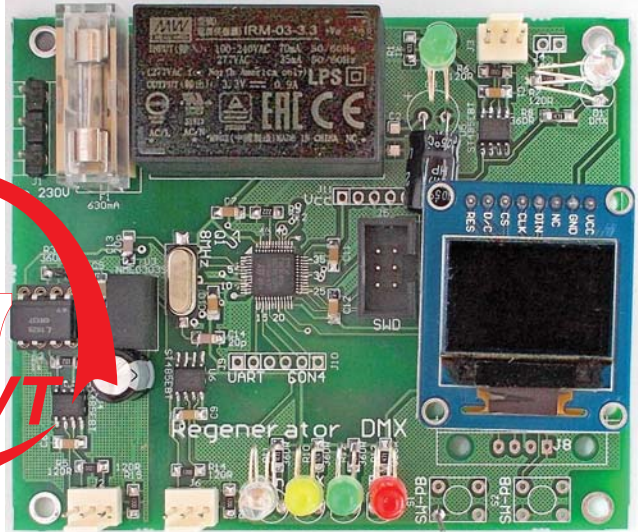
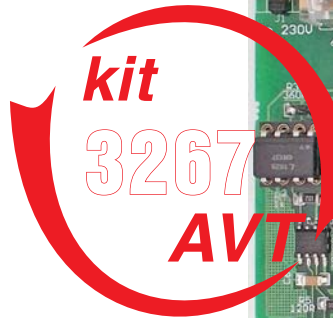




Regenerator sygnału DMX



Niezawodność transmisji łączem RS-485, np. DMX o znacznej długości zależy od wielu czynników. Jednym z nich jest jakość przewodów połączeniowych. Kolejnym czynnikiem wpływającym na transmisję sygnału są splittery, zwłaszcza z izolacją galwaniczną. Rozwiązaniem problemu jest opisany w artykule regenerator, który dodatkowo ma izolację galwaniczną.

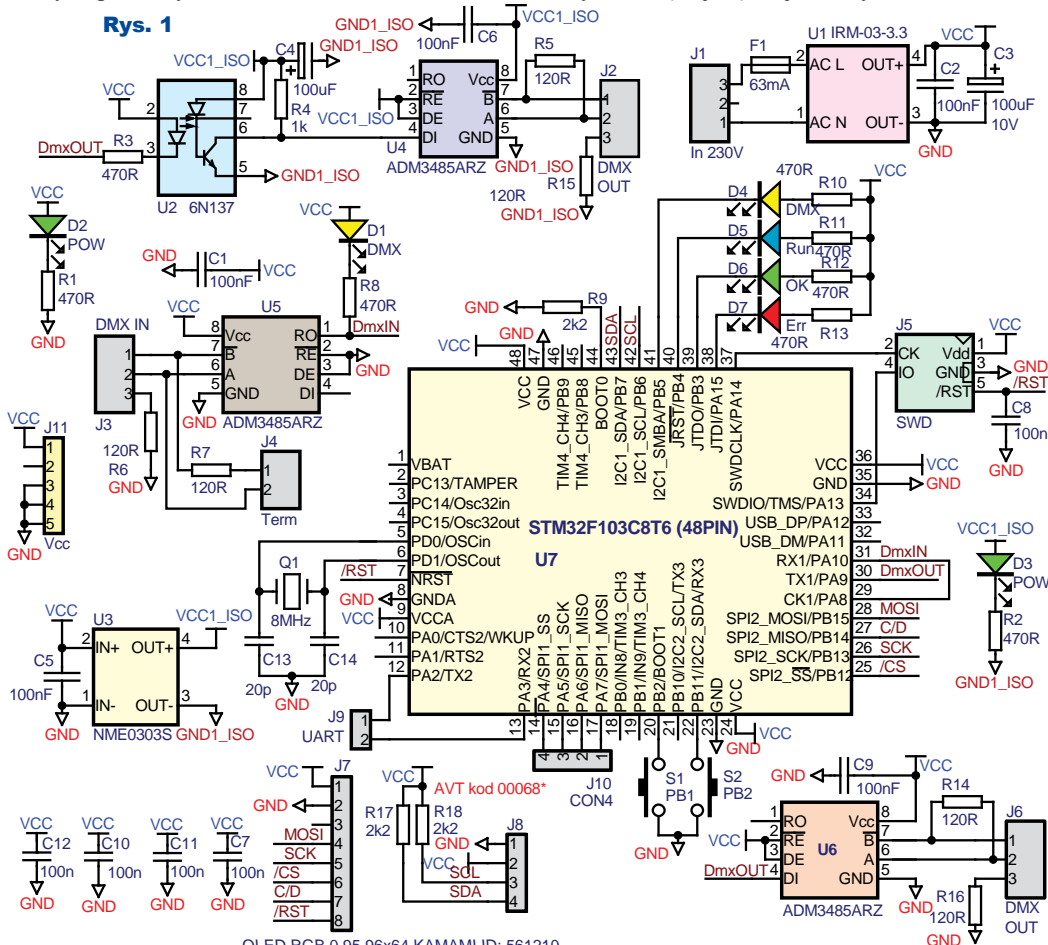
Opisywane urządzenie służy do poprawiania pewności transmisji sygnału DMX, który może zostać zniekształcony przez złą jakości okablowanie lub urządzenia dołączone do magistrali. Największe zniekształcenia mogą wprowadzić obwody izolacji galwanicznej. Często zawierają one transoptory, co może wydłużyć czas zmiany sygnału, zwłaszcza jednego ze zboczy, przez co czas trwania bitu ulega zmianie. Kilka takich urządzeń połączonych szeregowo może na tyle zmienić czas trwania bitu, że komunikacja będzie niemożliwa. Włączenie opisywanego regeneratora za splitterem rozwiąże problem. Regenerator ma dwa wyjścia, więc pełni także funkcję splittera, przy czym jeden kanał jest izolowany galwanicznie.

tem zasilacza zbudowanego klasycznie, ale sprawność zdecydowanie wyższa, czasem nawet dwukrotnie. Sygnał wejściowy DMX o poziomach zgodnych z RS485/422, doprowadzony do J3, jest konwertowany na poziomy TTL-LV driverem U5 (ADM3485ARZ). Skonwertowany sygnał trafia na wejście UART mikrokontrolera oraz steruje diodą D1. Mikrokontroler odebrane informacje wysyła na swoje wyjście. Dane wyjściowe są ponownie konwertowane na poziomy RS485/422 w U6. Dodatkowo,

ten sam sygnał steruje transoptorem U2, który przekazuje go dalej do drivera U4. Zasilanie U4 zapewnia U3 – przetwornica DC-DC NE0303S.

Aby zrozumieć zasadę działania urządzenia, trzeba zaznajomić się ze sposobem transmisji sygnału DMX. Sygnał jest przesyłany z prędkością 250kb/s, format ramki 8N2 przy wykorzystaniu poziomów napięć standardu RS485/422. Z tego powodu, w urządzeniach DMX używane są najczęściej drivery MAX485,

Rys. 1



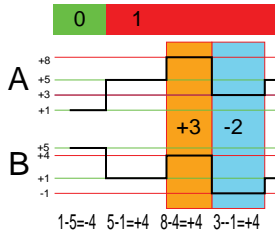
Opis układu

Schemat ideowy pokazany jest na rysunku 1. Układ zasilany jest przetwornicą AC/DC obniżającą napięcie sieciowe do 3,3V, z którego zasilane są wszystkie układy urządzenia. Przetwornica ma wyższą sprawność niż transformator małej mocy, mniejsze wymiary i nie wymaga dodatkowego stabilizatora, często z radiatorem. W konsekwencji koszt przetwornicy porównywalny jest z kosz-

OLED RGB 0,95 96x64 KAMAMI ID: 561210

MAX3485 lub inne, kompatybilne z nimi. W RS485 użyta jest transmisja różnicowa, odporna na zakłócenia.

Z czego wynika ta odporność? Na **rysunku 2** przedstawiono przykładowy przebieg wychodzący z nadajnika różnicowego. Górny przebieg to sygnał prosty, niezanegowany (wyjście A nadajnika), dolny – zanegowany (wyjście B). Przyjęto, że napięcia na wyjściu nadajnika mogą przyjmować poziomy +1 i +5V. Gdy nadawane jest „zero”, na linii A pojawia się napięcie 1V, na B +5V i różnica napięć wynosi -4V. W czasie nadawania „jedynek” na A panuje napięcie 5V, na B 1V i różnica wynosi +4V. Na prawej części rysunku przedstawiono sytuację, gdzie pojawia się impuls zakłócający o wartości +3V. Impuls ten pojawia się na obu liniach, przez co na A wystąpi napięcie 8V (5+3), na B 4V (1+3), jednak nadal różnica napięć wynosi +4V (8-4). Gdy impuls zakłócający ma wartość -2V, na A pojawi się napięcie +3V (5-2), na B -1V (1-2). Także w tym przypadku różnica nadal wynosi +4V.



Rys. 2

Napięcia na magistrali RS485/422 zależą od napięcia zasilania driverów, dlatego będą inne w przypadku układów zasilanych z 5V, inne dla 3,3V, a jeszcze inne w pozostałych przypadkach. Dla odbiornika same poziomy sygnałów nie są istotne, ważne by nie przekraczały dopuszczalnych wartości (najczęściej -5V...+12) względem masy układu. Odbiornikowi wystarczy różnica poziomów 200mV, aby zinterpretować sygnał.

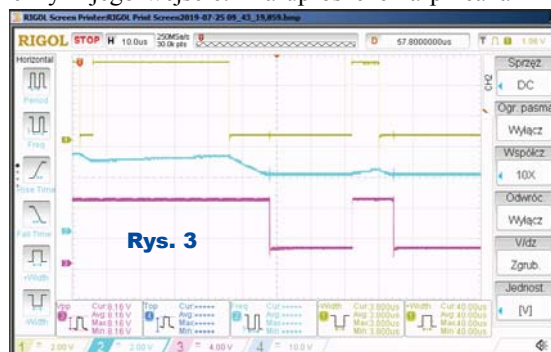
Istotne jest to, by przewód transmitujący dane był skrętką, która niejako „sprzętowo” tłumi zakłócenia w sposób podobny, jak to robi wzmacniacz różnicowy. Warunkiem dużego tłumienia sygnałów wspólnych jest użycie odpowiednich kabli połączeniowych. Kabel do systemów alarmowych/domofonów (YTDY) nie nadaje się, bo żyły nie są ze sobą skręcone, tylko ułożone równolegle. Dużo lepszy będzie kabel telefoniczny (YTKSY), gdzie żyły skręcone są co kilka cm, ale te kable nie zawsze skręcane są parami, tylko czasem czwórkami. Kable czwórkowe przeważnie mają większe pojemności (10 par i więcej) ale na przykład YTKZY1×4×0,5 ma dwie pary, a jest kablem czwórkowym. Trzeba to mieć na uwadze, bo różnica w oznaczeniu YTKSY i YTKZY jest niewielka, ale pełny symbol wszystko wyjaś-

nia, przykładowo parowy YTKSY2×2×0,5 i czwórkowy YTKZY1×4×0,5. Kable wyglądają praktycznie tak samo. Jeśli nie widać oznaczeń (częsta sytuacja, gdy kabel był używany i podczas wyciągania z koryt kablowych oznaczenia starły się), taki kabel należy rozszyć na odcinku 10...15cm i zobaczyć jak są skręcone przewody.

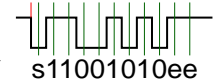
Dlaczego wspominam o kablach czwórkowych? W takim kablu w miarę skutecznie można wykorzystać połowę możliwości (po dwie żyły w czwórce). Jeszcze lepsza jest popularna „skrętka komputerowa” (kabel UTP), która może, ale nie musi być ekranowana. Najlepszy jest dedykowany kabel przeznaczony do transmisji DMX. Ma on najczęściej grubą izolację, czego typowy kabel UTP nie ma. **Nie powinien to być kabel mikrofonowy** z indywidualnie ekranowanymi żyłami, a jego wykorzystanie kusi, bo ma złącza takie same jak DMX. Kabel mikrofonowy i indywidualnie ekranowanymi żyłami nie jest przeznaczony do przesyłania szybkich sygnałów. Spowodowane jest to tym, że ma on stosunkowo dużą pojemność, która wynika z obecności ekranów.

Przy zastosowaniu skrętki może pojawić się problem tłumienia sygnału. Każdy przewód zachowuje się jak dolnoprzepustowy filtr RC. Użycie w urządzeniach DMX rozwiązań, które wprowadzają niesymetrię w narastaniu i opadaniu sygnału (przykładowo transoptory, kable o dużej pojemności), może powodować błędy transmisji. Przykładowy niesymetryczny przebieg pokazano na **rysunku 3**. Żółty przebieg to sygnał wejściowy, niebieski na transoptorze splittera, fioletowy to sygnał wyjściowy. Widać wyraźnie, że czas trwania jedynki logicznej uległ wydłużeniu.

Aby zrozumieć, dlaczego taki sygnał może powodować błędy, należy wiedzieć, jak UART odczytuje dane trafiające na jego wejście. Dla uproszczenia przeana-

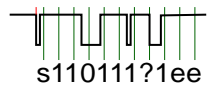


lizujemy sytuację, gdy brana jest jedna próbką, jak na przykład w software'owym UART dla Arduino. Na **rysunku 4** pokazano przykładowy sygnał. Czerwona kreska oznacza miejsce, od którego UART zacznie próbować sygnał wejściowy, pionowe zielone (pierwsza po połowie czasu trwania bitu) reprezentują kolejne momenty próbkowania. Litera „s” i „e” to bit startu i bity stopu. Dla ułatwienia analizy uwzględniłem próbkowanie w jednym punkcie, jak miało to miejsce w starszych konstrukcjach mikrokontrolerów (np. 8051). W nowszych sygnał próbkowany jest 3 (AVR) a nawet 8 czy 16 (ARM STM32) razy, co zwiększa odporność na błędy, ale ich nie eliminuje. Jak można wywnioskować, gdy sygnał narasta lub opada zbyt wolno, odbierana informacja zostanie sfalszowana. **Rysunek 5** przedstawia sygnał z **rysunku 2** z tym, że czas trwania jedynki jest wydłużony. Zależnie od właściwości UART, pierwszy impuls może nie być zinterpretowany jako bit startu, ale programowa realizacja UART najczęściej uzna go za start. Widać, że interpretacja poszczególnych bitów jest niepoprawna, a w przypadku przedostatniego bitu nie wiadomo, jak UART czy program zinterpretuje dane. Prawdopodobieństwo wystąpienia problemu zwiększa połączenie urządzeń w szereg. Każde z nich wprowadzi niewielkie opóźnienie ale sumaryczne może być na tyle duże, że dojdzie do błędów w komunikacji. W takiej sytuacji regeneratory mogą pomóc, bo błędy transmisji do około 2,5% UART odczyta poprawnie. Dane na wyjściu mają praktycznie 0% odchyłkę (najgorszy kwarc ma tolerancję 50ppm – 0,05%).



Rys. 4

W przypadku wystąpienia błędów w systemie DMX regeneratory należy włączyć w takim miejscu magistrali, gdzie dioda Error nie świeci. Tylko wtedy spełni on swoje zadanie.



Rys. 5

Program mikrokontrolera nie jest zbyt skomplikowany. Po zainicjowaniu UART, w przerwaniu dane zapisywane są do bufora „DmxIn”. Obecność sygnału BREAK stwierdza się na podstawie ustawienia bitu błędu ramki (FE). Procedura jest dość prosta, więc pozwolę ją sobie pokazać na **listingu 1**.

Procedura analizuje poprawność sygnału CS, mierzy czas trwania ramki, paazy pomiędzy ramkami, liczbę

```
void irqUart1(){
    uint8_t static err;

    //----- Czy BREAK ? -----//
    if ( USART1->SR & USART_SR_FE ){
        //USART1->SR &= ~USART_SR_FE; // INFO: HAL skasuje flage

        //----- Długość ramki (razem z CS )
        lenDmxCopy = lenDmxCopy + prtdmxCnt; prtdmxCnt = 0;

        //----- Sprawdzenie sygnału CS
        if( DmxCnt[ 0 ] ){
            dmxCnt |= ERR_DMxCNT;
            err |= ERR_DMxCNT;
        }
        DmxCnt[ 0 ] = 0;

        //----- Pomiar czasu trwania ramki, okresu, przerwy
        TimDmxCnt = TimDmxCntCnt; TimDmxCntCnt = 0;
        TimDmxCntPause = TimDmxCntPauseCnt; TimDmxCntPauseCnt = 0;
        TimDmxCntRamka = TimDmxCntOkres - TimDmxCntPause;

        //----- Sprawdzenie czy ramka ma wymagana długość
        if( lenDmxCopy < 24+1 ) {
            dmxCnt |= ERR_DMxCNT_LEN24;
            err |= ERR_DMxCNT_LEN24;
            lenDmxCopy = 24+1;
        }
        else{
            TimDmxCntTimeout = TimRestart = 100; // Ramka co najmniej 10 razy na sekundę
        }

        //----- Sprawdzenie ile kanałów w użyciu -----//
        for( int16_t x=lenDmxCopy-1; x>0; x--){
            if( DmxCnt[x] ) { uzywaneDmxCnt = x; break; } // jeśli wartość różna od 0
        }

        memcpy( DmxCntPrev, DmxCntCopy, lenDmxCopy ); // Zapamiętaj poprzednia ramkę
        memcpy( DmxCntCopy, DmxCnt, lenDmxCopy );
        if( !err ){
            err = 0;
            memcpy( DmxCntOut, DmxCntIn, lenDmxCopy ); // Gdy nie ma błędów kopiuje
        }
        uint16_t len = lenDmxCnt;
        if( len > (512+1) ) len = 513;
        dmxCntRoznica = 0;
        for(uint16_t x=0; x<lenDmxCnt; x++) if ( DmxCntPrev[x] != DmxCntCopy[x] ) dmxCntRoznica++;

        FL_RamkaComplete = true;
    }
    else if ( USART1->SR & USART_SR_RXNE ){ //----- Czy dana odbiorcza ? -----//
        //USART1->SR &= ~USART_SR_RXNE; // INFO: HAL skasuje flage

        TimDmxCntPauseCnt = 0;

        if( prtdmxCnt < DMxCNT_LEN ){
            DmxCntIn[prtdmxCnt++] = USART1->DR;
            err = 0;
        }
        else{
            dmxCntErr |= ERR_DMxCNT_LEN512;
            err = ERR_DMxCNT_LEN512;
        }
    }
}
```

```
if( FL_RamkaComplete ){
    FL_RamkaComplete = false;

    //----- Generowanie BREAK -----//
    // PA9 - TX USART1
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    GPIO_InitStructure.Pin = GPIO_PIN_9; // PA9
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure); // Port A

    DWT_Delay_us(88);

    GPIO_InitStructure.Mode = GPIO_MODE_AF_PP; // PA9 - funkcja alternatywna - USART1
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure); // Port A

    //----- Inicjacja wysłania danych -----//
    HAL_UART_Transmit_DMA( &uart1, DmxCntOut, lenDmxCntCopy );
}
```

łów i urządzenia, które akceptują transmisję. Gdy dwie są takie same, mogłyby działać z większą wydajnością. Taka opcja

delay, ale w razie potrzeby można zaangażować do tego celu timer. Po 88us przywracana jest funkcja alternatywna i uruchamiana trans-

jest łatwa do wprowadzenia, ponieważ w zmiennej „uzywaneDmxCnt” przechowywana jest informacja o ostatnim kanale, który zawiera wartość różną od zera. Oczywiście sama zmienna nie wystarczy do tego celu, trzeba jeszcze w inteligentny sposób zapamiętać tę wartość, aby ustawienie ostatnio używanego kanału na 0 nie zmniejszało liczby transmitowanych kanałów.

Listing 1

Pozostałe procedury programu są mniej istotne, bo zajmują się tylko wizualizacją danych na diodach LED i wyświetlaczu. Wysyłaniem danych do wyświetlacza zajmuje się DMA, więc w tym czasie program może bez problemu realizować inne funkcje (głównie spać po rozkazie WFI).

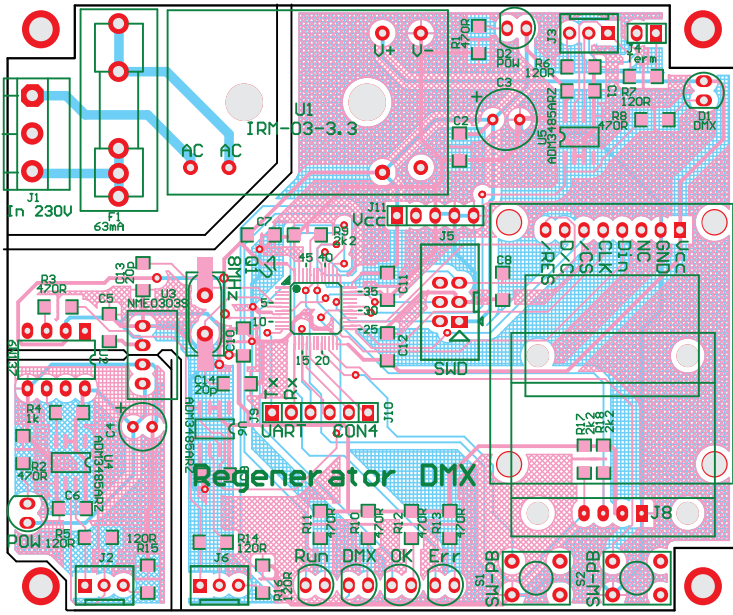
Montaż i uruchomienie

Układ można zmontować na płytce drukowanej, której projekt pokazany jest na **rysunku 6**. Standardowo montujemy układ, zaczynając od elementów najmniejszych, a kończąc na największych. Jeśli izolacja galwaniczna nie jest potrzebna, elementy z nią związane nie muszą być montowane, co znacznie obniży koszt urządzenia. **Rys. 6**

danych, czy zawiera się w dopuszczalnej granicy od 24 do 512 bajtów. Jeśli ramka ma poprawną budowę, zerowany jest timer programowy odliczający *timeout* po czym szacowane jest, ile kanałów jest używanych. Ramka jest następnie kopiowana do bufora „DmxCntCopy”. Jeśli nie wykryto błędów, dane są kopiowane do bufora „DmxCntOut”. Prosta funkcja określa, ile z kanałów zmieniło wartość od poprzedniej transmisji. W pętli głównej, co najmniej raz na milisekundę, wykonywany jest fragment programu pokazany na **listingu 2**.

W nim, gdy flaga odebrania całej ramki jest ustawiona, zostanie wygenerowany sygnał BREAK o czasie 88us. Realizowane jest to przez wyłączenie funkcji alternatywnej portu PA9, na którym ustawiony jest stan niski. Ze względu na prostotę programu i krótki czas sygnału BREAK pozwoliłem sobie na

misja z wykorzystaniem DMA. Warto zwrócić uwagę, że dane są wysyłane z bufora „DmxCntOut” gdzie znajdują się dane, w których błędy nie stwierdzono. Dzięki temu błędne ramki zapisane w buforze „DmxCntCopy” nie są transmitowane i zastępowane są przez ostatnią ramkę, w której błędów nie stwierdzono. Pozwala to wyeliminować pojedyncze błędy. Program można wyposażyć w trochę więcej „inteligencji”. Wiele programów komputerowych wysyła dane 512 kanałów, choć zwykle używa się ich mniej. Program mógłby przeprowadzać transmisję tylko do kanału, który jest modyfikowany. Pozwoliłoby to zwiększyć częstotliwość odświeżania kana-



Przyciski S1, S2 oraz złącza CON4, J8, J9 i J11 nie są używane. J9 przewidziano do podłączenia wyświetlacza monochromatycznego, ale program obsługuje tylko wyświetlacz kolorowy. Zworka na J3 pozwala włączyć

terminator linii, gdy regenerator jest ostatnim urządzeniem w łańcuchu DMX. Płytką drukowaną zaprojektowaną jest do obudowy KM-35. Fotografia wstępna pokazuje model. Uruchomienie układu polega na sprawdzeniu napięć zasilających. **Podczas uruchamiania należy zachować ostrożność, ponieważ w urządzeniu występuje napięcie niebezpieczne dla życia.** Jeśli zasilanie jest poprawne, należy wgrać program złączem J5. Do zaprogramowania mikrokontrolera można wykorzystać ST-Link V2. Jeśli program pracuje poprawnie, dioda D5 (Run) będzie migać z częstotliwością 1Hz, na wyświetlaczu pojawi się obraz – **rysunek 7**. Jeśli tak jest, do wejścia regeneratora można doprowadzić sygnał DMX. Na ekranie zobaczymy obraz podobny do tego na **rysunku 8**. W pierwszym wierszu wyświetlona jest liczba transmitowanych kanałów. Następnie po znakach „<>” liczba kanałów, których dane uległy zmianie od poprzedniej transmisji. Jeśli liczba kanałów będzie błędna, to zostanie wyświetlona na czerwono **rysunek 9**, dane jednak zostaną wysłane po dopełnieniu zerami jeśli liczba kanałów jest zbyt mała albo zostaną obcięte jeśli przekroczą 512. Gdy z powodu dużych liczb w wierszu nie zmieszczą się wszystkie teksty, zostanie usunięty znak „=” (**rysunek 10**), lub w razie konieczności ciąg „L=” (**rysunek 11**).

W drugim wierszu wyświetlana jest informacja o okresie powtarzania ramki danych oraz wyliczana częstotliwość jej odświeżania. Zależnie od niej różny jest kolor wiersza. Od zielonego dla odświeżania co najmniej 40Hz, przez cyan (rysunek 9) dla 33Hz, żółty (rysunek 10) dla 25Hz, pomarańczowy dla 20Hz, do czerwonego dla odświeżania poniżej 20Hz, którego niechlubnym przykładem jest program „MasterPeace OpenDmx”, a efekt jego pracy widać na **rysunku 12**.

| Oznaczenie | Nazwa | Kolor | Funkcja |
|------------|-------|-----------|--|
| D2 | POW | Zielona | Sygnalizuje obecność napięcia zasilającego |
| D1 | DMX | Żółta | Odwzorowuje sygnału na wejściu DMX |
| D4 | DMX | Żółta | Świeceniem sygnalizuje odbiór ramki DMX (obecność sygnału BREAK) |
| D5 | Run | Niebieska | Miga z częstotliwością 1Hz |
| D6 | OK | Zielona | Świeci gdy nie wykryto błędów w ramce DMX |
| D7 | Err | Czerwona | Świeci gdy wykryto błędy komunikacji lub błąd programu |

Tabela 1

```
L=-1/0<>0
t=0ms 0Hz
D=0ms P=0
Timeout
```

```
L=512/12<>0
t=26ms 38Hz
D=25ms P=1
```

```
L=12/12<>0
t=26ms 38Hz
D=25ms P=1
```

```
L512/189<>0
t=39ms 25Hz
D=24ms P=15
```

```
512/124<>112
t=40ms 25Hz
D=25ms P=15
```

```
L=512/0<>0
t=70ms 14Hz
D=24ms P=46
```

```
Error
Hard Fault
```

Rys. 7...13

Program regeneratora sygnalizuje wiele błędów. Jeden z nich można zobaczyć na **rysunku 13**. Błędów jest kilkanaście, ale nie ma sensu ich wszystkich tu wymienić, bo oznaczają one poważny błąd w programie urządzenia, którego sygnał jest nieprawidłowy. W przypadku ukazania się błędu należy skontaktować się z autorem takiego programu. Błąd jest wyświetlany przez około 10 sekund, po czym mikrokontroler jest resetowany. W czasie wyświetlania błędu miga napis „Error” oraz dioda D7 „Err”.

Urządzenie może pracować bez wyświetlacza. W takiej sytuacji stan urządzenia można odczytać z diod LED według **tabeli 1**.

W **tabeli 2** podany jest zestaw czasów transmisji danych oraz paazy pomiędzy transmisjami różnych programów. Pauza pomiędzy transmisjami wynika z niedoskonałości programu (systemu), który generuje sygnał DMX:

Proste programy nie są zbyt dobre, nie można ustawić liczby obsługiwanych kanałów, break jest bardzo długi (od 200us do kilku ms), przerwy pomiędzy paczkami danych są rzędu ms lub dziesiątek ms, co niekorzystnie wpływa na częstotliwość odświeżania danych. Ponadto większość wysyła dane do pierwszego znajdującego układu FTDI, którym ze względu na jego popularność może być

| Nazwa programu / urządzenia | Czas transmisji danych | Czas paazy między ramkami | Odświeżanie |
|-----------------------------|------------------------|---------------------------|-------------|
| DMX512 Light Control | 25ms | 15ms | 39ms 25Hz |
| MasterPeace OpenDmx | 25ms | 47ms | 70ms 14Hz |
| USB2DMX512DEMO | 25ms | 2ms | 27ms 38Hz |
| AVT5456 (24kanały) | | 0...1ms | 2ms 500Hz |

Tabela 2

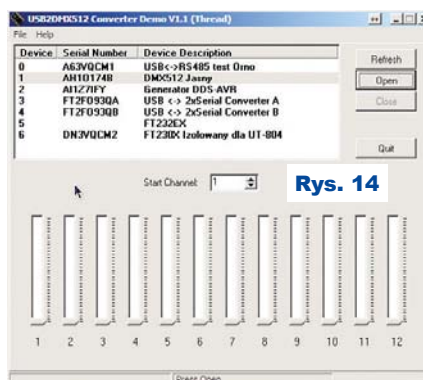
inne urządzenie niż konwerter USB-DMX. Ponadto nie sprawdza, czy konwerterem jest układ z UART i mimo że FT22x (USB-SPI) i FT20x (USB-I2C) nie mogą odczytać sygnału break, wysyłają dane do nich. Wyjątek stanowi program „USB2DMX512DEMO”, który pozwala wybrać interfejs, z którym program będzie się komunikował, a dodatkowo pokazuje tak zwane przyjazne nazwy – **rysunek 14**. Przy okazji ten program ma najkrótszy czas paazy między ramkami spośród testowanych programów dla Windows. Niestety, multitasking Windows jest, jaki jest i gdy komputer jest obciążony, wydajność programu drastycznie spada. Na tyle, że zakłócone jest wysyłanie danych. Inne programy w takiej sytuacji sprawują się lepiej. W przypadku zainteresowania szczegółami proszę o kontakt.

SaS
sas@elportal.pl

Wykaz elementów

Rezystory 1206:

- R1,R2,R3,R8,R10,R11,R12,R13 470R
- R4 1k
- R9,R17,R18 2k2
- R5,R6,R7,R14,R15 R16 120R
- C1,C2,C5-C12 100n kond. ceramiczny 1206
- C3,C4 100uF/10V kond. elektrolityczny ce8/35
- C13 C14 20p kond. ceramiczny 1206
- U1 IRM-03-3.3 przetwornica AC/DC
- U2 6N137
- U3 NME0303S
- U4 U5 U6 ADM3485ARZ
- U7 STM32F103C8T6
- D1 D4 Dioda LED 5mm żółta
- D2 D3 Dioda LED 5mm zielona
- D5 Dioda LED 5mm niebieska
- D6 Dioda LED 5mm zielona
- D7 Dioda LED 5mm czerwona
- Q1 Kvarc 8MHz
- F1 Bezpiecznik 63mA z oprawą
- J1 ARK3
- J2 J3 J6 SN25-W3P
- J4 Goldpin 2x1
- J5 T821-1-06-S1
- J7 OLED RGB 0,95 96x64 KAMAMI ID: 561210



Rys. 14

Płytką drukowaną jest dostępna w Sklepie AVT jako AVT3267