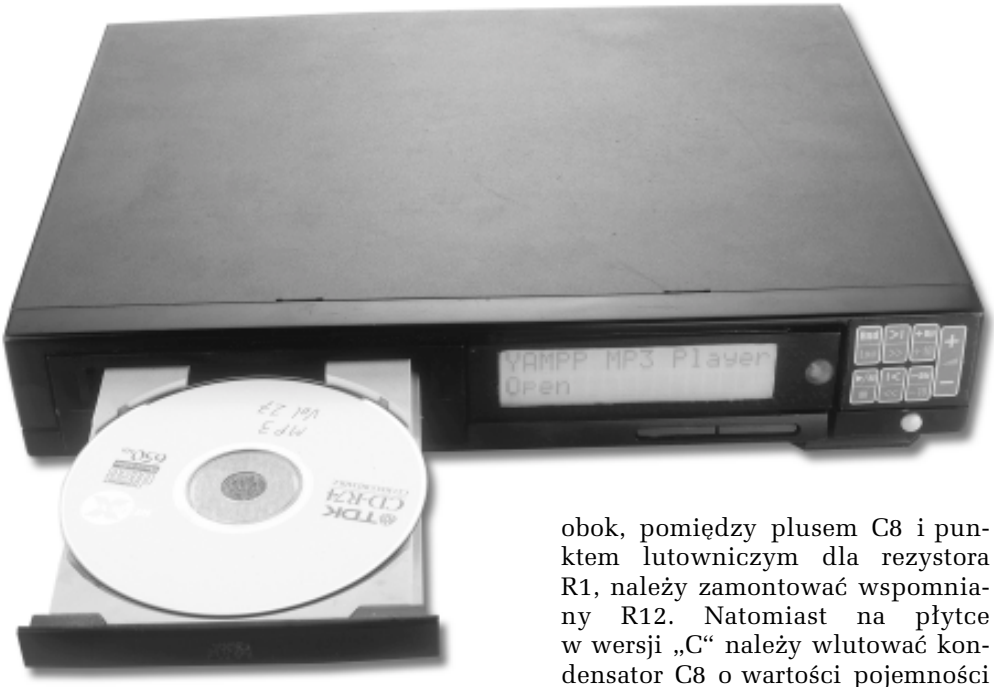
Yampp 3
Sprzętowy odtwarzacz MP3, część 2



W drugiej części artykułu opisujemy montaż i uruchomienie odtwarzacza yampp. Dzięki specjalnemu oprogramowaniu testowemu nie jest to zadanie bardzo trudne, wymaga jednak pewnej wiedzy. Skorzystamy zatem z porad i opisu przygotowanego przez twórcę oprogramowania yamppa - Romualda Białego.
Rekomendacje: *sprzęt dla fanów „empetrójki" lubiących znać urządzenia od środka.*

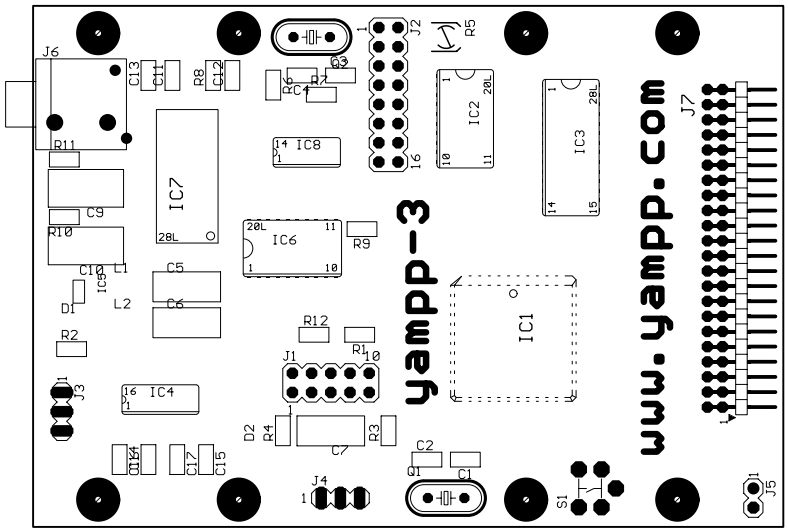
Montaż i uruchomienie

Ze względu na zastosowanie elementów montowanych powierzchniowo, montaż układu należy przeprowadzić szczególnie starannie. Najlepiej jest zastosować podaną poniżej kolejność montażu, co ułatwi uruchomienie odtwarzacza oraz usunięcie ewentualnych błędów. Sposób ten nieco odbiega od tego, który został zaproponowany przez Jespera Hansena, ale jest prostszy i został sprawdzony w praktyce.

Niestety, Jesper Hansen nie ustrzegł się kilku drobnych błędów. Po pierwsze, powstało małe nieporozumienie dotyczące sygnału zerującego mikroprocesor. Na schemacie i płytce drukowanej w wersji „B" było miejsce na kondensator C8 włączony pomiędzy masę układu a wejście zerujące. W wersji „C" zamiast kondensatora C8 zastosowano rezystor R12 włączony pomiędzy VCC a wejście sygnału zerującego. Praktyka dowodzi, iż najlepiej jest zastosować oba wspomniane elementy, więc na płytce w wersji „B" należy zamontować kondensator C8 na swoim miejscu i zaraz obok, pomiędzy plusem C8 i punktem lutowniczym dla rezystora R1, należy zamontować wspomniany R12. Natomiast na płytce w wersji „C" należy wlutować kondensator C8 o wartości pojemności ok. 2,2 µF pomiędzy masę i wejście sygnału zerującego (minus do masy). Prawdę mówiąc, najlepszym wyjściem jest jednak zastosowanie specjalizowanego układu scalonego (np. trójkońcówkowy DS1813 lub DS1233) zamiast wspomnianych elementów C8 i R12, ale nie jest to konieczne.

Kolejnym błędem jest brak kondensatorów blokujących zasilanie +5 V, które autor projektu po prostu zapomniał umieścić na płytce. Przy zasilaniu *yamppa* z niezaszumionego zasilacza nie stwarza to problemów, lecz przy zasilaniu z zasilacza impulsowego może być przyczyną zakłóceń dźwięku lub samoistnego zerowania się *yamppa*. Dobrze jest więc zamontować przynajmniej dwa kondensatory ceramiczne 100 nF i jeden tantalowy 22 µF/10V pomiędzy VCC i masę. Jeden z nich najlepiej przylutować bezpośrednio do wyprowadzeń mikrokontrolera.

Pozostawienie „w powietrzu" nieużywanych wejść bramek układu IC8 nie powinno mieć miejsca w przypadku zastosowania układów wykonanych w technologii CMOS. Najprostszym rozwiąza-



Rys. 6. Rozmieszczenie elementów na płytce drukowanej

niem tego problemu jest połączenie kroplami cyny odpowiednich nóżek IC8. Łączymy więc ze sobą nóżki 12, 13 i 14, a następnie 9, 10 i 11 oraz 4 i 5, po czym krótkim odcinkiem przewodu łączymy je do masy, czyli do wyprowadzenia 7 IC8.

Rozmieszczenie elementów na płytce drukowanej *yamppa* pokazano na **rys. 6**. Na początku należy zamontować procesor IC1, stabilizator IC5, diody LED D1 i D2, dławiki L1 i L2, kondensatory C1, C2, C5, C6, C8, rezystory R1, R2, R4, R12, przycisk resetu S1, kwarc Q1 oraz złącza J1 i J5. Można też zamontować pamięć IC3 oraz zatrzask IC2. Po zamontowaniu elementów podłączamy zasilanie 5 V do złącza J5. Należy użyć zasilacza z ograniczeniem prądowym ustawionym na ok. 100 mA. Powinna się zaświecić dioda D1. Jeśli się nie świeci, to oznacza, że została zamontowana odwrotnie lub jest jakieś zwarcie w układzie. Następnie należy zmierzyć napięcie za stabilizatorem IC5 (na dodatnich końcówkach kondensatorów C5 lub C6). Powinno ono wynosić 3 lub 3,3 V, w zależności od wersji zastosowanego stabilizatora. Następnie należy zaprogramować mikrokontroler programem testowym. W tym celu podłączamy interfejs programujący do portu drukarkowego komputera, 10-stykową wtyczkę interfejsu wkładamy do złącza J1, włączamy zasilanie *yamppa* i uruchamiamy program ładujący na komputerze. Jeżeli korzystamy z programu *Yapp* napisanego przez autora projektu (opisaliśmy go w EP7/2002), należy nacisnąć klawisz identyfikacji procesora. W okienku obok powinna pojawić się informacja o wykryciu procesora AT90S8515. Jeśli wszystko przebiega poprawnie, to otwieramy nowy projekt, zaznaczamy okno z zawartością pamięci Flash i wczytujemy program testowy (*File>Load File>yampp_3_test.rom*), który jest dostępny zarówno w Internecie (adresy podajemy na końcu artykułu), jak i na płycie CD-EP10/2002B. Teraz wystarczy kliknąć na ikonę z wykrzyknikiem, czyli *Autoprogram*, i po chwili program testowy będzie załadowany do procesora. W przypadku korzystania z innego programu ładującego, procedurę ładowania pliku *yampp_3_test.rom* przeprowadzamy zgodnie z jego instrukcją obsługi.

Jeżeli programator nie wykryje procesora, należy sprawdzić poprawność montażu *yamppa* i interfejsu programującego oraz pracę generatora z kwarcem Q1.

Po poprawnym zaprogramowaniu procesora odłączamy wtyczkę programatora od złącza J1 i naciskamy przycisk *reset*. Powinniśmy zauważyć trzy wolne błyski diody świecącej D2, co świadczy o poprawnej pracy procesora. Jeżeli została zmontowana pamięć RAM wraz z zatrzaskiem, to po sekundzie powinny się pojawić trzy szybkie błyski, a po następnej półsekundowej przerwie kolejne 3 szybkie mrugnięcia diody D2. Jeżeli te dwie serie krótkich błysków będą rozdzielone jednym, dwoma lub trzema wolnymi błyskami, to oznacza, że występuje jakiś problem z pamięcią RAM lub zatrzaskiem adresów IC2. Jeden długi błysk oznacza, że jest błąd na magistrali danych, dwa błyski - błąd na magistrali adresowej, a trzy błyski świadczą o uszkodzeniu któregoś z układów IC2 lub IC3. W takim przypadku należy ponownie sprawdzić poprawność zamontowania IC1, IC2 i IC3.

Jeżeli wstępne testy przebiegły pomyślnie, możemy przejść do kolejnego etapu, w którym wlutowujemy IC4, kondensatory C14...C17, złącza J2 i J3 oraz potencjometr do regulacji kontrastu wyświetlacza R5. Aby przetestować działanie wyświetlacza LCD, należy go podłączyć do złącza J2. Do złącza J3 podłączamy kabel łączący *yamppa* z portem RS232 w komputerze, na którym uruchamiamy dowolny program terminalowy (np. *Hyper Terminal*) z ustawionymi parametrami transmisji na 19200 bodów, 8 bitów i brak parzystości, po czym włączamy zasilanie *yamppa*. Po zakończeniu sekwencji błysków diody D2 powinniśmy zobaczyć tekst powitalny zarówno w oknie terminala, jak i na wyświetlaczu LCD. Jeśli w oknie terminala nic się nie pojawi, to należy sprawdzić poprawność podłączenia kabla oraz poprawność montażu. Można też spróbować odwrócić wtyczkę złącza J3 i ponownie wyzerować *yamppa*. Aby wiadomość powitalna ukazała się na wyświetlaczu LCD, należy za pomocą potencjometru R5 odpowiednio ustawić jego kontrast.

Jeśli nadal wszystko przebiega pomyślnie, nadszedł czas na zamontowanie pozostałych elementów. Tak więc montujemy układy IC6, IC7, IC8 oraz wszystkie pozostałe elementy bierne. W tym miejscu mam ważną uwagę dotyczącą elementów generatora kwarcowego dla układu IC7.

Począwszy od produkcyjnej wersji „K“ układu VS1001, firma VLSI zmieniła sposób podłączenia elementów współpracujących z kwarcem Q2. Ze względu na to, że obecnie są dostępne już tylko wersje „K“ tego układu, modyfikacja staje się niezbędna. Polega ona na zmianie wartości kondensatora C4 na 33 pF, rezystora R7 na 1 MΩ oraz na niemontowaniu rezystora R6. W miejsce J4 może-

my bezpośrednio wlutować odbiornik podczerwieni lub złącze do jego podłączenia. Gdy wszystko jest już na swoim miejscu, do gniazda minijack podłączamy słuchawki, do złącza J3 kabel terminala, a do złącza J4 odbiornik podczerwieni. Jeżeli go nie podłączymy, należy tymczasowo zewrzeć skrajne styki złącza J4, aby zapobiec wnikaniu zakłóceń interpretowanych przez procesor jako przypadkowe komendy zdalnego sterowania. Po włączeniu zasilania powinniśmy usłyszeć cichy trzask w słuchawkach oznajmiający pojawienie się zasilania układu IC7. Po pojawieniu się w oknie terminala tekstu powitalnego powinien się pojawić też znak zachęty do pisania. Wciskając na klawiaturze PC znak zapytania „?“, powinniśmy otrzymać spis możliwych do przeprowadzenia testów dekodera VS1001. Wciskając klawisz 4, powinniśmy usłyszeć w słuchawkach trzy dźwięki. Jeżeli mamy podłączony odbiornik podczerwieni, możemy go również przetestować, wysyłając w jego kierunku sygnał z dowolnego pilota pracującego w standardzie REC80 (np. Panasonic). Na terminalu powinny się pojawić kody odpowiadające poszczególnym klawiszom pilota.

Następnie do złącza J7 podpinamy sformatowany dysk twardy z systemem plików FAT32 i klastrem o wielkości 4kB, a do złącza J5 podłączamy zasilacz, który powinien mieć wydajność prądową około 1A i możemy wykonać sprawdzenie interfejsu ATA, uruchamiając test numer 8. Jeżeli jako źródło plików MP3 chcemy zastosować CD-ROM, to wykonanie powyższego testu jest bezcelowe, ponieważ program testowy obsługuje jedynie dyski twarde.

Podłączenie 3,5-calowego dysku lub CD-ROM-a

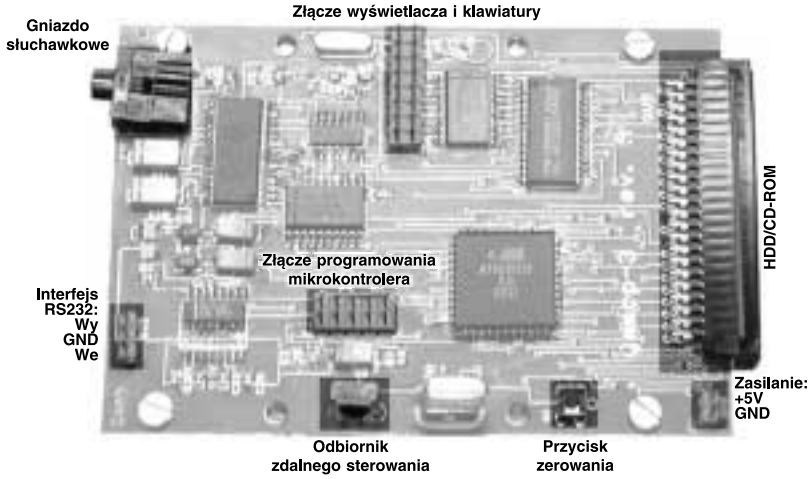
Jeśli nie dysponujemy dyskiem z notebooka, to możemy z powodzeniem zastosować w *yamppie* standardowy dysk 3,5-calowy. Do tego celu można wykorzystać standardową przejściówkę IDE pozwalającą na podłączenie 2,5-calowego dysku w zwykłym PC. W tym przypadku wykorzystujemy ją „na odwrót“. Można też zrobić specjalną taśmę, która po jednej stronie będzie miała 40-stykowe złącze do 3,5" dysku, a po drugiej stronie złącze 44-stykowe do *yamppa*. Styki o numerach od 1 do 40 z 44-stykowego złącza łączymy z odpowiadającymi im stykami złącza 40-stykowego. Pozostałe 4 pozostawiamy wolne. W ten sam sposób można do *yamppa* podłączyć CD-ROM z interfejsem ATA-PI, przy czym należy pamiętać, że *yampp* wymaga urządzenia IDE skonfigurowanego jako MASTER, i trzeba odpowiednio przełączyć zworkę w CD-ROM-ie lub dysku.

Gdy stosowany dysk jest 3,5" lub CD-ROM, będzie potrzebne dodatkowe źródło zasilania tych dysków o napięciu 12 V. W wykonaniu stacjonarnym - takim jak w modelu pokazanym na zdjęciu - można wykorzystać np. zasilacz pochodzący z obudowy od PC typu *slim*. Posiada on między innymi odpowiednie napięcia +5 V i +12 V oraz ma niewielkie wymiary. Przy tak niewielkim obciążeniu jakie powoduje *yampp* i twardy dysk lub CD-ROM może on bez problemu pracować bez wymuszonego chłodzenia wentylatorem.

Oprogramowanie

Oprogramowanie do *yamppa* tworzone jest na licencji GPL, co oznacza, że każdy może do woli zmieniać i publikować swoją wersję pod warunkiem publikacji kodu źródłowego oraz zachowania oryginalnych wpisów dotyczących autorów projektu i wersji bazowej. W ten sposób powstało kilka wersji oprogramowania różniących się oferowanymi funkcjami. Opiszę tutaj najnowsze wersje dostępne w momencie tworzenia artykułu. Piszę w liczbie mnogiej, bo istnieją oddzielne programy obsługujące dyski twarde i CD-ROM-y.

Na wstępie trzeba zaznaczyć, że odpowiednie skonfigurowanie programu do własnych potrzeb, czyli do posiadanego przez nas pilota zdalnego sterowania, odpowiedniego ułożenia klawiszy, zastosowanego w układzie kwarcu, czy choćby zmiany wielkości klastra na dysku wymaga posiadania kompilatora języka C na procesory AVR. Oprogramowanie zostało napisane przy wykorzystaniu darmowego kompilatora AVR-GCC, którego najnowszą wersję można pobrać ze strony *www.avrfreaks.net*. Do poprawnej kompilacji wymagana jest wersja 3.0 (lub o wyższym numerze) wspomnianego kompilatora. Można oczywiście załadować do procesora gotowy skompilowany program, lecz wtedy jesteśmy skazani na takie ustawienia, jakie zostały zaproponowane przez jego autora, w związku z czym prawdopodobieństwo posiadania pilota wysyłającego odpowiednie kody jest niewielkie. W większości przypadków nie jest konieczna umiejętność programowania w języku C, ponieważ program został napisany tak, aby odpowiednią konfigurację można było dostosować poprzez wstawienie lub usunięcie znaków komentarza przy odpowiednich definicjach w pliku *constants.h* lub poprzez zmianę wartości odpowiednich stałych (również w tym pliku). Znakiem komentarza w języku C jest podwójny *slash* (//) wstawiony na początku linii. Niestety, z powodu ograni-



Rys. 7. Rozmieszczenie najważniejszych elementów i złącz na płytce *yamppa*

czonej pojemności pamięci na kod programu trzeba wybierać, które z dostępnych funkcji oprogramowania chcemy włączyć, a które są nam zbędne, żeby program po kompilacji zmieścił się w pamięci procesora.

Pierwszą ważną modyfikacją jest ustawienie odpowiedniej częstotliwości kwarcu zastosowanego w generatorze taktującym układ VS1001 - *#define F_VS1001*. Należy pozostawić bez znaków komentarza tylko jedną z możliwych wartości lub dopisać swoją. Jeśli przez przypadek posiadamy starszą niż „K" wersję układu VS1001, to należy usunąć komentarz z linii *#define OLD_VS1001*.

Następnie poprzez wybór jednej z pięciu dostępnych definicji *LCD_TYPE* wybieramy rodzaj posiadanego wyświetlacza LCD. Następnie należy zdefiniować wielkość klastra dysku twardego. Określamy go, wpisując w linii *#define CLUSTER_SIZE* jego wielkość w bajtach, tak więc 4096 to klaster 4 kB, 8192 to 8 kB, 16384 to 16 kB itd. Jeżeli chcemy korzystać z dysku sformatowanego w systemie FAT16, to musimy usunąć komentarz z linii *#define FAT16*. Jeżeli nie chcemy, żeby dysk zatrzymywał się po naciśnięciu klawisza *Stop*, to musimy wstawić komentarz w linii *#define HDD_SPINDOWN*. Oczywiście te definicje występują tylko w wersji przeznaczonej do HDD. W wersji CD zamiast sekcji *ATA & FAT* mamy *CD-ROM and ATAPI options*, w której należy ustawić preferowaną prędkość odczytu płyt CD (definicja *#define READ_SPEED x*, gdzie x to prędkość w kB/s). Zalecana jest prędkość x4 lub x8, czyli 706 lub 1430 kB/s. Mniejsza prędkość może powodować przerwy w trakcie odtwarzania, a większa sprawia, że napęd CD-ROM zaczyna pracować zbyt głośno, zmniejszając komfort odsłuchu.

Sekcja *SONG NAVIGATION* umożliwia dokonanie wyboru interesujących nas funkcji sterujących. Wybór ten może się okazać konieczny, jeśli będziemy chcieli włączyć jakąś inną funkcję programu, która nie zmieści się w pamięci procesora.

AVR-GCC i *yampp*

Pozostaje jeszcze tylko zdefiniowanie kodów pilota zdalnego sterowania i klawiatury lokalnej. To zadanie zostawmy na później.

Teraz należy zapisać wprowadzone zmiany w pliku *constants.h* i skompilować program do postaci „strawnej" przez programator. Do tego wykorzystamy wspomniany wyżej kompilator AVR-GCC. Po jego zainstalowaniu w systemie powinniśmy dopisać do pliku startowego *Autoexec.bat* ścieżkę dostępu do plików wykonywalnych GCC, czyli w linii *PATH* dopisać na końcu np.;*C:\GCC\bin* - jeśli kompilator został zainstalowany w katalogu *C:\GCC*. Następnie otwieramy katalog, w którym mamy gotową do kompilacji wersję źródłową oprogramowania i piszemy *make* lub *make all*. Po skompilowaniu zobaczymy komunikat o tym, ile zajmuje miejsca wygenerowany kod. Jeśli wartość HEX jest większa niż *1FFFh* lub jeśli dostaniemy komunikat *Region TEXT is full*, to oznacza, że program nie zmieści się w procesorze i należy wyłączyć jakąś funkcję (w pliku *constants.h*). Zdarzyć się to może tylko wówczas, jeśli próbowaliśmy włączyć coś więcej niż to, co było włączone w oryginalnym pliku. Po skompilowaniu dostajemy plik *Yampp3.hex* (lub *Yampp3CD.hex*) gotowy do wprowadzenia do procesora. Sposób programowania procesora jest identyczny z opisanym przy okazji uruchamiania *yamppa*. Po prawidłowym zaprogramowaniu procesora otrzymujemy w pełni funkcjonalny odtwarzacz.

Zdefiniowanie kodów pilota

Aby przystosować jeden z posiadanych pilotów zdalnego sterowania do współpracy z *yamppem*, musimy nauczyć go odpowiednich kodów klawiszy tego pilota. W tym celu możemy wykorzystać jeden ze zdefiniowanych już zestawów kodów lub, jeśli żaden nie działa prawidłowo, określić swój własny. Obecnie *yampp* obsługuje pięć protokołów wysyłania kodów zdalnego sterowania: REC80, NEC80, SONY15, SONY12 i RC5. Począwszy od jednej z ostatnich wersji oprogramowania, w dość drastyczny sposób został zmieniony sposób konfigurowania pilota, a miało to na celu uproszczenie tej procedury. W poprzednich wersjach definicje kodów pilota były umieszczone w jednym dużym pliku *rec80.h*, co powodowało trochę zamieszania przy próbie utworzenia własnego zestawu kodów. Obecnie każdy model pilota posiada swój własny plik definiujący go, czyli protokół komunikacji oraz wymagany zestaw kodów odpowiadających naciskanym klawiszom. Pliki te (o rozszerzeniu.*def*) znajdują się wewnątrz katalogu *REMOTES*, w którym można też znaleźć plik *!_SAMPLE_BASE.def*. Może on posłużyć jako szablon do utworzenia pliku definiującego nasz konkretny model pilota.

Żeby skorzystać z jednego z gotowych plików, wystarczy go skopiować do katalogu z plikami źródłowymi oraz zmienić jego nazwę na *remote.def* (zastępując poprzedni plik o takiej nazwie)

i ponownie skompilować kod. Jeśli nasz pilot jest zgodny z wybranym modelem, to po załadowaniu kodu do *yamppa* powinniśmy uzyskać pełną możliwość sterowania jego pracą.

Jeśli nie posiadamy pilota zgodnego z predefiniowanymi modelami lub posiadamy nadajnik pracujący w nieznanym standardzie, musimy sami stworzyć dla niego odpowiedni plik definiujący. W tym celu należy odszukać w pliku *constants.h* linie *#define SETUP_REMOTE_CODES* i usunąć z niej znaki komentarza. Następnie należy skompilować kod i załadować do *yamppa*. Po jego wyzerowaniu na wyświetlaczu LCD oraz na ekranie terminala R2232 powinniśmy zobaczyć komunikat o tym, że *yampp* próbuje rozpoznać protokół naszego pilota. Żeby mu to umożliwić, należy kilkakrotnie nacisnąć dowolny klawisz naszego pilota. Jeśli pilot nadaje w jednym z obsługiwanych protokołów, to na wyświetlaczu pojawi się numer i nazwa rozpoznanego protokołu. Jeśli na wyświetlaczu będą się jedynie zmieniać numery i nie pokaże się nazwa protokołu, to niestety nasz pilot nie nadaje się do sterowania *yamppem* i musimy poszukać innego.

Załóżmy jednak, że *yampp* rozpoznał protokół i jesteśmy gotowi do stworzenia pliku zawierającego definicje kodów. W tym celu najłatwiej jest skorzystać ze wspomnianego już pliku wzorcowego (szablonu) *!_SAMPLE_BASE.def*. Najpierw tworzymy jego kopię, zmieniając przy okazji jego nazwę na taką, żeby umożliwić łatwą identyfikację pilota, którego będziemy opisywać. Następną czynnością będzie wpisanie w linii *#define REM_STD* numeru protokołu naszego pilota, który będzie wyświetlony w pierwszej linii wyświetlacza *yamppa* (lub na ekranie terminala).

Teraz możemy przystąpić do definiowania kodów poszczególnych funkcji. Naciskając poszczególne klawisze pilota, otrzymujemy na wyświetlaczu (i terminalu) odpowiadające im kody w postaci dwubajtowej liczby zapisanej w kodzie heksadecymalnym, które należy wpisywać w kolejnych liniach opisujących każdą z dostępnych funkcji odtwarzacza, zastępując dotychczasowe wpisy *0xffff*. Pamiętajmy, że w języku C liczby szesnastkowe poprzedzone są prefiksem 0x, więc odczytaną z wyświetlacza liczbę *7cf1* zapisujemy jako *0x7cf1*. Jeśli nie chcemy zdefiniować jakiejś funkcji, pozostawiamy przy niej kod *0xffff*.

Po zakończeniu definiowania kodów zapisujemy nasz plik i kopiujemy do katalogu z plikami źródłowymi, zmieniając mu nazwę na roboczą, czyli *remote.def*. Pozostało jeszcze przywrócenie znaków komentarza w linii *#define SETUP_REMOTE_CODES* pliku *constants.h*, skompilowanie kodu i ponowne załadowanie do *yamppa*.

W ten oto sposób dostosowaliśmy *yamppa* do współpracy z naszym pilotem oraz otrzymaliśmy plik opisujący owego pilota, który może być wykorzystany w kolejnej wersji oprogramowania lub przesłany autorowi celem powiększenia istniejącej bazy gotowych definicji pilotów.

Zdefiniowanie klawiatury lokalnej

Jeżeli nie odpowiada nam zaproponowane rozmieszczenie klawiszy na klawiaturze lokalnej, możemy to zmienić. Jak już wcześniej wspomniałem, każdy klawisz ma dwa znaczenia. Znaczenie krótkiego naciśnięcia definiujemy w linii *#define KBD_SHORT*, a długiego w linii *#define KBD_LONG* wewnątrz pliku *constants.h*. Pierwszą definicją musi pozostać *EV_IDLE* - oznacza brak akcji, jeśli nie jest naciśnięty żaden klawisz. Kolejne definicje oznaczają funkcje klawiszy oznaczonych od K1 do K8. Jeżeli w obu liniach dany klawisz ma przypisaną tę samą funkcję, to będzie on posiadał tylko jedno znaczenie niezależnie od czasu jego naciśnięcia (tak powinno być dla klawiszy regulacji głośności). Należy w tym miejscu zaznaczyć, że dłuższe przytrzymanie klawisza powoduje wysyłanie kodu *LONG* co około 0,8 sekundy, czyli samopowtarzanie klawisza. Jest to przydatne szczególnie przy regulacji głośności i szybkim przewijaniu w przód i w tył. Samopowtarzanie tych funkcji zostało specjalnie przyspieszone w stosunku do pozostałych. Po ustaleniu znaczenia wszystkich ośmiu klawiszy wystarczy zapisać zmodyfikowany plik, ponownie skompilować kod i wgrać do procesora.

Miłej zabawy!

Romuald Biały

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/pazdziernik02.htm*** *oraz na płycie CD-EP10/2002B w katalogu* ***PCB****.*

Yampp w Internecie

W Internecie można znaleźć sporo stron zawierających opisy, porady i przeróbki dotyczące *yamppa*. Pozwolę sobie przytoczyć adresy najważniejszych:
Strona główna - http://www.yampp.com
Najnowsze wersje oprogramowania - http://www.yamppsoft.prv.pl
Strona firmowa VLSI Solutions Oy - http://www.vlsi.fi
Kompilator AVR-GCC - http://www.avrfreaks.net
Forum dyskusyjne poświęcone *yamppowi* - http://www.myplace.nu/mp3/yabb/YaBB.cgi

Układy VS1001 nie będą oferowane przez AVT, można je będzie natomiast zakupić w polecanym przez Jespera Hansena sklepie internetowym Jelu (*http://www.jelu.se/shop.php*). Płatności można dokonać za pomocą karty kredytowej lub przelewem na konto - wszystkie informacje znajdują się na stronie sklepowej.
Ważne jest, żeby nie zamawiać na raz podzespołów na kwotę większą niż 70 EUR, ponieważ grozi to odprawą celną, która potrwać może (z doświadczeń redakcyjnych) nawet do 3 tygodni. Wiąże się ona także ze sporymi kosztami dodatkowymi.
Przykładowe ceny podzespołów (z połowy września 2002):
- VS1001 - 20 USD,
- 74LVC245 - 1,50 USD,
- kwarc 7,3728 MHz - 2,30 USD,
- kwarc 12,288 MHz - 2,70 USD.