Mirlight

Sterownik podświetlania monitora

AVT 5277

Zadaniem układu jest podświetlanie otoczenia monitora w kolorze zależnym od aktualnie wyświetlanego obrazu. Jest to efektowny bajer zmniejszający kontrast pomiędzy wyświetlanym obrazem a tłem.

Rekomendacje: *podświetlacz uatrakcyjni każdy zestaw komputerowy.*

Podświetlacz składa się ze sterownika, zbudowanego z użyciem mikrokontrolera ATMega16 i połączonych z nim modułów diodowych, po trzy diody połączone równolegle. Mirlight ma 8 niezależnych kanałów generujących światło w barwie zbliżonej do „wypadkowego" światła na brzegach ekranu. Łączność z komputerem zapewnia konwerter USB/RS232 firmy FTDI. Oprogramowanie sterujące zostało napisane w języku Python i działa z powodzeniem pod systemem Windows i Linux. W programie komputerowym jest możliwe zdefiniowanie obszarów „uśredniania" koloru dla każdego z kanałów oraz ustalenie szybkości przejść kolorów, gdy obraz na ekranie dynamicznie się zmienia. Dzięki wykorzystaniu biblioteki QT4 aplikacja na komputerze działa bardzo sprawnie i zbytnio nie obciąża procesora. Podświetlacz ma też możliwość odbioru transmisji RC5 i sterowania odtwarzaczem filmów. Oprogramowanie dla mikrokontrolera zostało napisane w języku Bascom AVR i do jego kompilacji z powodzeniem wystarczy wersja demonstracyjna z ograniczeniem kodu wynikowego do 4 kB. Dzięki temu każdy, w oparciu o dokładny opis programu, będzie mógł wprowadzić dodatkowe funkcje lub zmodyfikować już istniejące.

Opis działania układu

Schemat ideowy sterownika pokazano na **rysunku 1**. Mikrokontroler U1 (ATMega16-16PU) jest taktowany sygnałem oscylatora z rezonatorem kwarcowym X1 (16 MHz). Program zawarty w mikrokontrolerze steruje podświetlaniem. Pozostałymi elementami są bufory wyjściowe i konwerter USB/RS232 (U3, FT232RL). Złącze prog (goldpin) umożliwia zaprogramowanie mikrokontrolera U1 bez wyjmowania go z podstawki.

AVT-5277 w ofercie AVT:
AVT-5277A – płytka drukowana
AVT-5277B – płytka drukowana + elementy

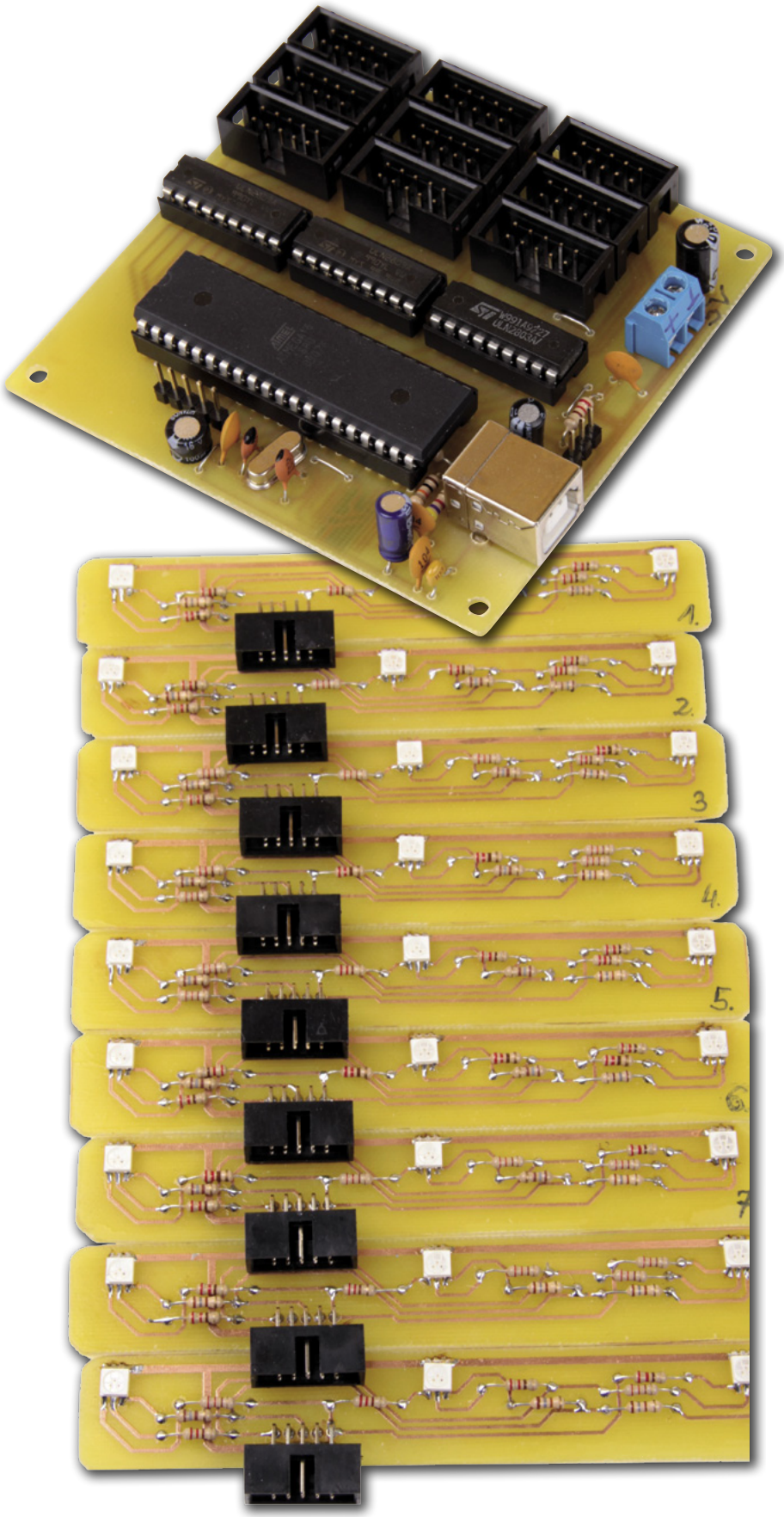
Podstawowe informacje:
- Sterownik podświetlenia z mikrokontrolerem ATMega16
- Podświetlenie za pomocą 9 modułów z diodami LED (8 kanałów, jeden kanał zdublowany)
- Zasilanie z zewnętrznego zasilacza 5 V/1,6 A
- Współpraca z systemami Windows i Linux

Dodatkowe informacje:
Najnowsza wersja programu sterującego jest dostępna na dysku CD oraz na stronie *https://github.com/grizz-pl/mirlight*, natomiast wszelkie inne informacje na jego temat znajdują się na stronie autora *http://grizz.pl/mirlight*.

Dodatkowe materiały na CD i FTP:
ftp://ep.com.pl, user: 17855, pass: 4s406qj2
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

Złącze USB1 jest gniazdem USB typu B. Kondensatory C9...C11 filtrują napięcie zasilania dla U3 i zostały zastosowane zgodnie z zaleceniami producenta układu. Rezystory R4 i R5 są pomocne przy badaniu podłączenia przewodu do portu USB, gdyż cały układ jest zasilany z zewnętrznego zasilacza 5 V (ze względu na duży pobór prądu nie da się za-

silać układu bezpośrednio z portu USB komputera). Rezystor R4 wymusza niski poziom na doprowadzeniu portu mikrokontrolera, gdy wtyczka USB jest odłączona. Rezystor R5 ogranicza przepływ prądu z gniazda USB do pinu PD5 mikrokontrolera w przypadku, gdy zewnętrzny zasilacz jest odłączony. Rezystory R2 i R3 tworzą dzielnik rezystancyjny dopasowujący poziomy logiczne układów zasilanych napięciami 5 V i 3,3 V oraz polaryzują wejście RESET# układu U3. Wszystkie układy są zasilane z zewnętrznego zasilacza, a napięcie pojawiające się po włożeniu wtyczki USB powoduje wykonanie restartu konwertera USB/RS232.



Dzięki temu niezależnie od kolejności włączenia komputer prawidłowo wykryje urządzenie w systemie.

Zasilanie 5 V jest dołączone za pomocą złącza Z1 (ARK), przeważnie bezpośrednio z zasilacza w komputerze, gdyż taka liczba diod może pobierać prąd większy niż 1,6 A. Ze względu na znaczną liczbę diod nie mogą one być wysterowane bezpośrednio z mikrokontrolera i jest konieczne zastosowanie buforów wyjściowych. W tym celu zastosowano układy U4...U6 (ULN2803). Ponadto umożliwiają one zasilanie diod napięciem wyższym niż 5 V. Mirlight pozwala na wysterowanie 9 modułów w 8 niezależnych kanałach (ostatni jest zdublowany, aby lepiej oświetlić spód monitora). Rozmieszczenie kanałów zostanie przedstawione w dalszej części artykułu.

Moduły oświetlające są dołączone do sterownika za pomocą tasiemek 10-żyłowych z wykorzystaniem popularnych złączy FD10 i wtyczek FC10 (FC1- FC8, FC8B). Tak duże złącze zostało wybrane ze względu na łatwość montażu tasiemek i brak konieczności lutowania pojedynczych przewodów.

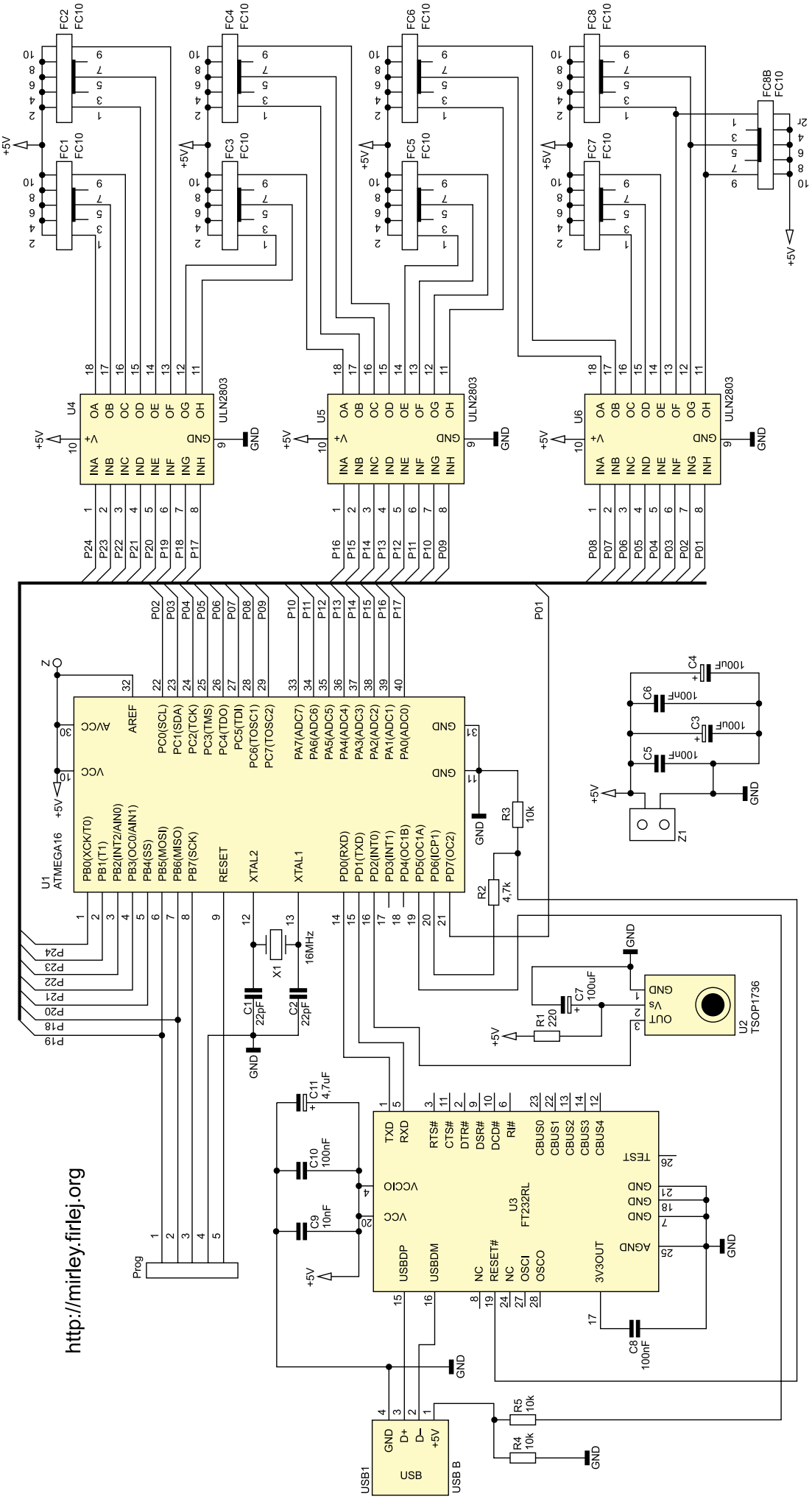
Zastosowanie układu odbiornika podczerwieni U2 (TSOP1736) umożliwia zdalne sterowanie z wykorzystaniem pilota w standardzie RC5 i upraszcza jego implementację. Funkcja ta ma większe znaczenie w przypadku pracy w systemie Linux, gdyż łatwo można skojarzyć przycisk na pilocie z poleceniem w konsoli, a tym samym wykonać dowolną akcję w systemie.

Schemat ideowy modułu diodowego przedstawiono na **rysunku 2**. Zastosowano diody D1-D3 przeznaczone do montażu SMD. Rozpraszanie barw w takich obudowach jest znacznie lepsze niż w odpowiednikach do montażu przewlekanego. Rezystory ograniczające prąd R1, R4 i R7 mają rezystancję 180 Ω, która jest mniejsza niż pozostałych, aby zapewnić jednakową jasność świecenia wszystkich barw (widzianą przez oko). Pozostałe rezystory mają rezystancję 220 Ω. Najlepiej jednak rezystory ograniczające prąd dobrać własnoręcznie, pamiętając, aby nie przekroczyć przy tym maksymalnego prądu danej diody. W opisy-

wanym układzie diody pracują z prądem rzędu 16...18 mA, przy maksymalnym prądzie używanych diod wynoszącym 20 mA. W podświetlaczu zastosowano diody o jasności 250 mcd, jednak czym większa jasność świecenia diod, tym uzyskuje się lepsze efekty podczas korzystania z urządzenia.

Kilka słów o FT232RL

Układ FT232RL jest kompletnym interfejsem umożliwiającym konwersję USB/RS232. Zapewnia bezproblemową pracę we wszystkich systemach operacyjnych. Wewnątrz niewielkiej obudowy zostały zamknięte między innymi: kompletny transceiver USB, generator przebiegu zegarowego, bufor wejściowy i wyjściowy oraz logika sterująca. Do pracy układu FT232RL wymagane jest tylko dołączenie kilku biernych elementów zewnętrznych, a po podłączeniu do portu USB i zainstalowaniu sterowników widziany jest on w systemie jako wirtualny port szeregowy. Sterowniki dla odpowiedniej wersji systemu operacyjnego można pobrać ze strony *http://www.ftdichip.com/Drivers/VCP.htm*. Po ich zainstalowaniu w menadżerze urządzeń (Windows) powinien pojawić się nowy port szeregowy, podobnie jak na **rysunku 3**. W przypadku systemu Linux po wydaniu w konsoli polecenia „dmesg” pojawią się informacje na temat nowo wykrytego urządzenia, między innymi jego nazwa (prawdopodobnie ttyUSB0) i czasami także modułu, który należy załadować do jądra. Przykładowy rezultat tego polecenia można zobaczyć na **rysunku 4**.



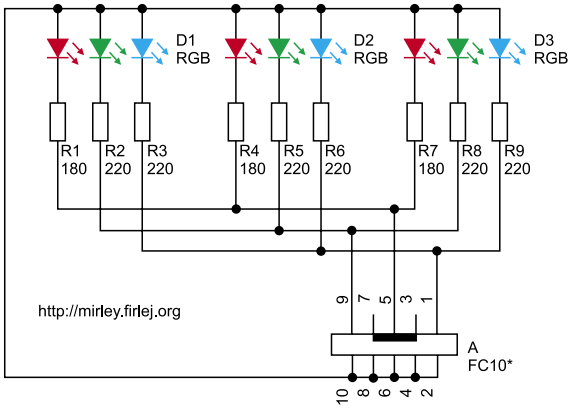
Rysunek 1. Schemat ideowy sterownika

Budowa i uruchomienie

Widok płytki drukowanej sterownika zamieszczono na **rysunku 5**. Montaż sterownika nie jest trywialny ze względu na układ U3 w bardzo małej obudowie SSOP28. Trzeba tutaj bardzo uważać przy lutowaniu układu, aby uniknąć zwarć i należy go przylutować w pierwszej kolejności. W drugiej kolejności należy wlutować wszystkie zworki (6 sztuk) oraz rezystory. Pod układ U1 dobrze jest zamontować podstawkę, podobnie jak pod bufory U4...U6. Kolejność montażu pozostałych elementów jest dowolna, z tym że złącza FD10 dobrze jest zamontować na samym końcu. Wygląd gotowego sterownika pokazano na **fotografii 6**.

Do wykonania płytek z diodami zaleca się zastosowanie cienkiego laminatu, np. 0,8 mm, gdyż wtedy będzie można lekko wygiąć dany moduł i dopasować do krzywizny monitora. Montaż modułów paneli diodowych (9 sztuk) nie jest trudny, ale pracochłonny. Można wlutować rezystory od strony elementów, a diody od strony ścieżek lub wszystkie elementy umieścić po stronie ścieżek. Taki montaż jest nietypowy dla elementów przewlekanych, jednak umożliwi łatwe przyklejenie płytek z tyłu monitora. Złącze na taśmę (kątowe) musi być przylutowane od strony druku. Na płytce modułu diodowego nie zastosowano elementów SMD (poza diodami), gdyż wymagany dystans między diodami wyznacza wymiary laminatu znacznie przewyższające powierzchnię potrzebną na umieszczenie pozostałych elementów.

Jak wspomniano, układ jest zasilany napięciem 5 V bezpośrednio z zasilacza komputerowego. W tym celu należy wyprowadzić



*złącze to przylutowane jest na płytce od strony druku, numery pinów odpowiadają tym ze sterownika.

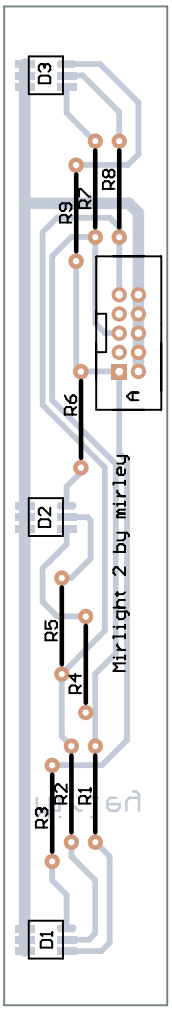
Rysunek 2. Schemat modułu LED

Rysunek 3. Dev_manager

Rysunek 4. Linux console

z komputera kabel zasilający jednym z wolnych złączy MOLEX (do zasilania dysków ATA, CDROM itp.). Konstrukcja podświetlacza Mirlight umożliwia jego zamontowanie z tyłu monitora bez żadnych dodatkowych elementów. Nie trzeba też stosować żadnego materiału rozpraszającego światło. Zastosowane diody mają bardzo duży kąt świecenia (120°), barwy znakomicie mieszają się, tworząc jednolity kolor pośredni. Układ nie wymaga obudowy, a płytki modułów diodowych i sterownik można przykleić do monitora za pomocą dwustronnej taśmy klejącej do luster. Rozmieszczenie modułów diodowych (kanałów podświetlacza) przedstawiono na **rysunku 7**. Rysunek pokazuje miejsca zamontowania płytek z diodami w widoku od przodu monitora. Kanał 1. znajduje się po lewej stronie u dołu, natomiast płytka z kanałem 8. jest zdublowana, aby lepiej oświetlić dół monitora.

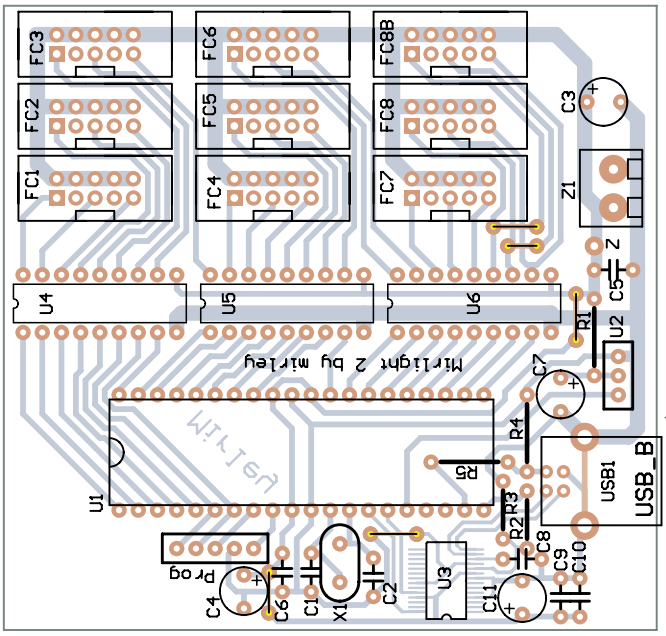
Przed przystąpieniem do połączenia całości warto sprawdzić działanie każdego z modułów diodowych – czy wszystkie diody świecą i czy ich jasność jest jednakowa. Tasiemki należy tak zacisnąć, aby przewodziły 1 do 1 (pierwszy pin jednej wtyczki był połączony z pierwszym pinem drugiej), zgodnie z **rysunkiem 8**. Podczas montażu płytek na monitorze pomocna będzie **fotografia 9** przedstawiająca gotowe urządzenie. Przy prowadzeniu taśm należy zwrócić uwagę na otwory wentylacyjne monitora, których nie można zasłaniać.



Rysunek 6. Schemat montażowy płytki modułu diodowego

Transmisja PC ->uC

Komunikacja między komputerem a mikrokontrolerem odbywa się w trybie transmisji szeregowej. Należy przesłać 26 bajtów danych. Strukturę pakietu danych transmitowanego do mikrokontrolera przedstawiono na **rysunku 9**. Składa się on z bajtu startowego, który jako jedyny przyjmuje wartości większe od 127. Wartość 128 oznacza, że są to informacje o kolorach, natomiast inne wartości bajtu są zarezerwowane dla ewentualnego konfigurowania sprzętu. Na przykład w tej wersji oprogramowania przesyłane bity konfiguracyjne są wykorzystywane do automatycznego wykrywania portu, do którego jest dołączony Mirlight. Przesłanie wartości 130 zmusza układ do odesłania tej samej wartości i tym samym pozwala aplikacji na komputerze zapisać aktualnie odpytywany port jako prawidłowy. W dalszej kolejności odbierane są 24 bajty (wartości 0–100) zawierające informacje o kolorach w kolejnych kanałach. Pakiet danych zakończony jest sumą kontrolną (wartość 0–127), która powstaje poprzez zsumowanie wartości bajtów całej paczki, pobranie reszty z dzielenia tej warto-



Rysunek 5. Schemat montażowy płytki sterownika

ści przez 256 i podzielenie wyniku tej operacji przez 2.

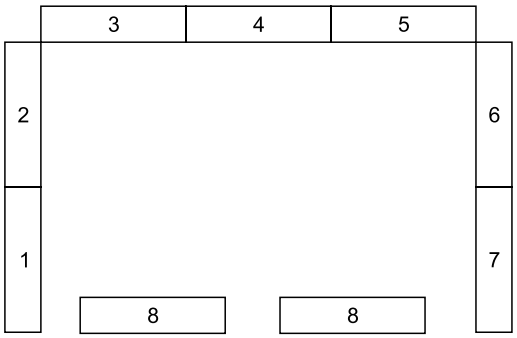
Oprogramowanie mikrokontrolera

Oprogramowanie dla mikrokontrolera zostało napisane w Bascomie AVR. Kod źródłowy *.bas oraz pliki wynikowe *.bin i *.hex są dostępne w materiałach dodatkowych dołączonych do tego numeru EP. Program po kompilacji zajmuje mniej niż 3 kB, dzięki czemu może być modyfikowany z użyciem wersji demonstracyjnej Bascoma, która ma ograniczenie wielkości kodu wynikowego do 4 kB.

Na **rysunku 10** pokazano ustawienia fusebitów mikrokontrolera (zrzut z programu Burn-O-Mat (*http://avr8-burn-o-mat.aaabbb.de/avr8_burn_o_mat_avrdude_gui_en.html*). Mikrokontroler musi być skonfigurowany do pracy z zewnętrznym rezonatorem kwarcowym o częstotliwości 16 MHz. Interfejs JTAG musi być wyłączony, aby zwolnić wszystkie linie portu C mikrokontrolera i umożliwić ich używanie jako zwykłych wyprowadzeń I/O.

Na **listingu 1** przedstawiono deklaracje konfigurujące mikrokontroler. W ich skład wchodzą między innymi deklaracja typu procesora ($regfile) i częstotliwości dołączonego rezonatora kwarcowego ($crystal), konfiguracja i stan początkowy wszystkich wyprowadzeń, ustawienie pracy timerów, przerwania zewnętrznego od odbiornika RC5. Ponieważ komunikacja z komputerem odbywa się za pomocą portu COM, a ściślej wirtualnego portu szeregowego, ważne jest zdefiniowanie rozmiaru bufora wejściowego za pomocą polecenia *Config Serialin* oraz ustawienie szybkości transmisji ($baud). Wielkość bufora wejściowego ustalono na 26 bajtów.

Na **listingu 2** umieszczono pętlę główną. Jej zadaniem jest odbieranie i wysyłanie danych z/do portu szeregowego oraz sprawdzanie poprawności otrzymanego (26 bajtowej) pakietu. Użycie w pierwszej linii polecenia *Portd.6 = Pind.5* w nieskończonej pętli *Do Loop* może być niejasne. Stan logiczny z pinu D5 jest tutaj przepisywany na pin D6. Polecenie to w istocie stanowi bramkę AND, której sygnałami wejściowymi są stan pinu D5 mikrokontrolera i zasilanie mikrokontrolera. Zapewnia to prawidłowe zerowanie konwertera FT232 zarówno podczas podłączenia układu do USB (przy włączonym zasilaniu), jak i podczas włączenia zasilania (przy podłączonym USB). Przez cały czas mikrokontroler realizuje nieskończoną pętlę, oczekując na pojawienie się w buforze wejściowym pierwszego znaku. Pierwszy bajt powinien mieć wartość 128 i oznacza początek paczki danych. Wyzerowany zostaje licznik odebranych bajtów *I*, a zmienna przechowująca wartość sumy kontrolnej *Tab_sum* przyjmuje wartość pierwszego bajtu. Każdy kolejny bajt zostaje wpisany do tymczasowej tablicy *Tab_temp* i na bieżąco jest obliczana suma kontrolna. Dzieje się tak do momentu odebrania 25 bajtów, po czym suma kontrolna wyliczona podczas odbioru jest porównywana z tą odebraną w ostatnim bajcie. Jeśli wszystko jest w porządku, to dane z tablicy tymczasowej przepisywane są do tablicy *Color* przechowującej ustawione wypełnienie przebiegów prostokątnych podawanych na diody



Rysunek 7. Rozmieszczenie modułów diodowych na monitorze, widok od przodu

Rysunek 8. Sposób zaciskania gniazd IDC na przewodach

Listing 1. Konfigurowanie mikrokontrolera

```
$regfile = „m16def.dat”
$crystal = 16000000
$baud = 38400
Config Serialin = Buffered , Size = 26
Config Porta = &B11111111 : Porta = &B11111111
,.....
Config Timer1 = Timer , Prescale = 8
Enable Timer1 : On Timer1 Prztimer1
On Int0 Przzew0 Nosave
Config Int0 = Low Level
Enable Int0
Config Rc5 = Pind.2
Enable Interrupts
```

Fotografia 9. Widok oświetlacza zamontowanego na monitorze

Rysunek 10. Pakiet przesyłanych danych

Wykaz elementów

Sterownik

Rezystory:
R1: 220 Ω
R2: 4,7 kΩ
R3...R5: 10 kΩ

Kondensatory:
C1, C2: 22 pF
C3, C4, C7: 100 µF/16 V
C5, C6, C9, C10: 100 nF
C11: 4,7 µF/25 V

Półprzewodniki:
U1: ATMega16-16PU
U2: TSOP1736
U3: FT232R
U4...U6: ULN2803

Inne:
X1: 16 kwarc MHz
Z1: ARK5
USB1: złącze USB B
FC1...FCFC8B: wtyk IDC10 do druku

Płytka LED (pojedyncza)
R1, R4, R7: 180 Ω
R2, R3, R5, R6, R8, R9: 220 Ω
D1...D3: dioda LED RGB, SMD
A: wtyk IDC10 do druku

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Listing 2. Pętla główna

```
Do
 Portd.6 = Pind.5

 If Ischarwaiting() = 1 Then
   Temp = Inkey()
   If Temp > 127 Then
     I = 0
     Tab_sum = Temp
     If Temp <> 128 Then Conf_temp = Temp And &B01111111 Else Conf_temp = 0
   Else
     Incr I
     Tab_temp(i) = Temp
     Tab_sum = Tab_sum + Temp
     If I = 25 Then
       Tab_sum = Tab_sum - Tab_temp(25)
       Tab_sum = Tab_sum / 2
       If Tab_sum = Tab_temp(25) Then
         Stop Timer1
         For I = 1 To 24
           Color(i) = Tab_temp(i)
         Next I
         Start Timer1
         If Conf_temp <> 0 Then
           Conf = Conf_temp
           Conf_temp = 130
           If Conf = 2 Then Printbin Conf_temp
         End If
       End If
     End If
   End If
 End If
 If Send = 1 Then
   Send = 0
   If Adres = 0 Then Printbin Komenda
   Enable Int0
 End If

Loop
End
```

Listing 3. Przerwanie timera

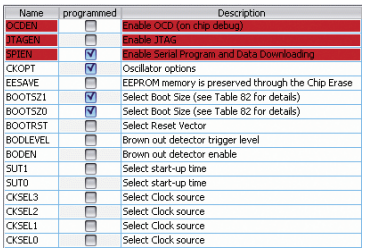
```
Prztimer1:
 Timer1 = 65286
 Incr Licznik
 If Licznik = 100 Then Licznik = 0
 If Licznik < Color(1) Then Set B.1 Else Reset B.1
,......sterowanie Color (2...23)
 If Licznik < Color(24) Then Set C.1 Else Reset C.1
 Porta = A
 Portb = B
 Portc = C
 Portd.7 = D7
Return
```

Listing 4. Obsługa pilota Rc5>

```
Przzew0:
  Disable Int0
  Enable Interrupts
  Getrc5(adres , Komenda)
  Komenda = Komenda And &B01111111
  Send = 1
Return
```

LED. Gdy suma kontrolna się nie zgadza, parametry wyświetlanych kolorów nie ulegają zmianie, a program czeka na kolejny bajt startowy. Gdy bajt startowy jest równy 130, urządzenie interpretuje paczkę jako rozkaz kontrolny i odsyła do komputera wartość 130. Takie działanie wykorzystywane jest do automatycznego wykrywania, do którego z portów podłączono podświetlacz. Program na komputerze rozsyła pakiety kontrolne na wszystkie dostępne porty szeregowe i czeka na odpowiedź.

Pętla główna zajmuje się także odsyłaniem komend sterujących z pilota. Jeżeli zmienna Send zostanie ustawiona na 1, to jeśli *Adres* jest równy 0, komenda zostaje wysłana do portu szeregowego. Adres 0 jest związany z pilotem od telewizora i może być zmieniony w przypadku innego pilota. Ważne jest tylko, aby pilot pracował w standardzie RC5.



Name	programmed	Description
OCDEN	☐	Enable OCD (on chip debug)
JTAGEN	☐	Enable JTAG
SPIEN	☑	Enable Serial Program and Data Downloading
CKOPT	☑	Oscillator options
EESAVE	☐	EEPROM memory is preserved through the Chip Erase
BOOTSZ1	☑	Select Boot Size (see Table 82 for details)
BOOTSZ0	☑	Select Boot Size (see Table 82 for details)
BOOTRST	☐	Select Reset Vector
BODLEVEL	☐	Brown out detector trigger level
BODEN	☐	Brown out detector enable
SUT1	☐	Select start-up time
SUT0	☐	Select start-up time
CKSEL3	☐	Select Clock source
CKSEL2	☐	Select Clock source
CKSEL1	☐	Select Clock source
CKSEL0	☐	Select Clock source

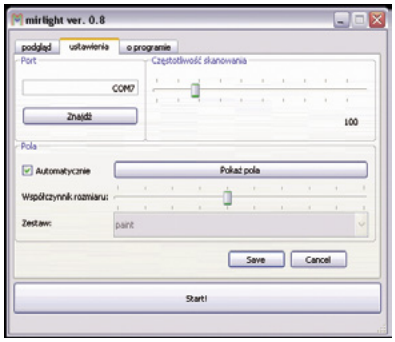
Rysunek 11. Ustawienia fusebitów mikrokontrolera ATMega16

Jedną z ważniejszych procedur w programie jest obsługa przerwania od timera zaprezentowana na **listingu 3** (listing uproszczony, nie zawiera sterowania wszystkimi pinami). Przerwanie wywoływane jest co około 125 µs, odpowiada to częstotliwości 8 kHz. Po każdym wywołaniu obsługi przerwania, cyklicznie zwiększana jest zmienna Licznik (od 0 do 99). Dalej następuje porównanie wartości licznika z wartością odpowiedniej komórki tablicy *Color*, a rezultatem tego porównania jest ustawienie odpowiedniego bitu w jednej ze zmiennych pomocniczych: A, B lub C. Procedura realizuje funkcję programowego PWM z krokiem co 1% wypełnienia. Jeśli dla przykładu zmienna Color(1) będzie równa 50, to dla wartości licznika 0...49 zmienna B.1 przyjmie wartość 1, a dla wartości licznika 50...99 zmienna B.1 będzie równa 0. Da to w efekcie przebieg prostokątny o wypełnieniu 50%. Podobnie będzie dla innych wartości zmiennych *Color*, a zatem wartość ta wprost odpowiada wypełnieniu przebiegu na konkretnym wyjściu (24 wyjścia = 8 kanałów po 3 kolory). Każdemu z wyjść odpowiada jeden bajt z 24-elementowej tablicy *Color*. Na końcu procedury obsługi przerwania wartości zmiennych pomocniczych są przesyłane do portów mikrokontrolera w postaci całych bajtów. Częstotliwość przebiegów na wyjściach programowych PWM-ów wynosi 80 Hz i wynika z podziału częstotliwości 8 kHz przez 100 za pomocą zmiennej Licznik.

Procedurę obsługi pilota RC5 zamieszczono na **listingu 4**. Wykorzystano w niej przerwanie zewnętrzne INT0 mikrokontrolera. Po odebraniu transmisji układ TSOP1736 ustawia

Listing 5. Funkcja odczytująca kolor z ekranu
```
def getColor(self, px, py, w, h ):
    self.originalPixmap = QtGui.QPixmap.grabWindow(QtGui.QApplication.desktop().winId(), px, py, w, h)
    self.destPixmap = self.originalPixmap.scaled(1, 1, QtCore.Qt.IgnoreAspectRatio, QtCore.Qt.SmoothTransformation)
    self.destImage = self.destPixmap.toImage()
    value = self.destImage.pixel(0,0)
    return value
```

Rysunek 12. Menu główne



Rysunek 13. Ustawienia

na swoim wyjściu poziom niski, co powoduje wywołanie procedury obsługi przerwania i w efekcie wywołanie polecenia *Getrc5*. Kodu RC5 użyto głównie ze względu na prostotę obsługi i fakt, że sterowanie pilotem nie należy do głównych zadań układu. Po odebraniu komendy i adresu z pilota zostaje ustawiona zmienna *Send = 1*, aby pętla główna mogła zająć się wysyłaniem komendy z pilota w dogodnej chwili (z punktu widzenia transmisji danych).

Oprogramowanie sterujące

Zaprogramowany mikrokontroler czy nawet gotowy układ sprzętowy Mirlighta nie będzie działał bez specjalistycznego oprogramowania na komputerze. Na **rysunku 11** przedstawiono wygląd okna głównego programu (zakładka „Podgląd"), natomiast **rysunek 12** ilustruje zakładkę ustawień.

Do najważniejszych funkcji programu należy odczytywanie kolorów wyświetlanych na monitorze, ich „uśrednianie" i odpowiednie przeliczanie na wypełnienie przebiegu prostokątnego. Zaraz po pierwszym uruchomieniu programu należy wejść na zakładkę ustawień (rys. 12) i wybrać odpowiedni port, do którego został dołączony układ podświetlacza. Można skorzystać z funkcji automatycznego wyszukiwania portu, co w większości przypadków załatwia sprawę. W razie wątpliwości nazwa portu jest widoczna w menadżerze urządzeń (dla użytkowników Windows: rys. 3) lub po wydaniu

Listing 6. Wysyłanie pakietu danych
```
def sendColors(self, colors):
    kod = chr(128)
    global sum
    sum = 0
    self.addSum(128)
    for color in colors:
        red = float(QtGui.qRed(color))/255
        green = float(QtGui.qGreen(color))/255
        blue = float(QtGui.qBlue(color))/255
        verbose(„k: %d" % (colors.index(color)+1), 3)
        verbose(„\trgb: %f, %f, %f" % (red, green, blue), 3)
        red = int(red*red*100)
        green = int(green*green*100)
        blue = int(blue*blue*100)
        verbose(„\t\trgb: %f, %f, %f" % (red, green, blue), 3)
        kod += chr(red)
        kod += chr(green)
        kod += chr(blue)
        self.addSum(red)
        self.addSum(green)
        self.addSum(blue)
    kod += chr(sum/2)
    try:
        ser.write(kod)
    except:
        verbose(„--\nCannot send to device. Check your configuration!",1)
    time.sleep(0.009)
```

w konsoli polecenia „dmesg" (dla użytkowników Linuksa: rys. 4). W oknie ustawień można wybrać „częstotliwość skanowania", którą należy utożsamiać z czasem (mierzonym w milisekundach) między kolejnymi odczytami kolorów z ekranu. W większości przypadków wartość 100 ms jest odpowiednia. Wartości mniejsze mogą przyczynić się do problemów z wydajnością komputera, ale można popróbować.

Przycisk „Pokaż pola" umożliwia podgląd ustawionych pól odczytowych. W większości przypadków tryb automatyczny dostrajania pól jest wystarczający, jednak wyłączenie tego trybu daje możliwość dopasowania miejsc odczytu koloru w sposób manualny. Ręcznie ustawiony schemat odczytu można zapisać do pliku.

Uruchomienie podświetlacza następuje po kliknięciu przycisku „Start!" w oknie głównym (zakładce podglądu) programu. Następuje wtedy otwarcie ustawionego wcześniej portu oraz rozpoczęcie odczytu kolorów, przeliczania i wysyłania pakietów danych do układu sprzętowego.

Aplikacja sterująca została napisana w języku Python z wykorzystaniem biblioteki QT4 i jest rozpowszechniana na licencji GPL. W niniejszym opracowaniu zostaną przedstawione tylko ważniejsze procedury programu, gdyż cały kod zająłby zbyt wiele miejsca. Na listingu 5 pokazana została funkcja pobierająca „średni" kolor z określonego miejsca ekranu. Najpierw pobierana jest z pulpitu mapa bitowa o współrzędnych ekranowych px,py oraz szerokości w i wysokości h. W dalszej kolejności mapa bitowa zostaje zmniejszona do rozmiarów jednego piksela z wykorzystaniem uśredniania kolorów i pominięciem proporcji w rozmiarze. Z tak uzyskanej mapy bitowej funkcja zwraca jeden piksel reprezentujący średni kolor całego pola.

Za wysyłanie pakietu danych przez port szeregowy odpowiada funkcja przedstawiona na **listingu 6**. Pakiet danych jest formowany na podstawie tablicy (listy) *colors* zawierającej informacje o kolejnych kanałach. Dokładniej jest to tablica zawierająca piksele (reprezentujące całe kanały), tworzone za pomocą kilkukrotnego wywołania funkcji z listingu 5. Na początku do zmiennej pomocniczej kod (stanowiącej pakiet danych) wpisywany jest znak początku pakietu (wartość 128), zerowana jest wartość sumy kontrolnej i za pomocą funkcji *addSum* dodawana jest do niej wartość 128. Funkcja *addSum* zastępuje w tym przypadku „zwykłe" dodawanie, ale dodatkowo nie pozwala, aby wartość zmiennej *sum* wyszła poza zakres bajta. W dalszej kolejności z wykorzystaniem pętli po wszystkich elementach (pikselach) tablicy *colors* wykonywane są następujące czynności: z elementu są pozyskiwane informacje o nasyceniu każdej składowej koloru (RGB) w postaci liczb zmiennoprzecinkowych o wartości 0–1. W drugiej kolejności każda ze składowych podnoszona jest do kwadratu i mnożona przez 100, co pozwala odwzorować zależność natężenia świecenia od współczynnika wypełnienia przebiegu prostokątnego (jest to funkcja kwadratowa). W dalszej kolejności przeliczone składowe koloru w postaci znaków są dodawane do pakietu przeznaczonego do wysyłki oraz do sumy kontrolnej. Po wyjściu z pętli do zmiennej kod dodawana jest także suma kontrolna podzielona przez 2 i całość zostaje wysłana na port szeregowy.

Mirosław Firlej
elektronika@firlej.org
http://mirley.firlej.org/mirlight2