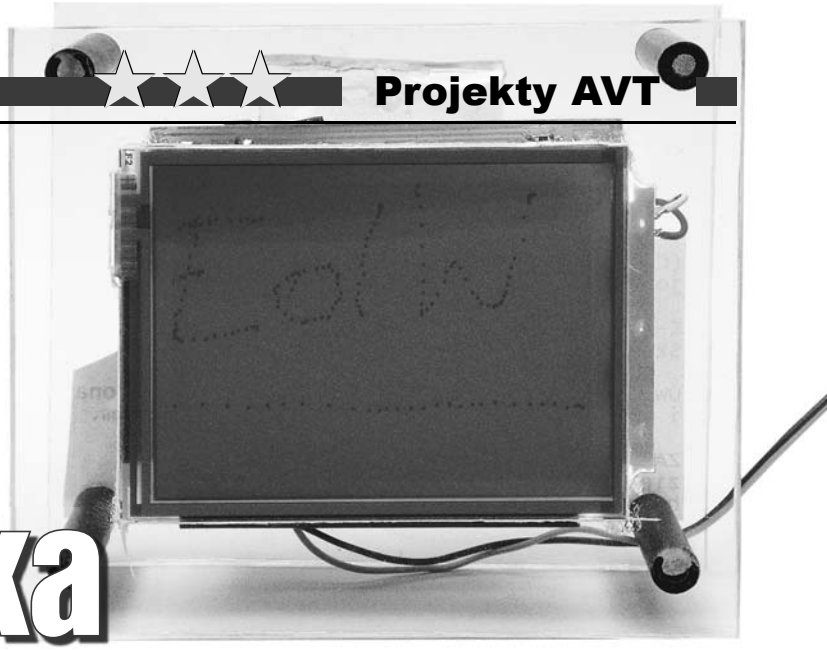


kit 2893 AVT

E-karteczka

Zapominanie o różnych drobiazgach, takich jak kupienie gazety, odpisanie na list czy wyniesienie śmieci, przytrafia się chyba każdemu. Sposobów na radzenie sobie z tym problemem jest wiele. Mój ulubiony polega na zapisywaniu krótkiej informacji na karteczce i przyczepianiu jej do lampki stojącej na biurku. Po pewnym czasie zbiera się takich karteczek sporo i trzeba pamiętać (znowu!) o ich sukcesywnym wyrzucaniu. Zainspirowany zadaniem 148 Szkoły Konstruktorów postanowiłem przygotować elektroniczną wersję takiej karteczki, której nie trzeba będzie wyrzucać. W założeniu miało to być urządzenie pozwalające robić odręczne notatki, wymazywać je, robić następne i tak dalej. Zadanie to spełnia ekran LCD wyposażony w panel dotykowy i w tym kierunku podążyły moje działania. Na następnych stronach przedstawiam sposób realizacji czegoś na kształt elektronicznego papieru – urządzenia, na ekranie którego można pisać zwykłym rysikiem i wielokrotnie to kasować. Możliwe jest również nanoszenie informacji palcem (niepraktyczne) lub innym spiczastym przedmiotem (śrubokręt albo wypisany długopis – grozi to jednak porysowaniem szybki).

Być może zastanawiasz się Czytelniku ile może kosztować taki „magiczny" wyświetlacz. Podczas realizacji tego projektu był on dostępny w jednym ze sklepów internetowych i kosztował w granicach 60zł. Pozostałe elementy wykorzystane w urządzeniu również są łatwo dostępne w polskich sklepach elektronicznych, co powinno pozwolić na zbudowanie e-karteczki wszystkim zainteresowanym. Na stronie Elportalu dostępny jest kod źródłowy oraz kod wynikowy, który można wgrać do mikrokontrolera bez żadnych modyfikacji.

Działanie urządzenia jest bardzo proste – przyłożenie rysika do ekranu spowoduje włączenie pod nim pikseli. Przesuwając rysik, można rysować linie, krzywe, kółka i wszelkie inne dziwaczne figury, które Czytelnik sobie zażyczy. Zwracam tylko uwagę, aby nie robić tego zbyt szybko, gdyż układ nie reaguje natychmiast – większa część mocy obliczeniowej idzie na obsługę wyświetlacza. Kiedy ekran będzie w całości zapełniony, wystarczy przytrzymać rysik w lewym, górnym rogu ekranu przez kilka chwil, aby go wyczyścić. Potem można dalej rysować... Uwaga, to wciąga :).

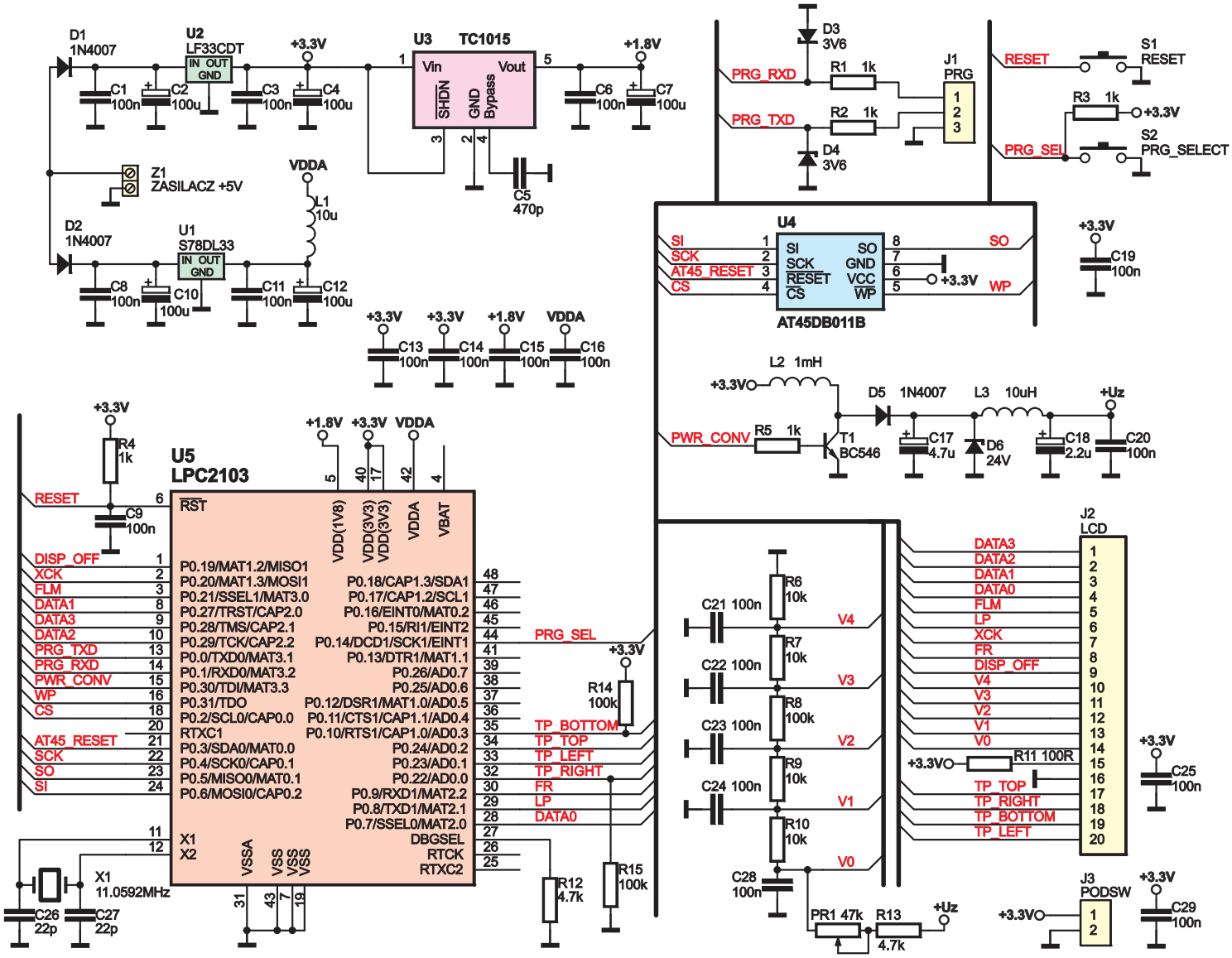
Opis układu

Schemat urządzenia został przedstawiony na **rysunku 1**. Najważniejszym elementem jest oczywiście sam wyświetlacz, który ma oznaczenie LGMJM160240A6WLW3-TP (wpisanie tego symbolu w Google pozwoli dotrzeć do sklepu). Ma on rozdzielczość 160x240 punktów, wbudowane podświetlanie białymi diodami LED oraz rezystancyjny panel dotykowy. Panel ten jest przymocowany do ekranu, a jego zasada działania opiera się na zmianie rezystancji, która jest zależna od miejsca przyłożenia rysika. Największą wadą omawianego panelu LCD jest brak kontrolera – wyświetlanie obrazków odbywa się w sposób multipleksowany, tzn. poprzez zapis kolejnych wierszy (szczegóły dalej). Prostą implikacją tego jest konieczność wygospodarowania pamięci obrazu. Przyjmując jednobitową paletę „kolorów", czyli piksel włączony lub wyłączony, otrzymujemy 160*240*1=38400 bitów. Po przeliczeniu na bajty wychodzi, że pamięć obrazu wymaga aż 4,8kB. Jest to wartość, która od razu dyskwalifikuje praktycznie wszystkie popularne mikrokontrolery AVR. W związku z tym zastosowano procesor z rdzeniem ARM, konkretnie dość popularny układ LPC2103. Ma on 8kB pamięci RAM, co pozwala bez trudu przechować cały obraz. Za tym wyborem idzie konieczność zapewnienia napięcia zasilającego 3,3V (stabilizator U2) oraz zasilania rdzenia, do czego wymagane jest napięcie 1,8V (układ U3). Pewne zdziwienie może budzić obecność jeszcze trzeciego stabilizatora – U1 (nawiasem mówiąc, do kupienia w tym samym sklepie co i wyświetlacz. Również kup tu złącze ZIF). Obsługa panelu dotykowego bazuje na pomiarach przetwornikiem analogowo-cyfrowym i od dokładności pomiaru zależy dokładność rysowania punktów na ekranie. Na otrzymywane wyniki wpływa m.in. dokładność napięcia odniesienia przetwornika (pin VDDA). Podanie tam napięcia z głównego stabilizatora mogłoby spowodować, że do układu ADC trafią wszystkie zaburzenia generowane przez pracujący procesor. Problem ten nie występuje właśnie w sytuacji, gdy te dwa napięcia pochodzą z różnych stabilizatorów. Wymagana duża dokładność pomiaru zdecydowała również o dodaniu dławika L1, którego zadaniem jest dodatkowe zmniejszenie zakłóceń.

Elementy R5, L2, T1, D5, C17 oraz D6 tworzą prostą przetwornicę podwyższającą napięcie zasilania do wartości 24V. Napięcie to jest wymagane przez sterownik wyświetlacza, a jego wartość decyduje m.in. o kontraście. Wymusza to zastosowanie możliwie stabilnego napięcia, aby uchronić wyświetlacz przed niepoprawnym wyświetlaniem obrazu. Przetwornice mają to do siebie, że generują dużo śmieci, stąd dodatkowy filtr złożony z elementów L3, C18, C20. Uzyskane w ten sposób napięcie +Uz trafia na dzielnik złożony z potencjometru PR1 i rezystorów R6...R10. Napięcia z poszczególnych sekcji dzielnika trafiają do wyświetlacza i decydują o kontraście i sposobie świecenia pikseli. Ponownie stabilność tego napięcia ma kluczowe znaczenie i jej zapewnienie zdecydowało o dołączeniu kondensatorów C21...C24 oraz C28. Rezystory R11 i R13 ograniczają maksymalny prąd, jaki może płynąć do wyświetlacza. Zapobiega to uszkodzeniu wyświetlacza w przypadku odłączenia napięcia 3,3V przed odłączeniem +Uz.

Złącze JP1 oraz przyciski S1 i S2 zostały przeznaczone do programowania procesora w systemie i kalibracji panelu dotykowego – szczegóły w sekcji poświęconej montażowi i uruchomieniu.

Na koniec jestem winien Czytelnikom pewne wyjaśnienie, konkretnie dotyczące pamięci zewnętrznej Flash. Niestety uruchomienie wyświetlacza okazało się zbyt pracochłonne i nie zostało czasu na



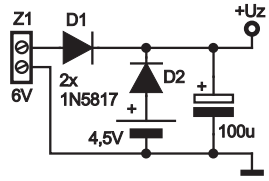
Rys. 1

zaimplementowanie obsługi pamięci Flash. W założeniach miała być ona wykorzystana do przechowywania obrazu i ochrony przed jego utratą w wyniku zaniku napięcia zasilania. Z tego względu NIE MA POTRZEBY lutowania układu U4. Oczywiście płytka została wykonana dużo wcześniej i postanowiłem jej nie zmieniać, gdyż funkcjonuje prawidłowo. Dla pełnego obrazu postanowiłem również pozostawić zewnętrzną pamięć Flash na schemacie. Czytelnicy, którzy chcą mieć możliwość podtrzymania pamięci, w przypadku problemów z zasilaniem mogą zastosować rozwiązanie z **rysunku 2**.

Oprogramowanie

Omawianie oprogramowania warto rozpocząć od przybliżenia idei sterowania wyświetlaczem LCD. Jest on zbudowany z segmentów w postaci pojedynczych pikseli. Każdy z nich składa się z dwóch elektrod, między którymi znajduje się kryształ. Mówiąc najprościej: przyłożenie do tych elektrod napięcia większego od progowego powoduje, że dany segment staje się widoczny (robi się czarny). Brak takiego napięcia sprawia, że piksel staje się niewidoczny i w efekcie w danym punkcie widoczne jest światło emitowane przez diody podświetlające. W rzeczywistości napięcie przyłożone do elektrod to przebieg zmienny, który nie może mieć składowej stałej. Składowa stała jest szkodliwa dla kryształów i powoduje degradację modułu LCD. Problem ten rozwiązuje się poprzez zmianę polaryzacji napięcia przyłożonego do elektrod. Ujmując rzecz obrazowo, można powiedzieć, że raz na górnej elektrodzie jest napięcia dodatnie, a raz masa, natomiast na dolnej – odwrotnie. Obserwując wtedy to napięcie względem dolnej elektrody, otrzymujemy przebieg prostokątny o amplitudzie od –V do +V, co po uśrednieniu daje zero (brak składowej stałej).

Rys. 2



Wysterowanie wyświetlacza sprowadza się do zapisu rejestru kolumn i rejestru wierszy, które mają postać rejestru przesuwającego – zapisywane dane są sukcesywnie przesuwane w takt sygnału zegarowego. Zapisanie zera powoduje, że elektroda będąca kolumną lub wierszem ma napięcie opowiadające napięciu masy. Zapisanie jedynki skutkuje podaniem jednego z napięć sterujących: V43, V5, V12, V0 lub V5, które są pobierane z dzielnika rezystorowego R6...R10. Które z tych napięć trafi na kolumnę, a które na wiersze, decydują stany logiczne obecne na wejściach FR, LP oraz DISP_OFF. Określają to dwie tabelki widoczne na **rysunkach 3** i **4**. Są one bardzo podobne, jednakże występują tu pewne różnice, konkretnie dla pozycji drugiej i czwartej w podanych tabelkach. Jeżeli w wybranych wierszach i kolumnach wpiszemy jedynkę, to do elektrody wiersza

zostanie dołączone napięcie V5, a do elektrody kolumny V0 (przy założeniu, że FR=L, LP=H, DISP_OFF=H). Gdyby teraz zmienić stan wejścia FR na przeciwny, to sytuacja się odwróci: elektroda kolumny będzie miała napięcie V5, a elektroda wiersza V0. Na przecięciu tych elektrod pojawi się napięcie prostokątne o amplitudzie 2*V0 i w efekcie włączy się stosowny piksel. Zakładając, że do kolumny zapiszemy jedynkę logiczną, natomiast w wierszu logiczne zero, na przecięciu tych dwóch elektrod piksel nie włączy się – napięcie V0 jest niewystarczające. Właśnie to jest kontrast wyświetlacza LCD – regulacja amplitudy V0. Im jest ona większa, tym ciemniejsze są włączone piksele, jednakże na ekranie pojawiają się „cienie", bo wyłączone piksele zaczynają się włączać – napięcie V0 zbliża się do wartości progowej, umożliwiającej włączenie segmentu. Dobrze wyregulowany kontrast ma dwie cechy: włączone piksele są dobrze widoczne, a wyłączone są niewidoczne.

Ta zasada leży u podstaw algorytmu wykorzystanego do sterowania LCD. Podając jedynkę na wybrany wiersz włączamy całą poziomą linię, jeżeli włączona będzie więcej niż jedna kolumna – zaświeci się kilka pikseli. Nie da się w ten sposób narysować, np. kwadratu, który nie byłby w środku wypełniony – pokazuje to w uproszczeniu **rysunek 5**.

Problem ten można obejść, włączając w danym momencie tylko jedną kolumnę, co spowoduje zaświecenie pikseli w jednej, pionowej linii. Po chwili zmieniamy napięcia na elektrodach wierszy i wyświetlamy kolejną linię. W taki sposób, linia po linii tworzymy obraz. Cała operacja musi być powtarzana cyklicznie – mamy tu do czynienia z odświeżaniem podobnym do tego spotykanego w monitorach komputerowych.

Znając już algorytm pracy, możemy się przyjrzeć konkretnej realizacji oprogramowania. W tym miejscu chciałbym zauważyć, że zastosowany mikrokontroler jest 32-bitowy i z tego względu pamięć obrazu jest zorganizowana w formie tablicy zmiennych 32-bitowych – *unsigned int*. Wyświetlacz ma 160 linii, co po podzieleniu przez 32 daje 5 – tyle komórek pamięci jest potrzebnych do przechowania informacji o pojedynczej linii obrazu. LCD ma 240 kolumn, stąd pamięć obrazu stanowi tablica *lcdMemory[240][5]* znajdująca się w pliku *panel.h*.

Zapis wierszy odbywa się w trybie równoległym, 4-bitowym. Zadanie to wykonuje funkcja *SetDataPort*, która pobiera zmienną 32-bitową, dzieli ją na fragmenty 4-bitowe i kolejno zapisuje do sterownika. Po każdym takim zapisie podawane jest zbocze opadające na wejściu XCK. Funkcję tę można było napisać trochę „sprytniej", np. wykorzystując pętlę *for*, jednakże wprowadzała ona zbyt duże opóźnienie i obraz widocznie migotał.

Funkcja *SetDataPort* jest wywoływana z funkcji *refresh*, która zajmuje się odświeżaniem całego obrazu. Aby obraz był widoczny, musi być ona wywoływana cyklicznie. Po zapisaniu wszystkich wierszy, na wejście LP podawane jest zbocze opadające. Ma ono dwa zadania – „zatrzasnąć" dane zapisane do wierszy oraz umożliwić zapis kolejnej kolumny. Wartość kolumny zapisuje się poprzez wejście FLM. Dokonuje się to w pętli *for* wykonującej się 240 razy. W pierwszej iteracji do kolumny wpisywana jest jedynka (podanie napięcia zmiennego na elektrody) i zapisywany jest wiersz. W pozostałych iteracjach pętli zapisywane jest zawsze zero, a kolejne zbocza na wejściu LP powodują, że zapisana do kolumny jedynka „przesuwa się" cyklicznie po wszystkich kolumnach. Zależnie od jej położenia, włączane są odpowiednie wiersze. Funkcja *refresh* widoczna jest na **listingu 1**.

Te dwie funkcje stanowią esencję sterowania wyświetlaczem. Wspomniano wcześniej, że konieczny jest przebieg prostokątny na wejściu FR. Za jego generację odpowiedzialny jest... sprzętowy generator PWM! Po skonfigurowaniu, które odbywa się w konstruktorze klasy *panel*, daje on na wyjściu symetryczny przebieg (wypełnienie 50%) o częstotliwości około 85Hz. W tym miejscu ustawiany jest również drugi generator PWM do sterowania tranzystora przetwornicy wytwarzającej wymagane napięcie 24V. Przy okazji w konstruktorze dokonywana jest inicjacja portów sterujących pracą wyświetlacza (są konfigurowane jako wyjścia), czyszczona jest pamięć obrazu oraz ustawiany jest „obszar roboczy" panelu dotykowego (o tym za chwilę).

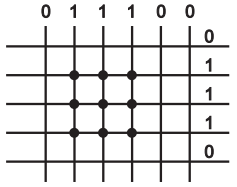
Z wyświetlaczem związane są jeszcze cztery inne funkcje: *clear*, *setPixel*, *clearPixel* oraz *rect*. Pierwsza z nich wymazuje cały ekran i umożliwia zapisywanie na nim obrazów od początku. Funkcja *setPixel*, jak wskazuje nazwa, umożliwia włączenie pojedynczego punkt na LCD. Operuje ona na pamięci obrazu (tablica *lcdMemory*) i na podstawie podanych argumentów określa kolumnę oraz wiersz i ustawia w nim żądany bit. Określenie wiersza jest bardziej skomplikowane niż określenie kolumny, gdyż konieczne jest, po pierwsze, wybranie jednej zmiennej spośród pięciu (obraz przechowywany w komórkach 32--bitowych), a po drugie, określenie, na której pozycji wybranej zmiennej ustawić jedynkę. W przypadku polecenia *clearPixel* sytuacja jest analogiczna z tym, że wybrany piksel jest wygaszany. Ostatnie funkcja, *rect*, rysuje prostokąt w oparciu o podane współrzędne dwóch naprzeciwległych wierzchołków.

Pozostałe trzy funkcje, tj. *kalibracja*, *tpGetX* oraz *tpGetY*, są przeznaczone do obsługi panelu dotykowego. Pierwsza z nich służy do określenia „obszaru roboczego" panelu dotykowego. Po dotknięciu ekranu tworzy się para dzielników „rezystorowych", gdzie wartości poszczególnych „rezystorów" zależne są od miejsca wystąpienia nacisku – ilustruje to poglądowy **rysunek 6**. Chcąc określić miejsce przyłożenia rysika, należy zasilić jedną parę rezystorów, a na wyjściu drugiej dokonać pomiaru (**rysunek 7**). Otrzymany pomiar informuje o położeniu rysika w osi Y, więc

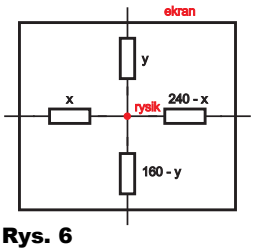
```
//odswieza widok
void panel::refresh(){
 //zmienne
 volatile int u = 0 ;
 volatile int p = 0 ;
   //zapis wszystkie linie poziome
   for(p=0 ; p<240 ; p++){
   for(u=0 ; u<5 ; u++){
      //zapisz slowo danych
      setDataPort(lcdMemory[p][u]) ;
   }
   //zatrzask
   if(p==0){LCD_FLM_H;}
   LCD_LP_H ;
   LCD_LP_L ;
   LCD_FLM_L ;
 }
 LCD_LP_H ;
 LCD_LP_L ;
}
```

Listing 1

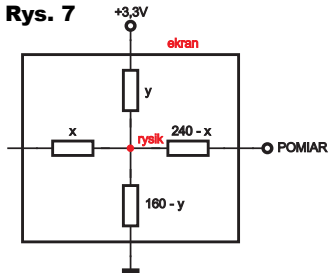
Rys. 5



Rys. 6



Rys. 7



Rys. 3

Common Mode

FR	Latch Data	/DISPOFF	Driver Output Voltage Level (Y1~Y160)
L	L	H	V43
L	H	H	V5
H	L	H	V12
H	H	H	V0
X	X	L	V5

Here, VSS ≤ V5 < V43 < V12 < V0, H: VDD (+2.5 to +5.5V), L: VSS (0V), X: Don't care

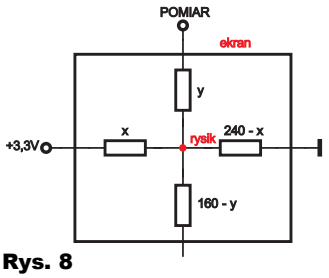
Rys. 4

Common Mode

FR	Latch Data	/DISPOFF	Driver Output Voltage Level (Y1~Y160)
L	L	H	V43
L	H	H	V0
H	L	H	V12
H	H	H	V5
X	X	L	V5

Here, VSS ≤ V5 < V43 < V12 < V0, H: VDD (+2.5 to +5.5V), L: VSS (0V), X: Don't care

konieczny jest drugi pomiar, aby określić położenie w osi X. Wymaga to zmiany zasilanej pary rezystorów i pomiaru na zasilanej uprzednio – **rysunek 8**. Podstawowym problemem jest to, że napięcie na wyjściu dzielnika nie ma wartości skrajnych, tzn. od 0V do 3,3V. W przypadku zastosowanego przeze mnie wyświetlacza przedział ten wynosił od około 0,5V do około 2,8V. Nie testowałem niestety innych egzemplarzy, więc nie wiem, jaka jest powtarzalność wykonania. Właśnie ta niepewność zdecydowała o dodaniu funkcji kalibracyjnej. Ujmując rzecz w skrócie, dokonuje ona pomiaru napięcia przy wszystkich czterech, skrajnych położeniach rysika i zapamiętuje je. Sposób wykonania kalibracji został przedstawiony w sekcji poświęconej montażowi i uruchomieniu urządzenia.

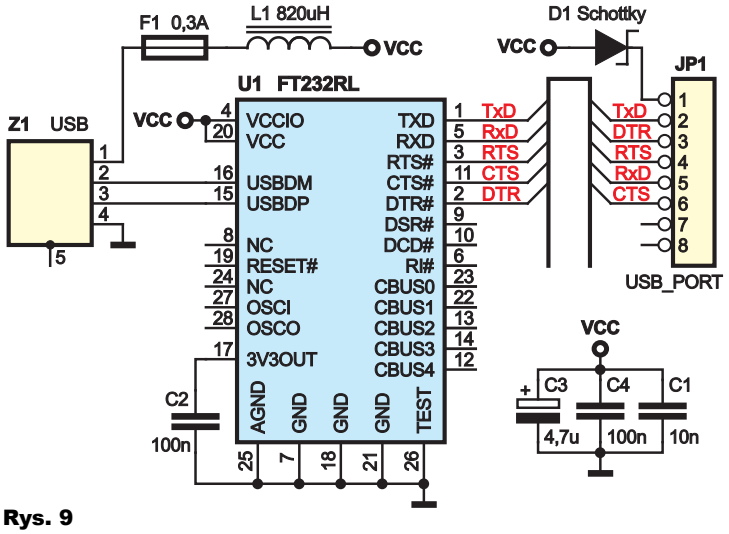


Rys. 8

Pozostałe funkcje, czyli *tpGetX* oraz *tpGetY*, służą do określania położenia rysika. Zależnie od przesłanego do funkcji argumentu zwracają wartość zmierzoną przetwornikiem ADC (wykorzystywane podczas kalibracji) lub położenie na ekranie zawierające się w zakresie 0...239 dla osi X oraz 0...159 dla osi Y (wykorzystywane podczas pisania na wyświetlaczu). Jeżeli rysik nie dotyka wyświetlacza, zwracana jest wartość 255 (poza zakresem), co jest znakiem dla funkcji *setPixel*, że ma nic nie rysować. Warto wspomnieć, że w obu funkcjach na bieżąco są rekonfigurowane porty mikrokontrolera: dwa ustawiane są jako wyjścia cyfrowe ze stanem zero oraz jeden (zasilania dzielnika), a jeden do pomiaru przetwornikiem zależnie od tego, czy mierzona jest współrzędna X, czy Y. Po ustawieniu portów wykonywany jest pomiar napięcia i w oparciu o znane, po procesie kalibracji wartości skrajne, wyznaczana jest aktualna pozycja rysika.

W pliku *main.cpp* znajduje się pętla główna programu. Jej konstrukcja nie jest szczególnie złożona. Przed rozpoczęciem jej wykonywania konfigurowany jest układ PLL – mnożnik częstotliwości, co pozwala zwiększyć częstot-liwość zegara ponad wartość znamionową rezonatora kwarcowego. Poprzez rejestr *SCS* włączana jest obsługa portów w trybie szybkim, który umożliwia wystawianie stanów logicznych w znacznie krótszym czasie niż w przypadku dostępu tradycyjnego. Tworzone są tu również dwa obiekty: *panel* obsługujący wyświetlacz oraz *port*, umożliwiający komunikację z komputerem przez port RS232 (co zostało wykorzystane na etapie uruchamiania urządzenia).

W pętli głównej następuje pobranie współrzędnych z panelu dotykowego. Są one pobierane dwa razy pod rząd, aby zmniejszyć wpływ przełączania napięcia przy zmianie „mierzonej" osi. Kolejnym krokiem jest odświeżenie zawartości ekranu za pomocą omówionej funkcji *refresh*. Następnie wywoływana jest procedura kalibracji, która sprawdza, czy naciśnięto przycisk S2, i wykonuje ewentualną kalibrację. Ostatnim zadaniem głównej pętli jest sprawdzenie, czy należy wykasować wyświetlacz. Jeżeli kilkadziesiąt razy pod rząd stwierdzone zostanie, że rysik znajduje się w lewym, górnym rogu, to nastąpi wyczyszczenie wyświetlacza. Zmiana położenia poza wyznaczony obszar lub zdjęcie rysika z ekranu spowoduje, że zmienna *przytrzymanie* zostanie wyzerowana i zliczanie rozpocznie się od początku.



Rys. 9

Montaż i uruchomienie

Jednym z najważniejszych elementów e-karteczki jest mikrokontroler, który niestety wymaga zaprogramowania... Okazuje się, że to zadanie nie jest takie trudne i wystarczy użyć zwykłego portu szeregowego komputera. Jedyną komplikacją jest konieczność zastosowania konwertera poziomów, np. popularnego układu MAX232. Następnie podpinamy się do złącza J1 (masę również!) i można programować. Uprzednio należy jeszcze wprowadzić mikrokontroler w tryb ładowania programu (uaktywnić bootloader), co sprowadza się do naciśnięcia przycisku S2, następnie S1 (reset) i puszczenia po chwili S1. Teraz można załadować program. Oprogramowanie można wgrać za pomocą firmowego programu (NXP Flash Utility – Google wie, gdzie on jest) lub darmowego Flash Magica (pytać Google o *NXP flash magic*).

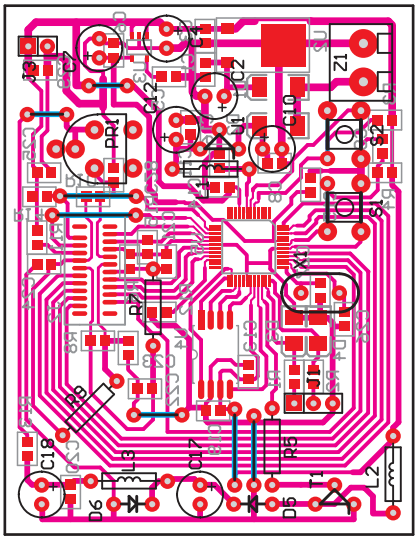
Co zrobić, jeśli nie posiadamy portu RS232 w komputerze? Zastosować inny układ, mianowicie FT232. Czytelnikom pragnę przypomnieć, że prosty konwerter USB<->RS232 opisany był w numerze 3/2008 EdW przy okazji omawiania *Elektronicznego dyrygenta*. Podczas tworzenia oprogramowania sam z niego korzystałem, tutaj nie potrzeba żadnej konwersji poziomów: należy połączyć złącze J1 bezpośrednio z konwerterem (masę również) i można zapisywać program po uprzednim wykonaniu sekwencji inicjacji bootloadera. Schemat tej przejściówki (nawiasem mówiąc, mojego autorstwa, więc zapożyczony legalnie:)) pokazany jest na **rysunku 9,** a wzory PCB można odnaleźć na stronie Elportalu (klikając na stronie głównej link *Wcześniejsze numery EdW*).

Do poprawnej pracy urządzenia potrzebna jest kalibracja – pozwala ona określić zakres pomiarowy dla przetwornika ADC. Sprowadza się ona do kilku prostych kroków. Na początek należy nacisnąć przycisk S2 co spowoduje wymazanie zawartości ekranu i wyświetlenie w jego górnej części małego kwadratu. Rysikiem należy go dotknąć lub jeszcze lepiej „wjechać" na niego rozpoczynając przesuwanie rysika od skraju wyświetlacza. Po najechaniu, kwadrat zniknie i pojawi się na dole, tu sytuacja jest analogiczna, przykładamy rysik do skraju ekranu i powoli go przesuwamy na kwadrat. Potem pojawi się on, kolejno, po lewej i prawej stronie. Za każdym razem metoda działania jest ta sama. Po zakończeniu kalibracji na kilka sekund wyświetli się symbol uśmiechniętej

buzi, która następnie zniknie, pozostawiając cały ekran do dyspozycji Czytelników.

Pragnę zwrócić uwagę na jeszcze jeden szczegół. Podczas uruchamiania urządzenia zdarzało się, że panel dotykowy nie reagował lub kwadraty znikały samoistnie w czasie przeprowadzania kalibracji. Winę za to ponosiło niepewne połączenie pomiędzy taśmą wyświetlacza a pinami złącza. W razie podobnych problemów warto delikatnie poruszać taśmą łączącą panel LCD z urządzeniem i obserwować efekty.

Zasadniczo dobór obudowy pozostawiam Czytelnikom. Model widoczny na zdjęciu umieszczono między dwiema płytkami z pleksiglasu. W górnej płytce wycięty został otwór na wyświetlacz, który został wciśnięty na styk i dodatkowo posmarowany czterema kropelkami kleju Super Glue. Obie płytki zostały połączone tym samym klejem, a pomiędzy nie wstawiłem cztery, plastikowe tulejki dystansowe. Fragment pleksiglasu (pasek o szerokości paru cm) podgrzałem zapalniczką i wygiąłem w kształt cyfry 1, co stworzyło podpórkę dla całej konstrukcji. Podpórka również została przytwierdzona klejem.



Rys. 10

Możliwości zmian

Układ można zmontować na płytce drukowanej pokazanej na **rysunku 10**.

Zasadniczo do projektu zostały dołączone pełne kody źródłowe, które były kompilowane z użyciem darmowego pakietu WinARM. Warto choćby dla samej ciekawości trochę „pomieszać" w kodzie, aby dodać własne funkcje, efekty graficzne, takie jak np. rozlewanie „atramentu" przy dłuższym przytrzymaniu rysika. Jest to również dobra okazja do „dotknięcia" procesorów 32-bitowych, tym bardziej że do zapisania programu nie jest potrzebny żaden skomplikowany programator, a kompilacja odbywa się za pomocą dołączonego skryptu *start.bat*. Jedyną trudnością jest pełne zainstalowanie kompilatora, gdyż trzeba zmodyfikować zmienną środowiskową systemu. Dokonuje się tego poprzez wybranie ikony *System* z *Panelu sterowania*, a następnie zakładki *Zaawansowane*, dalej klikając *Zmienne środowiskowe* i dopisując w polu ścieżkę dostępu do kompilatora (**rysunek 11**). Bardziej szczegółowy opis można znaleźć w artykule poświęconym *Cyfrowej Iluminofonii LED* (EdW 12/2006). Gorąco zachęcam do własnych eksperymentów na tym polu:).

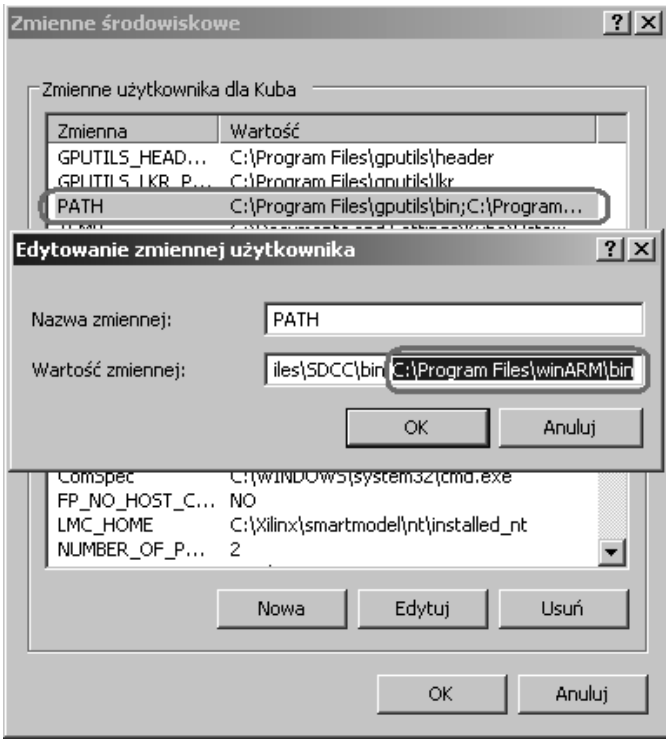
Po zainstalowaniu pakietu WinARM można wykorzystać skrypt programujący dołączony do kodów źródłowych. Konieczna jest jednak zmiana portu komunikacyjnego programatora. W tym celu należy w pliku *makefile* odnaleźć linię:

LPC21ISP_PORT = com5

i ustawić port szeregowy wykorzystywany przez programator lub port, pod który podłączył się układ FT232.

Jakub Borzdyński
jakub.borzdynski@elportal.pl

Rys. 11



Wykaz elementów

(SMD 2012 lub 0805)

R1-R4	1kΩ SMD	D5	1N4007
R5	1kΩ	D6	24V
R6,R10	10kΩ SMD	LCD	LGMJM160240A6WLW3-TP
R7,R9	10kΩ	T1	BC546
R8,R14,R15	100kΩ SMD	U1	S78DL33
R11	100Ω SMD	U2	LF33CDT
R12,R13	4,7kΩ SMD	U3	TC1015
PR1	47kΩ	U4	AT45DB011B
C1,C3,C6,C8,C9,C11,C13-C16,C19-C25,		U5	LPC2103
C28,C29	100nF SMD	J1	goldpin x3
C2,C4,C7,C10,C12	100µF	J2	ZIFNZ0320CV
C5	470pF SMD	J3	goldpin x2
C17	4,7µF	L1	10µF
C18	2,2µF	L2	1mH
C26,C27	22pF SMD	L3	10µH
D1,D2	1N4007 SMD	S1,S2	µswitch
D3,D4	3V6 SMD	X1	11,0592MHz
		Z1	ARK2

Płytka drukowana jest dostępna w sieci handlowej AVT jako kit szkolny AVT-2893.