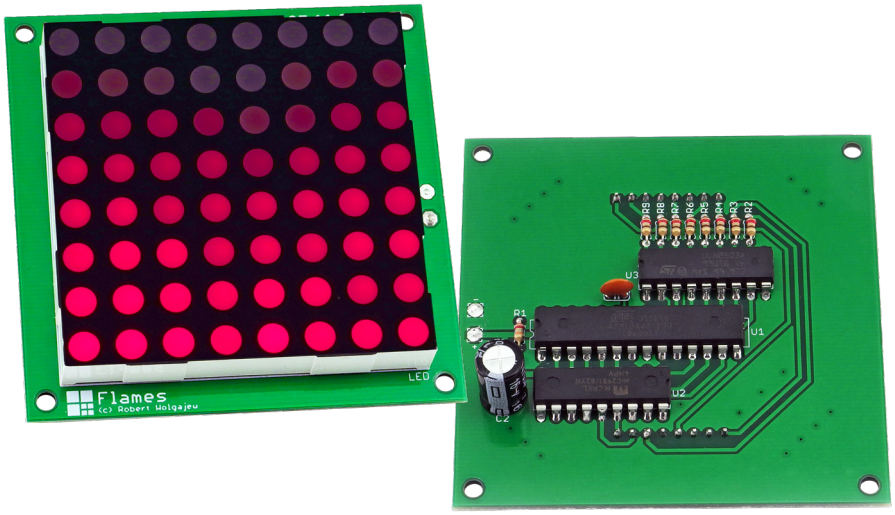
Flames

W mojej długoletniej praktyce elektronika-konstruktora wielokrotnie podejmowałem wyzwanie konstruowania różnorodnych systemów mikroprocesorowych, które zawierały wszelkiego rodzaju wyświetlacze. Nie powinno to specjalnie dziwić, gdyż elementy tego typu realizują niezwykle efektywny i efektowny sposób współpracy urządzenia z użytkownikiem. Tym razem zaprojektowałem bardzo prosty, lecz niezmiernie efektowny gadżet, którego jedynym zadaniem jest prezentacja dość wyszukanej animacji imitującej płomień.



Wyświetlacze matrycowe są bardzo ciekawe, jednak trzeba uczciwie przyznać, że zakres ich zastosowań jest dość specyficzny. Zwykle znajdują one zastosowanie we wszelkiego rodzaju systemach prezentacji danych, banerach reklamowych czy też systemach wielkoformatowej wizualizacji. Niestety żaden taki system w ostatnich latach nie stał się źródłem moich inspiracji. Postanowiłem to jednak zmienić. Tak powstał projekt, który nazwałem w dość wymowny sposób, a mianowicie flames.

Budowa i działanie

Schemat systemu został pokazany na **rysunku 1**. Jest to bardzo prosty system mikroprocesorowy, którego sercem a zarazem elementem odpowiedzialnym za realizację całej, założonej funkcjonalności jest niewielki mikrokontroler firmy Microchip (dawniej Atmel) o oznaczeniu ATmega48. Wybór tego konkretnego modelu z szerokiej palety układów mikroprocesorowych wspomnianego producenta podyktowany był wyłącznie potrzebą zastosowania komponentu w obudowie przewlekanej mającego wymaganą liczbę wyprowadzeń (portów komunikacyjnych).

Mikrokontroler jest odpowiedzialny wyłącznie za realizację jednego zadania, a mianowicie za obsługę wyświetlacza matrycowego o organizacji 8×8 diod LED. Zadanie to realizuje z użyciem przerwań systemowych, których potrzeba stosowania wynikała z multipleksowanego mechanizmu sterowania pracą wyświetlacza. Sterowanie takie polega na okresowym wystawianiu na port wierszy (PORTB, aktywny stan wysoki) kombinacji stanów odpowiadających poszczególnym diodom LED w wybranej kolumnie i odpowiednim, cyklicznym sterowaniu portem kolumn (PORTD, aktywny stan wysoki) tak, aby sekwencje wystawione na port wierszy odpowiadały żądanej kolumnie. Mikrokontroler jest taktowany wewnętrznym, wysokostabilnym generatorem RC o częstotliwości 8 MHz. To wystarczy, aby cykliczne przełączanie portu kolumn odbywało się 400 razy na sekundę (co 2,5 ms), co implikuje częstotliwość odświeżania każdej kolumny na poziomie 50 Hz. To w zupełności wystarczy, aby zjawisko migotania diod LED (będącego wynikiem ciągłego ich przełączania) nie było widoczne.

Opisany mechanizm jest niezwykle prosty i stosowany bardzo często, gdy zachodzi potrzeba sterowania wieloma wyświetlaczami LED (7-segmentowymi czy matrycowymi) przy użyciu ograniczonej liczby wyprowadzeń mikrokontrolera. Wszystko wyglądałoby dokładnie tak, jak opisano powyżej, gdyby nie jeden szczegół. Sytuacja komplikuje się, gdy zachodzi potrzeba regulacji jasności każdej z diod LED wymuszona chęcią jak najbardziej realistycznej symulacji efektu płomienia. Załóżmy, że chcemy, aby istniała możliwość regulacji jasności każdej diody LED w wierszu, powiedzmy w 16 krokach. Co należy zrobić? To całkiem proste, a rozwiązanie nasuwa się automatycznie. Cały czas, przez jaki procesor steruje wyświetlaczem matrycowym, podzielono na 8 następujących po sobie slotów, każdy o długości 2,5 ms, powtarzanych 50 razy na sekundę. Należy każdy taki slot podzielić na 16 mniejszych slotów, każdy trwający ok. 156 µs. W ramach tych mniejszych slotów trzeba zdecydować, jak długo wystawiany jest na port wierszy każdy ze stanów logicznych poszczególnych bitów wybranej kolumny wyświetlacza. Stosowne przerwanie systemowe, w naszym przypadku od porównania licznika Timer0 z rejestrem porównania OCR0A, musi być wywoływane nie 400, a 6400 razy na sekundę (wspomniane wcześniej 400 Hz × 16 poziomów jasności). W ramach funkcji

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT5846

Podstawowe parametry:
- efektowny gadżet świetlny – imitacja płomienia,
- obsługa wyświetlacza matrycowego LED o organizacji 8×8,
- niezależne sterowanie jasnością każdej z diod matrycy, w 16-stopniowej skali,
- zasilanie 5 V, min. 0,5 A.

Projekty pokrewne na www.media.avt.pl:

----	Modułowa matryca LED (EP 4/2019)
AVT-5606	Magic Matrix (EP 10-11/2017)
AVT-3184	Zegar/termometr z wyświetlaczem matrycowym (EdW 11/2017)
AVT-5561	Efektowny sterownik oświetlenia EP 12/2016)
AVT-2784	Zegar matrycowy (EdW 4/2006)

* Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!
Podstawową wersją zestawu jest wersja [B] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [B] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [C] – zmontowany, uruchomiony i przetestowany zestaw [B] (elementy wlutowane w płytkę PCB)
- wersja [A] – płytka drukowana bez elementów i dokumentacji

Kity w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- wersja [A+] – płytka drukowana [A] + zaprogramowany układ [UK] i dokumentacja
- wersja [UK] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! http://sklep.avt.pl. W przypadku braku dostępności na http://sklep.avt.pl, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.

Wykaz elementów:

Rezystory: miniaturowe 0,25 W
R1: 22 kΩ
R2…R9: 120 Ω (wartości rezystorów należy dobrać z zakresu 62…120 Ω w zależności od oczekiwanej, maksymalnej jasności diod LED)

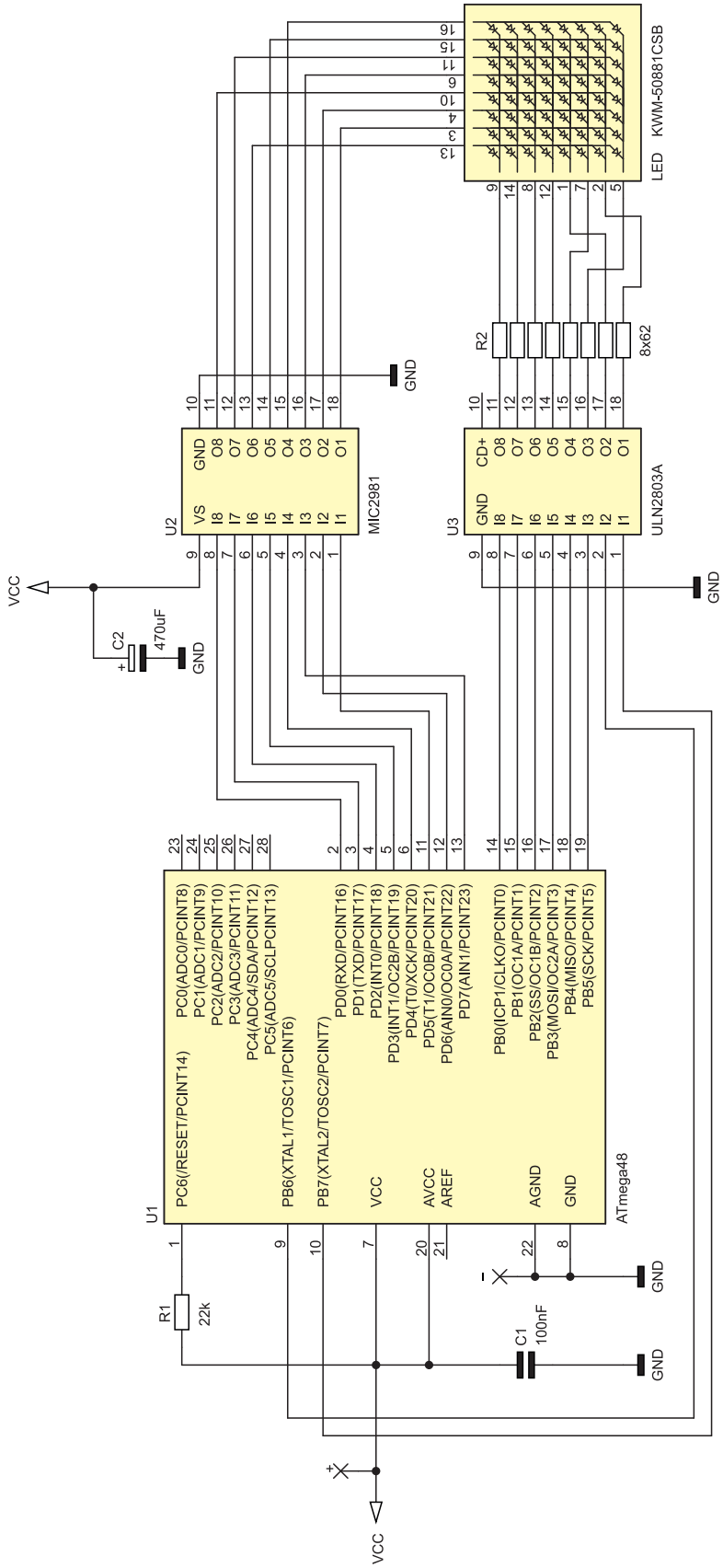
Kondensatory:
C1: 100 nF ceramiczny
C2: 470 µF/10 V

Półprzewodniki:
LED: wyświetlacz matrycowy KWM-50881CSB
U1: ATmega48 (DIL-28)
U2: MIC2981 (DIL-18)
U3: ULN2803A (DIL-18)

obsługi przerwania każde 16 kolejnych wywołań obsługuje jedną kolumnę wyświetlacza LED, realizując regulację jego jasności w zakresie od 0/16 do 15/16 (osobno dla każdego z bitów portu).

Wspomniane 16 kolejnych wywołań składa się na jeden 2,5 ms slot powtarzany 400 razy na sekundę. Jeśli brzmi to trochę zawile, to **rysunek 2** wyjaśni wszelkie wątpliwości. Jak widać, dla przykładowej, pierwszej kolumny wyświetlacza matrycowego pokazano sygnały sterujące na porcie kolumn i wierszy dla przypadku, gdy każda kolejna dioda tej kolumny ma coraz większą jasność (od 0/16 do 7/16). Podobnie dla drugiej kolumny pokazano sygnały sterujące na porcie kolumn i wierszy dla przypadku, gdy każda kolejna dioda tej kolumny ma coraz większą jasność (od 8/16 do 15/16). Myślę, że dalszy komentarz jest zbyteczny.

Pozostaje jeszcze jedna ważna kwestia. Każda kolumna wyświetlacza jest załączana maksymalnie na 125 ms (2,5 ms × 50). Każda z jej diod świeci z jasnością średnią odpowiadającą 1/8 wartości, gdyby była załączona na stałe. Wynika z tego, że aby nie pogorszyć znacząco i tak niezbyt jasnych diod matrycy LED (8…12 mcd), należy sterować je większym prądem. To z kolei oznacza, że możliwości portów I/O mikrokontrolera są w tym przypadku niewystarczające i to nawet biorąc pod uwagę fakt, że wartość tego



Rysunek 1. Schemat ideowy urządzenia

Listing 1. Definicje połączeń portów oraz tablic umieszczonych w pamięci Flash ułatwiających dostęp do tych portów

```
//Definicje połączeń portów wierszy i kolumn
#define ROWS 8
#define COLS 8
#define GRAYSCALE_LEVELS 16

#define ROW_PORT_REG PORTB
#define ROW_DDR_REG DDRB
#define COL_PORT_REG PORTD
#define COL_DDR_REG DDRD

#define ROW0 PB5
#define ROW1 PB7
#define ROW2 PB4
#define ROW3 PB6
#define ROW4 PB3
#define ROW5 PB2
#define ROW6 PB1
#define ROW7 PB0

#define COL0 PD2
#define COL1 PD5
#define COL2 PD6
#define COL3 PD0
#define COL4 PD7
#define COL5 PD1
#define COL6 PD3
#define COL7 PD4

//Definicje zmiennych operujących
//na portach wierszy i kolumn
const uint8_t rowBits[ROWS] PROGMEM = {
(1<<ROW0), (1<<ROW1), (1<<ROW2), (1<<ROW3),
(1<<ROW4), (1<<ROW5), (1<<ROW6), (1<<ROW7)};

const uint8_t colBits[ROWS] PROGMEM = {
(1<<COL0), (1<<COL1), (1<<COL2), (1<<COL3),
(1<<COL4), (1<<COL5), (1<<COL6), (1<<COL7)};
```

Listing 2. Definicje tablic opisujących zawartość ekranu wyświetlacza matrycowego

```
//Bufor wyświetlanych danych
uint8_t displayBuffer[ROWS][COLS];
//Bufor edytowanych danych
uint8_t editBuffer[ROWS][COLS];
```

Listing 3. Funkcja inicjująca mechanizm obsługi wyświetlacza matrycowego

```
void initPWM(void){
    //Port kolumn, jako wyjściowy
    //z nieaktywnym stanem 0x00
    COL_DDR_REG = 0xFF;
    //Port wierszy, jako wyjściowy
    //z nieaktywnym stanem 0x00
    ROW_DDR_REG = 0xFF;
    //Konfigurujemy Timer0
    //by przepełniał się ok. 6400 razy na sekundę
    //Tryb CTC
    TCCR0A = (1<<WGM01);
    //Porównanie ok. 6400 razy na sekundę
    OCR0A = 155;
    //Preskaler = 8
    TCCR0B = (1<<CS01);
    //Zezwolenie na przerwanie
    //Timer0 compare match A
    TIMSK0 = (1<<OCIE0A);
}
```

prądu ustawiono na niewiele, bo tylko 50 mA (z dostępnych i bezpiecznych dla wyświetlacza 100 mA, przy wypełnieniu równym 1/10). Dotyczy to zarówno portu wierszy (PORTB), jak i tym bardziej portu kolumn (PORTD), gdzie sumaryczny prąd może sięgać wartości 400 mA. Z tego powodu zastosowano drivery (U2 i U3), które rozwiązują ten problem. Dla dociekliwych Czytelników podpowiem, iż maksymalną jasność wbudowanych diod LED możemy regulować poprzez zmianę wartości rezystorów R2…R9 z zakresu 62…120 Ω, co jednocześnie wpłynie na sumaryczny prąd pobierany przez urządzenie ze źródła zasilania.

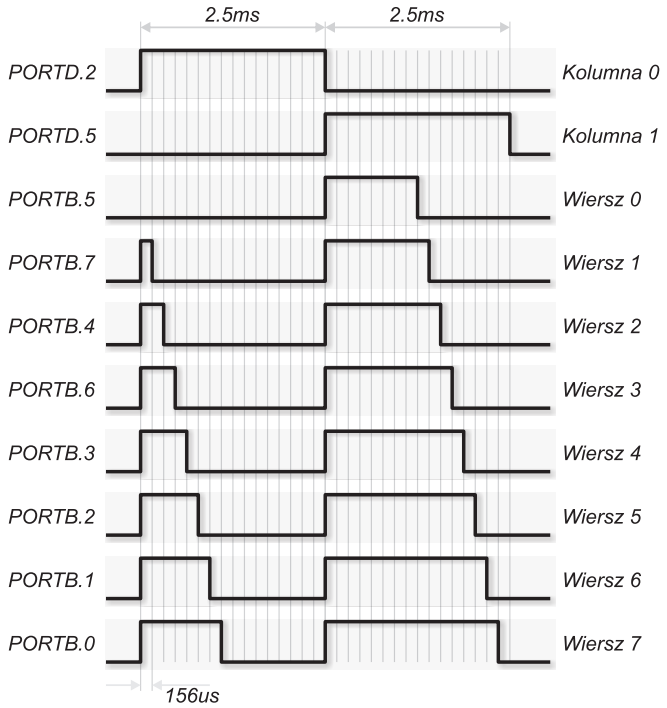
Program sterujący

Definicje połączeń portów oraz tablic umieszczonych w pamięci Flash ułatwiających dostęp do portów zostały pokazane na **listingu 1**.

Teraz powołamy do życia dwie zmienne tablicowe (tzw. bufory) opisujące zawartość pokazywaną przez wyświetlacz matrycowy. Dlaczego dwie? Pierwsza z nich to tablica, na której operuje funkcja obsługi przerwania odpowiedzialna za obsługę wyświetlacza matrycowego. Jest to bieżąca zawartość ekranu, do której dostęp z poziomu funkcji *ISR* dokonywany jest dość często, bo 6400 razy na sekundę. Druga tablica służy do wykonywania wszelkich operacji obliczeniowych, które utworzą nową zawartość ekranu. Jednym ruchem, za pomocą prostej funkcji narzędziowej, możemy przepisać jej zawartość do tablicy bieżącej (pierwszej). Definicje obu tablic pokazano na **listingu 2**. Warto podkreślić, że choć typ elementów tablicy jest zadeklarowany, jako *uint8_t* (czyli bajt bez znaku), to zakres ich wartości musi się mieścić w przedziale 0…15, gdyż opisuje on jasność każdej z diod wyświetlacza.

Na **listingu 3** została pokazana prosta funkcja inicjująca mechanizm obsługi wyświetlacza matrycowego. Ustawia ona odpowiednie kierunki portów wierszy i kolumn oraz inicjuje układ czasowo-licznikowy Timer0 mikrokontrolera, który jest odpowiedzialny za mechanizm multipleksowania.

Najważniejsza funkcja, czyli funkcja obsługi przerwania od porównania wartości licznika Timer0 z rejestrem porównania OCR0A, jest odpowiedzialna za obsługę multipleksowania wyświetlacza matrycowego oraz regulację jego jasności. Kod funkcji został pokazany na **listingu 4**.



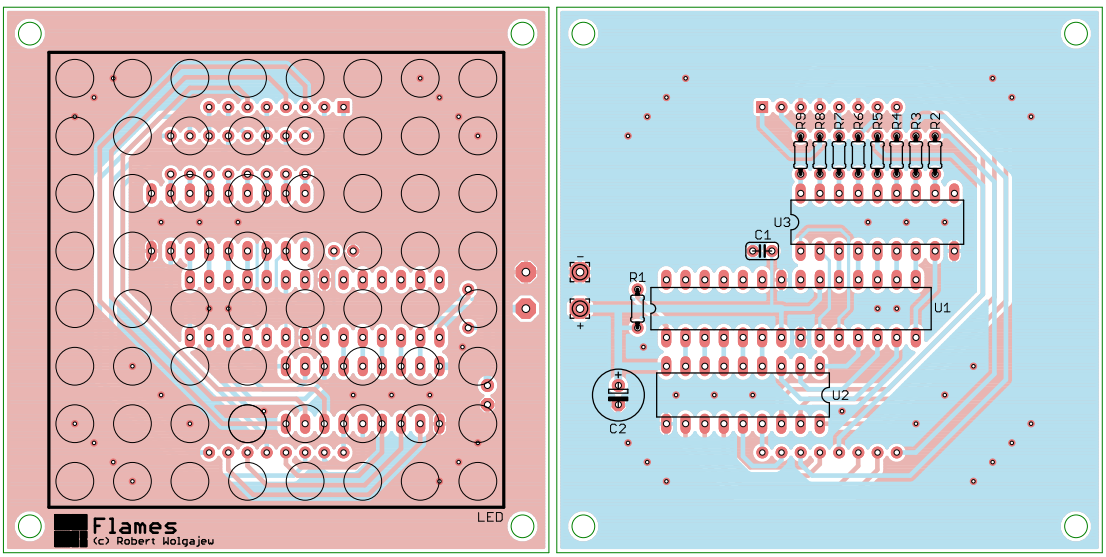
Rysunek 2. Diagram prezentujący sposób regulacji jasności poszczególnych diod matrycy LED

Listing 4. Funkcja obsługi przerwania odpowiedzialna za obsługę multipleksowania wyświetlacza matrycowego oraz regulację jego jasności

```
ISR(TIMER0_COMPA_vect){
    static uint8_t Column;
    static uint8_t Counter;
    uint8_t rowValue = 0;

    //Obliczenie wartości portu wierszy
    //na podstawie wartości PWM
    //dla poszczególnych diod wyświetlanej kolumny
    for(uint8_t Row=0; Row < ROWS; ++Row){
        if(Counter < displayBuffer[Row][Column]){
            rowValue |= pgm_read_byte(&rowBits[Row]);
        }
    }
    //Wystawienie wartości na port wierszy
    ROW_PORT_REG = rowValue;
    //Załączenie odpowiedniej kolumny
    //na początku każdego slotu 2.5ms
    if(Counter == 0){
        COL_PORT_REG = pgm_read_byte(&colBits[Column]);
    }

    if(++Counter == GRAYSCALE_LEVELS){
        Counter = 0;
        if(++Column == COLS) Column = 0;
        //Wyłączenie portu kolumn,
        //gdyż za chwilę będziemy pokazywać kolejną kolumnę
        COL_PORT_REG = 0x00;
    }
}
```



Rysunek 3. Schemat płytki PCB

Listing 5. Funkcja narzędziowa operująca na buforach obrazu wyświetlacza matrycowego

```
void showFrame(void){
    ATOMIC_BLOCK(ATOMIC_RESTORESTATE){
        memcpy(displayBuffer, editBuffer, sizeof(editBuffer));
    }
}
```

Ustawienia Fuse-bitów:

SUT1: 1	CKSEL1: 1
SUT0: 0	CKSEL0: 0
CKSEL3: 0	CKDIV8: 1
CKSEL2: 0	

Listing 6. Funkcja generująca imitację płomienia na wyświetlaczu matrycowym

```
void drawFlames(void){
    uint8_t Brightness;

    //Animacja ognia
    for(uint8_t Row=0; Row<ROWS-1; ++Row){
        for(uint8_t Col=0; Col<COLS; ++Col){
            Brightness = editBuffer[Row+1][Col] << 1;
            Brightness += editBuffer[Row][Col] >> 1;
            Brightness += (Col? editBuffer[Row+1][Col-1] : editBuffer[Row+1][COLS-1]) >> 1;
            Brightness += (Col == COLS-1? editBuffer[Row+1][0] : editBuffer[Row+1][Col+1]) >> 1;
            Brightness >>= 2;
            editBuffer[Row][Col] = Brightness;
        }
    }

    //Dodanie przypadkowych punktów na górze animacji
    for(uint8_t Col=0; Col<COLS; ++Col){
        Brightness = (rand()>0xB0)? rand()&0x0F : (editBuffer[ROWS-1][Col]?     editBuffer[ROWS-1][Col]-1 : 0);
        editBuffer[ROWS-1][Col] = Brightness;
    }
}
```

Wreszcie na **listingu 5** pokazano prostą funkcję narzędziową umożliwiającą przepisanie zawartości bufora obrazu poddawanego edycji do bufora pamięci ekranu wyświetlacza matrycowego, na którym operuje funkcja obsługi przerwania, pokazana wcześniej na listingu 4. Jak widać, przepisanie zawartości buforów ujęto w sekcję *ATOMIC_BLOCK*, przez co cała operacja ma charakter atomowy z punktu widzenia programu aplikacji (nie zostanie przerwana przez jakiekolwiek przerwanie systemowe).

Mamy już wszystkie funkcje pozwalające na obsługę wyświetlacza matrycowego zastosowanego w naszym projekcie. Brakuje wyłącznie jednej bardzo ważnej funkcji, a mianowicie funkcji generującej imitację płomienia na wspomnianym wyświetlaczu LED. Funkcja bazuje na materiale źródłowym dostępnym pod adresem: http://bit.ly/2YWbMR1, a jej kod został pokazany na **listingu 6**. Uważny i bardziej doświadczony Czytelnik zauważy zapewne, że w mechanizmach sterowania jasnością wyświetlacza matrycowego, które bazują na modulacji PWM, nie zaimplementowano korekcji gamma. Funkcjonalność ta poprawiłaby rozróżnienie poszczególnych odcieni czerwieni diod LED przez oko ludzkie. Nie została zastosowana z prostego powodu. Efekt płomienia wydawał się zdecydowanie lepszy dla wersji bez korekcji gamma, więc ostatecznie porzucono to opcjonalne rozwiązanie.

Montaż i uruchomienie

Schemat montażowy urządzenia został pokazany na **rysunku 3**. Zaprojektowano dwustronną, niewielką płytkę drukowaną z zastosowaniem wyłącznie elementów przewlekanych montowanych po obu stronach obwodu drukowanego. Budowę urządzenia rozpoczynamy od warstwy BOTTOM, gdzie w pierwszej kolejności montujemy wszystkie rezystory, następnie kondensatory, a na samym końcu układy scalone. Następnie przechodzimy na warstwę TOP, gdzie montujemy nasz wyświetlacz matrycowy, pamiętając o odpowiednim jego pozycjonowaniu (pin oznaczony jako 1 wyróżniono kwadratowym kształtem pola lutowniczego). Poprawnie zmontowany układ nie wymaga żadnych regulacji i powinien działać od razu po włączeniu zasilania. Dobierając stosowny zasilacz, należy mieć na uwadze niezbędną wydajność prądową (minimum 410 mA, w przypadku rezystorów R2…R9=62 Ω) oraz dostarczany poziom napięcia (5 V). Fotografie nie oddają prawdziwego efektu działania urządzenia, dlatego warto zobaczyć nagranie dołączone do materiałów dodatkowych.

W tym miejscu warto podkreślić, iż urządzenie flames oprócz swojej podstawowej funkcji gadżetu wizualnego może być niezłą bazą układową do własnych eksperymentów z tak sterowanym wyświetlaczem matrycowym, zwłaszcza że wprowadzono możliwość niezależnego sterowania jasnością każdej z diod matrycy.

Robert Wołgajew, EP