcje często są bardzo ciekawe, ale obsługa ich przez użytkownika pozostawia wiele do życzenia. Zdarzają się nawet sytuacje humorystycznie, na przykład gdy na materiale filmowym widać wielki niezdarny palec próbujący pstrykać maleńkimi klawiszami *microswitch*. A przy okazji widać jak na dłoni, że prezentowana obsługa jakiegoś menu co chwilę przeskakuje do innej opcji, trzeba się cofnąć znowu, starając się kliknąć maleńki klawisz. Zwykle bywa to efektem złej obsługi programowej i niewyeliminowania drgań styków.

Reasumując, zbudowanie nawet prostej klawiatury, składającej się choćby z kilku klawiszy, staje się największą bolączką na etapie projektowania. Ale nie tylko projektowania. Często bowiem powstaje najpierw projekt urządzenia amatorskiego, a dopiero w późniejszym czasie konstruktor czy projektant stara się dopasować do niego obudowę. Wtedy dochodzi do poważniejszych trudności związanych z doborem i dopasowaniem obudowy. Na końcu tej drogi może okazać się, że urządzenie nie znajdzie praktycznego zastosowania z powodu… braku obudowy albo z powodu jej wątpliwej estetyki. Trudno wówczas wymagać akceptacji przez potencjalnych odbiorców, klientów itp.

W takich sytuacjach zwykle zastanawiam się, dlaczego tak rzadko wykorzystywana jest podczerwień i bardzo wygodne klawiatury znajdujące się na pilotach zdalnego sterowania. Domyślam się, że być może powodem jest niechęć do konstruowania we własnym zakresie pilota zdalnego sterowania, gdyż to również jest spore wyzwanie, może nawet większe niż wykonanie obudowy i klawiatury pilota. Uważam jednak, że jest to błędne podejście. Wystarczy się rozejrzeć wokoło. Obecnie mamy do dyspozycji setki, jak nie tysiące przeróżnych gotowych pilotów zdalnego sterowania, posiadających nawet ciekawe pod względem designu obudowy, mających od kilku do kilkunastu klawiszy. Co więcej, piloty posiadające więcej klawiszy mają cenną zaletę z naszego punktu widzenia, jeśli chodzi o ciekawe pomysły na ich praktyczne zastosowanie. Zwykle jest to zorga-

nizowana i wydzielona klawiatura numeryczna (klawisze 0–9). Piszę o tym, ponieważ wiele razy spotkałem się z brakiem pomysłu na to, aby zastosować taką klawiaturę do własnych potrzeb w zamian za rozbudowę klawiatury sprzętowej w samym urządzeniu. Wielu twórców amatorów ma też obawy z tego powodu, że to niezbyt elegancko wygląda, gdy do własnego urządzenia wykorzystuje się „jakiś tam" pilot od telewizora czy magnetowidu. Tymczasem to znowu tylko brak wyobraźni narzuca takie ograniczenia w pomysłach. Spotkałeś się zapewne z wieloma już rodzajami pilotów zdalnego sterowania na podczerwień, tzw. no name. Bez nazwy typu i producenta, a nawet z takim opisem czy symbolami klawiszy funkcyjnych (poza częścią numeryczną), że nie sposób stwierdzić, czy jest to pilot do TV, czy do VCR, czy może do własnego urządzenia. Dlaczego nie? Niektórych pedantów może nadal martwić jednak sposób opisu klawiszy funkcyjnych. Nie zechcą zastosować gotowego pilota tylko dlatego, że jeśli już miałoby to nastąpić, to musiałyby być na nim własne opisy, a z tym już gorzej, bo często namalowane są one wprost na plastikowej obudowie, więc zmiana ich we własnym zakresie może wyjść równie nieelegancko, albo i gorzej, jak wykonanie własnej klawiatury sprzętowej. Ale ja i na to mam radę. Wystarczy poświęcić nieco więcej czasu na poszukanie takiej wersji pilota, który opisy ma wykonane nie bezpośrednio na obudowie, lecz na specjalnej laminowanej okleinie, którą co ciekawe, można w dosyć prosty sposób podmienić na swoją własną, drukując uprzednio opisy na zwykłej drukarce laserowej i po zalaminowaniu wymienić oryginalną okleinę. W efekcie końcowym można bardzo tanio uzyskać zdecydowanie lepszy ekwiwalent takiej „klawiaturki" sprzętowej, przy całej gamie zalet, jakie zostały wymienione w punktach na początku rozdziału. Na rysunku 1 przedstawione są dwa piloty z różnymi obudowami, które dają możliwość wymiany okleiny na swoją własną. Nie jest to może pomysł, który nadaje się do zastosowania na szeroką skalę, ale przy małoseryjnej produkcji może być opłacalny. Zastosowanie takiego pilota może przynieść wymierne korzyści nie tylko klientowi, lecz także samemu programiście, który przy takiej okazji zyska wiele nowych możliwości. Nie wspomnę już tutaj o czysto hobbystycznych rozwiązaniach, w których obniżanie kosztów wykonania projektów, ale i estetyka powinny stać zawsze na najwyższym miejscu.

A

B

Rysunek 1. A – pilot w pełni zgodny ze standardem RC5 (możliwość zmiany opisów klawiszy). B – pilot uniwersalny, nadający w wielu różnych standardach do wyboru za pomocą konfiguracji (RC5, Sony, Daewoo itp.). Piloty dostępne na: www.sklep.atnel.pl

Wbrew pozorom piloty tego typu są bardzo tanie, a dodatkowo mają możliwość pracy w popularnym standardzie RC5, z którym są w pełni zgodne.

Dlatego jeszcze raz bardzo cię zachęcam do szerszego korzystania z tego typu gotowych klawiatur, i to klawiatur na podczerwień. Być może jest to określenie nieco na wyrost, ale nie do końca. Chodzi tutaj przecież o najzwyklejszą wygodę użytkowania, oprogramowania, jak i estetykę wykonania całego projektu. Cały rozdział na temat podczerwieni ma na celu pokazanie, że wykorzystanie praktycznie każdego, wręcz dowolnego pilota podczerwieni, nawet gdy nie znasz jego producenta i danych na temat jego standardu, będzie można w prosty sposób wykorzystać do swoich potrzeb. Nieważne, w jakim standardzie nadaje, i czy jest to pilot Sony, Philips, JVC czy wspomniany wyżej „no name".

Wyobraź sobie także, że obsługa każdego z nich, podkreślam – każdego, będzie zawsze taka sama z punktu widzenia twojego programu. Spróbujmy zatem przyjrzeć się w szczegółach, co tak naprawdę odstręcza wiele osób w oprogramowaniu pilotów. Ja mógłbym to wymienić w kilku punktach:

1. Przeświadczenie, że obsługa pilota w języku C sprowadza się zawsze do pisania prawie od początku całej implementacji jego obsługi.
2. Brak jednoznacznych standardów korzystania w sposób uniwersalny z różnych pilotów.
3. Przeświadczenie graniczące z pewnością, że próba użycia innego pilota niż RC5 będzie się wiązała z tak ogromnym nakładem pracy, że od razu lepiej przejść do budowy klawiatury na mikroprzełącznikach.
4. Brak dostępnych uniwersalnych bibliotek z uwagi na setki, a wręcz tysiące różnych sposobów kodowania dla różnych pilotów.
5. Zniechęcenie do tej technologii po kilku nieudanych próbach opanowania jej za pomocą własnego lub cudzego programu.

Po zapoznaniu się z przyczynami najczęstszych niepowodzeń związanych z pomysłami praktycznego wykorzystania podczerwieni, warto określić założenia, jakie powinna spełniać własna i zarazem uniwersalna biblioteka do obsługi pilotów IR. Mowa tu oczywiście o szerokim zastosowaniu pilotów jako najprostszych form klawiatur. Zapoznanie się ze szczegółami zagadnień z tym związanych pozwoli także tobie pomyśleć o innych jeszcze zastosowaniach podczerwieni. Można bowiem wykorzystać ją w niektórych przypadkach do zwykłej transmisji danych pomiędzy kilkoma urządzeniami, o ile znajdują się „w zasięgu wzroku z ich punktu widzenia". Podczerwień to także czujniki odległości, bariery optyczne itp. Poniżej kilka najważniejszych założeń, jakie będziemy chcieli uzyskać podczas wykorzystania zdalnych klawiatur:

1. Jedna biblioteka z możliwością konfiguracji parametrów pracy jeszcze przed kompilacją za pomocą pliku konfiguracyjnego typu *.h.
2. Możliwość korzystania z tej samej biblioteki na różnych mikrokontrolerach AVR.
3. Uniezależnienie pracy biblioteki od częstotliwości taktowania mikrokontrolera.
4. Zawsze taka sama obsługa w programach od strony użytkownika – nowy standard.
5. Możliwość obsługi co najmniej kilku rodzajów pilotów wybieranych na etapie konfiguracji przed kompilacją.

6. Działanie w sposób „nieblokujący" pracy pętli głównej – oparte na zdarzeniach.
7. Jak najmniejsze zużycie zasobów sprzętowych mikrokontrolera.
8. Prostota pod względem wprowadzania własnych modyfikacji.
9. Minimalne obciążenie procesora (praca w tle).

Wystarczy. Sam przyznasz, że gdyby udało się spełnić powyższe założenia, to nagle zaczyna się rodzić w głowie coraz więcej pomysłów na zastosowania praktyczne. Poczekaj jednak jeszcze chwilę, aż przybliżone zostaną techniczne szczegóły obsługi i możliwości. To otworzy ci jeszcze szersze pole do popisu wyobraźni.

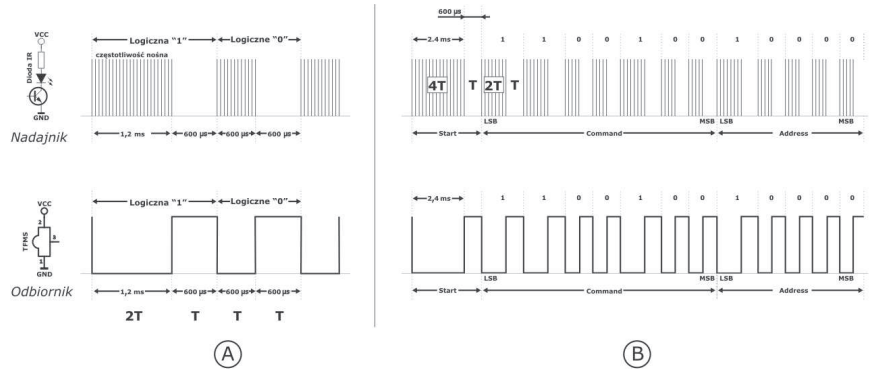
Jak sobie zapewne przypominasz, standard RC5 to inaczej kodowanie bifazowe znane także pod nazwą kodowania „Manchester". W mojej poprzedniej publikacji „Mikrokontrolery AVR. Język C. Podstawy programowania" w rozdziale dotyczącym odbioru i dekodowania kodów pilota w tym standardzie użyłem sprzętowego przerwania ICP, wykorzystując do tego moduł 16-bitowego timera nr 1. Tutaj także skupię się na takim pisaniu procedur dla obsługi innych standardów/protokołów, które zapewnią eliminację błędnych/uszkodzonych ramek, zwiększając przez to niezawodność. Uszkodzenia ramek mogą wynikać z kilku powodów. Najbardziej trywialnym jest prawdopodobieństwo odbicia emitowanej wiązki podczerwieni od ścian lub przedmiotów, które spowoduje (co ławo wyobrazić sobie), że na końcu do odbiornika może nadlecieć zmodyfikowany i zakłócony sygnał. Trzeba poradzić sobie z jego odrzuceniem, podobnie jak w przypadku zakłóceń spowodowanych między innymi świetlówkami, które emitują spore ilości podczerwieni na przykład podczas włączania. Także niekorzystne mogą być zakłócenia spowodowane innymi rodzajami pilotów. Zapewne nie chciałbyś zrobić urządzenia sterowanego własnym pilotem i nawet własnym protokołem w podczerwieni w taki sposób, żeby ramki z całkiem innego pilota spowodowały nieprzewidziane reakcje, odbierając przypadkowe wartości poleceń. Łatwo sobie wyobrazić na przykład własnego robota startującego w zawodach, gdy nagle tracisz z nim kontakt z powodu emisji przez sonary podczerwieni przeciwnika na ringu.

Przed przystąpieniem do pisania programu, warto się zapoznać, przynajmniej w podstawowym zakresie, z pozostałymi standardami kodowania w podczerwieni. Wyróżniłbym trzy podstawowe rodzaje, z którymi będziesz spotykał się najczęściej:
– RC5 (znane już kodowanie bifazowe lub kodowanie Manchester),
– kodowanie typu Pulse – wykorzystywane na przykład przez Sony (w protokole SIRCS),
– kodowanie typu Space – wykorzystywane na przykład przez firmę JVC.

Podstawy działania RC5 przedstawiłem dokładnie w mojej poprzedniej książce, teraz omówię krótko dwa kolejne standardy. Dodam tylko, że bardzo często zarówno Pulse, jak i Space określane są jednym mianem, tylko Space. Spotkasz się z takim prostym podziałem na witrynie www.lirc.org/remotes/, serwującej dokładne opisy standardów nadawania setek, jeśli nie tysięcy najróżniejszych pilotów podczerwieni z całego świata. Strona ta będzie także dla nas nieocenionym źródłem informacji podczas dalszych prac.

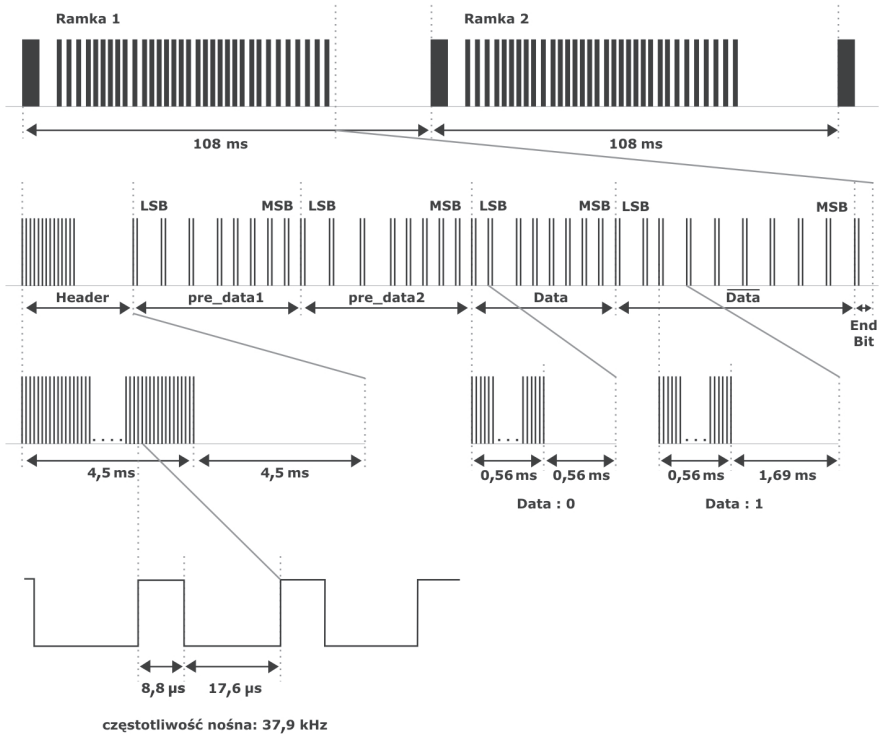
1.1. Kodowanie typu Pulse

Na rysunku 2 w części A przedstawiłem charakterystykę nadawania pojedynczych bitów w tym standardzie, a w części B – kompletny widok przesyłanej ramki danych. Kodowanie typu Pulse związane jest z tym, że występuje tu regulacja szerokości impulsu (nośnej) przy zachowaniu stałego czasu przerwy w ramach jednego bitu. Na rysunku widzisz przykładowy protokół SIRCS (Sony), w którym niezmienny czas przerwy występujący w każdym bicie równa się T = 600 µs. Bit prezentujący wartość 1 składa się zatem z nośnej emitowanej w czasie 2T = 1200 µs oraz wspomnianej przerwy T = 600 µs. Natomiast bit prezentujący wartość 0 składa się z nośnej emitowanej w czasie T = 600 µs oraz przerwy o czasie równym także T = 600 µs. Dla ułatwienia przedstawiłem widok emisji pojedynczych bitów od strony nadajnika, a także po stronie odbiornika.



Rysunek 2. Kodowanie Pulse

Widać, że sygnał odwrócony jest w fazie. Inne protokoły nadające w tym standardzie różnią się przede wszystkim czasem trwania przerwy, która wcale nie musi wynosić 600 µs, jak również czasem trwania nośnej w trakcie pierwszej części bitu o zmiennej długości. Wiesz już zatem, jak wygląda pojedynczy bit. Na rysunku 2 w części B, zgodnie z tym, co powiedzieliśmy, mamy rozrysowany widok już kompletnej ramki. Jak już napisałem w „Mikrokontrolerach AVR. Język C. Podstawy programowania", w standardzie RC5 mieliśmy do czynienia z 2 bitami startowymi, które miały za zadanie pomóc w synchronizacji odbiornika, ale także pomóc w odróżnieniu standardu ramki. Stanowiły one swego rodzaju stały nagłówek ramki. W przypadku każdego protokołu mamy podobną sytuację. Na początku pojawia się nagłówek, z tym, że tutaj jest on zwykle odmienny niż w standardzie RC5. Jak widzisz na rysunku, w protokole SIRCS nagłówek składa się z dwóch części, przy czym pierwsza to nośna emitowana przez czas 4T = 2400 µs, a druga to przerwa o czasie trwania T = 600 µs. Podczas dekodowania sygnałów tego typu rozpoznanie charakterystycznych długości nagłówka decyduje o tym, z jakim protokołem mamy do czynienia. W dalszej części, już po nagłówku, emitowane są kolejne bity. Ta część ramki także mocno zależy od protokołu, a nawet od konkretnej wersji pilota nadającego w takim samym protokole. Na przykład piloty nadające w standardzie SIRCS mogą emitować od 12 do 16 bitów w jednej ramce. Najczęściej spotykany jest



Rysunek 4. Ramka pilota firmy Samsung

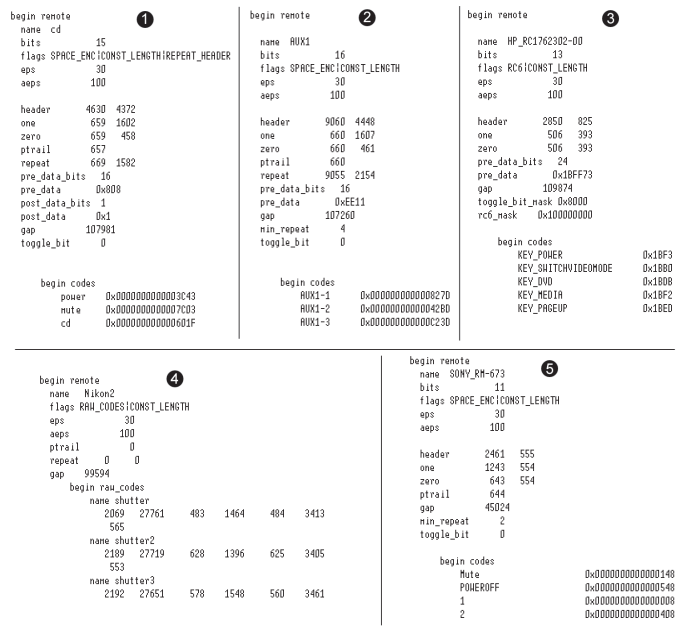
zwanych `pre_data`. Posiadają one zawsze stałą i niezmienną wartość, która dodatkowo, poza samym nagłówkiem, charakteryzuje wybrany wręcz model pilota/urządzenia. W pewnym sensie to jakby odpowiednik pola `address` z poprzednich protokołów, tyle że nie zmienia się on w nadawanych ramkach, nawet jeśli ten sam pilot ma klawisze służące do sterowania zarówno urządzeniem TV, jak i VCR. W dalszej części ramki mamy 2 kolejne pełne bajty, czyli 16 bitów. Wbrew pozorom nie oznacza to jednak, że występuje tutaj większa liczba kombinacji niż 256, jaka dostępna na jednym bajcie. Zwróć uwagę na to, że drugi bajt o nazwie DATA ma poziomą kreskę nad nazwą. To oznacza, że jest on po prostu zanegowaną wartością wcześniejszego bajtu DATA. Wszystkie te zabiegi związane są ze zwiększeniem niezawodności przesyłanych ramek, tak aby nie zdarzały się przypadkowe wartości. Z tego też powodu zawsze należy sprawdzać wartość tych dwóch odebranych bajtów. Na dole rysunku podano także dokładną częstotliwość nośnej oraz, co ważne, także stosunek wypełnienia używany w tym konkretnym protokole. Rzadko jednak będziemy mieli do dyspozycji tak dokładne instrukcje dotyczące interesujących nas protokołów. Będzie się można posiłkować najczęściej danymi prezentowanymi na witrynie lirc.org. Zagadnienie to jest szerzej omówione w następnym podrozdziale.

1.4. Analiza dokumentacji z witryny lirc.org

Jak zapewne zauważyłeś, w już kilkakrotnie powoływałem się na dokumentację do przeróżnych pilotów podczerwieni, jaka znajduje się w witrynie www.lirc.org/remotes/ (lub http://lirc.sourceforge.net/remotes/). Niestety, w ramach tej dokumentacji występuje kilka nieścisłości. Dotyczą one przede wszystkim tego, że:

– zarówno standard PULSE, jak i SPACE opisywany jest tylko za pomocą określenia space,
– impuls PTRAIL jest nieraz zbędnie podany (nie powinno go być),
– brak rozgraniczenia ramki na `command`, `address` lub `data`, w zamian podane są tylko wartości HEX dla poszczególnych klawiszy pilota zapisane w obydwu bajtach,
– nie dowiemy się nic na temat częstotliwości nośnej.

Zanim napiszę więcej na ten temat, warto przyjrzeć się najpierw przykładowym opisom kilku pilotów. Całkowicie różne ich typy wybrałem przypadkowo, jednak zwróciłem uwagę, aby występowała różnorodność w kwestii przedstawianych parametrów. Na pierwszy rzut oka opisy protokołów bardzo różnią się od siebie. Pomimo to myślę, że po wcześniejszych wyjaśnieniach nie będziesz już miał problemów z rozpoznaniem samych standardów kodowania. Mam na myśli te typowe, jak: RC5, Space oraz Pulse. Jedynie przykłady oznaczone numerami 3 oraz 4 mogą wydać ci się nieznajome. Nie wątpię, że na podstawie podanych informacji sam już poradzisz sobie z przykładem numer 3, prezentującym standard RC6 (następca RC5), ale w przypadku przykładu numer 4 należy ci się kilka słów wyjaśnienia.



Rysunek 5. Przykłady opisów z lirc.org

Opis numer 5 na rysunku 5 nie zawiera typowych informacji, jak *header*, *one*, *zero* itp. Spotykamy się za to z określeniem *Raw codes*:

```
gap   99594
begin raw_codes
  name shutter
    2069   27761    483    1464   484    3413   565
  name shutter2
    2189   27719    628    1396   625    3405   553
```

Cóż to oznacza? Czyżby jakiś nowy standard? Nie. Tak prezentowane są informacje, gdy nie występuje żaden znany standard nadawania. Podane są kolejno czasy, jakie zostały zmierzone dla nadlatujących z pilota impulsów. Widać nazwy *shutter*, *shutter2* itd. Czy to są kody klawiszy na pilocie? Nie. To są powtórzone wielokrotnie czasy dla tej samej ramki (tego samego klawisza), przy czym za każdym razem pomiar wykazuje minimalne różnice pomiędzy poszczególnymi odcinkami. Oznacza to tylko to, że występuje tutaj dosyć spora tolerancja. W rzeczywistości ten pilot do aparatu fotograficznego Nikon posiada fizycznie tylko jeden klawisz do wyzwalania migawki. Gdybyśmy chcieli wykonać we własnym zakresie takiego pilota należałoby wygenerować następującą sekwencję:

```
Nośna  – czas   ok. 2 100 µs
Przerwa – czas  ok. 27 730 µs
Nośna  – czas   ok. 550 µs
Przerwa  – czas ok. 1 430 µs
Nośna  – czas   ok. 550 µs
Przerwa  – czas ok. 3 409 µs
Nośna  – czas   ok. 550 µs
Przerwa między ramkami GAP – czas ok. 99 594 µs
```

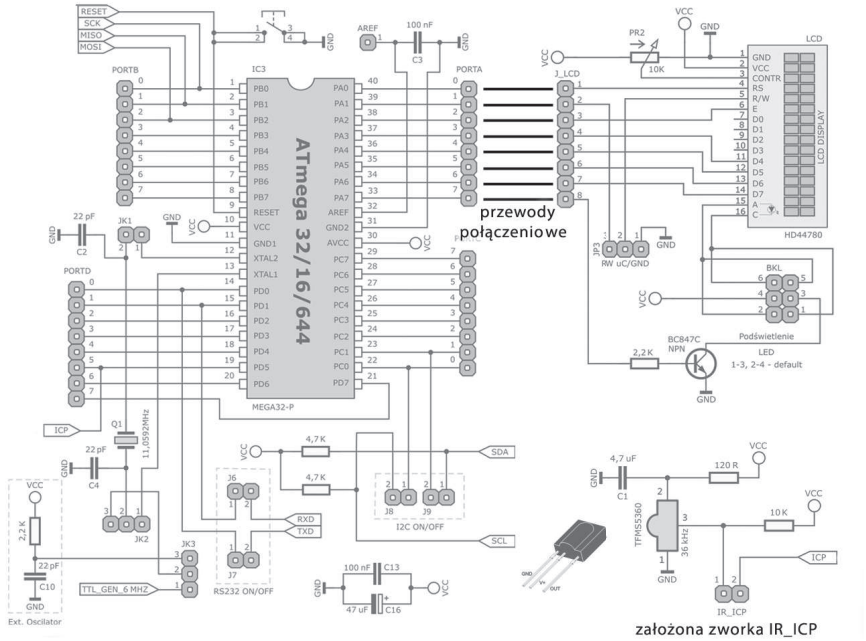
W ten prosty sposób mamy gotowy pilot do aparatu. Pozostanie jeszcze kwestia doboru częstotliwości nośnej, ponieważ szczególnie w przypadku aparatów może zachodzić potrzeba uzyskania jak największego zasięgu. Warto zacząć od nośnej 36 kHz i sprawdzić maksymalny zasięg przy tym samym prądzie diody nadawczej. Następnie zwiększyć częstotliwość do 38 kHz, następnie do 40 kHz itp. Jeśli przy jednej z częstotliwości uzyskasz największy zasięg, to sam empirycznie sprawdzisz, jaki to zakres. Zwykle częstotliwości nośne nie są mniejsze niż 32 kHz i nie większe niż 45 kHz. Zatem masz nieduży zakres do sprawdzenia. Ja bym go jednak ograniczył do 36–40 kHz.

Wracając do rysunku 5, w przykładach numer 1 oraz 2 mamy do czynienia z kodowaniem typu Space, a w przykładzie numer 5 – z kodowaniem Pulse. Po czym można to rozpoznać na podstawie prezentowanych opisów? Spójrz, proszę, na dwa fragmenty opisów z przykładu numer 1 i 5 (tabela 1).

Tabela 1. Kodowanie Space i Pulse

Nazwa	Nośna	Przerwa
header	4630	4372
one	659	1602
zero	659	458
ptrail	657	
repeat	669	1582

Nazwa	Nośna	Przerwa
header	2461	555
one	1243	554
zero	643	554
ptrail	644	
gap	45024	



Rysunek 6. Schemat połączeń do ćwiczenia podczerwieni w zestawie ATB

nagłówka oraz bitów wraz z impulsem PTRAIL. Ponieważ sami zrobimy odbiór kodów, na razie nie interesuje nas kwestia odstępu między ramkami „gap". Najpierw wykorzystane zostanie, podobnie jak w książce „Mikrokontrolery AVR. Język C…", przerwanie ICP, 16-bitowego timera. Całość zostanie uruchomiona w moim przypadku na zestawie uruchomieniowym firmy ATNEL, o symbolu ATB rev:1.02 (to ćwiczenie można także wykonać na bliźniaczym zestawie ATB rev:1.01). Użyję procesora ATmega32. Schemat połączeń w tym przypadku jest banalnie prosty. Cały port A mikrokontrolera wykorzystam do podłączenia wyświetlacza LCD wraz z podświetleniem w takim porządku, że sygnały RS, RW oraz E będą sterowane kolejno przez PA0, PA1 oraz PA2. Następnie od PA3 do PA4 podłączę 4-bitową szynę danych D3-D7. Natomiast podświetlenie LCD podłączę do PA7. Na rysunku 6 przedstawiony jest poglądowy schemat połączeń dokonanych za pomocą przewodów połączeniowych na zestawie. Wykorzystałem 8 sztuk przewodów, natomiast sam odbiornik podczerwieni można wygodnie podłączać do wejścia ICP mikrokontrolera za pomocą jednej zworki, oznaczonej IR_ICP, znajdującej się tuż obok odbiornika TFMS. Wyświetlacz LCD ma założoną domyślnie zworkę umożliwiającą podłączanie pinu R/W do mikrokontrolera. Ponieważ w poprzedniej publikacji udało nam się razem wypracować dosyć dobrze działającą bibliotekę do obsługi LCD, to w tej lekcji zostanie ona również użyta. Przed kompilacją zajmiemy się tylko skonfigurowaniem pinów w pliku nagłówkowym *.h w folderze LCD.

Tym razem, jako że chcemy połączyć przyjemne (poznanie IR od podszewki) z pożytecznym (kontynuacja nauki języka C dla AVR), przygotujemy całość, opierając się na prawdziwym mechanizmie zdarzeń. Wprawdzie od tego blisko już do języka C++, my jednak pozostaniemy jeszcze przy samym języku C. Zacznę od przedstawienia i omówienia pliku nagłówkowego *ir_sony.h*, który należy przygotować tak, aby później korzystać z obsługi protokołu Sony jak z biblioteki. Będzie w nim widać jak na dłoni, czego albo jakich właściwości możemy się spodziewać po takiej bibliotece oraz co i jak możemy ustawiać/konfigurować według własnych bieżących potrzeb:

```
#ifndef IR_SONY_H_
#define IR_SONY_H_

/* ****   KONFIGURACJA   **** */
/* ****   USTAWIENIA UŻYTKOWNIKA   **** */

#define IR_PIN (1<<PD6)  // numer pinu wejścia ICP
#define IR_DIR DDRD
#define IR_PORT PORTD

// podajemy wartość 1, jeśli chcemy użyć Virtual Toggle Bit, lub 0 jeśli nie
#define VIRTUAL_TOGGLE 1

// podajemy wartość 1, jeśli chcemy przemapować najważniejsze klawisze na
standard RC5
// lub wartość 0, jeśli chcemy oryginalne kody SONY
#define KEYS_REMAP 0

/* **** USTAWIAMY WARTOŚCI SPECYFICZNE DLA SONY SIRCS  **** */
// makro przeliczające czasy w µs w zależności od F_CPU (automatycznie)
#define ir_micro_s(num) ((num)*(F_CPU/1000000UL)/8)
// czas pierwszej charakterystycznej części nagłówka (headera)
#define SONY_HEADER ir_micro_s(2440)
// czas trwania bitu o wartości = 1
#define SONY_BIT_HIGH_MID ir_micro_s(1240)
// 2/3 czasu trwania pierwszej połowy bitu o wartości = 1
#define SONY_BIT_HIGH_MID ir_micro_s(925)
// czas trwania pierwszej połowy bitu o wartości = 0
#define SONY_BIT_LOW ir_micro_s(650)
// toleracja, jaką przyjmiemy dla naszego pilota. Jeśli będzie za duża
#define SONY_TOLERANCE ir_micro_s(150)
// ta procedura może zacząć odbierać niechcący czasem dziwne kody
// z pilotów innych producentów; jeśli będzie za mała, to może się
// zmniejszyć zasięg pilota albo będzie gorzej działał z odbicia np. od ścian itp.
// optymalne wartości zwykle zawierają się w granicach 100-250

/* **** KONIEC  USTAWIEŃ  UŻYTKOWNIKA **** */

// komunikaty na temat stanu ramki w trakcie jej dekodowania
#define FRAME_RESTART 0
```

stą eliminacją drgań styków i wygenerowanie dwóch różnych kodów klawiszy (`command`). Zmienna `address` posiada stałą wartość = 1, ale możesz to zmieniać według uznania, jeśli sam wykonujesz odbiornik. Kompilujemy program, uruchamiamy i proszę bardzo. Z jednej strony program reaguje na fabryczny pilot, wyświetlając poprzez funkcję `moja()` na LCD `address`, `command` oraz nawet `key_time`, gdyż załączyłem mechanizm *Virtual Toggle Bit*. (należy pamiętać tylko, że Timer0 zajęty jest tutaj na potrzeby generowania nośnej, więc w konfiguracji biblioteki podczerwieni wybrałem Timera2 do obsługi *Virtual Toggle*).

Okazuje się, że wszystko działa wyśmienicie. Mam nadzieję, że przerobienie takiego nadajnika na potrzeby na przykład pilota Samsung, nadającego w standardzie Space, nie będzie dla ciebie trudne. Funkcja nadająca ramkę będzie się tylko nieco różniła. Do zmiennej `data` wpiszemy do młodszego bajtu kod `argumentu adr`, natomiast do starszego – zanegowaną wartość bajtu `adr`. Następnie po wygenerowaniu nagłówka konieczne będzie wygenerowanie 16 bitów `pre_data`. Można sobie utworzyć dodatkową zmienną 16-bitową, do której ułożymy podane na lirc.org bity. W dalszej kolejności pozostanie nam już tylko wygenerować całą 16-bitową ramkę i ewentualnie sprawdzić, czy będzie konieczne wygenerowanie impulsu PTRAIL. Podobny sposób postępowania będzie obowiązywał z każdym pilotem. Chyba że będzie to coś w rodzaju Pentaxa, którego ramki pokazałem wyżej. W takim przypadku wystarczy wprost po kolei w krótkiej pętli wygenerować kolejne impulsy, pobierając nawet ich czasy na przykład ze zmiennej tablicowej typu `uint16_t`, którą uprzednio przygotujemy i wpiszemy wszystkie czasy z rysunków. To znaczy po jednej tabeli dla każdego klawisza. To jest chyba najprostszy sposób wygenerowania takich bardzo nietypowych ramek, które nie są związane z żadnym standardem.

1.10. Transmisja własnych danych w podczerwieni

Kolejnym ciekawym zastosowaniem podczerwieni we własnych projektach może okazać się transmisja różnego rodzaju danych pomiędzy różnymi urządzeniami. Tyle że tym razem będziemy musieli we własnym zakresie nauczyć się tworzyć ramki danych. Opracować własne standardy, zamiast korzystać z gotowych, jakie oferowane są w pilotach omówionych wyżej. W pierwszej kolejności trzeba będzie rozważyć to, jaki standard kodowania wybrać. Moim zdaniem najlepszy będzie tu popularny RC5, ponieważ mamy już opanowane zarówno nadawanie, jak i odbiór. Dodatkowo wykorzystamy zaletę sprzętowego bitu *Toggle*. (Sposób nadawania ramek RC5 szczegółowo opisałem w mojej poprzedniej książce „Mikrokontrolery AVR. Język C. Podstawy programowania", więc nie będę go tu przytaczał, za to na pewno będę się powoływał na wiedzę tam przekazaną).

Przymierzając się do tego zadania, najpierw musimy ustalić ogólne założenia odnośnie do potrzeb i sposobu przesyłu danych. Wiadomo, że nie będziemy korzystali z transmisji UART, jednak w równie wygodny sposób, bo w przerwaniach, będziemy w stanie odbierać nasze dane. Dobre pytanie na początek naszych rozważań jest takie: jak duże ilości danych/bajtów chcemy przesyłać oraz jaką przepustowość? Wydaje mi się, że to

bardzo mocno zależy od faktycznych potrzeb bieżących, więc na pewno nie będziemy dążyć tutaj do zorganizowania transmisji, która z powodzeniem zastąpi nam USART. Do porozumiewania się tą drogą z czujnikami temperatury, do sterowania silnikami w urządzeniach oraz do wielu innych tego typu zagadnień wystarczy przesyłanie prostych ramek danych zawierających przynajmniej dwa pełne bajty. Nasze założenia nie będą nastawione na uzyskiwanie dużych prędkości. Za to pokusimy się o przygotowanie komunikacji dwustronnej typu *half duplex* (możliwość albo nadawania, albo odbioru w jednym czasie). Należy pamiętać również, że skoro mowa o transmisji w podczerwieni, to zastosowania ograniczają się tylko do komunikacji pomiędzy urządzeniami znajdującymi się w zasięgu wzroku, bez żadnych przeszkód pomiędzy nimi oraz do niewielkich odległości (od kilku do kilkunastu metrów). Dodam jeszcze jeden powód, dla którego decydujemy się na standard kodowania RC5 (*Manchester*). Dzięki niemu można maksymalnie skrócić czas trwania przesyłanych ramek danych. Wszystkie czasy kolejnych impulsów są praktycznie równe, podczas gdy w innych standardach mamy do czynienia z bardzo długimi nagłówkami, a przez to występuje konieczność stosowania o wiele dłuższych przerw między ramkami. Biblioteka, którą stworzymy, nie będzie być może najbardziej zoptymalizowana. Ma ona jednak na celu pokazanie jednej z metod stworzenia własnego standardu komunikacji.

Podstawowe założenia:

1. możliwość przesyłania minimalnie pojedynczych ramek składających się z dwóch pełnych bajtów danych oraz bitu *Toggle*;
2. możliwość przesyłania krótkich łańcuchów tekstowych *(stringów*), maks. długość – 16 bajtów;
3. dwustronna komunikacja;
4. prosta obsługa z możliwością dowolnej modyfikacji;
5. odbiór oparty na przerwaniu ICP (Timer1);
6. nadawanie oparte na nośnej generowanej przez Timer0 (wyjście OC0) lub Timer2 (wyjście OC2).

Na początek przyjrzyjmy się możliwościom naszej nowej biblioteki, zaglądając do pliku nagłówkowego *ir_man.h*.

```
#ifndef IR_MAN_H_
#define IR_MAN_H_

/* **** K O N F I G U R A C J A ***** */
/* **** U S T A W I E N I A  U Ż Y T K O W N I K A  **** */
// Wybór timera 8-bitowego do generowania nośnej
// Timer0 - wyjście OC0
// Timer2 - wyjście OC2
#define TIMER_NR 2

#define IR_PIN (1<<PD6)  // numer pinu wejścia ICP
#define IR_DIR DDRD
#define IR_PORT PORTD
```

2. Transmisja radiowa 433/868 MHz i 2,4 GHz

Wspominałem wcześniej, że przejście z tematyki nadawania/kodowania w podczerwieni do nadawania/kodowania drogą radiową w popularnych pasmach 433/868 MH będzie płynne. Sporo już wiesz na temat podstawowych sposobów kodowania sygnałów, które musi zapewnić eliminację błędów podczas transmisji. Mam tu na myśli kodowanie typu Manchester. To właśnie na nim oprzemy się podczas omawiania tych zagadnień. Istnieje jednak podstawowa różnica w naszym podejściu do kodowania bifazowego, jeśli mamy użyć nadajnika i odbiornika pracującego na paśmie na przykład 433 MHz od tego, co było w podczerwieni. Poprzednio, jak dobrze pamiętasz, musieliśmy zadbać sami o częstotliwość nośną, którą po prostu modulowaliśmy, chcąc przesyłać pojedyncze bity. W tym przypadku częstotliwość nośną (433 MHz/868 MHz) zapewniają nam już z marszu komplety nadajnik/odbiornik lub transcievery (transciever, czyli odbiornik i nadajnik w jednym module). Zanim przejdę dalej, poinformuję tylko, że w ramach omawianego tematu bierzemy pod uwagę tylko takie nadajniki/odbiorniki, które mają proste wejścia/wyjścia służące do kluczowania naszego sygnału, który chcemy transmitować, oraz wyjście, które zachowuje się podobnie jak wyjście odbiornika podczerwieni typu TFMS. Nie są to zatem moduły mające na pokładzie własne mikrokontrolery, które zapewniają nam podstawowe mechanizmy kodowania/dekodowania, ale za to w zamian pozwalające na prostszą transmisję z wykorzystaniem już znanych ci protokołów transmisji, jak RS232, SPI oraz I2C. Dlaczego w takim razie zdecydowałem się opisać te, jakby się mogło wydawać, trudniejsze rozwiązania? Tutaj zaprotestowałbym, że są to trudniejsze rozwiązania. Każdy rodzaj modułów do zrealizowania transmisji radiowych ma swoje wady i zalety. Jest ich cała masa dostępnych na rynku, wielu producentów oraz mnóstwo rozwiązań dla użytkowników. O tym, jaki moduł wybierzemy, nie zawsze decyduje cena, choć jest to bardzo ważny parametr.

Myślę, że dokonując wyboru, warto poznać chociaż pobieżnie sposoby działania kilku różnych modułów i spróbować określić kilka czynników, jak: stopień skomplikowania procedur nadawczo-odbiorczych, cena za komplet lub transciever, liczba elementów zewnętrznych potrzebnych do ich uruchomienia, sposób doboru i skuteczności anteny, a także zasięgi i stopień komplikacji, jeśli chodzi o inicjalizację programową modułów (gdy jest wymagana). Ale i chyba najważniejszym zagadnieniem są potrzeby, gdyż nie każdy rodzaj komunikacji nadaje się wprost do pewnych zastosowań.

Dlatego wymienię tu w pewnej kolejności kilka różnych modułów i je scharakteryzuję (będzie to bardzo uproszczona charakterystyka, mająca na celu wyłącznie to, abyś mógł się jak najszybciej zorientować, w jakim kierunku iść podczas tworzenia prototypu, a nawet w fazie powstawania idei prototypu):

1. komunikacja Bluetooth oraz znane moduły BTM-222 lub BTM-112 (pasmo 2,4 GHz),
2. komunikacja WiFi – bezprzewodowe moduły kart sieciowych (pasmo 2,4 GHz),

3. scalone dwukierunkowe transcievery zapewniające komunikację poprzez standardowe łącza typu: RS232, SPI, I2C (pasma od 433 MHz do 2,4 GHz),
4. najprostsze nadajniki/odbiorniki pracujące z modulacją AM/FM, posiadające jedno wejście kluczujące w nadajniku oraz jedno wyjście kluczowanego sygnału w odbiorniku.

Nie bez powodu umieściłem moduły Bluetooth na pierwszym miejscu. Rzeczywiście posługiwanie się nimi od strony praktycznej to pestka. Jeśli chcemy pracować na domyślnie ustawionych parametrach, to w zasadzie niczego nie trzeba konfigurować ani ustawiać. Moduły działają po prostu jak zwykły kabel RS232, tyle że bezprzewodowy. Jeśli zaś chcemy zmienić ich konfigurację, na przykład nadać inną nazwę w otoczeniu sieciowym lub zmienić prędkość RS232, to konfiguracji takiej dokonujemy jednorazowo i wcale nie potrzebny jest nam do tego mikrokontroler. Wystarczy zwykła przejściówka USB/RS232 lub na układzie MAX232 i całą konfigurację dokonujemy z poziomu terminala w komputerze PC. Jest ona zapamiętywana na stałe w pamięci EEPROM modułu. Ich ogromną zaletą jest to, że mamy do dyspozycji w pełni dwustronną komunikację i to nawet w jednym czasie (*full duplex*), najbardziej naturalny i pospolity standard komunikacji RS232, a do tego do wyboru prawie dowolne prędkości. Czegóż więc jeszcze oczekiwać? A gdy dodamy do tego, że w celu komunikacji z komputerem nie musimy dokonywać zakupu dwóch takich modułów (wystarczy tylko jeden), ponieważ w komputerze można użyć albo wbudowanej (na przykład w laptopach) przejściówki USB/BT, albo dokupić taką przejściówkę za połowę ceny modułu BTM, to wydawać by się mogło, że to idealne rozwiązanie do każdego celu. Niestety, komunikacja tego typu pomimo szeregu zalet w ogóle nie nadaje się do niektórych zastosowań. Ma wręcz poważne wady z punktu widzenia prób zastosowania jej do zbudowania sieci czujników, które mogłyby porozumiewać się z układem nadrzędnym typu Master w sposób podobny do sieci RS485. Kolejna wada to dosyć spory pobór prądu, co od razu eliminuje długi szereg rozwiązań bateryjnych, na przykład pilotów zdalnego sterowania. Przy czym jeśli chodzi o pomysł wykorzystania ich w pilotach, to kolejną wadą byłoby to, że czas negocjacji i pierwszego nawiązania połączenia ze sobą dwóch modułów (bo tylko tak mogą pracować – połączenie jeden do jednego) stawiałby pod znakiem zapytania wykorzystanie pilota na przykład do zamykania lub otwierania bramy garażowej. Kolejna wada w porównaniu z innymi modułami radiowymi to zasięg. Tylko, proszę, dobrze mnie zrozum. Słabszy zasięg jako taki nie jest żadną wadą w ogólnym rozumieniu. W przypadku typowych zastosowań Bluetootha bywa to wręcz zaletą, dlatego też spotkasz się z modułami pracującymi w standardzie Class1 (zasięg do 100 m) na przykład BTM-222, ale także z modułami pracującymi w standardzie Class2 (zasięg tylko do 10 m). Po cóż bowiem taka na przykład myszka BT miałaby rozsiewać swoje informacje na 100 m i zakłócać pracę ew. innych urządzeń pracujących w tym samym paśmie. Jeśli jednak weźmiemy pod uwagę komunikację choćby małej stacji meteo, znajdującej się w odległości, powiedzmy, 150–200 m od miejsca centrali, gdzie po drodze nie ma żadnych zabudowań, przeszkód terenowych, to już moduły BT się nie przydadzą. Na pewno trzeba będzie sięgnąć po kolejne rodzaje modułów z listy podanej wyżej.

Punkt drugi obejmuje wszelkiego rodzaju moduły kart sieciowych WiFi, które nadają się do wykorzystania w niewielkich wbudowanych systemach opartych na małych mi-

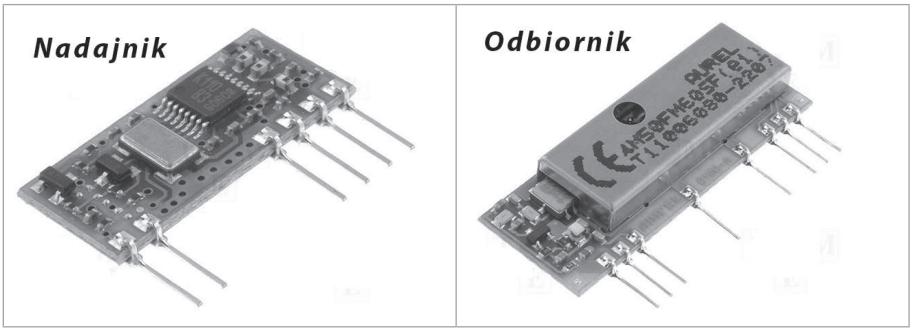
krokontrolerach. Przytoczę tu przykład jednego z czytelników, który napisał do mnie w e-mailu, że do połączenia bezprzewodowo swoich dwóch urządzeń na mikrokontrolerach użył właśnie tego typu karty. Wydawało mu się, że jej obsługa, jeśli chodzi o komunikację pomiędzy urządzeniami albo pomiędzy komputerem PC, sprowadzi się do przesłania kilku interesujących go bajtów. Jakież było jego zdziwienie, gdy dowiedział się, że do tego celu musiałby zaimplementować w swoim mikrokontrolerze jeszcze sporą część stosu TCP, w ogóle zapoznać się z taką implementacją, a także napisać własne drivery do karty sieciowej i stosu TCP – po to aby na końcu nauczyć się przesyłać informacje za pomocą jednej z warstw stosu czy też wyższych protokołów, jak UDP, http lub TCP.

Reasumując, rozwiązania takie są oczywiście wspaniałe, występują nawet karty z już wbudowanym stosem TCP lub możliwością jego obsługi za pomocą poleceń AT, tak jak to ma miejsce na przykład w modemach GSM przy wykorzystaniu łączności GPRS. Nadal jednak jest to dużo bardziej skomplikowane zagadnienie niż wykorzystanie najprostszych nadajników. I również w tym przypadku coraz niższe ceny lub co ważniejsze niesamowicie niski pobór prądu w stanie bezczynności nie są wyznacznikiem tego, że w każdym przypadku opłacalne jest użycie takiego rozwiązania. Poruszamy się zatem w dół listy.

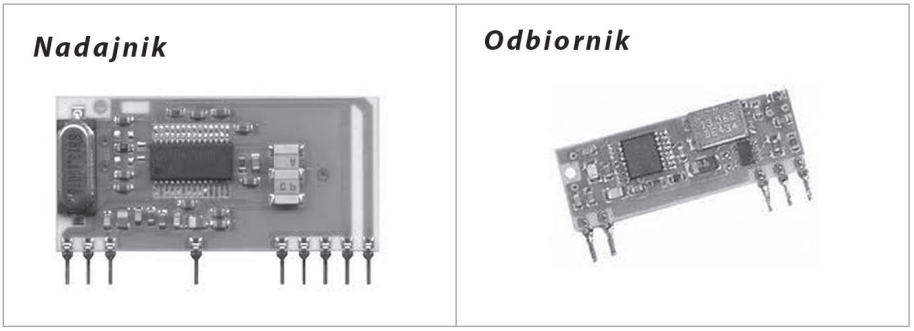
Punkt trzeci obejmuje moduły, które mogą się wydawać na pierwszy rzut oka proste i mało skomplikowane w użyciu. W tej klasie ceny bywają chyba najniższe z możliwych, jeśli spojrzymy na moduły transcievereów typu RFMxx firmy Hoperf. Akurat te modele najczęściej zapewniają komunikację za pomocą łącza SPI. Zatem mamy tu: bardzo niską cenę, niski pobór prądu i dobre zasięgi do 200–300 m w terenie otwartym. Gdzież więc tutaj są wady, czy w ogóle takie rozwiązanie ma wady? Wspomnę także o modułach typu CC100 firmy Chipcon. Tu znowu mamy podobną sytuację. O ile jednak ich konfiguracja programowa i przygotowanie do pracy, a także sama transmisja są nieporównywalnie prostsze od obsługi modułów kart WiFi, to jednak nadal pozostaje kilka kwestii, które powodują, że start w tej dziedzinie bez znajomości zasad komunikacji radiowej tak ogólnie bywa bardzo trudny. Przyznam, że wybierając pomiędzy akurat tymi dwoma rodzajami modułów, wolę ten drugi, z uwagi na prozaiczną rzecz, która nabiera nieraz niebagatelnie wielkiego znaczenia. Chodzi mianowicie o dokumentację producenta. Przypadek firmy Hoperef według mnie jest przykładem, jak nie należy postępować z klientem oraz jak nie należy przygotowywać dokumentacji technicznej pozwalającej na proste, łatwe i szybkie skorzystanie z modułów przez każdego. Nie dość, że oryginalne noty PDF, przynajmniej te dotyczące modułów RFM12/B, zawierały i chyba nadal zawierają błędy, to na stronach producenta trudno uzyskać jakiekolwiek wsparcie techniczne czy też inne wersje dokumentacji, przykładów implementacji itd. Nie chcę przez to powiedzieć, że są one bezwartościowe, ani w sensie ceny, ani przydatności, wręcz odwrotnie – czasem warto się pokusić o ich kupno, a następnie poświęcić kilka długich zimowych wieczorów na walkę z dokumentacją, własne próby, testy itd. Moduły tego typu charakteryzują się tym, że trzeba je prawidłowo zainicjalizować przed przystąpieniem do pracy. Przy czym producent daje bardzo wiele wariantów ustawień. Bywa to niestety dla mniej wtajemniczonych adeptów elektroniki cyfrowej jedną z największych bolączek. By wybrnąć z takiej sytuacji, przeszukują fora internetowe w celu znalezienia „gotowca", który mogliby zaadaptować bez wni-

kania w szczegóły działania. Kończy się to zwykle niepowodzeniem i zrażeniem się do tego typu źle udokumentowanych rozwiązań.

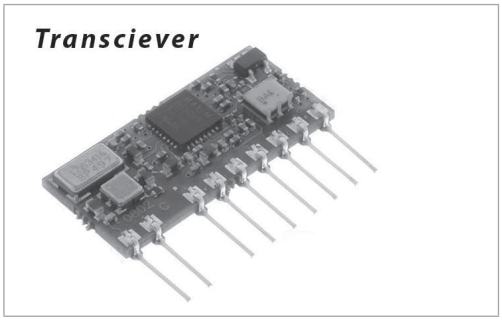
Wreszcie czwarty punkt. Mamy tu do czynienia z najprostszymi nadajnikami, odbiornikami i transcieverami, które nie wymagają od użytkownika wykonywania żadnych ustawień, mają ściśle zdefiniowane parametry pracy. Ich obsługa sprowadza się do podania na jeden pin wejściowy w nadajniku gotowego sygnału, który chcemy przesyłać, natomiast po stronie odbiornika do dyspozycji dostajemy pin, na którym ten sygnał się pojawia w niezmienionej postaci. Przynajmniej w założeniach. Mogłoby się wydawać, że to idealne rozwiązanie. Teoretycznie wystarczy bowiem podać stan wysoki na wejście nadajnika, aby na jego wyjściu uzyskać taki sam stan wysoki, i przeciwnie, na wejście podać stan niski, aby uzyskać także stan niski na wyjściu odbiornika. Niestety, w tym miejscu następuje zwykle ogromne rozczarowanie. Wprawdzie w przypadku podania stanu wysokiego w nadajniku otrzymamy taki sam stan na wyjściu odbiornika, to w przypadku podania stanu niskiego, na wyjściu odbiornika, jeśli podłączymy oscyloskop, zobaczymy ogromne ilości szumów, przebiegów, szpilek, po prostu jakieś różne częstotliwości nakładające się jakby na siebie. Jak zatem można cokolwiek przesłać taką drogą? Okazuje się, że najpierw trzeba zapoznać się z prostą specyfiką działania tego typu modułów, a wszystko stanie się jasne. Dla przykładu podam nazwy kilku kompletów nadajników/odbiorników oraz transcievera tego typu (rysunki 22–24).



Rysunek 22. Firma **Aurel**: nadajnik TX-FM-MID oraz odbiornik RX-4M50FM60SF



Rysunek 23. Firma **Telecontrolli**: nadajnik RTFQ2-433 MHz-R, odbiornik RRFQ1-433 MHz



Rysunek 24. Firma **Aurel**, transciever RTX-MID, wersje zasilania +3,3 V lub +5 V

Wybrałem te trzy typy nie dlatego, że są najlepsze spośród wszystkich możliwych dostępnych na rynku rozwiązań radiowych. Jak wspominałem – po prostu nie ma najlepszych, uniwersalnych, są tylko takie, które najlepiej nadają się do konkretnego rozwiązania, które realizujemy. Myślę, że omówienie komunikacji na trzech wymienionych przykładach pozwoli Ci już samodzielnie decydować się, jaki moduł wybrać spośród setek dostępnych na rynku. Chciałbym jak najszybciej przybliżyć ci podstawy komunikacji tą metodą, abyś mógł zastosować ją w praktyce.

Próbując zatem wybrać coś dla siebie, nie mając doświadczenia w tej dziedzinie, warto wiedzieć, na jakie parametry należy zwrócić uwagę. W tabeli 2 przedstawiam przykładowy moduł z parametrami pracy.

Tabela 2. Dane na temat modułu z noty PDF

Electrical Characteristics

Ta = 25°C unless otherwise specified

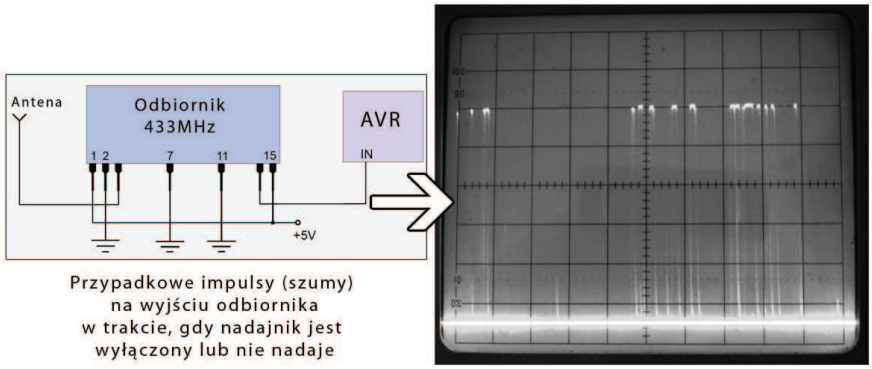
	CHARACTERISTICS	MIN	TYP	MAX	UNIT
V_{CC}	Supply Voltage (RTFQ2-XXX model)	2.2	3.3	4	VDC
	Supply Voltage (RTFQ2-XXX-R model)	2.5	5	12	VDC
I_S	Operating Supply Current		7	8	mA
F_W	Working Frequency		315/433.9/868.35		MHz
P_O	RF Output Power into 50Ω (Vcc = 3.3V)		+5 / +5 / +1		dBm
	Initial Frequency Accuracy	-50	0	+50	KHz
	FM Deviation		+/-30		KHz
	Harmonic Spurious Emission		-50		dBc
	Max Data Rate			9.6	Kbit/s
V_{IH}	Input High Voltage	1.5		5.5	V
	Power-Up Time to full RF			5	msec
T_{OP}	Operating Temperature Range	-25		+80	°C

Zakładając, że naszym celem będzie wykorzystanie takich modułów do prostych zadań w postaci pilotów radiowych lub transmisji danych z czujników rozmieszczonych w pomieszczeniu(-ach) do układu nadrzędnego, to jednym z najważniejszych parametrów będzie (szczególnie po stronie nadajnika) zakres napięcia zasilającego. Często bowiem w pilotach stosuje się w celu miniaturyzacji baterie o napięciu 3 V. Dlatego też spora część nadajników ma szerszy zakres napięcia lub występuje w dwóch wersjach,

tzn. zasilanych napięciem +3,3 V lub +5 V. To jest dosyć oczywiste i zapewne zdajesz sobie z tego dobrze sprawę. Myślę że podobnie podszedłbyś do parametru, jakim jest pobór prądu, szczególnie przez nadajnik zarówno w trakcie działania, jak i uśpienia. Dobierzesz sobie więc nadajnik bez problemów pod tym względem. Oczywistą rzeczą jest nie tylko to, że nadajnik i odbiornik muszą działać na tej samej częstotliwości, ale trzeba jeszcze skompletować (zakupić) je pod kątem rodzaju modulacji, jakiej używają obie strony. W bogatej ofercie wielu sklepów internetowych nie zawsze prezentowane moduły dobierane są w pary. Dlatego musisz zwrócić uwagę na rodzaj modulacji, czy jest typu AM, czy może FM. Ale to, co najbardziej będzie nas interesować, to przepustowość. Dzięki temu parametrowi będziemy w stanie oszacować prędkości, z jakimi będziemy mogli przesyłać dane. W tym miejscu zaznaczę, że tego typu rozwiązania nie są przeznaczone do transmisji dużej ilości danych. Najczęściej stosuje się je właśnie do różnego rodzaju pilotów, aby przesłać kody klawiszy lub krótkie informacje z danymi z czujników, albo do sterowania pracą innych urządzeń/modułów.

Wracając jednak do konkretnych modułów podanych wyżej, przyznam, że nie grzeszą one najniższymi cenami w porównaniu chociażby z transcieverami firmy Hoperf, które można nabyć już za kilkanaście złotych. Komplet nadajnik/odbiornik albo sama para transcieverów zwykle kosztuje kilkadziesiąt złotych (od 40 zł do 60 zł). Niemniej jednak, jak się przekonasz, prostota ich wykorzystania może spowodować, że taki wydatek warto ponieść. Jednocześnie uważam, że sposób komunikacji, który możemy ustandaryzować dokładnie tak jak w przypadku podczerwieni, zachowując przy tym spore zasięgi, umożliwia bardzo łatwą implementację tego typu nadajników/odbiorników w dowolnych urządzeniach.

Przejdźmy teraz do wyjaśnienia, dlaczego próby nadawania przez tego typu komplety zwykłych ramek RS232 kończą się zwykle niepowodzeniem albo w najlepszym przypadku znacznym zmniejszeniem zasięgu w trakcie przesyłania danych. Dlaczego tak się dzieje? Aby sobie na to pytanie odpowiedzieć, spójrzmy najpierw na wyjście takiego odbiornika, które podłączamy już do mikrokontrolera w trakcie, gdy nic nie nadajemy albo gdy nadajnik jest wyłączony (rysunek 25).



Rysunek 25. Szumy na wyjściu odbiornika

wanym wsadem odbiornika. Ich wysyłaniem zajmuje się omawiana wcześniej funkcja do przesyłania całej ramki z łańcuchem tekstowym.

Część przykładowego kodu programu ma te elementy, które wymieniłem tuż przed przedstawieniem powyższego kodu. Widać więc deklarację naszych dwóch funkcji do odbioru tekstów i wyświetlania ich na LCD oraz funkcję do prezentacji cyklicznie odbieranych ramek binarnych. A to, co najważniejsze, czyli korzystanie ze zdarzenia, sprowadza się jak zwykle do:

```
// **** pętla nieskończona
   while(1) {
      MAN_EVENT(); // **** ZDARZENIE OBSŁUGI ODBIORU
      // dalsza część własnego programu….
   }
```

Mam nadzieję, że podobnie jak mnie, korzystanie z tego typu sposobów tworzenia własnych bibliotek – choć na pierwszy rzut oka mogą się wydawać zbyt skomplikowane – przypadnie ci do gustu. A szczególną uwagę przykuje prostota posługiwania się tymi bibliotekami. Zachęcam więc do przeprowadzenia własnych ćwiczeń i modyfikacji. Na płycie DVD postaram się umieścić króciutki materiał wideo prezentujący działanie opisanego tutaj kodu w nadajniku i odbiorniku podczas prawdziwej, fizycznej komunikacji.

2.1. Transcievery – komunikacja

Aby trzymać się konwencji stopniowego i płynnego przechodzenia pomiędzy omawianymi tematami, poświęcę teraz kilka słów transcieverom firmy Aurel, o których wspomniałem na początku tego rozdziału, prezentując jeden model w tabelce. Niewątpliwą zaletą tych modułów jest przede wszystkim możliwość uzyskania dwustronnej komunikacji, ale także niska cena. Być może nadal nie aż tak niska jak transcieverów firmy Hoperf (typu RFM12B), jednak prostota ich stanowi tu niewątpliwie przewagę. Transcievery oparte są na układzie scalonym firmy Maxim 7030AH. W zasadzie aplikacja układu jest na tyle prosta, że przy odrobinie wysiłku można byłoby we własnym zakresie budować sobie tego typu transcieverki. W związku z powyższym wszystkie informacje zawarte w nocie aplikacyjnej transcieverów RTX-MID odpowiadają w większości informacjom zawartym w nocie aplikacyjnej samego chipu. Najważniejsze z naszego, czysto użytkowego punktu widzenia są informacje o podstawowych wyprowadzeniach modułu, ich funkcjach, a także o specyficznym sposobie przełączania modułu bądź to w tryb nadawania, bądź odbioru. Występują tu bowiem proste zależności czasowe, które wystarczy zachować w programie podczas sterowania pinami wejściowymi modułu, takimi jak TX/RX oraz ENABLE. W przypadku tych modułów jest jedna ciekawostka techniczna, chodzi o możliwość blokady szumów na wyjściu odbiornika w trakcie, gdy nic nie jest nadawane i odbierane. Mechanizm ten to opcja *Squelch* (blokada szumów), działająca podobnie jak ta, z której korzystają krótkofalowcy. W krótkofalarstwie ma ona za zadanie uchronić uszy przed obróbką zbędnego szumu, natomiast tutaj ma uchronić mikrokontroler przed nadmierną i niepotrzebną pracą, której celem jest analizowanie szumów i próba doszukania się w nich właściwych ramek da-

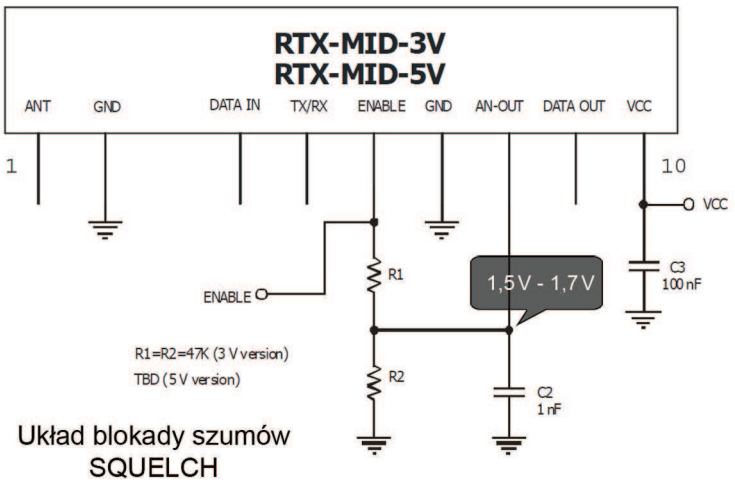
nych. O takim zjawisku wspominałem już w poprzednim podrozdziale, tzn. o szumie białym, który powoduje, że przerwanie ICP mikrokontrolera tak naprawdę reaguje na każdy impuls, próbując dekodować ramkę danych. Niestety, zwykle po kilku pierwszych sygnałach ramka zostaje odrzucana i proces rozpoczyna się od nowa przez cały czas, gdy nadajnik nie nadaje żadnych danych lub samej nośnej.

Z jakiegoś powodu blokada nie została wbudowana do modułu na stałe. Producent jedynie daje nam możliwość skorzystania z niej i ostrzega o skutkach ubocznych. Skutkiem ubocznym wprowadzania układu Squelch jest zmniejszenie czułości odbiornika o 3 dB i nieco opóźniony czas reakcji na nadlatujące ramki. I dlatego w nocie PDF znajdziemy informację, że jeśli nie zależy nam na wysokiej jakości toru nadawczo-odbiorczego, a co za tym idzie – na jak największym zasięgu oraz czułości i rozdzielczości, to możemy skorzystać z proponowanego układu blokady szumów. Jak widzisz, coś za coś. W praktyce może się rzeczywiście zdarzyć i tak, że transmisja będzie potrzebna w ramach na przykład jednego dużego pomieszczenia, więc nie będzie ci zależało na dużych zasięgach Może się za to okazać, że ważny zacznie być czas, który jest marnowany w przerwaniu na analizę szumu białego, wtedy to wykorzystasz układ Squelch. W innych przypadkach pominiesz go. A tak naprawdę nasze funkcje obsługi dekodowania ramek są skonstruowane na tyle optymalnie, że nie tracimy zbyt dużo czasu w przerwaniu ICP podczas prób dekodowania szumu i odrzucania ramek złych lub uszkodzonych z punktu widzenia programu. Oczywiście należy mieć na uwadze, że pojęcie „nie tracimy zbyt dużo czasu" jest względne i tak naprawdę w dużym stopniu zależy od indywidualnego projektu. Trudno to ocenić, jeśli się dopiero zaczyna przygodę z takimi zagadnieniami, dlatego mając już jakąś tam praktykę w tym zakresie, podpowiem ci bardzo ogólnie, że w zdecydowanej większości swoich pierwszych projektów określenie to będzie prawdziwe i nie narazisz procesora na duże obciążenie, realizując swoje projekty z wplecioną obsługą takich modułów czy samych odbiorników.

Cały układ składa się zaledwie z dwóch rezystorów i jednego kondensatora. Łączy się to wszystko pomiędzy wejściem Enable oraz wyjściem analogowym z modułu (w uproszczeniu mówiąc) według schematu podanego na rysunku 35.

Zasada działania polega na wprowadzeniu dodatkowego filtru dolnoprzepustowego na wyjściu analogowym, z którego dalej czerpie sygnał tzw. Data Slicer, czyli układ odpowiedzialny za formowanie już właściwego sygnału cyfrowego na wyjściu z samego modułu.

Działanie układu polega na podaniu napięcia około 1,6 V na wyjście ANALOG-OUT, które uzyskujemy z dzielnika rezystorowego R1 oraz R2 w trakcie, gdy odbiornik jest aktywny. Na wejściu Enable panuje wtedy stan wysoki. Na schemacie podane są wartości rezystorów dla napięcia zasilania 3,3 V, a dla napięcia zasilania 5 V widać tylko skrót TBD. Jest on dosyć tajemniczy, tym bardziej że w całej nocie PDF nie znajdziemy w tej sprawie ani słowa wyjaśnienia. To skrót angielskiego terminu *To Be Defined*, co znaczy „do ustalenia". Zatem tak naprawę należy samemu sobie ustalić, krótko mówiąc, dobrać sobie dzielnik tak, aby uzyskać na wyjściu napięcie w podanych wyżej granicach przy założeniu, że stan wysoki będzie odpowiadał napięciu zasilania VCC = +5V. Nie jest to chyba żaden problem.

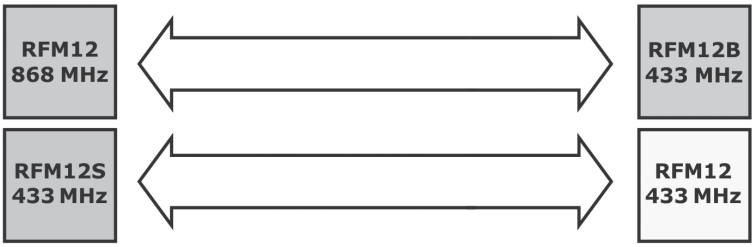


Rysunek 35. Transciever Aurel

Ja testowałem ogranicznik szumów z wartościami rezystorów: R1 = 33 K oraz R2 = 15 K. A także z innymi wartościami i nieco wyższym napięciem niż 1,7 V. Oczywiście efekt przy napięciu 1,5 V nie był zachwycający, gdy obserwowałem biały szum na oscyloskopie. Ale już przy wartości 2,2 V znikał całkowicie, przy czym zmieniała się w ogóle faza sygnału w stanie spoczynku. Natomiast poniżej 1,9 V faza była taka sama jak przed włączeniem układu blokady szumów. W stanie spoczynku nadal na wyjściu był stan wysoki. Dobór tego napięcia ma, jak widać, wpływ na dwie rzeczy: zbyt niska wartość doprowadzi do zbyt niskiej eliminacji szumów, nadal będą występowały, z kolei zbyt wysoka wartość odwróci fazę sygnału, ale tylko w stanie spoczynku. Czy odwrócenie tej fazy może mieć jakieś znaczenie dla nas? Jeśli będziemy korzystali z kodowania Manchester, to nie ma to żadnego znaczenia. Jeśli zaś chcielibyśmy skorzystać z transmisji poprzez UART/RS232, to jak się domyślasz, może to już mieć spore znaczenie. Wspominam o transmisji UART, ponieważ producent, czyli firma Aurel, na początku noty PDF zapewnia nas, że jest to możliwe i nie ma powodów do obaw o pojawianie się zakłóceń w trakcie nadawania bajtów, w których występuje sporo bitów o wartości 0 lub 1, następujących jeden po drugim. A to wszystko przy prędkości 9600 bps. Niestety, w dalszej części noty, przy opisie modułu *Data Slicer*, spotkamy informację: „istnieje możliwość bezpośredniej transmisji RS232, jednakże należy zadbać o to, aby w ramach bajtu pojawiła się przynajmniej jeden raz różnica stanów, czyli przynajmniej jeden bit o wartości = 1, jeśli pozostałe są = 0, i odwrotnie: przynajmniej jeden bit o wartości = 0, jeśli pozostałe są = 1".

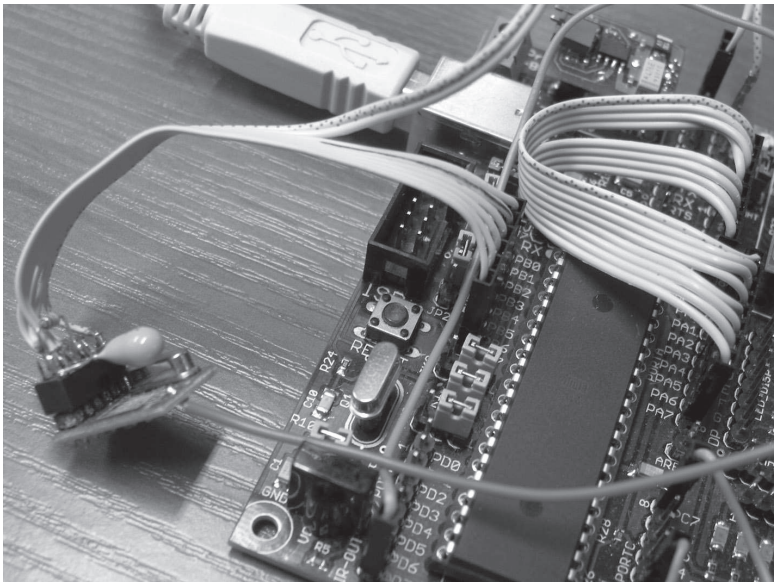
Jak widzisz, takie zastrzeżenie nieco ogranicza nam możliwości czystej transmisji binarnej. W praktyce nie jest tak źle, gdyż nawet przy włączonym układzie *Squelch* udawało się bez problemu transmitować dane za pomocą RS232 w obie strony przy zachowaniu rygorów czasowych związanych z ustalaniem kierunku nadawania. Nie były to jednak oszałamiające odległości/zasięgi w porównaniu z tymi, które pozwalają wy-

nej przeszkody. Chciałbym raz na zawsze wykluczyć spekulacje, że niektóre typy modułów RFM12 mogą ze sobą nie współpracować. Otóż z pełną odpowiedzialnością potwierdzam (będziesz to mógł sam empirycznie stwierdzić), że można nawiązać połączenie nawet pomiędzy modułem zakupionym na częstotliwość 433 MHz a drugim zakupionym na 868 MHz! Tak, każdy z nich można zaprogramować na dowolną częstotliwość pracy, ale z uwagi na dobór elementów wyjściowych RF, jak można się domyślić, zasięg pomiędzy nimi spadnie drastycznie. Nigdy jednak to nie spowoduje, że nie uda ci się uruchomić komunikacji między nimi.



Rysunek 41. Przykład możliwości komunikacji pomiędzy różnymi typami modułów

Na rysunku 41 pokazuję więc bardzo schematycznie różne kombinacje połączeń, jakie można zestawić między różnymi wersjami modułów RFM12. Proszę tylko pamiętać, że jeśli pokazana jest para 868 MHz oraz 433 MHz, to znaczy, że obydwa zostaną zaprogramowane na jedną z tych dwóch częstotliwości.



Zdjęcie 4. Moduł RFM podłączony do zestawu uruchomieniowego ATB za pomocą kabelka

3. Odtwarzanie plików dźwiękowych WAV z kart pamięci SD/MMC

Ponieważ w poprzednim rozdziale mieliśmy już do czynienia z dźwiękiem, dlatego kolejne zagadnienie, które omówię, także będzie dotyczyło dźwięku. Jak zapowiedziałem w tytule rozdziału, zajmiemy się odtwarzaniem plików dźwiękowych typu *.wav zapisanych na dowolnej karcie pamięci typu SD/MMC. Znasz już bowiem zagadnienia związane z generowaniem dźwięku za pomocą mikrokontrolera AVR. Poza tym poruszymy takie tematy, jak wykorzystanie znanych już z pierwszej części książki bibliotek do obsługi kart pamięci PetitFat oraz format nagłówka i danych plików dźwiękowych `wav`. Przy okazji podam informację dość istotną dla wszystkich użytkowników bibliotek PetitFat, ale także FatFS – obsługują one bez najmniejszych problemów najnowsze karty SDHC o pojemnościach większych niż 2 GB. Ja do przeprowadzenia tego ćwiczenia wykorzystuję kartę SDHC 8GB (szósta klasa prędkości). Zdecydowanie polecam korzystanie z kart pamięci posiadających właśnie minimalnie szóstą klasę prędkości.

Jako projekt wyjściowy do realizacji naszego zadania posłużył projekt ze wspomnianej już strony elm-chan.org. Jego autor przygotował oprogramowanie dla niewielkich mikrokontrolerów AVR, takich jak np. ATtiny85. W efekcie udało się maksymalnie zminiaturyzować tego typu odtwarzacz plików `wav`. Niestety, autor został niejako zmuszony do napisania części kodu w języku asembler, a to dlatego, że postawił sobie za cel odtwarzanie nawet plików `wav` z maksymalną częstotliwością próbkowania nawet 48 kHz. Z tego powodu wstawki asemblerowe dotyczą aż dwóch aspektów działania programu.

Podstawowym założeniem autora było zorganizowanie bufora cyklicznego, który posłużył do strumieniowego pobierania danych za pomocą funkcji bibliotek PetitFat. Efektem tego była konieczność dość znacznej modyfikacji pliku `mmc.c`, odpowiedzialnego za komunikację warstwy sprzętowej tej biblioteki z kartą pamięci. Dlatego też jej funkcja `pf_read()` odwołuje się wewnątrz do wstawki asemblerowej, która została umieszczona w jednym wspólnym pliku z pozostałymi wstawkami asemblerowymi. Tymczasem te pozostałe wstawki nie dotyczą już bezpośrednio bibliotek, lecz funkcji programu głównego. I tak kolejna ważna wstawka to pełna obsługa przerwania timera sprzętowego odpowiedzialnego za samplowanie. Nie wspomnę już o pozostałych drobniejszych wstawkach asm odpowiedzialnych za komunikację SPI.

Zaletą tych niewielkich mikrokontrolerów jest również to, iż można skorzystać z taktowania 16 MHz na potrzeby procesora, a także z jeszcze większego, bo aż 64 MHz, na

potrzeby modułów sprzętowych, jak np. Timer1. Takie podejście umożliwia z jednej strony wraz ze wstawkami asemblerowymi dużą prędkość działania programu i przerwań, z drugiej zaś daje do dyspozycji bardzo wysoką częstotliwość na potrzeby generowanego sygnału PWM.

Sam już wiesz z poprzedniego rozdziału, że im większa w tym przypadku częstotliwość PWM, tym lepsze efekty końcowe, jeśli chodzi o jakość odtwarzanego dźwięku. Skoro zatem taki projekt z powodzeniem odtwarza wszystkie dostępne odmiany plików `wav`, to dlaczego miałbym tutaj powielać taki opis? Niestety, w pewnych aspektach projekt ten ma wady. W tym rzecz. Bodajże największą wadą dla większości osób, które chcą w oparciu o ten projekt stworzyć własne tego typu urządzenie, jest stopień skomplikowania i trudność „przeportowania" (przeniesienia) całości na inne rodzaje mikrokontrolerów. I nie chodzi tylko o to, że same mikrokontrolery ATtiny85 czy ATtiny861 są trudno dostępne, bo nie jest w tym zakresie tak źle. Po prostu autor z uwagi na stopień trudności oprogramowania całego zagadnienia zmuszony był maksymalnie optymalizować cały kod, tak aby sprostać ciężkim wymaganiom.

Kolejną wadą tego rozwiązania, jeśli wykonamy tylko jego kopię bez modyfikacji kodu źródłowego, jest to, że nie uda nam się odtwarzać plików dźwiękowych `wav` z bardzo niskimi częstotliwościami samplowania, np. 6 kHz. Zastanowisz się może w tym miejscu, jaka to wada? I kto będzie korzystał z tak niskiej częstotliwości, skoro cały projekt został stworzony do odtwarzania dźwięków jak najwyższej jakości? Weź pod uwagę, że pliki `wav` nie są skompresowane, zatem te przechowujące próbki dźwiękowe najwyższej jakości zajmują dziesiątki megabajtów. Może się kiedyś okazać, że w jakimś własnym projekcie będziesz potrzebował np. tylko kilku prostych dźwięków kosztem nawet jakości, gdy najwyższym priorytetem będzie ich rozmiar. Warto wiedzieć, że można je umieścić albo ich część bez nagłówka wprost w pamięci Flash samego mikrokontrolera, albo w pamięci EEPROM/FRAM itp. Wtedy im niższa częstotliwość próbkowania i dźwięk monofoniczny, tym mniejszy rozmiar pliku, już rzędu kilobajtów, a nie megabajtów. Wszystko zależy od potrzeb.

Ja postanowiłem nieco inaczej podejść do tematu. Weźmy pod uwagę podstawowy fakt, że autor omawianego projektu wykonuje konwersję dźwięków zapisanych z rozdzielczością 16-bitową na rozdzielczość 8-bitową. Zatem już chociażby z tego powodu, jak się domyślasz, traci się nieco na jakości odtwarzanego dźwięku. Nie stanowi to jednak dużego problemu w tego typu projektach/rozwiązaniach, gdyż nie mają one na celu zastąpienia popularnych odtwarzaczy MP3, lecz jedynie umożliwienie realizacji wielu różnych ciekawych i nietypowych własnych projektów/konstrukcji, w których odtwarzanie wcześniej nagranych dźwięków byłoby bardzo pożądane. Wystarczy pomyśleć o różnego rodzaju zabawkach i urządzeniach, które mają generować proste komunikaty głosowe albo nietypowe sygnały dźwiękowe, a nawet prostych odtwarzaczach muzyki do zegarów, centralek telefonicznych itp.

W związku z powyższym miło byłoby mieć możliwość wykonania takiego projektu w pełni we własnym zakresie i dokładnie według własnych założeń. Szczególnie gdy sami możemy zdecydować się np. na obniżenie jakości dźwięku kosztem miejsca w pamięci. Poza tym odtwarzanie niższych częstotliwości próbkowania będzie umożliwia-

ło zastosowanie niższego taktowania mikrokontrolera, co z kolei pozytywnie może się przełożyć na zastosowania bateryjne i obniżenie poboru prądu w dłuższym czasie.

Postawiłem sobie zatem pytanie, czy próba zminiaturyzowania takiego urządzenia musi się koniecznie wiązać z wykorzystaniem tak małych mikrokontrolerów jak 8-nóżkowe ATtiny? Myślę, że zdecydowanie nie. Można bowiem skorzystać z pięknej i niesamowicie przydatnej na co dzień w każdym warsztacie elektronika, linii mikrokontrolerów: ATmega88 – ATmega168 – ATmega328. Gdy wybierzemy obudowę SMD TQFP, okaże się, że wraz z gniazdem na karty pamięci Micro SD uzyskamy bez najmniejszych problemów równie niewielkie rozmiary, jak prototyp urządzenia ze strony elm-chan.org. Oczywiście teraz do celów testowych wykorzystam zestaw uruchomieniowy ATB z mikrokontrolerem ATmega32 na pokładzie. Dlatego też z uwagi na większą liczbę wyprowadzeń postaramy się przygotować i oprogramować dodatkowe gadżety. Zaprzęgniemy zatem do pracy wyświetlacz LCD, który będzie nam prezentował nazwę odtwarzanego pliku dźwiękowego, stan aktywności: PLAY, NEXT, INSERT CARD, czas trwania utworu dokładnie w sekundach i minutach. Poza tym przygotujemy prosty wskaźnik wysterowania na ośmiu diodach LED. Dodatkowo układ będzie obsługiwał trzy przyciski. Jeden do zmiany utworu na następny, a dwa kolejne do zmiany prostych efektów dźwiękowych (obróbki dźwięku na żywo). Myślę, że ważnym rozwinięciem projektu będzie także możliwość wykrywania obecności karty pamięci w czytniku, dzięki czemu przygotujemy układ, który nie będzie wymagał ciągłego resetowania po przypadkowym wyjęciu i włożeniu karty. Zestaw uruchomieniowy ATB już od wersji 1.02 ma jeszcze jedną ciekawą możliwość – pozwala kontrolować w sposób całkowicie programowy zasilanie karty SD. Możemy zatem jeden pin mikrokontrolera przeznaczyć do tego celu. Czy rozwiązanie takie daje jakieś szczególne możliwości? Owszem, daje. Sam zauważyłeś zapewne, że czasem karta pamięci w wyniku np. błędnego działania programu lub w innych nieprzewidzianych okolicznościach potrafi się po prostu zawiesić. w takiej sytuacji pomaga zwykle dopiero wyjęcie jej i ponowne włożenie do czytnika. Oznacza to tak naprawdę wyłączenie i włączenie jej zasilania (zupełny reset). W efekcie jeśli zastosujemy taki mechanizm jak ten przedstawiony na schemacie zestawu uruchomieniowego ATB we własnych rozwiązaniach, możemy tworzyć jeszcze bardziej niezawodne urządzenia pod tym względem. Schematy zestawów ATB dostępne są także na płycie DVD dołączonej do książki, warto więc wesprzeć się nieraz sprawdzonymi rozwiązaniami. Dodam jeszcze, że nawet standardowe biblioteki FatFS do obsługi kart pamięci w przykładowych rozwiązaniach omówionych w mojej pierwszej książce korzystają z tych dwóch mechanizmów. Mowa tu o wykorzystaniu jednej linii mikrokontrolera do detekcji obecności karty w slocie, ale także drugiej linii do wyłączania i włączania zasilania całej karty. Z tym, że do czasu, kiedy pojawiła się wersja 1.02 płyty PCB, zestawy ATB nie były wyposażone sprzętowo w takie możliwości.

Przejdę teraz do najważniejszych założeń programistycznych odnośnie do tego projektu. Chciałbym uzyskać nie tylko wszystkie możliwości, jakie oferowane są w projekcie autora z elm-chan.org, ale także poszerzyć je o dodatkowe, opisane wyżej. Najważniejsze jest jednak to, że poszukiwałem drogi, aby tak dobrać sprzęt i na tyle zoptymalizować kod programu w C, aby nie były potrzebne nawet najmniejsze wstaw-

ki w asemblerze. Mam nadzieję, że to będzie istotna informacja dla wszystkich, którzy żywo zainteresowani są takim rozwiązaniem. Szczególnie że dzięki temu każdy będzie miał możliwość przeportowania takiego kodu programu nie tylko na dowolny procesor z rodziny AVR8, ale także na każdą inną platformę, obojętnie czy to będą także 8-bitowe mikrokontrolery PIC, czy może 32-bitowe mikrokontrolery ARM/STM.

Wspomniałem już, że będzie konieczne odpowiednie dobranie/zaprojektowanie części sprzętowej. Dlatego będziemy musieli zaopatrzyć nasz mikrokontroler w zewnętrzny rezonator kwarcowy o dość wysokiej częstotliwości, o ile będziemy chcieli skorzystać z najlepszych efektów. W przypadku mikrokontrolerów nie stanowi żadnego problemu zastosowanie maksymalnych częstotliwości taktowania, ponieważ będą one pracowały z pełnym powodzeniem, jeśli zechcemy je nawet delikatnie przetaktować. Przetaktowanie nie będzie jednak konieczne w każdym przypadku. Jeśli zastosujemy taką wersję mikrokontrolera ATmega, która standardowo może być taktowana maksymalną częstotliwością 20 MHz, to przetaktowanie będzie niepotrzebne. Jeśli zaś posiadasz np. tak jak ja do obecnych ćwiczeń mikrokontroler ATmega32 z maksymalną dopuszczalną częstotliwością taktowania 16 MHz, to spokojnie i bez obaw możesz użyć rezonatora nie tylko 20 MHz. Ja testowałem z pełnym powodzeniem taktowanie na nim: 24, 25, a nawet 27 MHz. W żadnym wypadku mikrokontroler nie odmówił posłuszeństwa, jeśli chodzi o normalną pracę. Być może mogłyby się pojawić problemy z zapisem do pamięci EEPROM, ale nie testowałem go pod tym względem. Mój procesor, i to nie jeden egzemplarz, pracował przez wiele dni i nocy z kwarcem 27 MHz.

Nie zrozum mnie jednak źle. Nie twierdzę, że należy tak projektować profesjonalne i komercyjne układy. Zawsze rozsądniej jest trzymać się jednak w tym zakresie zaleceń producenta. Natomiast jeśli chodzi o własne amatorskie rozwiązania można sobie na to pozwolić bez żadnego wahania.

Zresztą zastanówmy się, kiedy będzie nam potrzebne maksymalne taktowanie? Otóż tylko w przypadku gdy zechcemy w trakcie odtwarzania korzystać dodatkowo np. z wyświetlacza LCD, wskaźnika wysterowania lub innych gadżetów przy jednoczesnym odtwarzaniu plików dźwiękowych najwyższej jakości DAT 48 kHz! Myślę jednak, że rzadko kiedy do własnych prostych zastosowań będziesz chciał pokusić się o te wszystkie dodatki. Mogą się one okazać wręcz niepotrzebne. Wyłączając je zatem z kodu programu, możesz śmiało zmniejszać taktowanie mikrokontrolera już np. do 16 MHz przy próbkowaniu (samplowaniu 22,05 kHz!). Jeśli zaś uznasz, że wystarczy ci nawet niższe próbkowanie, np. na poziomie 11 kHz albo jeszcze niższe, 8 kHz lub 6 kHz, to możesz pokusić się o zmianę taktowania do wartości np. 12 MHz.

W tym miejscu chciałbym zwrócić uwagę na pewną ważną rzecz. Jeśli już zechcesz robić próby z oryginalnym kodem, a nie masz zestawu uruchomieniowego ATB, to poza zmianami w programie dotyczącymi dodatkowych linii sterowania karty SD musisz zwrócić uwagę na jeszcze jedną sprawę – otóż większość zestawów konkurencyjnych (w zasadzie prawie wszystkie, i nie wiem dlaczego tak jest), jeśli już ma wyświetlacz LCD na pokładzie, ma także podłączony (projektanci na siłę uszczęśliwiają klientów) pin RW tegoż wyświetlacza do GND. Co powoduje, że cała obsługa wyświetlacza

i wypróbować z kodu zawartego na płycie DVD. Dodam, że załączony kod nie zawiera tylko gołych bibliotek, ale również główną funkcję `main()`, czyli jakiś przykładowy program, który po kolei wyświetla coś nam na ekranie, przy okazji korzystając z różnych funkcji bibliotecznych, tak abyś miał wygodne przykłady ich praktycznego użytkowania.

4.1.1.1. Nieudokumentowane opcje sterownika SSD1963

Nadszedł moment, w którym mogę przejść do opisu i prezentacji pewnej nie do końca udokumentowanej opcji sterownika SSD1963. Wspominałem o tym na początku rozdziału, a chodzi tu o możliwość wykorzystania pamięci RAM sterownika na buforowanie dodatkowych obrazów, które mogą być wczytane w tle, a samo ich wyświetlenie może odbyć się po prostu błyskawicznie. Związane jest to z nietypowym zastosowaniem komend przeznaczonych, zgodnie z notą katalogową PDF, do przesuwania zawartości ekranu, tzw. scrolowanie. Ale po kolei. Zacznę od tego, że przypomnę oczywistą rzecz, którą można wyczytać w dokumentacji, że sterownik SSD1963 pozwala na obsługę ekranów LCD o różnej przekątnej, a co za tym idzie różnej rozdzielczości. Związane jest to bezpośrednio z ilością dostępnej pamięci RAM do buforowania obrazu. Maksymalna dostępna rozdzielczość to aż 864 x 480 pikseli przy 24-bitowej rozdzielczości kolorów. Łatwo sobie przeliczyć nawet, że w związku z tym dostępna pamięć RAM posiada: 864 x 480 x 3 = 1 244 160 bajtów (1,215 MB). Zatem przy obsłudze tak dużego ekranu LCD będzie ona wykorzystana w całości i na pewno nie uda się zastosować pewnego triku o którym chcę tu napisać. Wziąwszy pod uwagę to, że do naszych celów użyjemy dwóch rodzajów wyświetlaczy LCD TFT, o przekątnych ekranu 3,5" oraz 4,3", które mają rozdzielczość 320 x 240 pikseli oraz 480 x 272 pikseli, okaże się, że na pewno cała pamięć RAM sterownika nie będzie w normalnych warunkach wykorzystywana. Przecież ten o przekątnej 3,5" zużyje jej tylko 320 x 240 x 3 = 230 400 bajtów (225 kB), natomiast ten o przekątnej 4,3" – tylko 480 x 272 x 3 = 391 680 bajtów (382,5 kB).

Zgadza się, tak byłoby gdyby nie fakt, że my nie pozwolimy, by sterownik się za bardzo nudził, a spora część pamięci bufora leżała odłogiem. Postaramy się ją zagospodarować prawie w całości i dzięki temu uzyskać bardzo ciekawe efekty, które będą szczególnie przydatne przy zastosowaniu tak małych mikrokontrolerów jak 8-bitowe AVR. Podejrzewam, że po zastosowaniu tych nieudokumentowanych funkcji zaskoczysz niejednego znajomego efektami, które w normalnych warunkach można byłoby uzyskać programowo dopiero przy zastosowaniu dużo większych mikrokontrolerów, np. ARM, i to taktowanych dużo wyższymi częstotliwościami.

Wróćmy do zagadnienia pamięci. Skoro całość pamięci RAM jest równa 1,215 MB, to oznacza, że moglibyśmy zmieścić w takim buforze aż 5,4 obrazu z wyświetlacza o przekątnej 3,5", a także aż 3,1 obrazu z wyświetlacza o przekątnej 4,3". Naturalnie części ułamkowe nas w tym przypadku nie interesują, ale pewnie już wiesz, o co chodzi. Można bowiem doprowadzić do takiej sytuacji, że jeśli np. tworzymy przykładową ramkę zdjęciową, a zależy nam, aby przełączanie pomiędzy wyświetlanymi obraz-