KoNaR

KOŁO NAUKOWE ROBOTYKÓW

Sterownik diod RGB

Robert Budziński

Wrocław, 4 marca 2009

Spis treści

1. Wstęp

Różnego rodzaju efekty świetlne przykuwają uwagę i budzą zainteresowanie obserwatorów. Są stałym elementem stoisk, ekspozycji czy pokazów. Pojawienie się diod RGB umożliwiło projektowanie ciekawych efektów wizualnych przy stosunkowo małym nakładzie kosztów i pracy. Dioda RGB posiada strukturę umożliwiającą generowanie trzech podstawowych barw (czerwony, zielony, niebieski), a co za tym idzie, przez możliwość ich mieszania, praktycznie dowolnej barwy.

Celem projektu była realizacja prostego i wydajnego sterownika obsługującego większą ilość diod RGB. Zaprojektowane urządzenie jest przystosowane do obsługi 40 diod, jednak niewykorzystane zasoby użytego mikrokontrolera i zastosowana idea umożliwiają przystosowane sterownika do podłączenia kolejnych 24 diod.

W niniejszym raporcie opisałem ideę działania sterownika, jego budowę, obsługę programową oraz przykładowe efekty.

2. Idea działania

W projekcie założono wykorzystanie diod o wspólnej anodzie (CA - common anode). Diody połączono w matrycę o organizacji 8 wierszy i 5 kolumn. Katody struktur poszczególnych kolorów dołączono do wyprowadzeń wierszowych, natomiast anody diod do wyprowadzeń kolumnowych (schemat połączeń przedstawiono na rys.5).

Opisana konfiguracja umożliwia wysterowanie 40 (8x5) diod RGB, co w praktyce oznacza 120 diod (po 40 dla poszczególnych barw). Do tego celu wykorzystano 29 (8+8+8+5) linii mikrokontrolera. Konfiguracja ta wymusza konieczność sterowania multipleksowego. Podając stan wysoki na wyprowadzenie kolumny i stan niski na wyprowadzenie wiersza można zaświecić diodę leżącą na skrzyżowaniu tych linii. Każde z połączeń wierszowych danej barwy podłączone jest z osobnym ośmiobitowym portem mikrokontrolera. Wyświetlanie multipleksowe polega na wpisywaniu do rejestru wyjściowego danego portu sekwencji, która zostanie wyświetlona w pierwszej kolumnie diod, po czym załączona zostaje pierwsza kolumna. Po upływie odcinka czasu rzędu milisekund następuje wyłączenie kolumny, natomiast do rejestru zostaje wpisana sekwencja, która zostanie wyświetlona w drugiej kolumnie. Druga kolumna zostaje załączona na pewien czas. Czynność powtarza się kolejno dla trzeciej, czwartej i piątej kolumny, po czym cykl rozpoczyna się od początku. Odpowiednio duża częstotliwość załączania kolumn (min. $5 \cdot 25Hz = 125Hz$) sprawi, że będzie ono widoczne jako świecenie ciągłe.

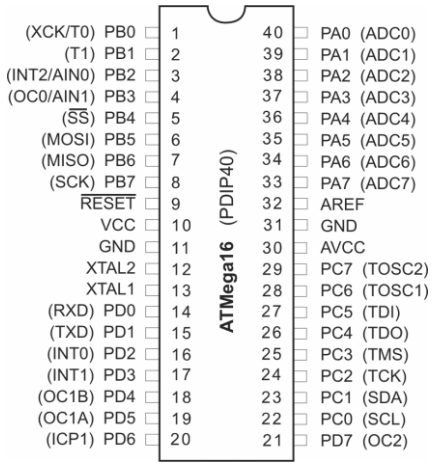
Multipleksowanie diod RGB odbywa się w tle (podczas realizowania programu głównego) a jego obsługa umieszczona jest w podprogramie obsługi przerwania generowanego przez Timer. Stan poszczególnych diod określa 15 zmiennych typu całkowitego (R1…R5, G1...G5, B1...B5). Dzięki użyciu przerwania, każda zmiana wartości danej zmiennej znajduje natychmiast odzwierciedlenie w sekwencji

zapalonych diod. Zapis binarny pozwala w prosty sposób definiować tę sekwencję, a wykorzystanie operacji bitowych upraszcza projektowanie efektów świetlnych.

Zastosowanie scalonego drivera mocy ULN2803 między portami mikrokontrolera a wyprowadzeniami wierszy matrycy powoduje, że zaświecenie diody uzyskujemy wysokim poziomem napięcia na liniach portu. Analogicznie, z powodu zastosowania tranzystorów p-n-p, załączenie kolumny odbywa się przez podanie poziomu niskiego. Rozwiązanie to, oprócz zabezpieczenia mikrokontrolera przed nadmiernym obciążeniem prądowym, umożliwia zastosowanie logiki dodatniej podczas sterowania diodami i projektowania efektów. Przykładowo wpisanie do rejestru wyjściowego portu wartości *0b10000011* spowoduje zapalenie diody pierwszej, drugiej i ostatniej.

3. Opis elektroniki

Sterownik oparty jest na mikrokontrolerze ATmega16 z rdzeniem AVR. Jest to ośmiobitowa jednostka z 16kB pamięci flash programowanej w systemie (ISP). Posiada 32 konfigurowalne linie wejścia/wyjścia. Taktowany jest zewnętrznym rezonatorem kwarcowym o częstotliwości 16MHz.



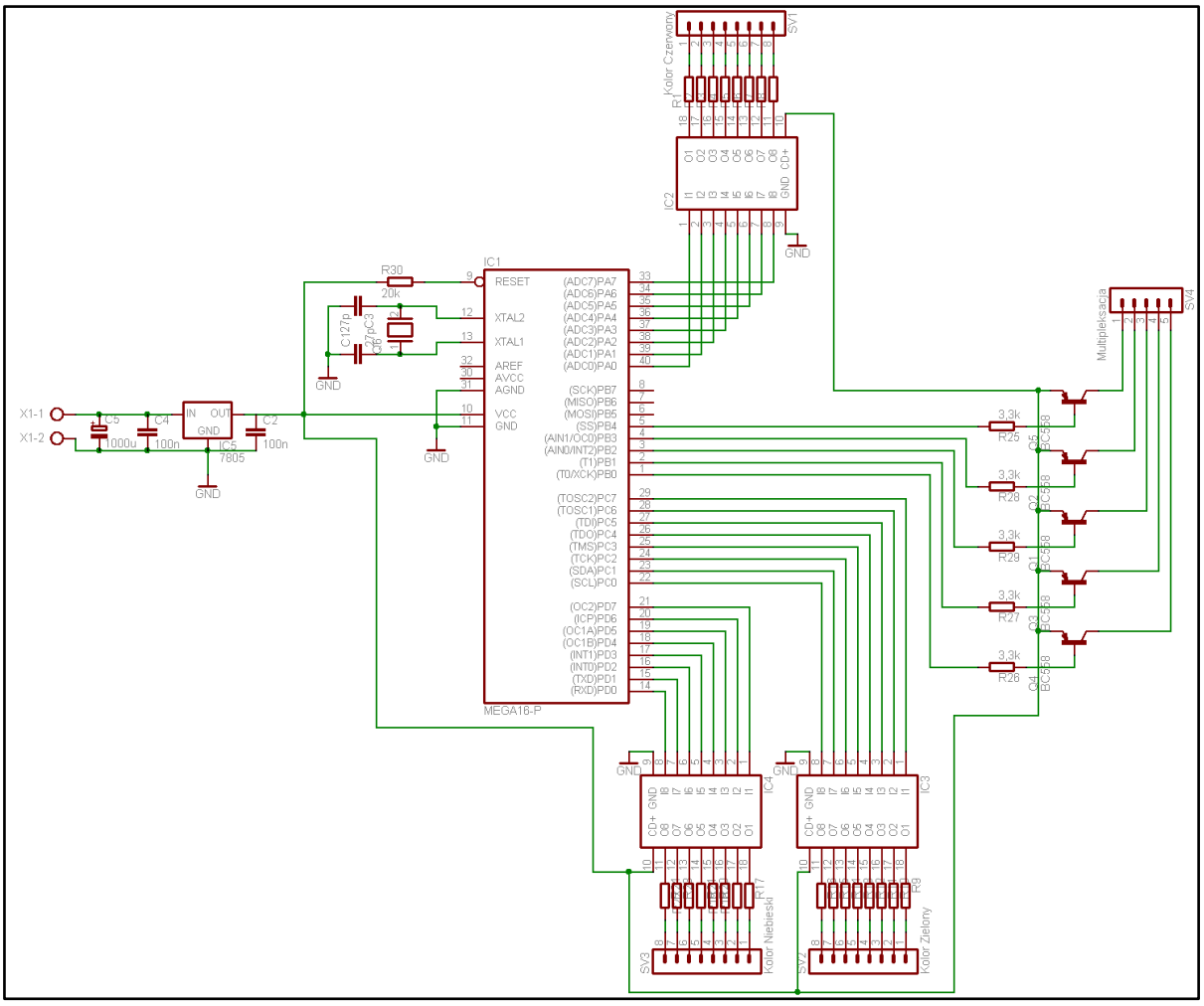
Rys.1. Opis wyprowadzeń mikrokontrolera ATmega16

W układzie wykorzystano 29 linii mikrokontrolera. Porty PORTA, PORTC i PORTD połączono z wyprowadzeniami wierszy matrycy diod odpowiednio do kolorów: czerwonego, zielonego i niebieskiego. Linie PB0…PB4 sterują kolumnami za pomocą tranzystorów BC558 (p-n-p). Linie PB5…PB7 (MOSI, MISO, SCK) pozostały wolne i służą wraz z liniami RESET, VCC i GND do podłączenia programatora szeregowego. Po drobnej modyfikacji układu można je wykorzystać do zwiększenia ilości obsługiwanych diod o kolejne 24 sztuki. W połączeniu portów mikrokontrolera z katodami diod pośredniczą układy ULN2803. Jest to scalony driver mocy. Składa się z ośmiu tranzystorów Darlingtona (n-p-n) zwierających obciążenie do masy. Podanie stanu wysokiego na wejście skutkuje wymuszeniem stanu niskiego na wyjściu. Obciążalność układu ULN2803 wynosi 500mA.

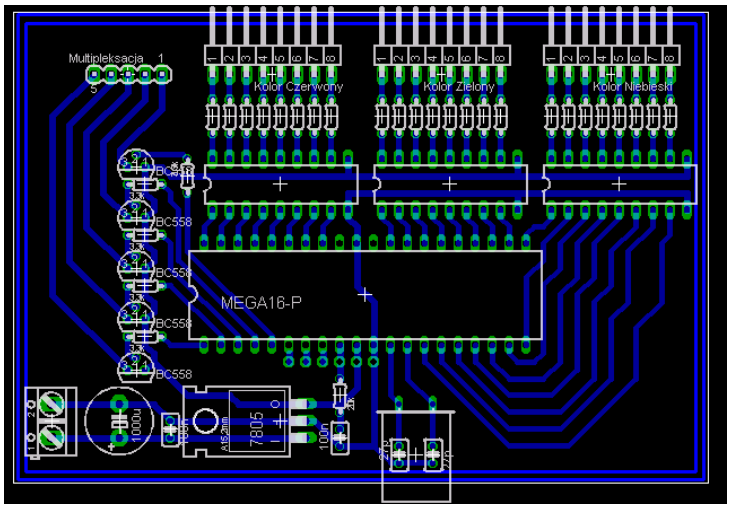
Przy sterowaniu multipleksowym, w celu zapewnienia odpowiedniej jasności diod, rezystory ograniczające prąd mogą mieć niższą wartość rezystancji niż w przypadku ciągłego zasilania pojedynczej diody. Ważne, by jeden kolor nie przeważał nad innymi, gdyż może to mieć negatywny wpływ na prawidłowe wyświetlanie barw dopełniających (żółty, różowy, biały itd.). Eksperymentalnie dobrano wartości 100Ω dla diody czerwonej i 270Ω dla diod zielonej i niebieskiej.

Mikrokontroler wyposażony jest w linię RESET. Układ zostanie zresetowany, jeżeli na tej linii pojawi się niski poziom trwający przynajmniej 1,5µs. Aby zapobiec resetowaniu się mikrokontrolera spowodowanego zakłóceniami, linię RESET podciągnięto rezystorem 10kΩ do VCC.

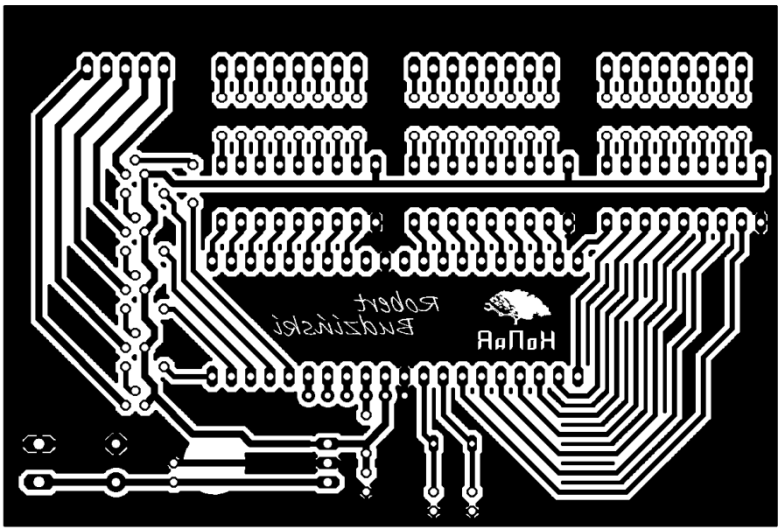
Zarówno elektronika jak i matryca diod RGB zasilane są napięciem o wartości 5V. W układzie zastosowano stabilizator 7805 z włączonym równolegle kondensatorem filtrującym 1000µF i dwoma kondensatorami odsprzęgającymi o wartości 100nF. Pobór prądu jest uzależniony od ilości świecących w danym momencie diod i wynosi maksymalnie 250mA (kolor biały – wszystkie diody zapalone). Nadmiar energii doprowadzonej do stabilizatora zostaje wypromieniowany w postaci ciepła, dlatego istotne jest zastosowanie radiatora.



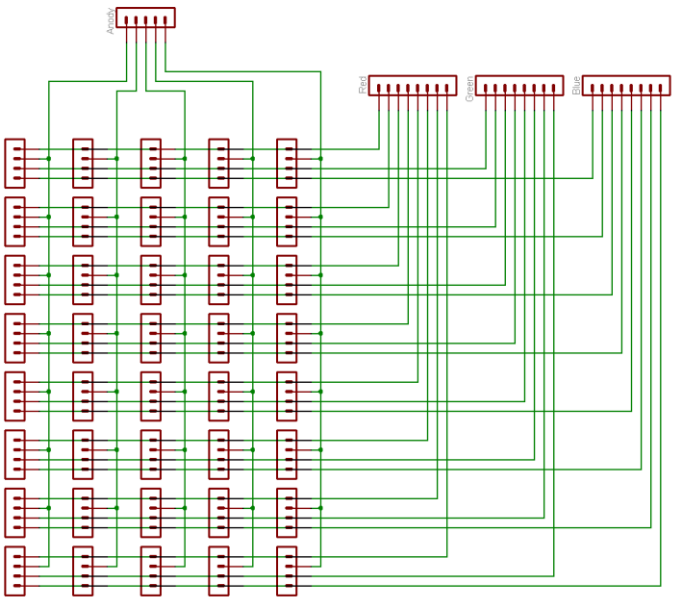
Rys.2. Schemat ideowy sterownika



Rys.3. Rozmieszczenie elementów na płytce



Rys.4. Mozaika ścieżek, wzór do wytrawienia



Rys.5. Schemat połączeń matrycy diod

4. Program sterujący, operacje bitowe i przykładowe efekty

Program sterujący zrealizowano w języku C a także w języku Bascom.

Multipleksowanie odbywa się w podprogramie obsługi przerwania generowanego po przepełnieniu Timera. Wszystkie wykorzystywane linie mikrokontrolera skonfigurowano jako wyjściowe.

Funkcja multipleksująca

w języku C

```
SIGNAL(SIG_OVERFLOW0)
{
  switch(kolumna)
  {
  case 1:
    PORTA=R1;
    PORTC=G1;
    PORTD=B1;
    PORTB=0b11111110;
    break;
  case 2:
    PORTA=R2;
    PORTC=G2;
    PORTD=B2;
    PORTB=0b11111101;
    break;
  case 3:
    PORTA=R3;
    PORTC=G3;
    PORTD=B3;
    PORTB=0b11111011;
    break;
  case 4:
    PORTA=R4;
    PORTC=G4;
    PORTD=B4;
    PORTB=0b11110111;
    break;
  case 5:
    PORTA=R5;
    PORTC=G5;
    PORTD=B5;
    PORTB=0b11101111;
    break;
  }
  kolumna=kolumna+1;
  if (kolumna==6)
    kolumna=1;
}
```

w języku Bascom

```
Wyswietl:
  Load Timer0 , 10
  Set Linia1
  Set Linia2
  Set Linia3
  Set Linia4
  Set Linia5
  Select Case kolumna
  Case 0:
    Porta = R1
    Portc = G1
    Portd = B1
    Reset Linia1
  Case 1:
    Porta = R2
    Portc = G2
    Portd = B2
    Reset Linia2
  Case 2:
    Porta = R3
    Portc = G3
    Portd = B3
    Reset Linia3
  Case 3:
    Porta = R4
    Portc = G4
    Portd = B4
    Reset Linia4
  Case 4:
    Porta = R5
    Portc = G5
    Portd = B5
    Reset Linia5
  End Select
  Incr kolumna
  If kolumna=5 Then kolumna = 0
Return
```

W programie zadeklarowano 15 zmiennych (R1...R5, G1...G5, B1...B5), których wartość na bieżąco odzwierciedlają diody RGB. Zmienna R1 określa pierwszą kolumnę diod czerwonych, R2 drugą kolumnę itd. Analogicznie zmienne G1…G5 odpowiadają za kolejne kolumny diod zielonych a B1…B5 niebieskich. Zapis binarny pozwala w prosty sposób definiować, które diody będą aktualnie zapalone. Zastosowanie operacji binarnych takich jak alternatywa (or, „|”), koniunkcja (and, „&”), alternatywa wykluczająca (xor, „^”), dopełnienie bitowe (toggle, „~”) i przesunięcie bitowe (shift, „>>”, „<<”) pozwala w prosty sposób projektować efekty świetlne. Kod efektu należy umieścić w programie głównym, najlepiej w nieskończonej pętli *while(1)*.

Przykład efektu zrealizowanego za pomocą operacji bitowych alternatywy i przesunięcia. Poniższy kod (w języku C) spowoduje kolejne zapalanie się diod pierwszej kolumny.

```
unsigned char temp = 0b00000001;
R1=0b00000001;
for (int i=0; i<7; i++)
{
  wait_ms(1000);
  temp <<= 1;
  if (i<8) R1=R1 | temp;
}
```

Zamiast załączania dwustanowego można zaimplementować płynną zmianę barw. Wykorzystano do tego programowy PWM (puls with modulation), czyli modulację szerokości impulsu. Poniższy kod spowoduje powolne rozświetlenie się czterech diod pierwszej kolumny wykorzystując 100 poziomów jasności.

```
unsigned int temp;
for (unsigned int i=0; i<100; i++)
{
    for (unsigned int j=0; j<3; j++)
    {
      R1=0b00000000;
      temp = 100-i;
      czekaj_us(temp);
      R1=0b00001111;
      czekaj_us(i);
    }
}
```

Poniższy listing przedstawia kompletny program sterownika. Zaimplementowany kod powoduje łagodną zmianę barw kolejnych poszczególnych diod. Ich kolejność została określona w programie. Kiedy już wszystkie diody będą świecić jednym kolorem, cykl rozpoczyna się od początku, lecz dla innej barwy. Zaimplementowano 6 kolorów (czerwony, żółty, zielony, turkusowy, niebieski, różowy).

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

unsigned int kolumna=1;
volatile unsigned char R1, R2, R3, R4, R5;
volatile unsigned char G1, G2, G3, G4, G5;
volatile unsigned char B1, B2, B3, B4, B5;
volatile unsigned int res_pwm=100; //rozdzielczość PWM
volatile unsigned int czas_zmiany=1;

void czekaj_us(unsigned int i)
{
   for(;i!=0;i--) _delay_us(50);
}

void czekaj_ms(unsigned int i)
{
   for(;i!=0;i--) _delay_ms(1);
}

SIGNAL(SIG_OVERFLOW0)
{
           switch(kolumna)
           {
                  case 1:
                    PORTA=R1;
                    PORTC=G1;
                    PORTD=B1;
                    PORTB=0b11111110;
                    break;
                  case 2:
                    PORTA=R2;
                    PORTC=G2;
                    PORTD=B2;
```

```
                    PORTB=0b11111101;
                    break;
                  case 3:
                    PORTA=R3;
                    PORTC=G3;
                    PORTD=B3;
                    PORTB=0b11111011;
                    break;
                  case 4:
                    PORTA=R4;
                    PORTC=G4;
                    PORTD=B4;
                    PORTB=0b11110111;
                    break;
                  case 5:
                    PORTA=R5;
                    PORTC=G5;
                    PORTD=B5;
                    PORTB=0b11101111;
                    break;
            }
            kolumna=kolumna+1;
            if (kolumna==6)
              kolumna=1;
}

int main(void)
{
          DDRA = 0b11111111;
          DDRC = 0b11111111;
          DDRD = 0b11111111;
          DDRB = 0b11111111;
          R1=0b00000000;
          R2=0b00000000;
          R3=0b00000000;
          R4=0b00000000;
          R5=0b00000000;
          G1=0b00000000;
          G2=0b00000000;
          G3=0b00000000;
          G4=0b00000000;
          G5=0b00000000;
          B1=0b00000000;
          B2=0b00000000;
          B3=0b00000000;
          B4=0b00000000;
          B5=0b00000000;

   TCCR0 = (1<<CS11)|(1<<CS01);
   TIMSK = (1<<TOIE0);
   sei();

   wszystkie_pojedynczo_rozjasnij(0); //czerwone
   czekaj_ms(1000);

   while(1)
   {
            wszystkie_pojedynczo_rozjasnij(5); //zolte
            czekaj_ms(1000);
            wszystkie_pojedynczo_zgas(0); //zielone
            czekaj_ms(1000);
            wszystkie_pojedynczo_rozjasnij(10); //turkusowy
            czekaj_ms(1000);
            wszystkie_pojedynczo_zgas(5); //niebieski
            czekaj_ms(1000);
            wszystkie_pojedynczo_rozjasnij(0); //rozowy
            czekaj_ms(1000);
            wszystkie_pojedynczo_zgas(10); //czerwony
            czekaj_ms(1000);
   }
   return 0;
}

//--------------- E F E K T Y ----------------
void wszystkie_pojedynczo_rozjasnij(unsigned int kolor)
{
          pojedynczo_rozjasnij(1+kolor,0b00000000, 0b00000001);
          pojedynczo_rozjasnij(3+kolor,0b00000000, 0b00000001);
          pojedynczo_rozjasnij(4+kolor,0b00000000, 0b00001000);
          pojedynczo_rozjasnij(2+kolor,0b00000000, 0b00010000);
          pojedynczo_rozjasnij(5+kolor,0b00000000, 0b00000010);
          pojedynczo_rozjasnij(1+kolor,0b00000001, 0b00100001);
          pojedynczo_rozjasnij(3+kolor,0b00000001, 0b00001001);
          pojedynczo_rozjasnij(4+kolor,0b00001000, 0b00001001);
          pojedynczo_rozjasnij(5+kolor,0b00000010, 0b00010010);
          pojedynczo_rozjasnij(1+kolor,0b00100001, 0b00101001);
          pojedynczo_rozjasnij(2+kolor,0b00010000, 0b00010010);
          pojedynczo_rozjasnij(4+kolor,0b00001001, 0b10001001);
          pojedynczo_rozjasnij(5+kolor,0b00010010, 0b00010110);
          pojedynczo_rozjasnij(3+kolor,0b00001001, 0b00101001);
          pojedynczo_rozjasnij(4+kolor,0b10001001, 0b11001001);
          pojedynczo_rozjasnij(1+kolor,0b00101001, 0b01101001);
          pojedynczo_rozjasnij(2+kolor,0b00010010, 0b01010010);
          pojedynczo_rozjasnij(5+kolor,0b00010110, 0b00110110);
          pojedynczo_rozjasnij(4+kolor,0b11001001, 0b11001101);
          pojedynczo_rozjasnij(3+kolor,0b00101001, 0b00101101);
          pojedynczo_rozjasnij(2+kolor,0b01010010, 0b01010011);
          pojedynczo_rozjasnij(5+kolor,0b00110110, 0b01110110);
          pojedynczo_rozjasnij(1+kolor,0b01101001, 0b01101101);
          pojedynczo_rozjasnij(2+kolor,0b01010011, 0b01010111);
          pojedynczo_rozjasnij(5+kolor,0b01110110, 0b01110111);
          pojedynczo_rozjasnij(3+kolor,0b00101101, 0b10101101);
          pojedynczo_rozjasnij(5+kolor,0b01110111, 0b11110111);
```

```
          pojedynczo_rozjasnij(1+kolor,0b01101101, 0b11101101);
          pojedynczo_rozjasnij(2+kolor,0b01010111, 0b01011111);
          pojedynczo_rozjasnij(4+kolor,0b11001101, 0b11011101);
          pojedynczo_rozjasnij(3+kolor,0b10101101, 0b10111101);
          pojedynczo_rozjasnij(1+kolor,0b11101101, 0b11111101);
          pojedynczo_rozjasnij(2+kolor,0b01011111, 0b01111111);
          pojedynczo_rozjasnij(4+kolor,0b11011101, 0b11111101);
          pojedynczo_rozjasnij(1+kolor,0b11111101, 0b11111111);
          pojedynczo_rozjasnij(3+kolor,0b10111101, 0b11111101);
          pojedynczo_rozjasnij(5+kolor,0b11110111, 0b11111111);
          pojedynczo_rozjasnij(3+kolor,0b11111101, 0b11111111);
          pojedynczo_rozjasnij(4+kolor,0b11111101, 0b11111111);
          pojedynczo_rozjasnij(2+kolor,0b01111111, 0b11111111);
}

void wszystkie_pojedynczo_zgas(unsigned int kolor)
{
          pojedynczo_rozjasnij(1+kolor,0b11111111, 0b11111110);
          pojedynczo_rozjasnij(3+kolor,0b11111111, 0b11111110);
          pojedynczo_rozjasnij(4+kolor,0b11111111, 0b11110111);
          pojedynczo_rozjasnij(2+kolor,0b11111111, 0b11101111);
          pojedynczo_rozjasnij(5+kolor,0b11111111, 0b11111101);
          pojedynczo_rozjasnij(1+kolor,0b11111110, 0b11011110);
          pojedynczo_rozjasnij(3+kolor,0b11111110, 0b11110110);
          pojedynczo_rozjasnij(4+kolor,0b11110111, 0b11110110);
          pojedynczo_rozjasnij(5+kolor,0b11111101, 0b11101101);
          pojedynczo_rozjasnij(1+kolor,0b11011110, 0b11010110);
          pojedynczo_rozjasnij(2+kolor,0b11101111, 0b11101101);
          pojedynczo_rozjasnij(4+kolor,0b11110110, 0b01110110);
          pojedynczo_rozjasnij(5+kolor,0b11101101, 0b11101001);
          pojedynczo_rozjasnij(3+kolor,0b11110110, 0b11010110);
          pojedynczo_rozjasnij(4+kolor,0b01110110, 0b00110110);
          pojedynczo_rozjasnij(1+kolor,0b11010110, 0b10010110);
          pojedynczo_rozjasnij(2+kolor,0b11101101, 0b10101101);
          pojedynczo_rozjasnij(5+kolor,0b11101001, 0b11001001);
          pojedynczo_rozjasnij(4+kolor,0b00110110, 0b00110010);
          pojedynczo_rozjasnij(3+kolor,0b11010110, 0b11010010);
          pojedynczo_rozjasnij(2+kolor,0b10101101, 0b10101100);
          pojedynczo_rozjasnij(5+kolor,0b11001001, 0b10001001);
          pojedynczo_rozjasnij(1+kolor,0b10010110, 0b10010010);
          pojedynczo_rozjasnij(2+kolor,0b10101100, 0b10101000);
          pojedynczo_rozjasnij(5+kolor,0b10001001, 0b10001000);
          pojedynczo_rozjasnij(3+kolor,0b11010010, 0b01010010);
          pojedynczo_rozjasnij(5+kolor,0b10001000, 0b00001000);
          pojedynczo_rozjasnij(1+kolor,0b10010010, 0b00010010);
          pojedynczo_rozjasnij(2+kolor,0b10101000, 0b10100000);
          pojedynczo_rozjasnij(4+kolor,0b00110010, 0b00100010);
          pojedynczo_rozjasnij(3+kolor,0b01010010, 0b01000010);
          pojedynczo_rozjasnij(1+kolor,0b00010010, 0b00000010);
          pojedynczo_rozjasnij(2+kolor,0b10100000, 0b10000000);
          pojedynczo_rozjasnij(4+kolor,0b00100010, 0b00000010);
          pojedynczo_rozjasnij(1+kolor,0b00000010, 0b00000000);
          pojedynczo_rozjasnij(3+kolor,0b01000010, 0b00000010);
          pojedynczo_rozjasnij(5+kolor,0b00001000, 0b00000000);
          pojedynczo_rozjasnij(3+kolor,0b00000010, 0b00000000);
          pojedynczo_rozjasnij(4+kolor,0b00000010, 0b00000000);
          pojedynczo_rozjasnij(2+kolor,0b10000000, 0b00000000);
}

void pojedynczo_rozjasnij(unsigned int ktory, unsigned char stan_wylaczenia, unsigned char stan_wlaczenia)
{
 unsigned int temp;
 for (unsigned int i=0; i<res_pwm; i++)
 {
          for (unsigned int j=0; j<czas_zmiany; j++)
           {
               if (ktory==1)  R1=stan_wylaczenia;
               if (ktory==2)  R2=stan_wylaczenia;
               if (ktory==3)  R3=stan_wylaczenia;
               if (ktory==4)  R4=stan_wylaczenia;
               if (ktory==5)  R5=stan_wylaczenia;
               if (ktory==6)  G1=stan_wylaczenia;
               if (ktory==7)  G2=stan_wylaczenia;
               if (ktory==8)  G3=stan_wylaczenia;
               if (ktory==9)  G4=stan_wylaczenia;
               if (ktory==10) G5=stan_wylaczenia;
               if (ktory==11) B1=stan_wylaczenia;
               if (ktory==12) B2=stan_wylaczenia;
               if (ktory==13) B3=stan_wylaczenia;
               if (ktory==14) B4=stan_wylaczenia;
               if (ktory==15) B5=stan_wylaczenia;
          temp = res_pwm-i;
          czekaj_us(temp);
          if (ktory==1)       R1=stan_wlaczenia;
                    if (ktory==2)       R2=stan_wlaczenia;
                    if (ktory==3)       R3=stan_wlaczenia;
                    if (ktory==4)       R4=stan_wlaczenia;
                    if (ktory==5)       R5=stan_wlaczenia;
          if (ktory==6)       G1=stan_wlaczenia;
                    if (ktory==7)       G2=stan_wlaczenia;
                    if (ktory==8)       G3=stan_wlaczenia;
                    if (ktory==9)       G4=stan_wlaczenia;
                    if (ktory==10)      G5=stan_wlaczenia;
          if (ktory==11)      B1=stan_wlaczenia;
                    if (ktory==12)      B2=stan_wlaczenia;
                    if (ktory==13)      B3=stan_wlaczenia;
                    if (ktory==14)      B4=stan_wlaczenia;
                    if (ktory==15)      B5=stan_wlaczenia;
          czekaj_us(i);
           }
  }
}
```

5. Bibliografia

- Marcin Wiązania, Programowanie mikrokontrolerów AVR w języku BASCOM
- Katedra Optoelektroniki i Systemów Elektronicznych, Dokumentacja mikrokontrolera Atmega16

6. Uwagi końcowe i podziękowania

Projekt przedstawiłem na łamach forum www.elektroda.pl pod adresem: http://www.elektroda.pl/rtvforum/topic1249129.html, gdzie umieściłem zdjęcia, film obrazujący przykładowy efekt zaimplementowany w sterowniku, kod źródłowy oraz schemat i projekt płytki PCB wykonany w programie Eagle.

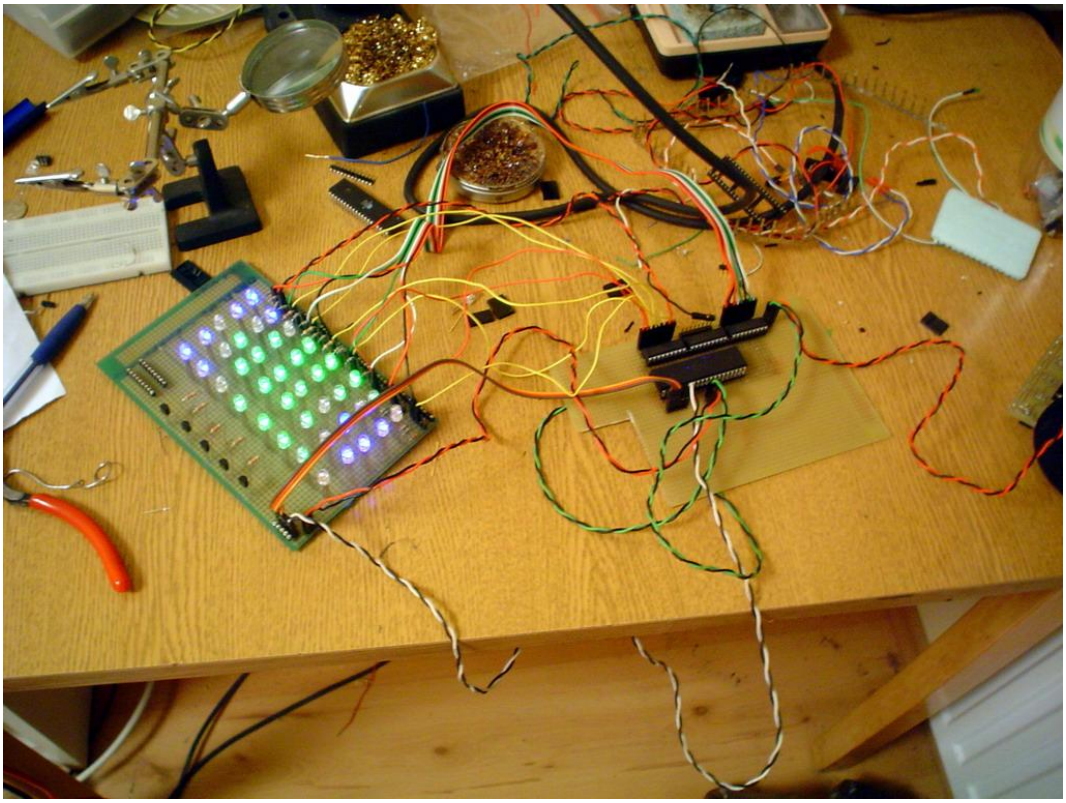
Będę niezmiernie wdzięczny za wszelkie uwagi i słowa krytyki. Mam nadzieję, że pozwolą mi one wyeliminować ewentualne błędy. Służę pomocą osobom zainteresowanym budową lub modyfikacją sterownika.

Kontakt:
Robert Budziński
gg: 4536153
e-mail: robi667[at]gmail.com

Pragnę podziękować wszystkim, którzy przyczynili się do zrealizowania tego projektu, a w szczególności:

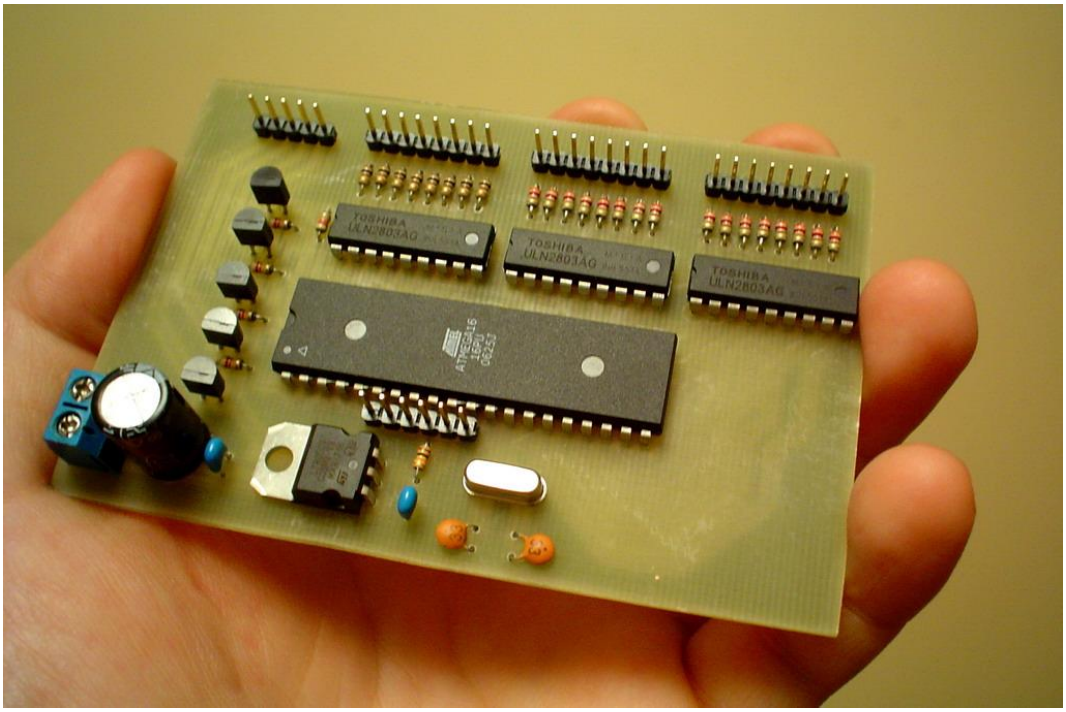
Pawłowi Bardowskiemu, Pawłowi Połeciowi:
– za cenne wskazówki natury technicznej,
Kamilowi Przychodzeniowi:
– za inspirację (www.digart.pl/praca/1813698/),
Marzenie Stachów:
– za znalezienie i podesłanie powyższego zdjęcia oraz wspieranie projektu,
Aleksandrze Harlender:
– za drukowanie wzorów płytek
Annie Kamińskiej, Julianie Pachli, Marcie Nowotnik, Monice Hyjek, Agacie Parasiewicz, Jarosławowi Likusowi:
– za szczególne zaangażowanie w zbieraniu kulek do „drzewka z fafkulców".

7. Zdjęcia z prac nad sterownikiem

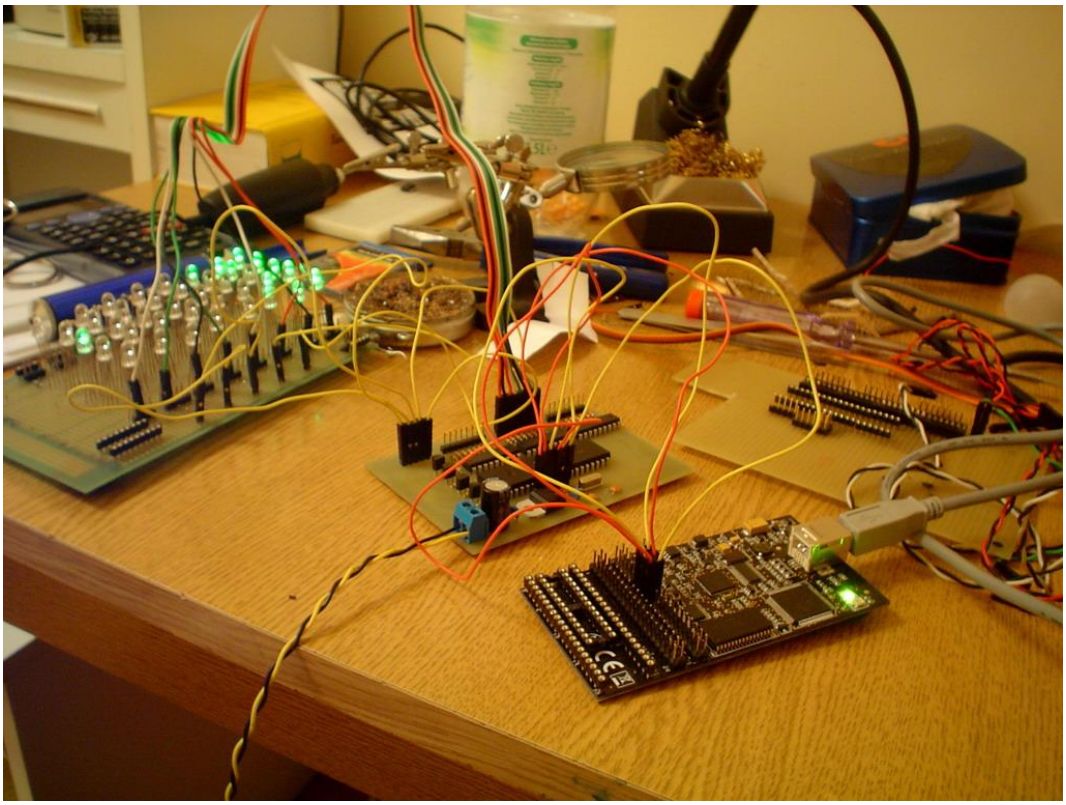


Fot.1. Wersja prototypowa sterownika

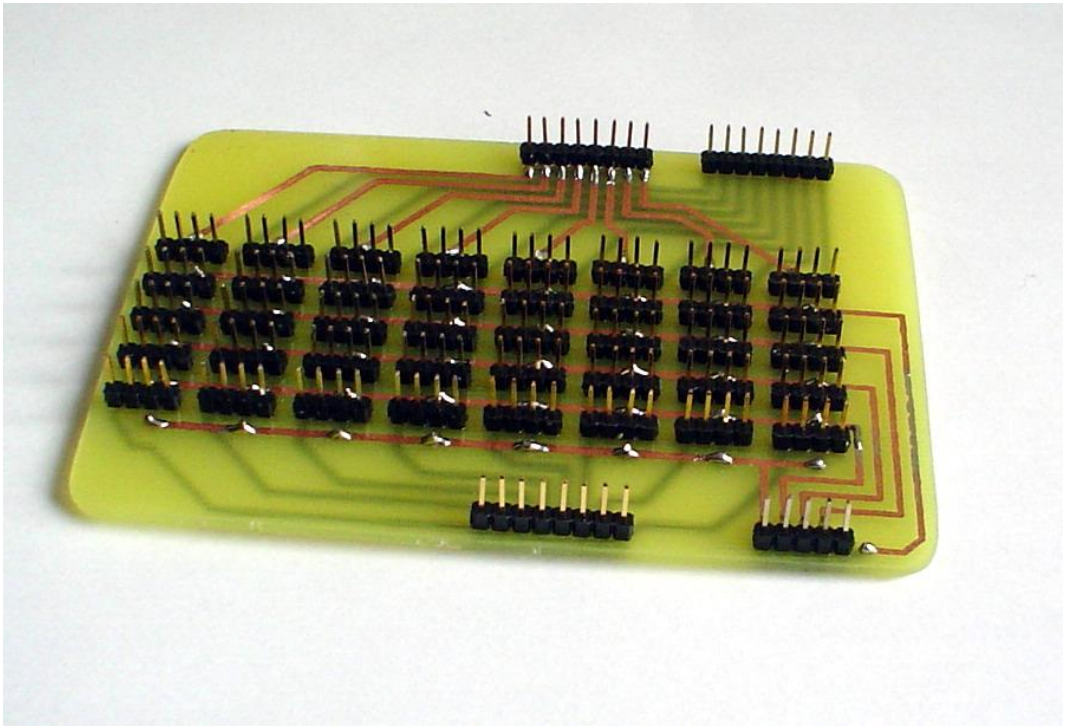
Fot.2. Wytrawiona płytka sterownika



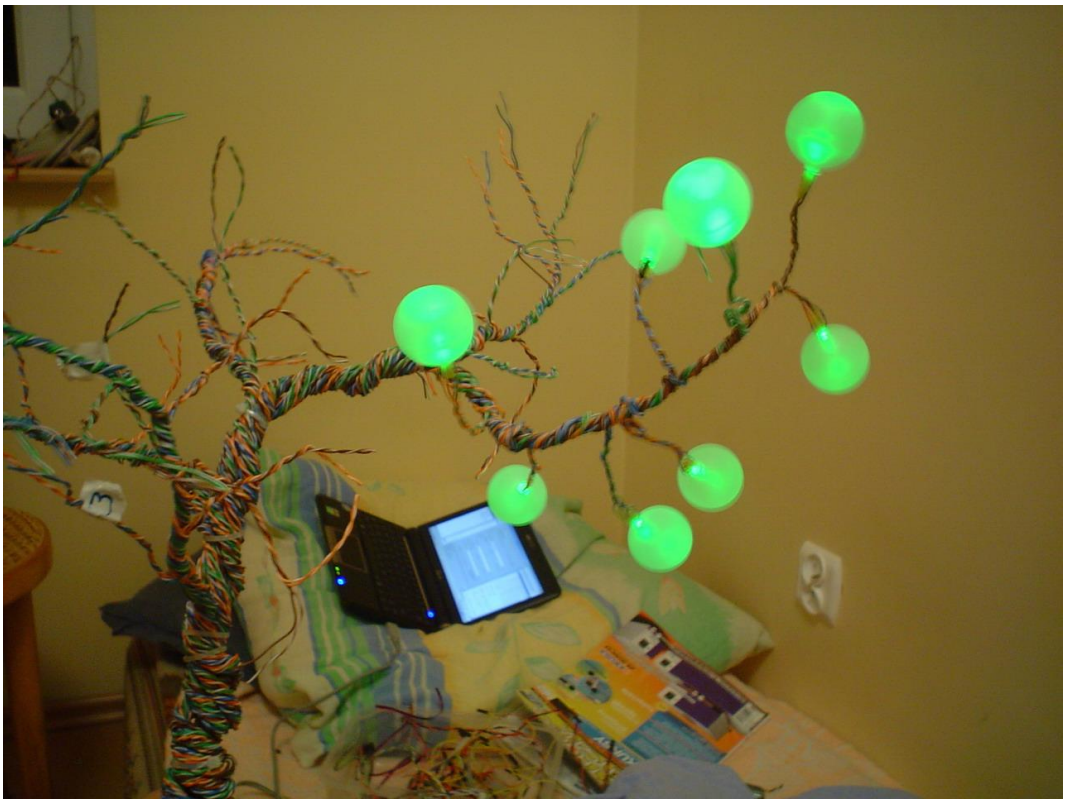
Fot.3. Polutowany sterownik



Fot.4. Pierwsze testy

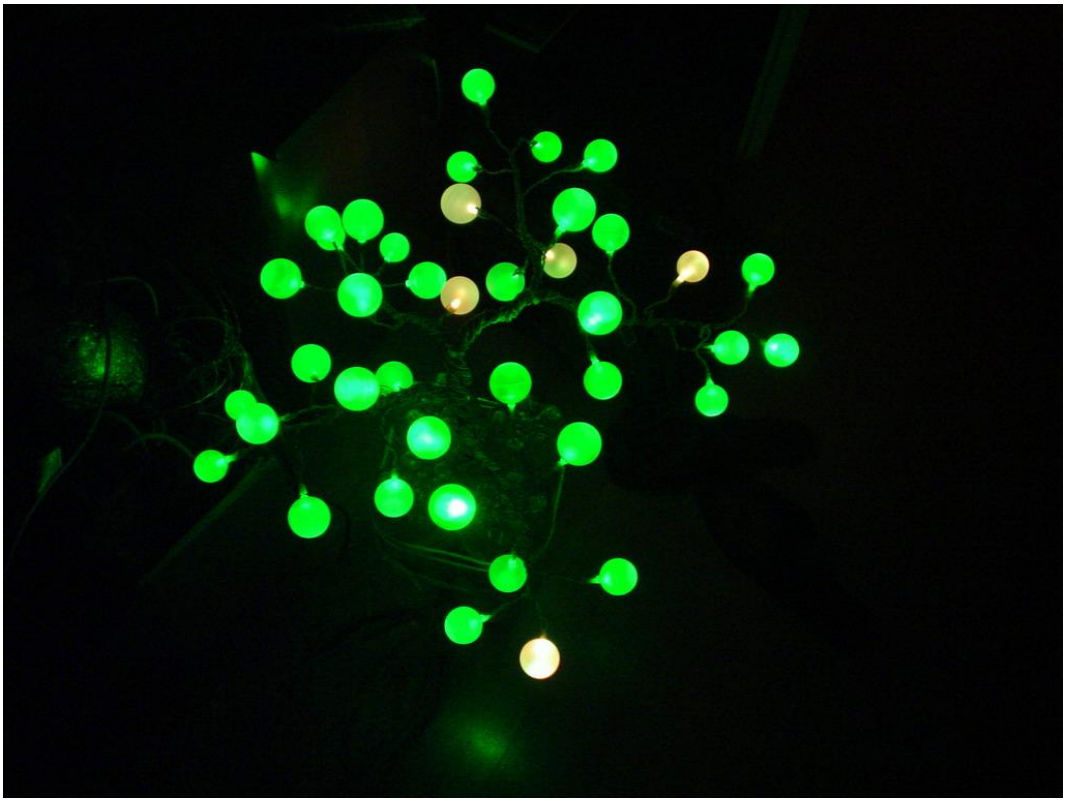


Fot.5. Płytka organizująca połączenia wierszy i kolumn



Fot.6. Konstrukcja diodowego „drzewka z fafkulców” – test jednej z pięciu gałęzi

Fot.7. Przykład zastosowania sterownika - diodowe „drzewko z fafkulców”



Fot.8. Przykład zastosowania sterownika - diodowe „drzewko z fafkulców”