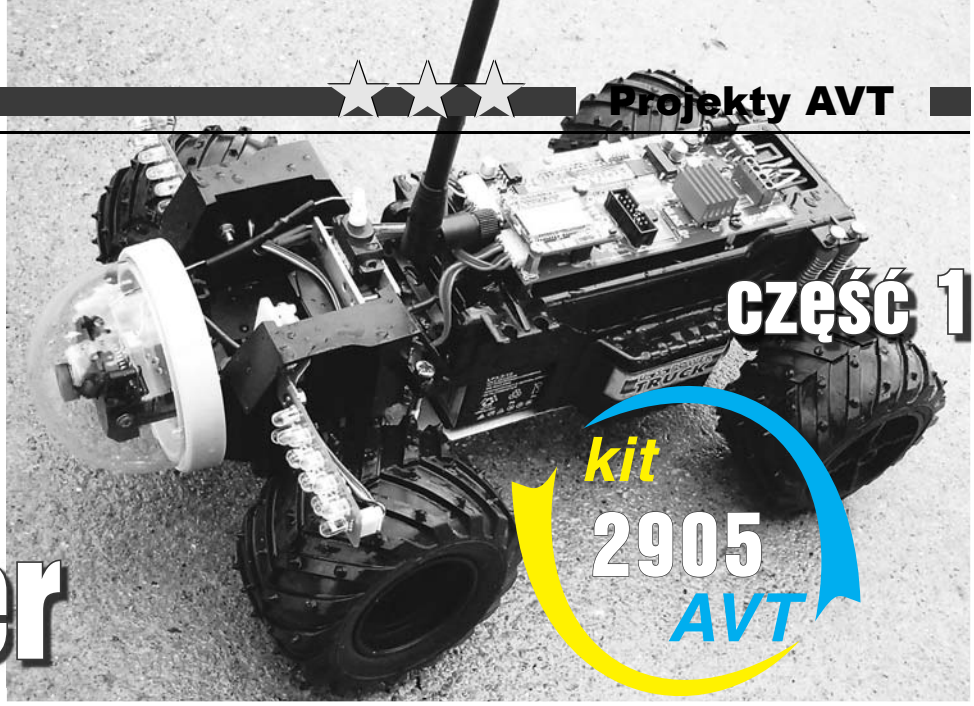
Robot MOS Voyager

część 1



AZ: Celem projektu była budowa mobilnego robota sterowanego drogą radiową. Dodatkowym założeniem było to, że pojazd ma być wyposażony w kamerę rejestrującą otoczenie. Nazwa robota wywodzi się z zainteresowania autorów eksploracją kosmosu, a w szczególności sondami kosmicznymi. Tak, autorów jest dwóch. Jeden to spec w dziedzinie mikroprocesorów, drugi ma smykałkę do mechaniki. Autorów sporo dzieli, ale również coś łączy: miłość do elektroniki, jak i marzenie z dzieciństwa, by zbudować własnego robota. Inicjały pokazują, kto jest Autorem poszczególnych fragmentów artykułu.

Młodsi Czytelnicy EdW mogą nie pamiętać tych czasów, ale nieco starsi stażem z pewnością przyznają rację: słynne raabowozy, czyli najróżniejsze elektroniczne „gryzonie", zawsze wzbudzały wiele emocji wśród Czytelników. Tych młodszych, bo to doskonała zabawa. I tych starszych, bo pod pretekstem poważnego hobby, za jakie uchodzi elektronika, mogli się na powrót zamienić w małych chłopców :-).

Co wyróżnia Voyagera pośród całej menażerii podobnych konstrukcji? Robot jest sterowany z poziomu aplikacji działającej w środowisku Windows. Komunikacja odbywa się drogą radiową za pomocą standardu Bluetooth, a do tego robot jest wyposażony w nietypowy digitalizer wideo, czyli układ rejestrujący w podręcznej pamięci ramki obrazu z analogowej kamery, które następnie mogą być przesłane do komputera.

Konstrukcja mechaniczna

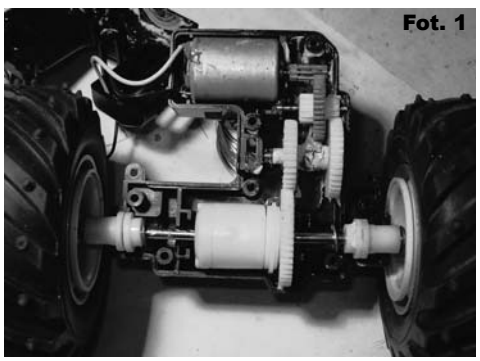
AZ: Robotyka jest niezwykle pociągającą dziedziną techniki. Nie ma jednak róży bez kolców, w szczególności dla nas elektroników – aspekty mechaniczne są bolączką dla wielu z nas. Trzeba sobie jednak jakoś radzić. Autorzy, chociaż mają pewne doświadczenie w dziedzinie cięcia, wiercenia, piłowania różnych rzeczy, to mechanikami nie są i do takiego miana nawet nie pretendują. Dlatego przy budowie robota zdecydowali się na łatwiejszą drogę – wykorzystali podzespoły z uszkodzonych, zdalnie sterowanych zabawek.

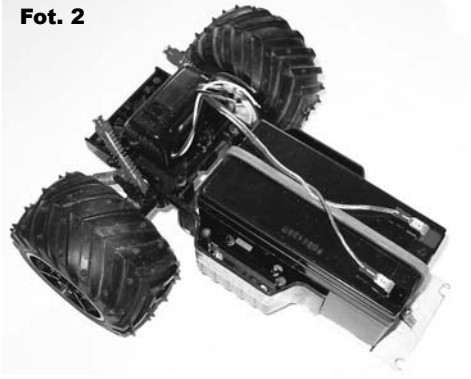
Podstawą do budowy ramy naszego robota były dwie zabawki. Przód z układem skrętnym pochodzi z jednej zabawki, a tył wraz z układem napędowym – z drugiej. **Fotografia 1** przedstawia budowę układu napędowego z przekładnią i dyferencjałem. Dyferencjał, czyli mechanizm różnicowy, ma za zadanie dostarczać do każdego koła jednakowy moment obrotowy. Ma to duże znaczenie np. na zakrętach, kiedy koła pokonują różne drogi. Dzięki przekładni jedno koło może obracać się szybciej, a drugie wolniej. Jednak suma prędkości obrotowej obu kół zawsze będzie równa podwójnej prędkości wału napędowego. Chroni to układ przeniesienia napędu przed zbytnimi naprężeniami.

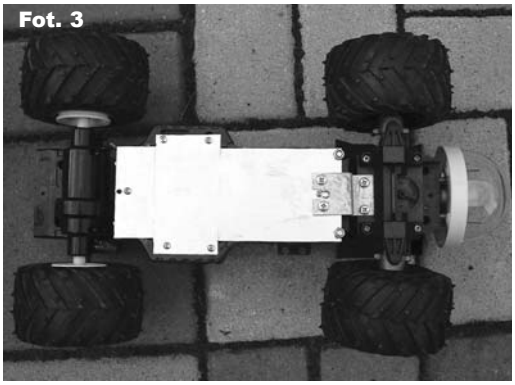
Podwozie robota zostało zrobione z jednego kawałka odpowiednio dociętej aluminiowej blachy. Do zasilania wykorzystaliśmy akumulator żelowy, który niestety dużo waży – ponad 1kg. **Fotografia 2** przedstawia sposób jego montażu (dodatkowo został on przyklejony do blachy za pomocą grubej dwustronnej taśmy, co w sumie by w zupełności wystarczyło), natomiast **fotografia 3** przedstawia ostateczny kształt podwozia. Gruby kątownik oraz liczne śruby mocujące były konieczne dla zapewnienia odpowiedniej sztywności ramy nośnej, szczególnie w miejscu łączenia szkieletów obu zabawek, które zostały praktycznie w całości „wypatroszone".

Zasilanie

MW: Voyager, ze względu na wagę oraz duży apetyt na prąd, potrzebował wydajnego źródła zasilania – stąd akumulator żelowy 12V 3,5Ah. Podzespoły robota wymagają różnych napięć zasilających: 12V dla silników, kamery wideo i oświetlenia, 5V dla serwomechanizmów i digitalizera wideo oraz 3,3V dla mikroprocesora MSP430 i modułu Bluetooth. Napięcie na akumulatorze, wbrew pozorom, bardzo się zmienia: od ponad 13V w stanie pełnego naładowania, do poniżej 10V w stanie rozładowania. Ponieważ jednak układy zasilane z 12V mają dosyć szeroki zakres napięcia pracy, nie było konieczności stosowania dodatkowych stabilizatorów i można je było zasilić wprost z akumulatora. Schemat części zasilającej można zobaczyć na **rysunku 1**. Serwomechanizmy zasilane z 5V mogą pobierać szczytowo prąd rzędu 1A. Idealnym rozwiązaniem okazała się pięciowoltowa przetwornica impulsowa LM2676 (układ U7) step down o prądzie 3A,

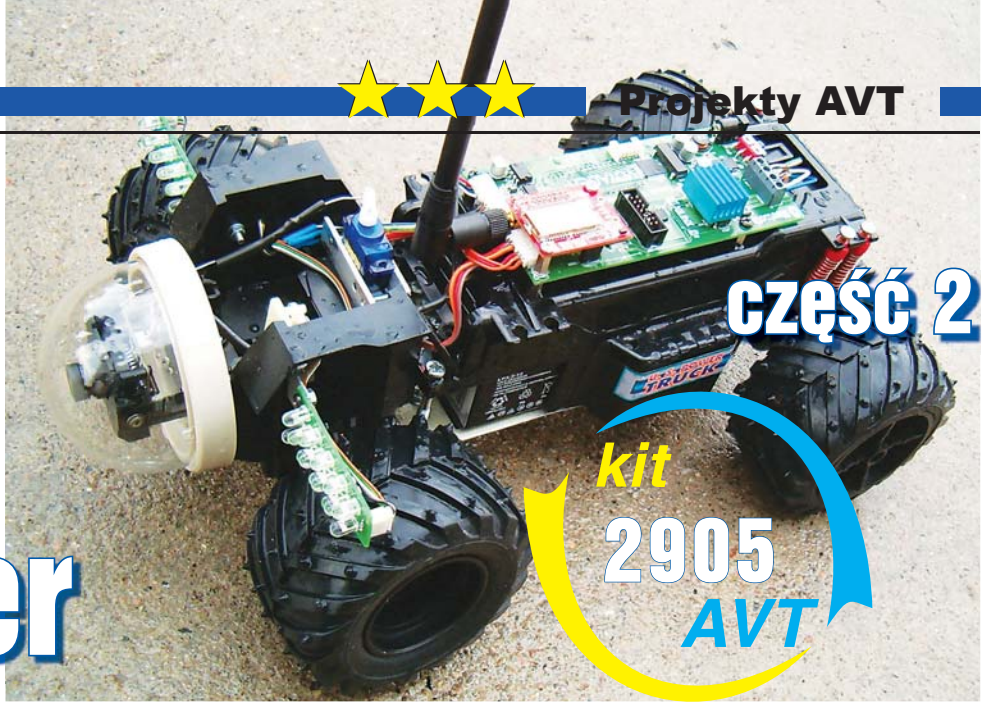

Fot. 1


Fot. 2


Fot. 3

Robot MOS Voyager

część 2



W poprzedniej części opisu robota skończyliśmy na prezentacji zdjęć zarejestrowanych pokładową kamerą. Istotną kwestią jest sposób przesyłania ich do operatora robota i od tego zaczniemy artykuł w części drugiej.

Transmisja obrazu

AZ: W trybie podstawowym digitalizer dostarcza zdjęć o rozdzielczości 416 na 288 pikseli, co oznacza, że jeden obraz zajmuje: 416 x 288 = 119808 bajtów = 117kB. W wersji oprogramowania, które przedstawiamy Czytelnikom w tym artykule, nie zastosowaliśmy żadnej kompresji obrazu. Co to oznacza w praktyce? Wirtualne łącze RS232 utworzone pomiędzy modułem Bluetooth a komputerem pozwala na przesyłanie danych z prędkością: 115200bps = 112,5kbps. Ile czasu zajmuje przesłanie jednego obrazu z kamery? W tym miejscu musimy się zastanowić, jak określa się prędkość transmisji w łączu szeregowym. Czy 115200bps (*bit per second*) oznacza, że w ciągu sekundy będziemy w stanie przesłać 115200/8=14400 bajtów danych? Nie, ponieważ jednostka bps określa maksymalną liczbę bitów możliwych do przesłania, ale łącznie z bitami kontrolnymi (czyli bitem startu, parzystości itd.). Inną jednostką określającą prędkość transmisji jest bod (ang. Baud), który „mówi nam", jak często może się zmieniać przesyłany sygnał (ile symboli możemy przesłać w ciągu sekundy). Dla łącza RS232 wyróżniamy dwa stany sygnału: logiczna jedynka (ok. –10V) oraz logiczne zero (ok. +10V), czyli jedna zmiana sygnału jest równoważna jednemu bitowi danych. Dla RS232 liczba bodów jest równa prędkości transmisji danych wyrażonej w bps.

Port szeregowy w robocie jest skonfigurowany następująco: 115200-8-N-1, co oznacza, że jedna ramka danych składa się z bitu startu, 8 bitów danych oraz jednego bitu stopu (bez bitu parzystości) – sumarycznie ramka składa się z 10 bitów. Dysponując łączem 115200 bodów, możemy w ciągu sekundy wysłać: 115200 / 10 = 11520 znaków (bajtów danych), czyli 11,5kB danych. Wcześniej policzyliśmy „wagę" jednego zdjęcia, tj. 117kB. Przesłanie go zajmie nam: 117kB / 11,5kB = 10,2 sekundy. Też nie byliśmy zachwyceni tym wynikiem ;-). Po prostu, przystępując do budowy robota, byliśmy pewni, że posiadany przez nas moduł BT (a dokładniej wirtualny COM) będzie pracował z prędkością 460,8kbps, która „czarno na białym" widnieje w dokumentacji modułu. Przy takiej prędkości oraz przy przesyłaniu zdjęć o mniejszej rozdzielczości (288x144), moglibyśmy uzyskać jedną klatkę obrazu na sekundę. Do transmisji wideo, wciąż byłoby daleko, niemniej dawałoby to już jakąś namiastkę.

Kompresja obrazu

AZ: Stare powiedzenie głosi: „Jak się nie ma, co się lubi, to się lubi, co się ma!". Zważywszy na niską dzielność terenową robota oraz brak transmisji obrazu na żywo, uznaliśmy, że robot świetnie nadaje się do robienia zdjęć pamiątkowych :-). I na tym moglibyśmy poprzestać, gdyby nie dobiegające nas z widowni gromkie okrzyki: „Kompresuj!". Dobry pomysł, tylko jaki algorytm zastosować? Do naszych rozważań weźmy na początek jeden z bardziej popularnych: JPEG. Ze standardem tym każdy z nas spotyka się na co dzień, chociażby przeglądając strony w Internecie. Kompresja JPEG jest kompresją stratną, co oznacza, że po kompresji część informacji z obrazu jest bezpowrotnie tracona. Jak to się dzieje, że pomimo usunięcia części danych, obraz niewiele traci na jakości? Kompresja JPEG wykorzystuje właściwości naszego narządu wzroku – ludzkie oko jest mało wrażliwe na wysokie częstotliwości w obrazie (dokładniej rzecz ujmując: na amplitudę szybkich zmian jasności). Usunięcie tych składowych z obrazu będzie dla oka prawie niezauważalne.

W tym miejscu każdy, kto miał do czynienia z edukacją na politechnice, powinien przeczuwać, że bez transformaty Fouriera się nie obejdzie :-). Transformata Fouriera jest jednym z podstawowych narzędzi służących do dekompozycji sygnału, czyli wydobycia z sygnału informacji o jego parametrach częstotliwościowych. Fourier udowodnił (prawie 2 wieki temu!), że każdy, dowolnie złożony przebieg okresowy da się „złożyć" z odpowiedniej liczby przebiegów sinusoidalnych oraz składowej stałej. Transformata (nazwa na jego cześć) określa stopień korelacji badanego sygnału z funkcjami harmonicznymi (reprezentującymi jakby poszczególne częstotliwości w widmie sygnału). Przekształcenie całkowe Fouriera operuje na sygnale ciągłym. W praktyce jednak korzystamy z sygnałów dyskretnych (skwantowane próbki dźwięku czy piksele obrazu). Dlatego też, wprowadzono dyskretną transformatę Fouriera (DFT). Obliczenie jej wymaga jednak działań na liczbach zespolonych (bardziej złożonych obliczeniowo). Dlatego w standardzie JPEG wykorzystuje się dyskretną transformatę cosinusową (DCT), która jest specjalnym przypadkiem DFT. Jej zaletą jest m.in. to, że operuje jedynie na liczbach rzeczywistych. Wzór tej transformaty można zobaczyć na **rysunku 16**. Jako dane wejściowe podajemy jej obraz, traktowany jako macierz (tablicę)

$$G_{u,v}=\alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g_{x,y}\cos\left[\frac{\pi}{N}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{N}\left(y+\frac{1}{2}\right)v\right]$$

Rys. 16

$$Q=\begin{bmatrix}16&11&10&16&24&40&51&61\\12&12&14&19&26&58&60&55\\14&13&16&24&40&57&69&56\\14&17&22&29&51&87&80&62\\18&22&37&56&68&109&103&77\\24&35&55&64&81&104&113&92\\49&64&78&87&103&121&120&101\\72&92&95&98&112&100&103&99\end{bmatrix}$$

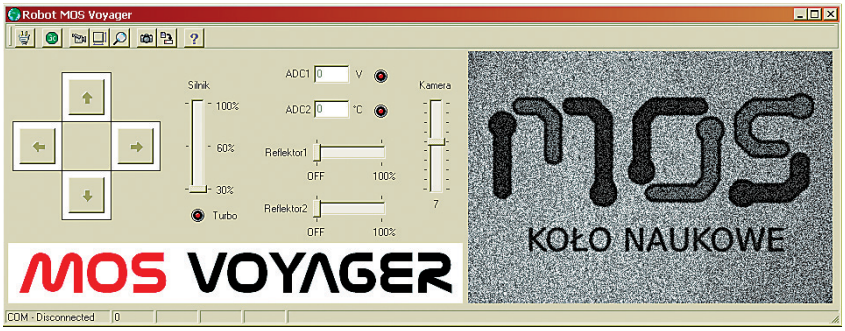
```
//jeśli w buforze odbiorczym znajduje się jakiś znak
to...
if (IFG1 & URXIFG0)
{
  //odczytaj nowy znak
  Rx_char = RXBUF0;

  //rozkazy to kody ASCII od 128 do 255
  if (Rx_char > 0x7F)
  {
    //odebrano nowy rozkaz
        function = Rx_char;
  }
  else //dane to kody ASCII od 0 do 127
  {
    //zdekodowanie rozkazu:

        //ustaw serwo kamery
    if (function == 0xF1) set_servo(0, Rx_char);
    ...
        //rozkaz zatrzymania silnika
        if (function == 0xF5)
    {
      new_order=2;  //zatrzymaj się
      speed=0;
    }
        //Nowa prędkość maksymalna
        if (function == 0xF6) speed=Rx_char;
    //steruj 1. reflektorem
        if (function == 0xF7) set_light(0, Rx_char);
    ...
        //zmierz napięcie na wybranym kanale
        if (function == 0xF9) get_ADC(Rx_char);
    ...
        //przechwyć jedną ramkę obrazu
        if (function == 0xFC) grab_picture();

    function = 0;
        }
}
```

Listing 4



Rys. 17

o wymiarach N na N. Zmienna g_{XY} oznacza kolor piksela o współrzędnych x, y, natomiast u oraz v to numer składowej harmonicznej (u, v = 0,1,..., N–1). Część tego wzoru, jak np. cosinusy i współczynniki skalujące a(u)a(v), można stablicować.

W standardzie JPEG obraz dzielimy na bloki 8 na 8 pikseli i poddajemy je DCT. W naszym przypadku przy rozdzielczości zdjęcia 416x288 musielibyśmy wykonać taką operację dla 1872 bloków. W wyniku DCT w miejsce pikseli otrzymujemy współczynniki transformaty. W lewym górnym rogu macierzy wynikowej G (o rozmiarach 8x8) umieszczone są współczynniki odpowiadające amplitudom wolnych zmian jasności, natomiast pod przekątną znajdują się współczynniki wysokoczęstotliwościowe. Na tym etapie obraz można jeszcze całkowicie odtworzyć. Faktyczna kompresja następuje dopiero przy kwantyzacji. Współczynniki otrzymane w wyniku transformaty dzielimy przez odpowiadające im elementy macierzy Q i zaokrąglamy w dół. Po przyjrzeniu się macierzy Q, łatwo zauważyć, że największe współczynniki znajdują się w prawym dolnym rogu. Powoduje to wyzerowanie składowych o wyższych częstotliwościach (faktyczne usunięcie ich z obrazu). Tak otrzymane macierze kompresujemy na koniec bezstratnie i zapisujemy do pliku.

Z punktu widzenia mikrokontrolera jest to ogromna liczba działań arytmetycznych, szczególnie typu mnożenie i akumulacja przy liczeniu DCT. Chociaż to domena procesorów sygnałowych, część mikrokontrolerów z rodziny MSP430 ma sprzętowy mnożnik, który realizuje taką operację (mnożnik ten nie jest częścią rdzenia – obsługuje się go poprzez rejestry tak jak inne peryferie).

Nie będę tutaj przedstawiał innych algorytmów kompresji, ponieważ wszystkie są złożone obliczeniowo. Tym, którzy chcą się dowiedzieć czegoś więcej, na temat kompresji JPEG polecam dokument „Matematyczne podstawy kompresji JPEG", autor P. Przybyłowicz, a zainteresowanych przetwarzaniem sygnałów odsyłam do książki będącej swoistym kompendium: „Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań", autor T. Zieliński (zbieżność nazwisk przypadkowa).

Na koniec warto jeszcze rozważyć pewien bardzo prosty algorytm kompresji bezstratnej, z którym poradziłby sobie nasz mikrokontroler. W obrazie może się zdarzyć, że wystąpi ciąg pikseli o tym samym kolorze. Gdyby wyszukiwać takie ciągi i kodować je w bardziej oszczędny sposób? Np. tak: <pierwszy piksel w ciągu><bajt sygnalizujący wystąpienie ciągu><liczba powtórzonych pikseli>. Takie kodowanie ma sens dla co najmniej czterech pikseli w ciągu. Napisałem prosty program (na PC-ta), który analizował pod tym kątem zdjęcia zrobione przez Voyagera. Zysk z takiej kompresji (przy założeniu, że badamy tylko linie obrazu) był na poziomie do 4%. Dużą poprawę dało zmniejszenie palety do 16 odcieni szarości (co w tym przypadku nie wpływa drastycznie na pogorszenie jakości obrazu). Powodowało to zysk kompresji na poziomie 10...40%. To jednak wciąż za mało, by transmitować obraz na żywo. Ostatecznie poprzestaliśmy na robieniu zdjęć bez kompresji „w pełnej" rozdzielczości 416x288.

Protokół komunikacyjny

AZ: Protokół komunikacyjny robota jest bardzo prosty (**listing 4**). Każdy rozkaz składa się z dwóch bajtów, z czego pierwszy to polecenie (znaki ASCII > 127), a drugi to zmienna (znaki <128). Takie rozgraniczenie jednoznacznie identyfikuje polecenia oraz dane. Mogliśmy sobie pozwolić na takie uproszczenie protokołu, ponieważ nad bezpieczeństwem transmisji czuwają niższe warstwy protokołu Bluetooth. Wszystkie rozkazy dotyczące ruchu robota (jazda do przodu, do tyłu, skręt) wymagają ciągłego ponawiania, tzn. jeśli w ciągu 600ms nie przyjdzie ponownie polecenie nakazujące jazdę np. do przodu, to robot się zatrzyma. Jest to zabezpieczenie na wypadek utraty kontroli.

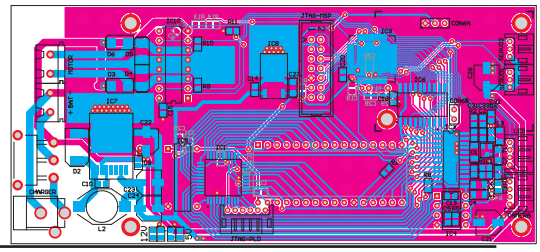
Oprogramowanie

AZ: Program dla MSP430 został napisany w języku C w środowisku IAR. Darmowa wersja tego oprogramowania pozwala tworzyć programy zajmujące do 4KB pamięci flash. W naszym przypadku ograniczenie to nie było problemem. Kody źródłowe są dostępne w Elportalu, tak więc każdy zainteresowany może po nie sięgnąć. Kod jest podzielony na moduły opatrzone sporym komentarzem, tak więc nie powinno być większego problemu ze zrozumieniem jego działania. Program na PC-ta powstał w środowisku Delphi 6. Można go zobaczyć na **rysunku 17**. Program prawie w całości jest sterowany z klawiatury – klikanie myszką nie na wiele by się zdało przy jeździe robotem. Aplikacja korzysta z komponentu ComPort, zapewniającego dostęp do portu szeregowego. Jest to biblioteka przeznaczona dla środowiska Borland, rozprowadzana na licencji **Public Domain. Do komunikacji z robotem od strony komputera używamy modułu Bluetooth na USB, który wg specyfikacji zapewnia transmisję do 100m. Do jego obsługi w systemie Windows używamy aplikacji BlueSoleil (niestety sprawiała ona czasem trochę problemów).**

Płytka drukowana

MW: Płytka drukowana (**rysunek 18**), jak już wcześniej zostało wspomniane, powstała przez rozbudowanie istniejącego projektu płytki prototypu digitalizera. Postanowiliśmy w całości zastosować montaż SMD, lecz z pewnymi wyjątkami. Driver silnika L293 jest trudno dostępny w wersji SMD, a wersja DIP

Rys. 18 Skala 50%



lepiej odprowadza ciepło i pozwoliła dodatkowo na przyklejenie podłużnego radiatora na grzbiecie układu (okazał się on niezbędny!). Drugim wyjątkiem jest pamięć RAM – cóż, tutaj wyszyło na jaw lenistwo autora. Pamięć do prototypu digitalizera została kupiona „za rogiem" w jednym z wrocławskich sklepów elektronicznych. Gdy powstawał ostateczny projekt płytki robota, to po prostu szkoda było zachodu na przeprojektowywanie układu dla pamięci w obudowie SMD. Układ LM1881 mimo swych niewątpliwych zalet także nie jest już najmłodszą konstrukcją – w jego przypadku wersja SMD jest praktycznie nie do zdobycia. Ostatnim wyjątkiem są wszystkie złącza. Jak można zobaczyć na zdjęciach, autor płytki postanowił zastosować dość fikuśne konektory w rastrze 2mm do podłączenia serwomechanizmów, reflektorów i kamery. Skłoniły go do tego ich małe rozmiary i brak miejsca na krawędzi płytki przeznaczonej na złącza. Jednak nie ma róży bez kolców: przewody jednego z serwomechanizmów okazały się zbyt grube i sprawiały dużo problemów z poprawnym zaciśnięciem na nich blaszek stykowych, czasami powodując malutkie zwarcia ;-). Z uwagi na złożoność płytki nie było sensu nawet myśleć o samodzielnym jej wytrawieniu – płytki zostały wykonane w firmie i mają metalizację.

Możliwość rozbudowy

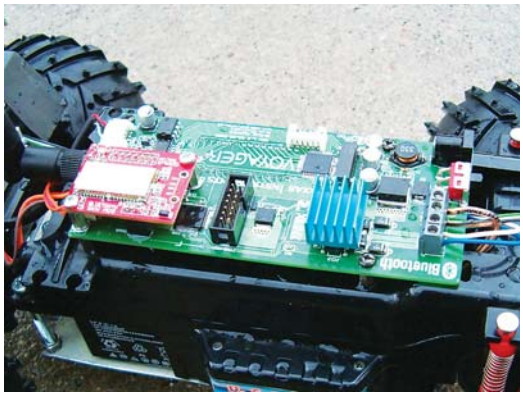
AZ: Możliwość rozbudowy robota zasygnalizowaliśmy już przy okazji omawiania procesora – na złączu JTAG wyprowadziliśmy kilka linii dowolnego przeznaczenia. Jedną z nich zastosowaliśmy do sterowania światłem stopu, którego projekt początkowo nie przewidywał. Do budowy tylnego światła wykorzystaliśmy zapasową płytkę do reflektora, z tym że wlutowaliśmy tam diody czerwone (superjasne). Światło zostało przykręcone z tyłu robota za pomocą dwóch tulejek dystansowych (**fotografia 13**). W czasie postoju oraz jazdy do przodu diody łagodnie się świecą – gdy jednak zatrzymamy robota lub będziemy cofać – reflektor zaświeci się pełną jasnością. Ponieważ wszystkie PWM-y sprzętowe w MSP430 były już zagospodarowane, musiałem do sterowania światłem stopu „napisać" bardzo prosty PWM programowy, który pracuje w głównej pętli programu. Wypełnienie przebiegu wynosi zaledwie 2%, jednak użyte diody są bardzo jasne i to w zupełności wystarcza do „podświetlenia" reflektora. W przypadku cofania diody są zasilane prądem stałym, czyli świecą z maksymalną jasnością.


Fot. 13

Do kolejnej, już bardziej zaawansowanej modyfikacji Voyagera przyczyniły się zawody Robotic Arena 2008 zorganizowane przez Koło Naukowe Robotyków „KoNaR". Impreza odbyła się 13 grudnia 2008 roku na terenie kampusu Politechniki Wrocławskiej. Postanowiliśmy wystartować w kategorii otwartej, w której nie obowiązują żadne wytyczne co do konstrukcji robota. Jest tylko jeden warunek – robot musi się spodobać publiczności, ponieważ to ona głosuje na zwycięzcę. Cóż... usiedliśmy przy herbacie i zaczęliśmy się zastanawiać, jak zainteresować publikę. Nasz Voyager robi pamiątkowe zdjęcia – to miłe, ale to jeszcze nic przebojowego. Tak narodził się pomysł rozbudowy robota o system nagłośnienia. Michał w ramach innego projektu zbudował swego czasu odtwarzacz muzyczny umiejący odgrywać płyty CD zgrane do plików *.wav, na kartę SD. Urządzenie wyposażone dodatkowo w graficzny wyświetlacz LCD, ze względu na swoje wymiary, dobrze się komponowało „na grzbiecie" Voyagera. Wtedy też postanowiliśmy przenieść antenę modułu BT z przodu na tył (zamontowana z przodu wyglądała zbyt ekscentrycznie, a poza tym stwarzała niebezpieczeństwo zahaczenia nią o przeszkodę, gdy robot pod czymś przejeżdżał – antena zamontowana z tyłu po prostu się „położy"). Odtwarzacz audio jest sterowany poprzez port szeregowy (oryginalnie miał dedykowaną klawiaturę). Ponieważ moduł BT dysponuje tylko jednym łączem szeregowym, odtwarzacz, jak i mikrokontroler zostały podłączone równolegle, z tym że są dla siebie niewidoczne. Kanały TxD obu urządzeń są podłączone do wejścia RxD modułu BT poprzez diody 1N4148 i rezystor podciągający (prosta bramka AND). Podobnie w drugą stronę. Odtwarzaczowi zostały przydzielone rozkazy, niekolidujące z oprogramowaniem Voyagera. Należało jedynie zadbać o to, by oba urządzenia nie nadawały jednocześnie.


Fot.14

Mamy już odtwarzacz – teraz potrzebny nam jest jeszcze jakiś głośnik. I najważniejsza kwestia – gdzie go zamontować? Po bokach robota jest trochę wolnego miejsca – można by tam ewentualnie zamontować jakieś niewielkie głośniki. Miałem jednak wizję, która kazała mi poszukać tuby głośnikowej (na wzór megafonu). Szybko się jednak okazało, że tego typu przetworniki są dosyć duże – nie ma mowy o montażu na robocie. Jeśli nie „na" to może „za"? Przyczepa okazała się najlepszym rozwiązaniem. Użyta przez nas tuba to zwykły głośnik w aluminiowej obudowie, o mocy 10W przy impedancji 8Ω. Oprócz głośnika na przyczepie zamontowałem także wzmacniacz – w tej roli wykorzystaliśmy układ TDA2006 w klasycznej konfiguracji zaczerpniętej z noty katalogowej. Jest to wzmacniacz pracujący w klasie AB. Ten leciwy układ okazał się w zupełności wystarczający w naszym zastosowaniu. Przy zasilaniu wprost z akumulatora dostarcza ok. 4W mocy. Układ jest chłodzony za pomocą niewielkiego radiatora zamontowanego na zewnątrz obudowy. Z tyłu robota



znajduje się gniazdo typu DIN, do którego podpięta jest „elektryka" przyczepy, tj. zasilanie oraz sygnał audio. Głośnik z powodu swej budowy ma dosyć wąskie pasmo przenoszenia dźwięku: 0,5...7 kHz, jest za to głośny. Dlatego, na wspomnianych zawodach, w czasie prezentacji „graliśmy" przede wszystkim soul i funk z lat 70. Taka muzyka nadawała całości dodatkowego klimatu, ale jest także inny powód – współczesna muzyka z dużą ilością syntetycznego basu, po odarciu z niskich częstotliwości, brzmi nijako. Efekty naszej pracy możesz zobaczyć na **fotografii 14**. Tak oto zrodził się najprawdziwszy pojazd propagandowy!

Zakończenie

Na koniec należałoby wspomnieć, iż robot tak naprawdę, poza spełnianiem marzeń z dzieciństwa, miał także inny cel, dla którego powołaliśmy go do życia. Tutaj możemy zdradzić, dlaczego zastosowaliśmy dość mało popularny w Polsce mikrokontroler MSP430. Napisanie programu pod ten procesor było naszym projektem zaliczeniowym z kursu Technika Cyfrowa 3 (głównym tematem kursu jest właśnie MSP430). Prezentując robota, udało nam się uzyskać lekki uśmiech na zawsze srogiej twarzy prowadzącego, a stwierdzenie „za zapał i poświęcenie – 5.0" dopełniło szczęścia ;-).

Wykaz elementów

Rezystory

R1	75Ω (SMD 0805)
R2,R4	680Ω (SMD 0805)
R3	75Ω (SMD 0805)
R5	680kΩ (SMD 0805)
R6-R13	10kΩ (SMD 0805)
R14,R16	1kΩ (SMD 0805)
R15	300Ω (SMD 0805)

Kondensatory

C1,C2,C4-C6,C8-C14,C16-C19,C30	100nF (SMD 1206)
C3	510pF (SMD 0805)
C7,C25,C26	10µF (tantal SMDA)
C21,C23,C24,C26	220µF (elektrolit SMD obudowaD)
C22	100µF (elektrolit SMD obudowaD)
C28,C29	22pF (SMD 0805)
C31	47pF (SMD 0805)
C32	10pF (SMD 0805)
C33	220pF (SMD 0805)

Półprzewodniki

D1	1N4148 (SMD MMELF)
D2	MBRS340T3 (SMD SMC)
D3-D6	FR2J (SMD SMB)
D7,D8	LED (SMD 0805)
T1	BC817 (SMD SOT23)
U1	XC9572XL – VQ45 (SMD TQFP44)
U2	dowolna pamięć SRAM 128KB (DIP32)
U3	TLC5540 (SMD SO24)
U4	LM1881 (DIP8)
U5	generator 16MHz 5V (DIP14)
U6	74HC244 (SMD SO20)
U7	LM2676 (SMD TO263)
U8	LF33 (SMD TO252)
U9	MSP430F134 (SMD TQFP64)
U10	L293 (DIP16)

Pozostałe

CON1	CRIMP6 (raster 2mm)
CON2,CON3	CRIMP4 (raster 2mm)
CON4,CON5	CRIMP3 (raster 2mm)
CON6A	GOLDPIN 3pin
CON6B	GOLDPIN 2pin
CON7	IDC14
CON8	ARK2 (raster 5mm)
CON9,COM10	złącze DC (bolec 2,1mm)
L1	10µH (SMD 1206)
L2	33µH (dławik mocy SMD, CSN084F)
L3	4,7µH (SMD 1206)
S1	przełącznik suwakowy (typ SL19123)
X1	8MHz (HC49/U)

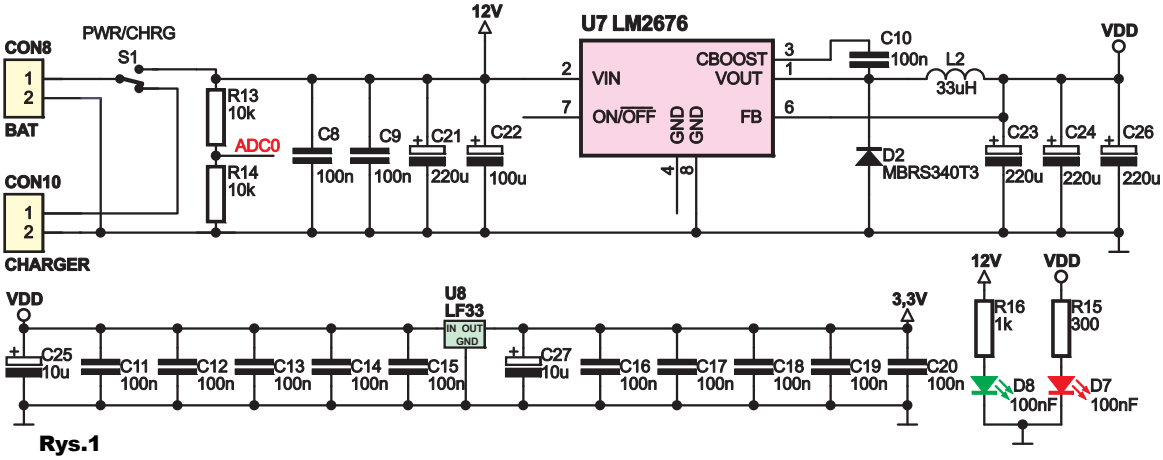
Płytka drukowana jest dostępna w sieci handlowej AVT jako kit szkolny AVT-2905.

Nie sądzimy, by znalazło się wielu chętnych do budowy robota, dokładnie wg naszego projektu. Realnie patrząc – to dosyć droga zabawka i w pewnej mierze skonstruowana z elementów, które mieliśmy pod ręką (szczególnie pod względem mechanicznym). Trudno byłoby odtworzyć dokładnie taką konstrukcję. Jesteśmy jednak przekonani, że artykuł ten będzie stanowił inspirację dla początkujących konstruktorów i zachęci ich do postawienia pierwszych kroków w robotyce. Bardziej zaawansowanych konstruktorów może zainspiruje pokładowa kamera i system audio, które pokazują, że robot może robić coś więcej niż tylko jeździć :-).

Więcej zdjęć Voyagera, nagrania video oraz inne materiały znajdziesz w Elportalu pod adresem: http://www.elportal.pl/index.php?module=ContentExpress&func=display&ceid=302

Projekt powstał w ramach Studenckiego Koła Naukowego MOS działającego przy Katedrze Metrologii Elektronicznej i Fotonicznej na Politechnice Wrocławskiej. Autorzy artykułu są odpowiedzialni m.in. za projekt P.I.W.O.2, czyli Potężnego Indeksowanego Wyświetlacza Oknowego (http://projekt-piwo.pl**).**

Michał Wysocki,
mos.wysocki@gmail.com
Arkadiusz Zieliński
mos.zielinski@gmail.com



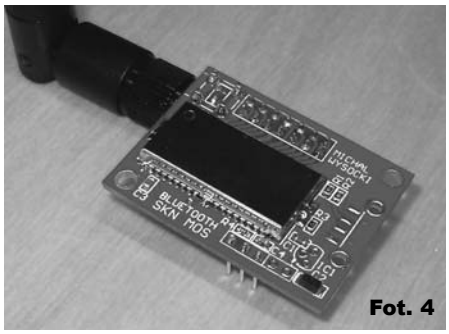
Rys.1

z dławikiem (L2), szybką diodą Schottky'ego (D2) oraz kondensatorami (C21–C24, C26). Więcej informacji – w nocie aplikacyjnej. Zachęcamy Czytelników do stosowania tego typu układów przetwornic – są bardzo proste w konstrukcji, a przy tym pozwalają na zaoszczędzenie dużych ilości energii i pozbycie się kłopotliwych radiatorów.

Napięcie 3,3V wytwarzane jest z 5V przez typowy stabilizator LF33 (U8). Diody LED D7 i D8 sygnalizują poprawne napięcia na liniach zasilania 12V i 5V. Akumulator został podłączony do złącza CON8, a złącze CON10 służy do wygodnego podłączenia ładowarki. Przełącznik S1 pozwala na odłączenie zasilania od układu i ładowanie akumulatora.

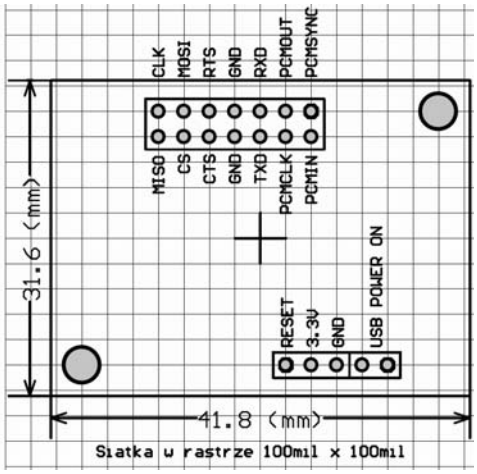
Bluetooth

MW: Standard Bluetooth jest bardzo popularnym łączem radiowym, pozwalającym na przesyłanie informacji na niewielkie odległości (od 10 do 100m). Pierwotnie został on stworzony jako dodatkowy sposób komunikacji pomiędzy telefonami komórkowymi (przesyłanie wizytówek, zdjęć oraz innych danych) w zasięgu jednego pomieszczenia oraz do podłączenia bezprzewodowych zestawów słuchawkowych z telefonem. Obecnie powstały nowsze wersje protokołu pozwalające na przesyłanie danych z większą prędkością (do 3Mbps) oraz na większe odległości (do 200m). Bluetooth jest dziś wykorzystywany w bardzo wielu dziedzinach elektroniki. Standard pozwala na przesyłanie różnorodnych typów danych: plików, strumieni audio, danych sieciowych TCP/IP, tworzenie bezprzewodowych łączy RS232 itp. Szczególnie ostatnia opcja jest dla nas, elektroników, bardzo kusząca. Możliwość zestawienia zdalnego, bezprzewodowego łącza RS232 pomiędzy komputerem a urządzeniem stwarza ogromne możliwości. Dodatkowym atutem standardu jest fakt, iż niższe warstwy protokołu Bluetooth odpowiadają za poprawność transmisji, dlatego możemy mieć pewność, że przesyłane dane zawsze trafią na drugi koniec łącza. Zwalnia to konstruktora z konieczności tworzenia własnych protokołów kodowania danych i korekcji błędów. W Voyagerze wykorzystaliśmy popularny w Polsce moduł Bluetooth BT222 firmy Rayson, który oferuje gotowy wirtualny port RS232 i może być konfigurowany za pomocą komend AT. Na **fotografii 4** można zobaczyć moduł przylutowany do płytki – przejściówki, wraz z typową tanią anteną od kart bezprzewodowych WiFi, pracujących na tej samej częstotliwości co Bluetooth (2,4GHz). **Rysunek 2** przedstawia zarys płytki modułu z opisem wyprowadzeń. Moduł może pracować jako master, czyli inicjować połączenia z innymi modułami, oraz jako slave (w tym trybie można mu przypisać unikalny adres MAC mastera, z którym będzie mógł nawiązać połączenie). Pracuje on w tzw. klasie I, czyli oferuje zasięg do 100...150m. Niestety, nie ma róży bez kolców. Pomimo iż nota aplikacyjna dumnie informuje o dostępnej prędkości do 3Mbps oraz wszystkich możliwych interfejsach, jakie moduł ma (USB, SPI, PCM, kodek audio, porty I/O, RS232), to tak naprawdę jedyne, co mamy do dyspozycji, to port COM o prędkości maksymalnej ok. 150Kbps (realnie 115Kbps). Wynika to z faktu, iż sprzedawane w Polsce moduły mają wewnętrzne oprogramowanie oferujące wyłącznie profil wirtualnego COM-a i niewykorzystujące wszystkich dostępnych możliwości. Producent użytego chipsetu pilnie strzeże innych wersji firmware'u i udostępnia je wyłącznie dużym producentom gotowych modułów, takim jak firma Rayson. Ograniczenie prędkości portu COM wynika z dwóch faktów. Po pierwsze, poprzez radio, poza danymi z linii RXD i TXD, przesyłany jest także stan wszystkich linii kontrolnych modemu (DTR, RTS, CTS etc.). Po drugie, łącze RS232 „obudowane" jest protokołem niższych warstw zajmujących się kontrolą, korekcją błędów i utrzymywaniem łącza. Wszystko to ogranicza realną przepustowość połączenia. Jednak do zdalnego sterowania robotem oraz przesyłania pojedynczych zdjęć z kamery jest to wystarczająca prędkość.



Fot. 4

Rys. 2



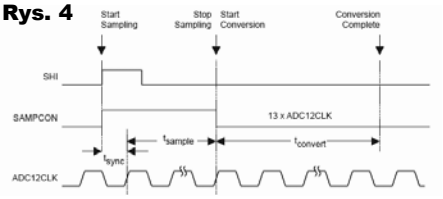
Jednostka centralna

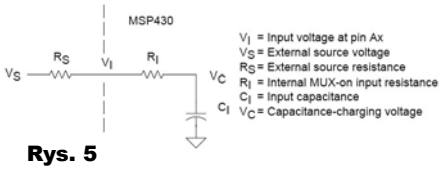
AZ: Z uwagi na wymagania stawiane przez digitalizer obrazu, mikroprocesor jest „napędzany" kwarcem 8MHz (**rysunek 3**) i ani przez chwilę nie przechodzi do trybu uśpienia. Na schemacie można wyróżnić złącza do podłączenia modułu Bluetooth, dwóch serwomechanizmów oraz reflektorów z diodami LED. Wolne wyprowadzenia w złączu JTAG (które służy do programowania mikrokontrolera i debugowania programu), zostały zagospodarowane w celu umożliwienia późniejszej rozbudowy układu. Wyprowadziliśmy tam (czerwona ramka na schemacie) trzy uniwersalne linie cyfrowe IO oraz dwa kanały przetwornika ADC. Linie mikrokontrolera opisane przedrostkiem P_ stanowią interfejs do obsługi digitalizera.

Przetwornik ADC

AZ: W układzie wykorzystano cztery kanały przetwornika, z czego pierwszy mierzy napięcie na akumulatorze (ADC0), drugi mierzy napięcie z wbudowanego w MSP430 czujnika temperatury, a dwa pozostałe zostały przewidziane do przyszłych zastosowań. Przetwornik pracuje w trybie „Pulse Sample Mode" (**rysunek 4**), co oznacza, że ustawienie flagi SHI (w naszym przypadku dzieje się to progra-

Rys. 4





Rys. 5

mowo), wyzwala wewnętrzny timer, który automatycznie odlicza czas zbierania próbki sygnału dla wybranego kanału. Przetwornik jest taktowany sygnałem z linii ACLK oraz włączony jest wewnętrzny dzielnik /8, co daje sygnał taktowania przetwornika ADC równy 1MHz. Jeden z rejestrów przetwornika pozwala wybrać, ile cykli z sygnału zegarowego ADC12CLK będzie przeznaczonych na próbkowanie. W spoczynku wejścia przetwornika znajdują się w stanie wysokiej impedancji. Gdy flaga SAMPCON zostaje ustawiona, wybrane wejście przetwornika można zasymulować jako filtr dolnoprzepustowy (**rysunek 5**), gdzie Ri to rezystancja zastępcza wewnętrznego multipleksera, a Rs to rezystancja źródła napięcia mierzonego. Karta katalogowa podaje wzór na minimalny czas potrzebny do naładowania pojemności Ci:

$t_{SAMPLE} > (Rs + Ri * \ln(213) * Ci + 800ns$

Kanał pierwszy przetwornika w robocie jest obciążony dzielnikiem napięcia (rezystory R13 i R14 na rysunku 1), którego rezystancja zastępcza Rs równa jest ok. 1kΩ. Po podstawieniu do wzoru otrzymujemy: $t_{SAMPLE} >$ 1,8us. Ponieważ jednak wewnętrzny czujnik temperatury wymaga czasu co najmniej 30us, dlatego wybrałem bezpieczną wartość 32 cykle, czyli t_{SAMPLE} = 32us. Pomiar na wybranym kanale jest przeprowadzany jednorazowo na żądanie operatora (aplikacja na komputerze PC). Przetwornik wykorzystuje tylko jeden rejestr pamięci – ADC12MEM0, w którym umieszczany jest wynik pomiaru. Po zakończeniu pomiaru przetwornik zgłasza przerwanie. Funkcja obsługująca przerwanie (**listing 1**) ustawia flagę, która zezwala na wysłanie wyniku pomiaru przez łącze szeregowe (odbywa się to w pętli głównej programu).

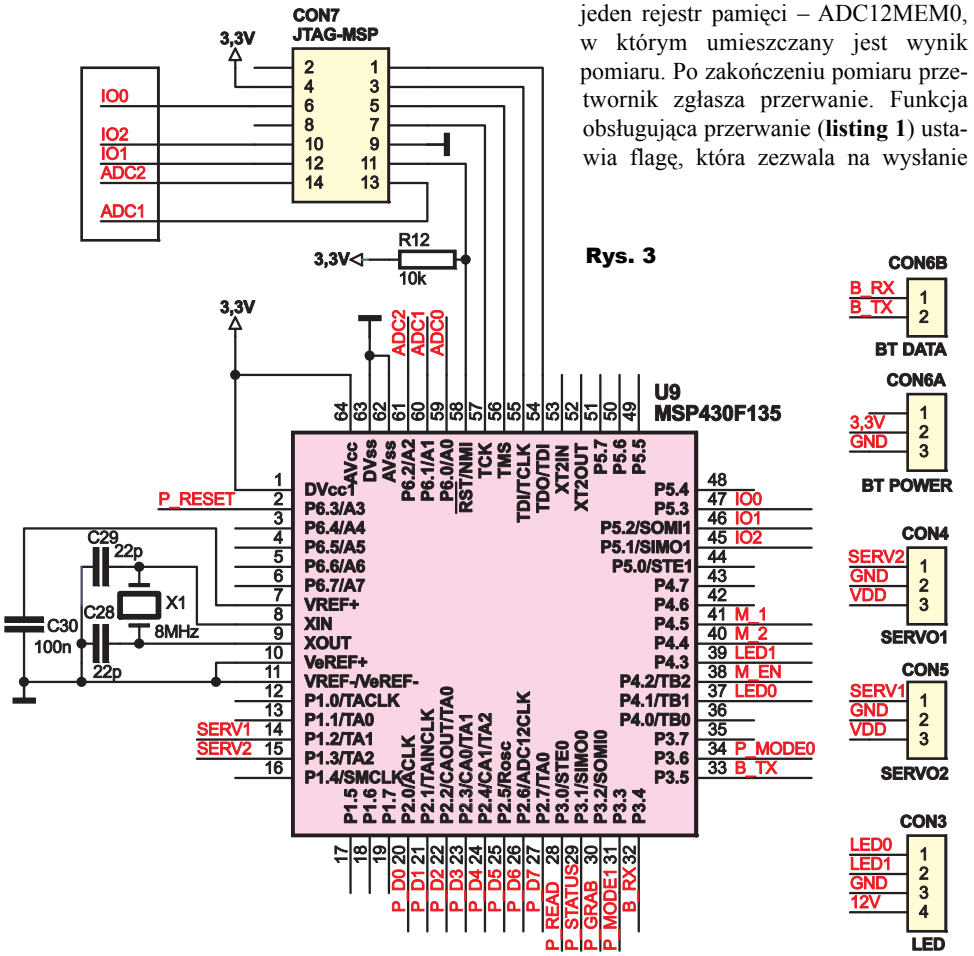
```
void get_ADC (char channel)
{
  //jeśli wynik poprzedniego pomiaru został wysłany to...
  if (send_adc_result_enable == false)
  {
    //wyłącz ADC, tylko wtedy można edytować rejestr ADC12MCTLx
        ADC12CTL0 &=~ ENC;

    switch(channel)
    {
                case 0: ADC12MCTL0 = SREF_1 + INCH_0; break;
                //kanał A0, wewnętrzne źródło referencyjne: Vr=1.5V
                case 1: ADC12MCTL0 = SREF_1 + INCH_10; break;
                //kanał A10 - pomiar temperatury, wewnętrzne źródło referencyjne: Vr=1.5V
                ...
                default: ADC12MCTL0 = SREF_1 + INCH_0;
    }

    //włącz przerwanie ADC12IFG.0
        ADC12IE = 0x01;
        //zezwól na konwersję
    ADC12CTL0 |= ENC;
    //Uruchom pomiar. Przetwornik zgłosi przerwanie, gdy pomiar zostanie zakończony.
        ADC12CTL0 |= ADC12SC;
  }
}

//przerwanie z przetwornika ADC
#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR (void)
{
  //wyłącz przerwanie z ADC
  ADC12IE = 0x00;
  //flaga zezwala na wysłanie wyniku pomiaru w głównej pętli programu
  send_adc_result_enable = true;
}
```

Listing 1



Rys. 3

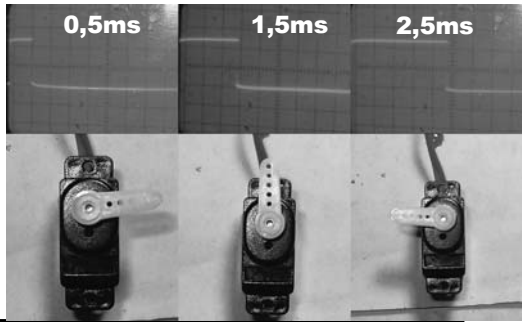
Serwomechanizmy

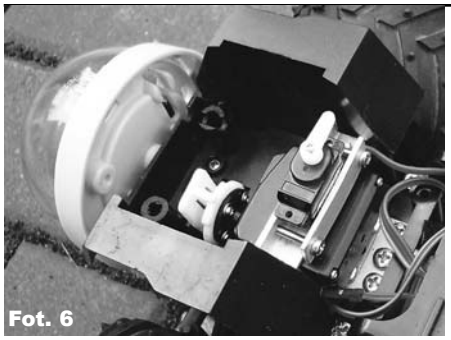
AZ: Typowy serwomechanizm modelarski składa się z:

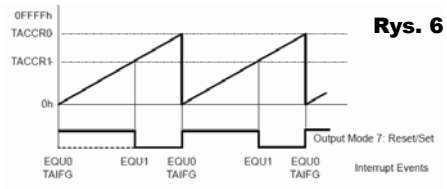
– wydajnego silnika prądu stałego,
– przekładni mechanicznej zamieniającej wysokie obroty silnika na duży moment obrotowy,
– potencjometru osadzonego na wale napędowym,
– układu elektronicznego, który porównuje napięcie z potencjometru z napięciem uzyskanym po przetworzeniu sygnału z układu sterującego, i tak steruje silnikiem, by oba napięcia były równe.

Serwo jest sterowane za pomocą impulsów o czasie trwania od 1ms do 2ms, co zapewnia obrót wału o 60°. Tanie serwa nie są zbyt restrykcyjne co do czasu trwania impulsów, co w praktyce pozwala na obrót wału o ponad 180°. Impulsy sterujące są wysyłane co 20ms. Tak duże przerwy pomiędzy kolejnymi impulsami pozwalają aparaturze nadawczej obsłużyć kolejne serwomechanizmy (patrz: „Koder i dekoder sterowania w systemie proporcjonalnym" EdW 8/98). Zasadę działania serwa można zobaczyć na **fotografii 5**, która

Fot. 5




Fot. 6

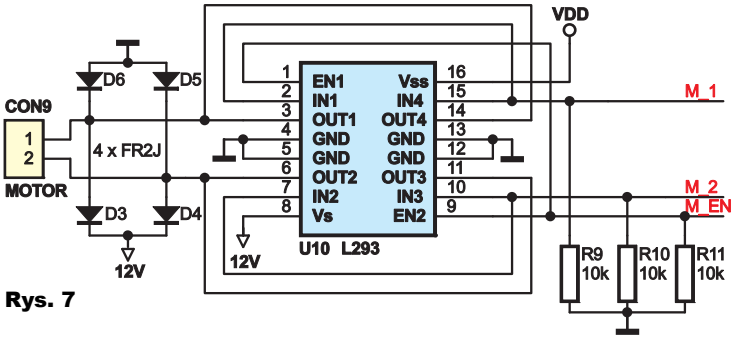


Rys. 6

przedstawia ekran oscyloskopu wyświetlającego impuls sterujący oraz odpowiadające mu położenie orczyka serwomechanizmu.

Nasz robot został wyposażony w dwa serwomechanizmy. Pierwszy (duży) steruje układem skrętnym, drugi (mniejszy) odpowiada za wychylenie kamery. Sposób ich montażu przedstawia **fotografia 6**. Widać na niej otwór wycięty w obudowie kamery, przez który będzie przechodzić cięgno połączone z orczykiem mniejszego serwa, co pozwali na sterowanie wychyleniem kamery w pionie (góra – dół). Do sterowania serwami został wykorzystany TimerA pracujący w roli generatora PWM zliczającego w górę do wartości zapisanej w rejestrze CCR0 (**rysunek 6**). TimerA jest taktowany sygnałem zegarowym 8MHz, przy czym jest włączony dzielnik /8, tak więc zlicza on z częstotliwością 1MHz. Timer ma trzy rejestry CCRx. Pierwszy z nich (CCR0) możemy wykorzystać do określenia górnej granicy zliczania. W takim przypadku mamy do dyspozycji dwa generatory PWM (rejestry CCR1, 2). Jeśli do rejestru CCR0 wpiszemy wartość 10000, da nam to okres generowanego przebiegu:
T = 1MHz/10000 = 10ms.
Jeśli teraz do rejestru CCR1 (CCR2) wpiszemy liczbę z zakresu 500...3000, uzyskamy na wyjściu TA1 (TA2) mikrokontrolera impulsy o czasie trwania 0,5...3ms, odpowiednie do sterowania serwomechanizmami.

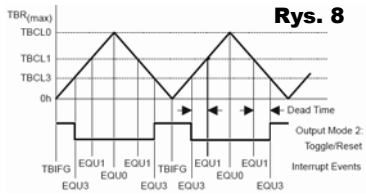
Serwo po zamontowaniu już w docelowym miejscu wymagają eksperymentalnego dobrania minimalnego i maksymalnego czasu trwania impulsów sterujących, tak by orczyk serwa operował w bezpiecznym zakresie (np. kamera może się odchylać tylko w pewnym zakresie). Funkcja sterująca (**listing 2**) przyjmuje jako argumenty dwie zmienne: numer serwa (0 lub 1) oraz wychylenie orczyka (znormalizowana wartość z zakresu 0x01..0x0F). W funkcji tej znajduje się pętla, która oblicza właściwy czas trwania impulsu sterującego, mając dane minimalny czas trwania impulsu i krok, które zostały zdefiniowane w programie dla obu serw w postaci stałych. Czyni to funkcję bardziej uniwersalną.



Rys. 7

Napęd

AZ: Silnik użyty w robocie został wraz z przekładnią pozyskany ze starej zabawki. Jest to silnik prądu stałego, o napięciu znamionowym 12V. Jego obroty są regulowane za pomocą PWM-a i popularnego układu L293, zawierającego w swojej strukturze cztery drivery, każdy o wydajności ±1A (prąd ciągły) oraz ±2A prąd w impulsie. Układ może pracować jako dwa mostki H do sterowania dwóch silników lub tak jak my zrobiliśmy, kanały można połączyć równolegle po dwa, dla zwiększenia wydajności (nota katalogowa dopuszcza taką konfigurację). **Rysunek 7** przedstawia schemat drivera dla silnika. Diody D3...D6 zabezpieczają przed przepięciami indukowanymi w silniku. Użyte diody FR2J to szybkie diody prostownicze o ciągłym prądzie pracy do 2A (w impulsie znacznie więcej). Podanie stanu wysokiego na wejście M_1 lub M_2 powoduje obroty silnika w lewo lub w prawo, gdy wejście zezwalające M_EN jest stanie wysokim – to wejście jest właśnie kluczowane przebiegiem PWM. Do jego wytworzenia został wykorzystany TimerB. Jest on taktowany sygnałem z linii zegarowej ACLK o częstotliwości 8MHz, przy czym jest włączony dzielnik /2. Timer pracuje w trybie up/down (**rysunek 8**), co oznacza, że zlicza najpierw w górę, do wartości zapisanej w rejestrze TBCL0, a następnie zlicza w dół do zera. W tym czasie porównuje zawartość licznika z rejestrami TBCL1 oraz TBCL2 i w razie wystąpienia zgodności, zmienia stan wyjścia TB1 (lub TB2) mikrokontrolera na przeciwny. Jest to tak zwany tryb poprawnej fazy, zalecany do sterowania silnikami. Po wpisaniu do rejestru TBCL0 wartości 127, daje nam to częstotliwość generatora PWM: f = 4MHz/(2*128) = 15,625kHz. Drugi generator PWM (wyjście TB2) został wykorzystany do sterowania jasnością reflektorów (białe diody LED).



Rys. 8

```
void set_servo (char channel, char position)
{
        //początkowe wartości dla generatora PWM:
        int PWM_servo1 = MIN_SERVO1;
        int PWM_servo2 = MIN_SERVO2;
        //zmień wychylenie pierwszego serwa
        if (channel == 0)
        {
                //wylicz długość impulsu dla generatora PWM
                for (ind=1; ind <= position; ind++)
                        PWM_servo1 += STEP_SERVO1;
                //zaktualizuj PWM
                CCR1 = PWM_servo1;
  }
        ...
}
```
Listing 2

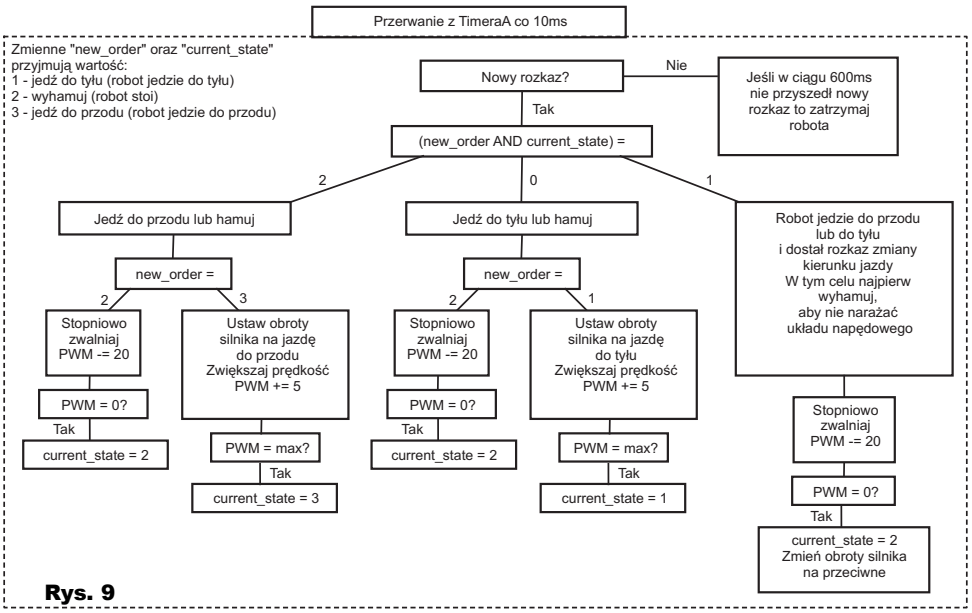
Algorytm sterowania silnikiem

AZ: TimerA (wytwarzający impulsy dla serwomechanizmów) generuje przerwania co 10ms. W przerwaniu tym zostało zaimplementowane sterowanie prędkością silnika. W przypadku wydania rozkazu jazdy do przodu algorytm stopniowo rozpędza silnik, aż do uzyskania zadanej prędkości. Natomiast w przypadku rozkazu zatrzymania (czy po prostu „puszczenia gazu") silnik robota jest stopniowo zatrzymywany. Takie sterowanie silnikiem jest bardziej intuicyjne i nie naraża układu napędowego na przeciążenia. Graf tego algorytmu można zobaczyć na **rysunku 9**. W pierwszej chwili algorytm może wydawać się zbyt rozbudowany do umieszczenia go w przerwaniu. W rzeczywistości jest dosyć szybki – przy każdym wywołaniu przerwania wykonywana jest tylko jedna gałąź drzewa (a po asemblacji jedna gałąź to zaledwie kilka poleceń).

Procedura obsługi silnika została umieszczona w przerwaniu ze względów bezpieczeństwa – zapewnia to priorytet w obsłudze silnika. Dodatkowo, jeśli w ciągu 600ms nie przyszedł nowy rozkaz, to procedura zatrzymuje silnik, tak aby w przypadku utraty łączności z nadajnikiem uniknąć ewentualnego wypadku.

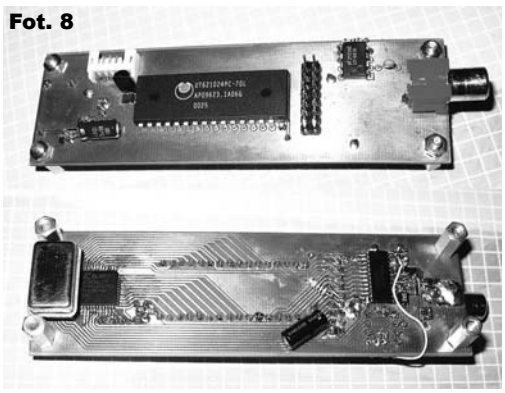
Kamera

MW: Kamera pokładowa to najbardziej nietypowy elementem naszego robota. Nie mamy tutaj do czynienia z tzw. wideosenderem, czyli nadajnikiem transmitującym analogowy sygnał wideo, pracującym na częstotliwości 2,4GHz. Co prawda takie rozwiązanie pozwala na łatwą i wygodną transmisję kolorowego obrazu, lecz wymaga po drugiej stronie dużego pudełka z odbiornikiem oraz karty telewizyjnej w komputerze. Jest to sprzeczne z naszymi założeniami – robot od początku miał być sterowany z małego przenośnego laptopa. Dlatego postanowiliśmy stworzyć własną „kartę telewizyjną", a mówiąc ściślej tzw. frame-grabber – „przechwytywacz" ramek.



Rys. 9

Fot. 7



Fot. 8

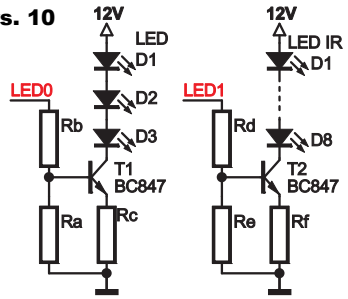
Jest to układ, który digitalizuje i rejestruje w pamięci RAM jedną klatkę ze standardowego obrazu PAL, generowanego przez kamerę. Następnie główny mikroprocesor może odczytać te dane i w dowolny sposób przesłać je do komputera. Układ będzie dokładnie opisany w dalszej części artykułu, a na razie napiszę kilka słów na temat samej kamery.

Dużo czasu zajęło poszukiwanie odpowiedniej kamery, najlepiej z cyfrowym wyjściem. Niestety tego typu wynalazki są trudno osiągalne i dość drogie. Co prawda można by spróbować wykorzystać moduł kamery np. z telefonu komórkowego, lecz mają one podstawową wadę – „wypluwają" dane z tak dużą prędkością, że nawet zastosowany tutaj PLD i pamięć nie poradziłyby sobie z tym zadaniem. Ostatecznie wybraliśmy łatwą do kupienia w Internecie kamerę typu „kopułowego", stosowaną powszechnie w prostych systemach monitoringu. Kamerę zamontowaną już w robocie można zobaczyć na **fotografii 7**. Jest to trochę lepsza kamera niż najtańsze modele dostępne na rynku, ponieważ posiada przetwornik CCD oraz tryb nocny – przy zbyt słabym oświetleniu kamera przełącza się z trybu kolorowego na czarno-biały o wielokrotnie większej czułości. Ponadto kamera nie ma w obiektywie filtru podczerwieni, przez co aż prosiła się o zamontowanie wokół niej podczerwonych diod LED i stworzenie prostego noktowizora.

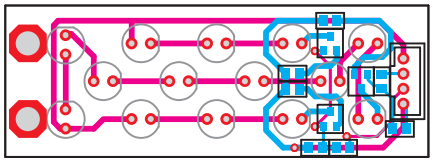
Reflektory

MW: Do oświetlenia drogi przed robotem przewidzieliśmy dwa reflektory zawierające po jednej sekcji ośmiu szeregowo podłączonych diod LED IR (co daje spadek napięcia: 8 x 1,2V = ok. 9,6V) oraz po dwie sekcje trzech szeregowo połączonych diod LED białych, ultrajasnych (spadek napięcia: 3 x 3V = ok. 9V). Jak widać, liczba diod w gałęzi została tak dobrana, aby suma spadków napięć na diodach była poniżej 10V, czyli przewidywanego minimalnego napięcia na akumulatorze. Zmiany napięcia na nim są na tyle duże, że niemożliwe było zasilenie diod tylko przez rezystory ograniczające prąd (wahania prądu mogłyby sięgać 500%). Zamiast tego zastosowaliśmy bardzo proste źródła prądowe w każdej gałęzi diod – schemat reflektora można zobaczyć na **rysunku 10**. Nadmiarowe zmiany napięcia zasilania odkładają się na tranzystorze, który utrzymuje zawsze stały prąd przepływający przed diody – takie źródło może być sterowanie wprost z portu mikroprocesora pracującego w standardzie napięć 3,3V. **Rysunek 11** przedstawia płytkę reflektora. Płytka była rysowana bez schematu, stąd brakuje na niej oznaczeń elementów. Układ jest jednak na tyle prosty, że nie nastręcza to problemów z montażem.

Rys. 10



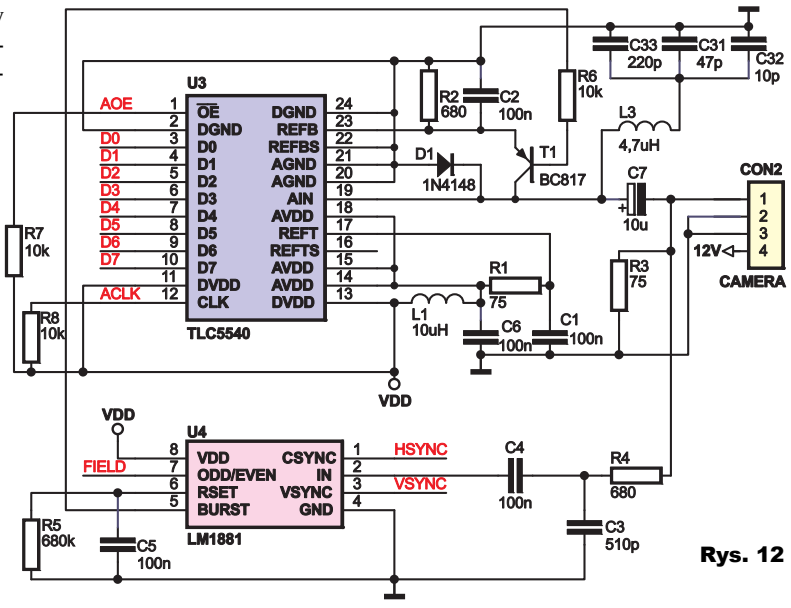
Rys. 11



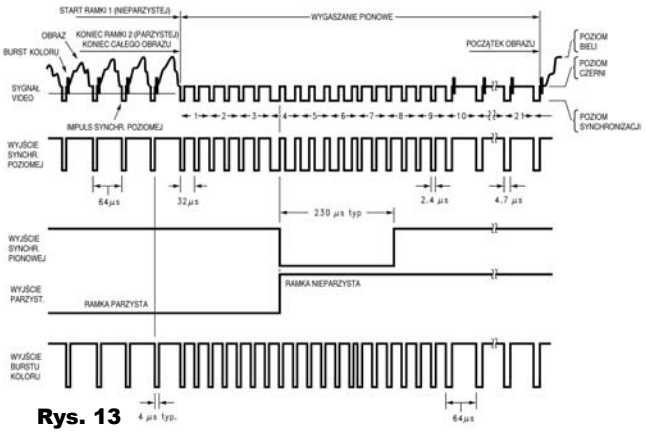
Digitalizer

MW: Układ digitalizera powstał wcześniej niż sam robot. Prototyp można zobaczyć na **fotografii 8**. Układ składa się z dwóch części: analogowej (układ wejściowy, przetwornik ADC oraz separator synchronizacji) i cyfrowej (układ logiczny CPLD, pamięć RAM, generator kwarcowy i bufor). W ostatecznej wersji, płytka główna Voyagera rozrosła się wokół pierwotnego projektu rejestratora.

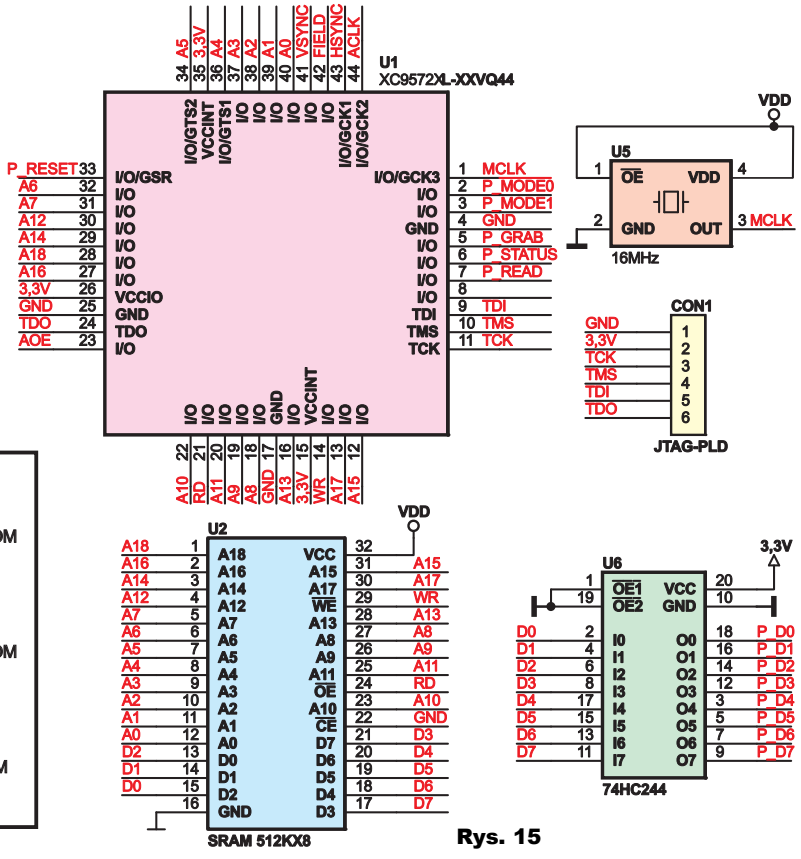
Schemat części analogowej można zobaczyć na **rysunku 12**. Jej bardzo istotnym elementem jest separator synchronizacji – układ



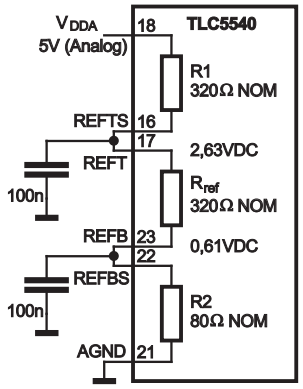
Rys. 12



Rys. 13



Rys. 15



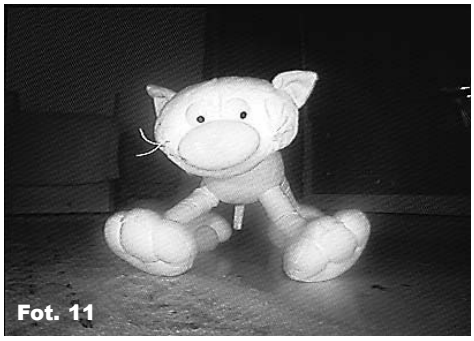
Rys. 14

LM1881 (U4). Na **rysunku 13** można zobaczyć przebiegi występujące na wyjściach układu. Rysunek ten również bardzo dobrze pokazuje szczegóły analogowego sygnału video w standardzie PAL. Typowy sygnał telewizyjny składa się z serii „półobrazów", a każdy z nich składa się z 312 linii i trwa 20ms. Półobrazy oznaczone są jako parzyste i nieparzyste, ponieważ linie z jednego półobrazu wyświetlane są na przemian z liniami drugiego, tworząc pełną ramkę obrazu mającą 625 linii. W naszym układzie będziemy rejestrować tylko jeden półobraz, co da rozdzielczość pionową ok. 288 pikseli. Przed każdym półobrazem występuje seria impulsów synchronizacji pionowej (VSYNC) informującej odbiornik o początku obrazu. W zależności od kształtu impulsów, telewizor jest w stanie odróżnić półobrazy od siebie. Każda z linii trwa równo 64us i posiada własny 4,7us impuls synchronizacji poziomej (HSYNC) wyznaczającej lewą krawędź obrazu. Część widzialna obrazu to ok. 52us, a 4us przednią zajmuje tzw. BURST – krótki impuls, który pomaga w synchronizowaniu układu dekodowania koloru – jest to kwestia bardzo złożona i nieistotna w tej aplikacji, ponieważ operować będziemy tylko na obrazie czarno-białym. Warto jeszcze wspomnieć o poziomach napięć sygnału. Jako punktu odniesienia używa się poziomu czerni, 0V. Kolor biały odpowiada napięciu ok. +0,7V, a impulsy synchronizacji mają potencjał ok. –0,3V. Jak widać, układ LM1881 potrafi wyłuskać z sygnału wszystkie interesujące nas informacje: impulsy synchronizacji pionowej, poziomej, typ ramki (parzysta/nieparzysta) oraz momenty występowania impulsów BURST, i dostarczyć je w postaci sygnałów cyfrowych TTL. Kamera podłączona jest do złącza CON2, gdzie dostarcza sygnał i skąd czerpie zasilanie 12V. Na samym początku sygnał trafia na rezystor terminujący R3 o wartości 75Ω, zapewniający dopasowanie impedancyjne wejścia. Układ LM1881 dołączony jest poprzez filtr dolnoprzepustowy (R4 i C3) oraz własny kondensator odcinający składową stałą C4. Kondensator C7 odcina składową stałą sygnału, który następnie trafia na elementy L3, C31–C33, będące filtrem, usuwającym zakodowany w obrazie sygnał koloru (bez tych elementów, na obrazie byłyby widoczne skośne paski na jaskrawo kolorowych płaszczyznach). Układ TLC5540 (U3) jest szybkim przetwornikiem ADC typu flash, pozwalającym na przetwarzanie z częstotliwością do 40MHz. Układ jest zasilany z napięcia 5V, a napięcia referencyjne dla mierzonego sygnału ustalane są poprzez wbudowany lub zewnętrzny dzielnik rezystancyjny (**rysunek 14**). Domyślne napięcia nie były odpowiednie do aplikacji, dlatego zamiast wewnętrznych rezystorów użyte zostały zewnętrzne R1 i R2, tworząc zakres pomiarowy od 2V do 2,7V. Jednak wciąż nie

odpowiada to poziomom napięć, o których była mowa wcześniej. Z pomocą przychodzi obszar impulsu BURST, który ma także inne zastosowanie – ponieważ składowa stała tego impulsu jest zerowa, możemy w tym obszarze przyjąć, że sygnał ma napięcie odpowiadające poziomowi czerni w obrazie. Jest to bardzo istotne, ponieważ poziomy napięć między nadajnikiem a odbiornikiem obrazu potrafią się po prostu rozjechać. Dlatego zarówno w odbiornikach, jak i nadajnikach sygnału wideo stosuje się kondensatory odcinające składową stałą, a układy muszą same dostosować się do referencyjnego poziomu czerni – jest to tzw. clamping. W naszym układzie jest to zrealizowane w banalny sposób za pomocą tranzystora T1, sterowanego bezpośrednio z wyjścia BURST układu LM1881. Emiter tranzystora podłączony jest do dolnego napięcia referencyjnego 2V, a kolektor do sygnału za kondensatorem odcinającym składową stała (C7). W momencie, gdy LM1881 wykryje obecność obszaru BURST w sygnale, tranzystor zaczyna przewodzić, ładując kondensator C7. Czas 4us jest wystarczający, aby kondensator został w pełni naładowany. W tym momencie, niezależnie od tego, jaką wartość ma poziom czerni, na wejściu przetwornika jest 2V, odpowiadające dolnemu zakresowi przetwarzania. Naładowany kondensator utrzymuje różnicę napięć, lecz przewodzi zmiany napięcia, będące właściwym sygnałem następującym po Burście. Czas trwania linii jest dostatecznie krótki, aby napięcie na kondensatorze utrzymało się prawie bez zmian. W każdym kolejnym cyklu proces się powtarza, automatycznie dostosowując się do możliwych zmian sygnału wejściowego. Ten prosty trik jest powszechnie stosowany także np. w przypadku sygnału VGA z komputerowej karty graficznej. Dioda D1 zabezpiecza wejście przetwornika przed wystąpieniem ujemnych napięć, gdy kondensator C7 nie jest wstępnie naładowany. Elementy C1, C2, C6, L1 zapewniają filtrację i odprzężenie zasilania i napięć referencyjnych przetwornika. Wynik pomiaru przetwornika dostępny jest na liniach D0-D7, a linia AOE pozwala na przełączenie wyjść w stan wysokiej impedancji. Ponieważ jest to przetwornik typu flash, wymaga on ciągłego sygnału taktującego ACLK (nie może być to pojedynczy impuls, ponieważ pomiędzy próbką aktualnie mierzoną a wynikiem dostępnym na D0-D7 istnieje przesunięcie o 4 pomiary, które są w tym czasie wewnętrznie przetwarzane w układzie). Rezystory podciągające R8 i R7 są konieczne, ponieważ układ PLD, który steruje liniami ACLK i AOE, ma poziomy wyjściowe 3,3V, a jak się okazało, minimalnym napięciem dla logicznej jedynki w ADC jest wartość ok. 4V. Jest to dość nietypowe, ponieważ mamy do czynienia z układem CMOS, gdzie poziom przełączania 0/1 powinien wynosić ½ VCC, czyli ok. 2,5V. Linie sterowane są przez wyjścia typu otwarty kolektor w PLD, a rezystory zapewniają poprawną „jedynkę".

Fot. 9 Autorzy projektu: Arek Zieliński (z lewej) i Michał Wysocki

Fot. 10



Fot. 11

Sercem rejestratora jest układ logiki programowalnej CPLD firmy Xilinx XC9572XL (**rysunek 15**), mogący pomieścić do 72 przerzutników oraz setki bramek logicznych. Użycie układu programowalnego wydawało się na miejscu, ponieważ ten sam układ zrealizowany na typowych kościach TTL/CMOS zabierałby dużo miejsca i byłby dość skomplikowany. Do przechowywania obrazu z kamery konieczna była pamięć SRAM o rozmiarze co najmniej 128KB. Rozdzielczość rejestrowanego obrazu to 416x288 pikseli. Rozdzielczość pozioma nie odpowiada co prawda klasycznym proporcjom obrazu 4:3, lecz jest akceptowalna. Wynika ona z przyjętej w układzie częstotliwości próbkowania pikseli 8MHz, dając 416 próbek w ciągu 52us. Linie adresowe oraz sterujące pamięci podłączone są do układu CPLD, lecz linie danych D0–D7 tworzą magistralę, do której, poza pamięcią, mamy podłączony przetwornik ADC, oraz bufor, który zmienia poziomy napięć z 5V na 3,3V i przesyła dane do głównego procesora. Jak widać, układ PLD nie ingeruje w rejestrowane dane, lecz pełni tylko funkcje sterującą. Należałoby w tym momencie napisać kilka słów na temat działania pamięci SRAM. W trybie odczytu WR = 1, RD = 0, pamięć w

sposób ciągły wystawia na liniach D0–D7 zawartość komórki spod adresu, który jest aktualnie podany na A0–A16. W trybie zapisu (RD = 1 oraz ujemny impuls na WR) zapisuje dane z linii D0–D7, będących wejściami, do komórki o adresie podanym na A0–A16. Stan niski na linii CE aktywuje operacje na pamięci, a stan wysoki sprawia, że pamięć ignoruje wszystkie pozostałe linie sterujące. Generator kwarcowy U5 wytwarza sygnał o częstotliwości 16MHz, niezbędny do taktowania wewnętrznej logiki PLD. Do układu dołączone są także sygnały z separatora synchronizacji: HSYNC, VSYNC oraz FIELD. Układ PLD jest zasilany napięciem 3,3V, lecz wszystkie jego piny akceptują poziomy logiczne 5V. Ponieważ linie pamięci nie mają tej właściwości, musi być ona zasilana z 5V tak jak przetwornik ADC. Złącze CON1 służy do podłączenia interfejsu JTAG, poprzez który można wgrać tzw. wsad układu PLD. Główny procesor steruje rejestratorem poprzez linie GRAB, READ, RESET oraz MODE0 i MODE1. Gdy procesor chce dokonać rejestracji ramki, ustawia stan wysoki na linii GRAB. Od tego momentu PLD czeka, aż pojawi się synchronizacja pionowa ramki nieparzystej i rozpoczyna rejestrację, sygnalizując to stanem wysokim na linii STATUS. Po zakończeniu wystawia na niej stan niski i przechodzi do trybu odczytu danych. Procesor musi ustawić stan wysoki na linii RESET i podać jeden impuls na linii READ w celu zresetowania licznika pamięci. Następnie podając kolejne impulsy na READ, zwiększa adres pamięci, każdorazowo odczytując bajt z D0-D7. W ten sposób można odczytać 288 linii po 416 pikseli każda. Linie MODE0 i MODE1 pozwalają na zmianę trybu odczytu, na co drugi piksel lub co drugą linie, dzięki czemu możemy od razu odczytać obraz o 2x niższej rozdzielczości 208x144.

Wsad układu PLD został stworzony w bezpłatnym środowisku ISE firmy Xilinx. Bezcelowe jest opisywanie tutaj środowiska programistycznego, ponieważ jest to dobry temat na całą serię artykułów, jednak postanowiłem przedstawić sam kod, napisany w języku Verilog. Jest to konkurencyjny dla języka VHDL sposób opisu układów logicznych. Można powiedzieć, iż istnieje święta wojna między zwolennikami Veriloga i VHDL-a – ja akurat należę do zwolenników tego pierwszego, głównie z powodu bardzo prostej i przejrzystej składni, szczególnie dla kogoś, kto zna i używa języka programowania C. Kod, wraz z bardzo obszernymi komentarzami, można znaleźć na **listingu 3** (można go ściągnąć z Elportalu). Zachęcam Czytelników nawet niebędących w ogóle w temacie, a posiadających podstawową wiedzę o układach logicznych, do przeanalizowania tego kodu. Warto tutaj dodać komentarz, iż tak naprawdę podczas rejestrowania obrazu układ w ogóle nie korzysta z sygnału synchronizacji poziomej HSYNC – jest ona używana tylko w parze z synchronizacją pionową VSYNC podczas pierwszego wyzwolenia. Pozwoliło to na znaczące uproszczenie logiki sterującej, a było możliwe dzięki temu, że sygnał pochodzący ze współczesnych źródeł jest bardzo dokładny pod względem czasowym, podobnie jak i generator 16MHz napędzający CPLD. Kolejne linie obrazu można rejestrować „ciurkiem" bez obawy o przesunięcia i rozsynchronizowanie.

Na sam koniec tego podrozdziału zostawiliśmy to, co najciekawsze, czyli efekt pracy digitalizera. **Fotografia 9** przedstawia autorów artykułu, natomiast **fotografia 10** prezentuje fotkę z wojaży po stancji. **Fotografia 11** została wykonana w całkowitej ciemności, z włączonymi reflektorami podczerwonymi.

ichał Wysocki,
mos.wysocki@gmail.com
Arkadiusz Zieliński
mos.zielinski@gmail.com

Ciąg dalszy w następnym numerze.