

# POLITECHNIKA WROCŁAWSKA

## WYDZIAŁ ELEKTRONIKI

---

KIERUNEK: Informatyka  
SPECJALNOŚĆ: (INS)

## PROJEKT INŻYNIERSKI

Obsługa paneli RGB z wykorzystaniem  
Raspberry Pi i magistrali SPI lub I<sup>2</sup>C

RGB panels control over SPI or I<sup>2</sup>C interface and  
Raspberry Pi

AUTOR:  
Mikalai Barysau

PROWADZĄCY PROJEKT:  
dr inż. Tomasz Surmacz

OCENA PROJEKTU:



# Spis treści

Wstęp	1
<b>1 Cel pracy</b>	<b>5</b>
<b>2 Opis sprzętu</b>	<b>7</b>
2.1 Układ <i>Texas Instruments tlc5947</i>	7
2.2 Mikrokomputer <i>Raspberry Pi</i>	8
<b>3 Opis wybranych technologii</b>	<b>9</b>
3.1 Biblioteka do obsługi układu <i>Texas Instruments tlc5947</i>	9
3.2 Aplikacja graficzna <i>LedSimulator</i> do symulacji działania systemu	9
3.3 Programowa konfiguracja <i>Raspberry Pi</i>	10
3.4 Interfejs komunikacyjny <i>SPI</i>	11
<b>4 Implementacja</b>	<b>13</b>
4.1 Biblioteka do obsługi układu <i>Texas Instruments tlc5947</i>	13
4.1.1 cleanRGB	14
4.1.2 getLedIndex	14
4.1.3 compileOddLedPattern	15
4.1.4 compileEvenLedPattern	15
4.1.5 insertLedRgb	16
4.1.6 getLedRGB	17
4.1.7 setLedRGB	18
4.1.8 setLedRGB	19
4.2 Aplikacja graficzna <i>LedSimulator</i>	19
<b>Spis rysunków</b>	<b>20</b>



# Wstęp

Rosnąca liczba urządzeń elektronicznych w segmencie budżetowym, wzrastające możliwości obliczeniowe komputerów i coraz mniejsze rozmiary procesorów, płyt głównych, źródeł zasilania i nośników danych umożliwiają stworzenie produktów, które stosunkowo niedawno wymagały znaczących inwestycji finansowych i zazwyczaj były mało mobilne.

Obecny świat elektroniczny sprzyja powstaniu licznych “inteligentnych” systemów i oddzielnych urządzeń. Niektóre z tych urządzeń już teraz stały się elementami naszego codziennego otoczenia, życia bez których nie możemy sobie wyobrazić. Spad kosztów, wzrost mocy jednostek obliczeniowych i gwałtowne zwiększenie rynku urządzeń elektronicznych powoduje coraz większe zapotrzebowanie społeczności w nowych rozwiązaniach sprzętowych i programowych.

Warto również zwrócić uwagę na to, że świat systemów operacyjnych również bardzo się rozwinął w ciągu ostatnich 20 lat. Jakiś czas temu najbardziej rozpowszechnionym system operacyjnym był system *Microsoft Windows*. Jednak po upływie czasu systemy, oparte na UNIX-ie, takie jak *GNU/Linux* i *OSX*, również zdobyły dużą popularność spośród użytkowników PC, co w tej chwili wymaga od twórców oprogramowania wyboru technologii, które pozwolą na stworzenie narzędzi, które nie będą wymagały od użytkownika zainstalowania dodatkowych bibliotek, konfigurowania odpowiedniego środowiska czy jakiegokolwiek ingerencji w proces funkcjonowania dostarczonego produktu. Dla użytkownika końcowego najważniejszym jest to, żeby produkt działał w taki sposób, jak od niego się oczekuje.

Dlatego rola programisty i inżyniera, który jest twórcą tego produktu, polega nie tylko na rozwiązaniu konkretnego problemu technicznego, ale również na przemyśleniu tego, czy dany produkt jest intuicyjny w obsłudze, o ile to jest możliwe, czy on jest sprawny i przetestowany, czy użytkownik końcowy nie będzie w stanie zakłócić działania programu przez nieodpowiednie korzystanie z dostępnych funkcjonalności.

Stworzenie takiego produktu jest zadaniem nietrywialnym i czasochłonnym. Opracowanie scenariuszy działania tworzonego systemu, przeprowadzenie analizy przypadków użycia, dobór odpowiednich technologii i efektywnych metod obliczeniowych jest niezbędną częścią procesu tworzenia oprogramowania o wysokiej jakości.



# Rozdział 1

## Cel pracy

Celem danej pracy było stworzenie biblioteki do sterowania panelami RGB za pomocą *Raspberry PI* przez interfejs *SPI* oraz aplikacji graficznej, umożliwiającej kalibrowanie kolorów i wykonanie symulacji działania systemu na zwykłym komputerze bez konieczności podłączenia sprzętu.

Wykorzystany w danej pracy sprzęt wymaga posiadanie pewnej wiedzy technicznej, dotyczącej zasad funkcjonowania pamięci układu *Texas Instruments tlc5947* specyfiki oraz opanowania technik programowania niskopoziomowego.

Stworzona biblioteka ma na celu znacznie ułatwić sterowanie podłączonym panelem LED RGB. Użytkownik biblioteki operuje na obiektach struktury *RGB* i funkcjach, które pozwalają zdefiniować lub pobrać kolor diody o wybranym numerze porządkowym. Dzięki temu nie jest konieczne dokładne zapoznanie się z modelem pamięci sterownika. Dla korzystania z biblioteki potrzebna jest jedynie podstawowa wiedza z programowania w języku *C*.

Z kolei rozwinięcie biblioteki, lub dostosowanie jej do innego modelu pamięci, będzie wymagało bardziej zaawansowanej wiedzy z dziedziny programowania niskopoziomowego. Nie powinno to, jednak, być dużym problem, ponieważ jedną z najważniejszych zasad tego projektu było bardzo szczególne dokumentowanie kodu, co pozwala osobie, nie posiadającej niezbędnej wiedzy o układzie *tlc5947*, w stosunkowo krótkim czasie zapoznać się z zasadami operowania na pamięci, definiowania kolorów i wyciągania danych z rejestru przesuwanego, wykorzystanego w danym układzie.





# Rozdział 2

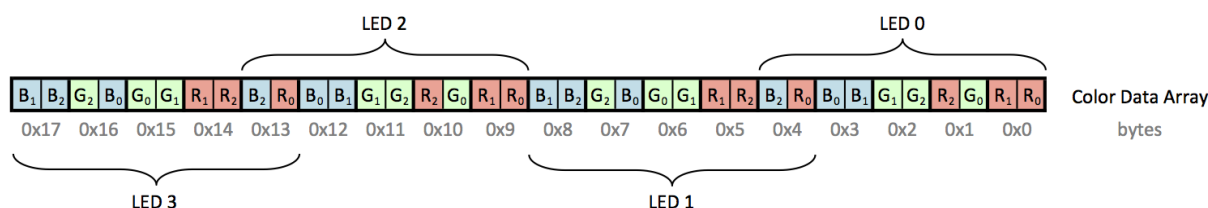
## Opis sprzętu

### 2.1 Układ *Texas Instruments tlc5947*

Przy wykonaniu danej pracy został wykorzystany sterownik *Texas Instruments tlc5947* – 24 kanałowy sterownik diod LED RGB[6]. Każdy z trzech kolorów odpowiada kolorom czerwonemu, zielonemu oraz niebieskiemu.

W środku sterownika znajduje się 36-bajtowy rejestr przesuwany. Każdej z ośmiu diod RGB przysługuje 4.5 bajta pamięci, co oznacza że wartość odpowiadająca każdemu kolorowi może mieć wartość w przedziale od 0 do 4095[6].

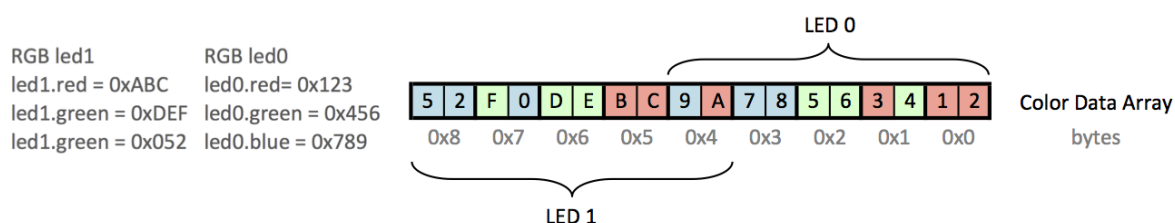
Specyfika modelu pamięci, zastosowanego w rejestrze przesuwным układu *Texas Instruments tlc5947*, polega na tym, że dane są uporządkowane w innej kolejności, niż tego może oczekiwać użytkownik. Schemat modelu pamięci jest przedstawiony na rysunku 2.1



Rysunek 2.1 Kolejność danych, odpowiadających dwom diodom, w rejestrze przesuwным sterownika *Texas Instruments tlc5947*

Litery  $R_xG_xB_x$  i kolor komurek odpowiadają kolorom oddzielnej diody. Indeksy  $x$  oznaczają kolejność liczb szesnastkowych, odpowiadających kodom poszczególnych kolorów. Adresy poszczególnych bajtów wypisane są szarą czcionką.

Następny przykład (rys. 2.2) ilustruje kolejność danych dla 2 diod w dziewięciobajtowym bloku pamięci.



Rysunek 2.2 Kolejność danych, odpowiadających dwom diodom, w rejestrze przesuwным sterownika *Texas Instruments tlc5947*

## 2.2 Mikrokomputer *Raspberry Pi*

*Raspberry Pi* – komputer jednopłytkowy, stworzony przez *Raspberry Pi Foundation*. Mikrokomputer oparty jest na układzie *Broadcom BCM2835 SoC*, który składa się z procesora *ARM1176JZF-S 700 MHz*, *VideoCore IV GPU* i 256 megabajtów (MB) pamięci RAM. Na potrzeby projektu na urządzeniu został zainstalowany system *Raspbian*.

Konfiguracja programowa mikrokomputera została omówiona w rozdziale 3.

Konfiguracja sprzętowa *Raspberry Pi model B*, który został użyty w danym projekcie, wygląda następująco:

<b>SoC</b>	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)
<b>CPU</b>	700 MHz ARM1176JZF-S core (ARM11 family)
<b>GPU</b>	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC high-profile decode
<b>Pamięć (SDRAM)</b>	256 MB (współdzielona z GPU) 256 MB (współdzielona z GPU)
<b>Porty USB 2.0</b>	2
<b>Nośnik danych</b>	złącze kart SD / MMC / SDIO MicroSD
<b>Połączenia sieciowe</b>	10/100 Ethernet (RJ45)
<b>Pozostałe złącza</b>	8 x GPIO, UART, szyna I <sup>2</sup> C , szyna SPI z dwiema liniami CS, +3,3 V, +5 V, masa
<b>Zasilanie</b>	700 mA (3,5 W)
<b>Źródło zasilania</b>	5 V przy pomocy złącza MicroUSB, ewentualnie za pomocą złącza GPIO
<b>Wymiary</b>	85,60 × 53,98 mm
<b>Waga</b>	45 g

# Rozdział 3

## Opis wybranych technologii

Wybór języka, który będzie głównym narzędziem do realizacji projektu, należy dokonywać biorąc pod uwagę specyfikę projektu, zasoby sprzętowe, przypadki użycia produktu, który powinien powstać, wymagania wydajnościowe oraz funkcjonalne. Niemniej ważny jest czas, którym dysponuje programista.

Dany projekt jest rozwiązaniem zarówno sprzętowym, jak i programowym. Dlatego wybrane technologie muszą spełniać zestaw określonych wymagań, oraz gwarantować możliwość dalszego utrzymania i rozwoju produktu.

Po przeanalizowaniu najbardziej popularnych obecnie technologii, zostały wybrane języki *C* i *C++* z wykorzystaniem zestawu bibliotek *Qt framework*.

### 3.1 Biblioteka do obsługi układu *Texas Instruments tlc5947*

Język *C* jest sprawdzonym narzędziem, które pozwala na bardzo szczegółowe manipulacje na pamięci oraz bezpośrednią kontrolę nad złożonością i wydajnością implementowanych funkcji i algorytmów. Język *C++*, z kolei, oprócz wsparcia niskopoziomowych operacji również pozwala na implementację interfejsów obiektowych i graficznej powłoki do sterowania programem oraz reprezentacji wyników działania i komunikacji z użytkownikiem.

### 3.2 Aplikacja graficzna *LedSimulator* do symulacji działania systemu

Zaletą języków *C/C++* jest duża wydajność i szybkość wykonywania skompilowanego kodu (pod warunkiem poprawnego posługiwania się możliwościami tych języków, systemu operacyjnego, rejestrami procesora, pamięcią operacyjną i innymi zasobami sprzętowymi). Od momentu stworzenia języków *C* i *C++* została opracowana liczna grupa bibliotek, dających prawie nieograniczone możliwości do tworzenia oprogramowania. Swobodny dostęp do dokumentacji, tutoriali, projektów z otwartym kodem źródłowym i ogromna społeczność programistów *C/C++* jest gwarancją tego, że napotkane problemy techniczne nie zablokują rozwoju produktu i nie pozostawią programistę sam na sam z ich rozwiązaniem.

Jako narzędzie do tworzenia interfejsu graficznego został wybrany *Qt framework*, gdyż pozwala on w bardzo wygodny sposób tworzyć interfejsy graficzne użytkownika. Oprócz tego

*Qt framework* posiada wyjątkowo dobrą dokumentację, która jest wbudowana w IDE *Qt Creator* i również jest dostępna na stronie internetowej projektu *Qt*:

`http://qt-project.org/doc/`

W dokumentacji do framework'u można znaleźć szczegółowe opisy funkcji bibliotecznych i także wiele przykładowych fragmentów kodu, które znacznie ułatwiają rozumienie mechanizmu działania sygnałów i slotów, zasad działania elementów interfejsu, kontenerów i innych narzędzi bibliotecznych. Kolejną zaletą użycia framework'u *Qt* jest możliwość stworzenia wieloplatformowej aplikacji graficznej bez konieczności utrzymania kilku wersji kodu dla różnych systemów operacyjnych.

### 3.3 Programowa konfiguracja *Raspberry Pi*

Jako środowisko systemowe na *Raspberry Pi* został zainstalowany *Raspbian* – system operacyjny *GNU/Linux* dla mikrokomputerów *Raspberry Pi*, oparty na dystrybucji *Debian*. System jest dostępny do ściągnięcia z oficjalnej strony projektu *Raspbian*:

`http://www.raspbian.org/`

lub z oficjalnej strony projektu *Raspberry Pi*:

`http://www.raspberrypi.org/downloads/`

Komunikacja z urządzeniami peryferyjnymi przez interfejs *SPI* odbywa się za pomocą sterownika *SPI*, wbudowanego w jądro systemu operacyjnego. Przed rozpoczęciem komunikacji między *Raspberry Pi* i urządzeniem peryferyjnym należy odpowiednio skonfigurować system operacyjny. Domyślnie sterownik *spi* znajduje się na “czarnej liście” modułów jądra systemu i dlatego nie jest ładowany podczas startu systemu. Aby dodać go do autostartu systemu należy otworzyć plik `/etc/modprobe.d/raspi-blacklist.conf` w edytorze tekstowym i zakomentować linię (umieścić znak '#' na początku linii), w której znajduje się nazwa sterownika *spi-bcm2708*

```
1 # blacklist spi and i2c by default (many users don't need them)
2 # blacklist spi-bcm2708
3 blacklist i2c-bcm2708
```

Listing 3.1 Zmodyfikowany plik `/etc/modprobe.d/raspi-blacklist.conf`

Również należy zmodyfikować plik `/etc/modules`, usuwając znak znać '#' na początku linii, w której znajduje się nazwa modułu *spi-dev*:

```

1  # /etc/modules: kernel modules to load at boot time.
2  #
3  # This file contains the names of kernel modules that should be loaded
4  # at boot time, one per line. Lines beginning with "#" are ignored.
5  # Parameters can be specified after the module name.
6  #
7  # sound devices
8  snd-bcm2835
9  # SPI devices
10 spi-dev
11 # I2C devices
12 # i2c-dev
13 # 1-Wire devices
14 # wl-gpio
15 # 1-Wire thermometer devices
16 # wl-therm

```

Listing 3.2 Zmodyfikowany plik */etc/modules*

Po wprowadzeniu zmian należy zrestartować *Raspberry Pi* (np. za pomocą wykonania polecenia *reboot* w linii komend systemu operacyjnego *Raspbian*).

Po ponownym załadowaniu można sprawdzić, czy moduł *SPI* został załadowany. W tym celu można skorzystać z polecenia *lsmod*.

Module	Size	Used by
snd_bcm2835	12808	0
snd_pcm	74834	1 snd_bcm2835
snd_seq	52536	0
snd_timer	19698	2 snd_seq, snd_pcm
snd_seq_device	6300	1 snd_seq
snd	52489	5 snd_seq_device, snd_timer, snd_seq, snd_pcm,
snd_bcm2835		
snd_page_alloc	4951	1 snd_pcm
spidev	5136	0
spi_bcm2708	4401	0

Listing 3.3 Wynik wykonania polecenia *lsmod*

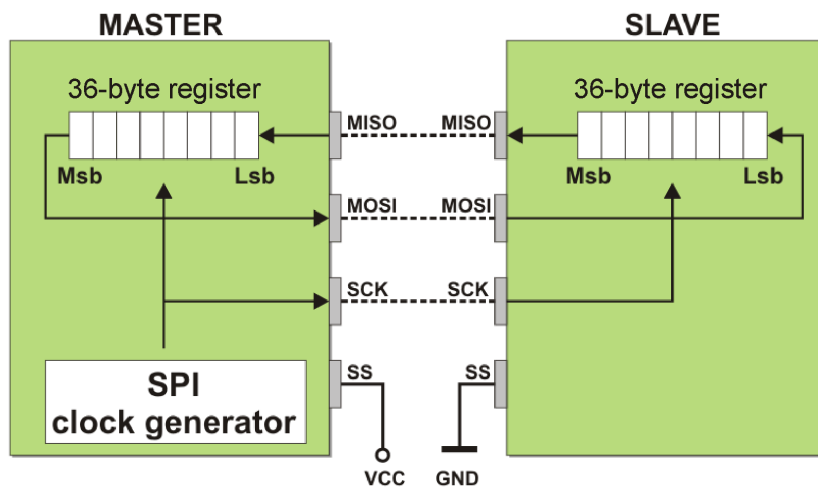
## 3.4 Interfejs komunikacyjny *SPI*

Interfejs *SPI* (*Serial Peripheral Interface*) umożliwia komunikację pomiędzy mikrokomputerem i urządzeniem peryferyjnym. Szeregowa transmisja danych odbywa się za pomocą trzech kanałów[5]:

- **MOSI** (*Master Out / Slave In*) - dane od jednostki nadrzędnej do podporządkowanej
- **MISO** (*Master In / Slave Out*) - dane od jednostki podporządkowanej do nadrzędnej
- **SCK** (*Serial Clock*) - zegar synchronizujący transmisję

Aktywacją wybranego urządzenia peryferyjnego odbywa się za pomocą dodatkowej linii **SS** (*Slave Select*).

Na rysunku 3.1 jest umieszczony schemat komunikacji między urządzeniem nadrzędnym i pojedynczym urządzeniem podporządkowanym.



Rysunek 3.1 Schemat komunikacji przez interfejs *SPI*

Po załadowaniu modułów jądra jest możliwa komunikacja z urządzeniem peryferyjnym przez interfejs *SPI*.

# Rozdział 4

## Implementacja

### 4.1 Biblioteka do obsługi układu *Texas Instruments tlc5947*

Biblioteka do sterowania układem składa się z trzech plików:

- *tlc5947\_controller.h*
- *tlc5947\_controller.c*
- *RGB.h*

Plik *RGB.h* zawiera strukturę *RGB*, która składa się z trzech pól:

- **uint64\_t** red
- **uint64\_t** green
- **uint64\_t** blue

Każde pole służy do przechowywania wartości odpowiedniego koloru. Struktura została przeniesiona do oddzielnego pliku w celu dalszej możliwości korzystania z niej bez konieczności dołączania całej biblioteki.

Plik nagłówkowy *tlc5947\_controller.h* zawiera makro definicje i nagłówki trzech funkcji, które stanowią interfejs zewnętrzny (API) biblioteki:

- **int** setLedRGB(**uint32\_t** ledNumber, **RGB** rgbSet, **uint8\_t\*** tab)
- **RGB** getLedRGB(**uint8\_t** ledNumber, **uint8\_t\*** tab)
- **void** printLedDataArray(**uint8\_t\*** tab)

Również plik *tlc5947\_controller.h* zawiera krótki komentarz, który wyjaśnia zasadę indeksowania diod w rejestrze przesuwным układu *Texas Instruments tlc5947*.

Plik *tlc5947\_controller.c* oprócz implementacji interfejsu zawiera również implementacje funkcji statycznych, które nie są widoczne dla użytkownika biblioteki, natomiast są wykorzystywane w mechanizmach adresacji i komunikacji z układem *tlc5947*. Lista zaimplementowanych funkcji wygląda następująco:

- **static void** cleanRGB(**RGB\*** rgb)
- **static uint64\_t** getLedIndex(**uint8\_t** ledNumber)
- **static uint64\_t** compileEvenLedPattern(**RGB** rgbSet)
- **static uint64\_t** compileOddLedPattern(**RGB** rgbSet)
- **static int** insertLedRgb(**uint32\_t** ledNumber, **uint64\_t** RGBpattern, **uint8\_t\*** tab)
- **RGB** getLedRGB(**uint8\_t** ledNumber, **uint8\_t\*** tab)
- **int** setLedRGB(**uint32\_t** ledNumber, **RGB** rgbSet, **uint8\_t\*** tab)
- **void** printLedDataArray(**uint8\_t\*** tab)

W dalszej części tego rozdziału przedstawiony jest szczegółowy opis funkcji z przykładami kodu.

#### 4.1.1 cleanRGB

Funkcja służy do czyszczenia obiektu struktury *RGB*.

Nazwa funkcji	<i>cleanRGB</i>
Funkcja statyczna	tak
Parametry wejściowe	<i>RGB*</i> rgb – wskaźnik na obiekt struktury <i>RGB</i>
Parametry wyjściowe	–

#### 4.1.2 getLedIndex

Funkcja służy do obliczania indeksu bloku pamięci diody w rejestrze przesuwным. W związku z tym. Dzięki tej funkcji użytkownik biblioteki nie musi uwzględniać modelu pamięci rejestru przesuwного układu *tlc5947*, korzystając z niego tak samo, jak ze zwykłej tablicy bajtowej.

Nazwa funkcji	<i>getLedIndex</i>
Funkcja statyczna	tak
Parametry wejściowe	<i>uint8_t</i> ledNumber – numer diody
Parametry wyjściowe	<i>uint64_t</i> ledAddress – indeks początku bloku pamięci diody w rejestrze przesuwным



### 4.1.3 compileOddLedPattern

Funkcja służy do kompilacji fragmentu pamięci dla diody o numerze nieparzystym, który będzie zawierał uporządkowane w odpowiedniej kolejności wartości dla kolorów czerwonego, zielonego i niebieskiego. Przykład działania funkcji: W przypadku, gdy Parametr wejściowy **RGB** *rgbSet* ma następujące wartości:

- *rgbSet.red* = 0x123
- *rgbSet.green* = 0x456
- *rgbSet.blue* = 0x789

Funkcja zwróci wartość 64-bitową, która stanowi następujący fragment pamięci:

**[00 00 00 89 67 45 23 01]**

Otrzymany w wyniku fragment jest później wkładany do tablicy *txdata*, która jest przesyłana do rejestru przesuwne. Z tego powodu, że otrzymamy fragment pamięci jest wkładany do tablicy *txdata* za pomocą operacji logicznej *OR*, wartości dla odpowiednich kolorów znajdują się na właściwych pozycjach, reszta fragmentu pamięci jest wypełniona zerami, żeby nie zmodyfikować wartości kolorów dla innych diod, które znajdują się w tablicy.

Operację na pamięci w większości wykonywane są za pomocą operatorów logicznych *AND*, *OR* oraz przesunięć bitowych. W kodzie biblioteki znajduje się szczegółowy opis zmian, które odbywają się w pamięci.

```

1 //compiling green color
2 hex = 0; // hex = [00 00 00 00 00 00 00 00]
3 hex = green & hex1Mask; // hex = [00 00 00 00 00 00 04 00]
4 hex = hex << 3 * POSITION; // hex = [00 00 00 00 00 40 00 00]
5 pattern = pattern | hex; // pattern = [00 00 00 00 00 40 23 01]
```

Listing 4.1 Fragment kodu, w którym przedstawiony jest komentarz ilustrujący efekt wykonywania operacji na pamięci

Nazwa funkcji	<i>compileEvenLedPattern</i>
Funkcja statyczna	tak
Parametry wejściowe	<b>RGB</b> <i>rgbSet</i> – obiekt struktury <i>RGB</i> , który zawiera wartości dla odpowiednich kolorów
Parametry wyjściowe	<b>uint64_t</b> <i>pattern</i> – spreparowany fragment pamięci, który następnie zostanie włożony do rejestru przesuwne

### 4.1.4 compileEvenLedPattern

Funkcja służy do kompilacji fragmentu pamięci dla diody o numerze parzystym, który będzie zawierał uporządkowane w odpowiedniej kolejności wartości dla kolorów czerwonego, zielonego i niebieskiego. Przykład działania funkcji: W przypadku, gdy Parametr wejściowy **RGB** *rgbSet* ma następujące wartości:

- `rgbSet.red = 0x123`
- `rgbSet.green = 0x456`
- `rgbSet.blue = 0x789`

Funkcja zwróci wartość 64-bitową, która stanowi następujący fragment pamięci:

**[00 00 00 90 78 56 34 21]**

Otrzymany w wyniku fragment jest później wkładany do tablicy *txdata*, która jest przesyłana do rejestru przesuwne. Z tego powodu, że otrzymany fragment pamięci jest wkładany do tablicy *txdata* za pomocą operacji logicznej *OR*, wartości dla odpowiednich kolorów znajdują się na właściwych pozycjach, reszta fragmentu pamięci jest wypełniona zerami, żeby nie zmodyfikować wartości kolorów dla innych diod, które znajdują się w tablicy.

Operację na pamięci w większości wykonywane są za pomocą operatorów logicznych *AND*, *OR* oraz przesunięć bitowych. W kodzie biblioteki znajduje się szczegółowy opis zmian, które odbywają się w pamięci.

```

1 //compiling blue color
2 hex = 0; // hex = [00 00 00 00 00 00 00 00]
3 hex = blue & hex1Mask; // hex = [00 00 00 00 00 00 07 00]
4 hex = hex << 5 * POSITION; // hex = [00 00 00 00 70 00 00 00]
5 pattern = pattern | hex; // pattern = [00 00 00 00 70 56 34 12]
```

Listing 4.2 Fragment kodu, w którym przedstawiony jest komentarz ilustrujący efekt wykonywania operacji na pamięci

Nazwa funkcji	<i>compileEvenLedPattern</i>
Funkcja statyczna	tak
Parametry wejściowe	<b>RGB</b> <i>rgbSet</i> – obiekt struktury <i>RGB</i> , który zawiera wartości dla odpowiednich kolorów
Parametry wyjściowe	<b>uint64_t</b> <i>pattern</i> – spreparowany fragment pamięci, który następnie zostanie włożony do rejestru przesuwne

#### 4.1.5 insertLedRgb

Funkcja służy do obliczania indeksu bloku pamięci diody w rejestrze przesuwne. W związku z tym. Dzięki tej funkcji użytkownik biblioteki nie musi uwzględniać modelu pamięci rejestru przesuwne układu *tlc5947*, korzystając z niego tak samo, jak ze zwykłej tablicy bajtowej.

Nazwa funkcji	<i>insertLedRgb</i>
Funkcja statyczna	tak
Parametry wejściowe	<ul style="list-style-type: none"> <li>• <i>uint32_t ledNumber</i> – number diody, której dotyczy działanie</li> <li>• <i>uint64_t RGBpattern</i> – spreparowany 64-bitowy fragment pamięci, zawierający odpowiednie wartości kolorów</li> <li>• <i>uint8_t* tab</i> – wskaźnik na tablicę <i>txdata</i>, która zostanie później przesłana do rejestru przesuwne</li> </ul>
Parametry wyjściowe	<i>int</i> – wartość 0 mówi o skutecznym wykonaniu operacji; w przypadku błędu zwracana jest wartość -1

#### 4.1.6 getLedRGB

Funkcja zwraca obiekt struktury **RGB**, który zawiera wartości kolorów wybranej diody. Wartości są pobierane z tablicy *txdata*, i wyciągane za pomocą operatorów logicznych *AND*, *OR* oraz przesunięć bitowych. W kodzie biblioteki znajduje się szczegółowy opis zmian, które odbywają się w pamięci.

```

1  //getting red color
2  hex = pattern & (0x0F);           //      hex = [00 00 00 00 00 00 00 01]
3  hex = hex << 2 * POSITION;         //      hex = [00 00 00 00 00 00 01 00]
4  ledRGB.red = ledRGB.red | hex;    //  ledRgb.red = [00 00 00 00 00 00 01 00]
5  hex = 0;                          //      hex = [00 00 00 00 00 00 00 00]
```

Listing 4.3 Fragment kodu, w którym przedstawiony jest komentarz ilustrujący efekt wykonywania operacji na pamięci

Nazwa funkcji	<i>getLedRGB</i>
Funkcja statyczna	nie
Parametry wejściowe	<ul style="list-style-type: none"> <li>• <i>uint32_t ledNumber</i> – number diody, której dotyczy działanie</li> <li>• <i>uint64_t RGBpattern</i> – spreparowany 64-bitowy fragment pamięci, zawierający odpowiednie wartości kolorów</li> <li>• <i>uint8_t* tab</i> – wskaźnik na tablicę <i>txdata</i>, która zostanie później przesłana do rejestru przesuwne</li> </ul>
Parametry wyjściowe	<i>RGB ledRGB</i> – obiekt struktury <i>RGB</i> , który zawiera odpowiednie wartości kolorów czerwonego, zielonego i niebieskiego wybranej diody

#### 4.1.7 setLedRGB

Funkcja służy do ustawiania koloru wybranej diody. Funkcja korzysta z funkcji statycznych biblioteki: *compileEvenLedPattern*, *compileOddLedPattern*, *insertLedRgb*

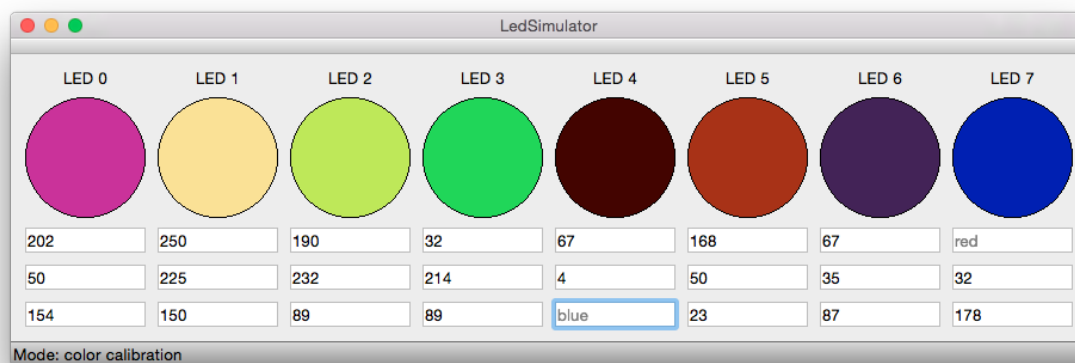
Nazwa funkcji	<i>setLedRGB</i>
Funkcja statyczna	nie
Parametry wejściowe	<ul style="list-style-type: none"> <li>• <i>uint32_t ledNumber</i> – number diody, której dotyczy działanie</li> <li>• <i>RGB rgbSet</i> – obiekt struktury <i>RGB</i>, który zawiera odpowiednie wartości kolorów czerwonego, zielonego i niebieskiego</li> <li>• <i>uint8_t* tab</i> – wskaźnik na tablicę <i>txdata</i>, która zostanie później przesłana do rejestru przesuwne</li> </ul>
Parametry wyjściowe	<i>int result</i> – wartość 0 mówi o skutecznym wykonaniu operacji; w przypadku błędu zwracana jest wartość -1

### 4.1.8 setLedRGB

Funkcja służy do wyświetlania zawartości tablicy *txdata* na standardowym wyjściu.

Nazwa funkcji	<i>printLedDataArray</i>
Funkcja statyczna	nie
Parametry wejściowe	<i>uint8_t* tab</i> – wskaźnik na tablicę <i>txdata</i> , która zawiera wartości kolorów wszystkich diod
Parametry wyjściowe	–

## 4.2 Aplikacja graficzna *LedSimulator*



Rysunek 4.1 Aplikacja graficzna *LedSimulator*: przykładowe kolory

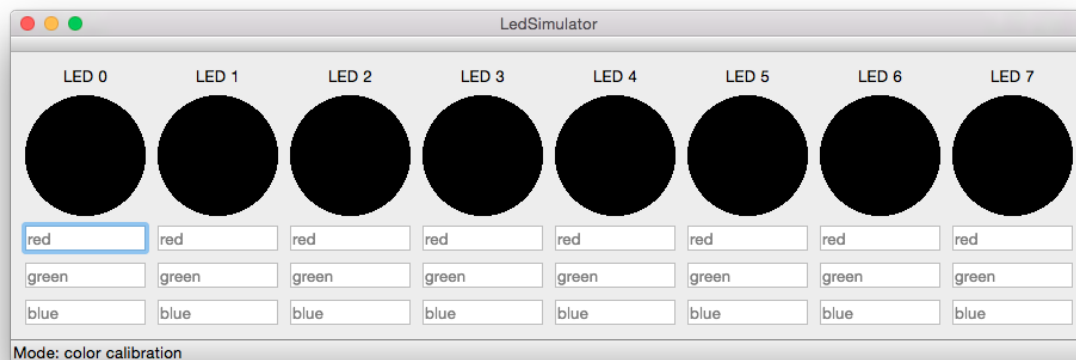
Na podanym zrzucie ekranu (rys. 4.1) jest przedstawione okno główne aplikacji graficznej *LedSimulator*. Aplikacja znajduje się w trybie *Color calibration*. Pokazany jest układ zawierający 8 diod RGB. Diody są ponumerowane od 0. Każdej diodzie przysługują 3 pola tekstowe:

- *red*
- *green*
- *blue*

W każde pole użytkownik może wprowadzić wartość od 0 do 255. Wprowadzane wartości są sprawdzane za pomocą wyrażeń regularnych, dzięki czemu nie jest możliwe wprowadzenie innych znaków, niż cyfry, oraz liczb, większych niż 255.

Kolory diod są odświeżane w momencie zmiany wartości w odpowiednim polu. Domyślną wartością, przechowywaną w każdym polu, jest 0, co oznacza brak koloru (rys. 4.2).

Aplikacja może pracować w dwóch trybach (rys. 4.3):

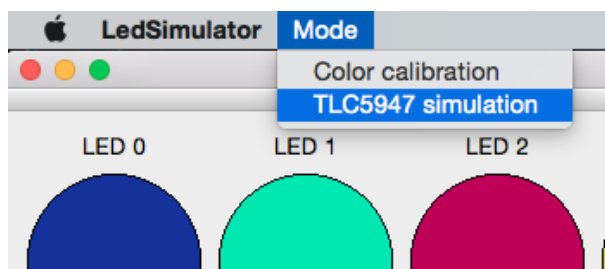


Rysunek 4.2 Aplikacja graficzna *LedSimulator*: domyślny stan diod

- *Color calibration*
- *TLC5947 simulation*

Tryb *Color calibration* został opisany wyżej.

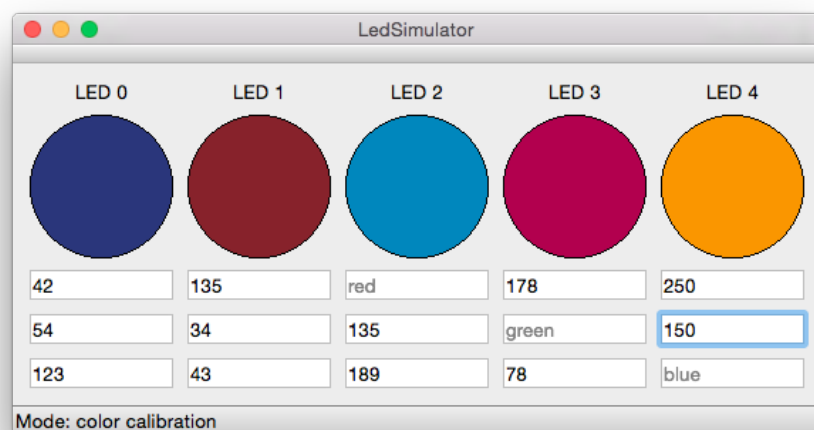
Tryb *TLC5947 simulation* służy do testowania działania biblioteki w przypadku braku dostępu do układu *Texas Instruments TLC5947*. Przy wyborze tego trybu uruchomiony zostanie scenariusz, umieszczony w funkcji *runLedScenario*.



Rysunek 4.3 Wybór trybu działania aplikacji *LedSimulator*

Aplikacja *LedSimulator* może być skonfigurowana odpowiednio dla różnych układów diod. Jak widać na rysunku 4.4, aplikacja może symulować działanie układu o dowolnej liczbie diod.

Również jest możliwe ułożenie diod w kilka rzędów. Na rysunku 4.5 jest przedstawiony widok aplikacji, skonfigurowanej dla symulowania działania układu diod 2x4



Rysunek 4.4 Aplikacja *LedSimulator*, skonfigurowana dla symulowania działania układu 5 diod



Rysunek 4.5 Aplikacja *LedSimulator*, skonfigurowana dla symulowania działania układu 2x4 diod





# Spis rysunków

2.1	Kolejność danych, odpowiadających dwom diodom, w rejestrze przesuw- nym sterownika <i>Texas Instruments tlc5947</i> . . . . .	7
2.2	Kolejność danych, odpowiadających dwom diodom, w rejestrze przesuw- nym sterownika <i>Texas Instruments tlc5947</i> . . . . .	7
3.1	Schemat komunikacji przez interfejs <i>SPI</i> . . . . .	12
4.1	Aplikacja graficzna <i>LedSimulator</i> : przykładowe kolory . . . . .	19
4.2	Aplikacja graficzna <i>LedSimulator</i> : domyślny stan diod . . . . .	20
4.3	Wybór trybu działania aplikacji <i>LedSimulator</i> . . . . .	20
4.4	Aplikacja <i>LedSimulator</i> , skonfigurowana dla symulowania działania układu 5 diod . . . . .	21
4.5	Aplikacja <i>LedSimulator</i> , skonfigurowana dla symulowania działania układu 2x4 diod . . . . .	21



# Literatura

- [1] *Loading I2C, SPI and 1-Wire drivers on the Raspberry Pi under Raspbian wheezy*. <https://www.modmypi.com/blog/loading-i2c-spi-and-1-wire-drivers-on-the-raspberry-pi-under-raspbian-wheezy> (data dostępu: 09.12.2014).
- [2] *Qt Documentation*. <http://doc.qt.io/> (data dostępu: 09.12.2014).
- [3] *Raspberry Pi Hardware Documentation. SPI*. <http://www.raspberrypi.org/documentation/hardware/README.md> (data dostępu: 09.12.2014).
- [4] *Using The SPI Interface*. <http://www.raspberry-projects.com/pi/programming-in-c/spi/using-the-spi-interface> (data dostępu: 09.12.2014).
- [5] *Broadcom BCM2835 ARM Peripherals*. 2012. PDF.
- [6] *Texas Instruments TLC5947: Datasheet*. 2014. 24-Channel, 12-Bit PWM LED Driver with Internal Oscillator (Rev. A), PDF.
- [7] G. Henderson. *Understanding SPI on the Raspberry Pi*. <https://projects.drogon.net/understanding-spi-on-the-raspberry-pi> (data dostępu: 09.12.2014).
- [8] B. W. Kernighan, D. M. Ritchie. *The C Programming Language*. wydanie 2, 1995.
- [9] B. Stroustrup. *The C++ Programming Language*. wydanie 4, 2013.