POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka
SPECJALNOŚĆ: (INS)

PROJEKT INŻYNIERSKI

Obsługa RGB

RGB panel

AUTOR:
Mikalai Barysau

PROWADZĄCY PROJEKT:
dr inż. Tomasz Surmacz

OCENA PROJEKTU:

WROCŁAW 2014

Spis treści

Wstęp

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. Analiza rynku i istniejących rozwiązań świadczy o tym, że wybrany temat jest w tej chwili bardzo aktualny.

Rosnąca liczba urządzeń elektonicznych w segmencie budżetowym, wzrastające możliwości obliczeniowe komputerów i coraz mniejsze rozmiary processorów, 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 swiąt 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 biliotek, konifgurowania odpowiedniego środowiska czy jakiejkolwiek ingerecji w process funkcjonowania dostarczonego produktu. Dla użytkownika końcowego najważnieszym 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 działanie 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 cześcią procesu tworzenia oprogramowania o wysokiej jakości.

Rozdział 1

Opis sprzętu

1.1 Układ *Texas Intruments tlc5947*

1.2 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*.

Konfiguacja programowa mikrokomputera została omówiona w rozdziale 2.

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

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

Rozdział 2

Opis wybranych technologii

Wybór języka, który będzie głównym narzedziem 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*.

2.1 Biblioteka do obsługi układu *Texas Intruments tlc5947*

Jężyk *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 nizkopoziomowych operacji również pozwala na implementację interfejsów obiektówych i graficznej powłoki do sterowania programem oraz reprezentacji wyników działania i komunikacji z użytkownikiem.

2.2 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 ne zablokują rozwoju produktu i nie pozostawią programistę sam na sam z ich rozwiązaniem.

Jako narzędzie do tworzenia interfejsu graficznego został wybrany *Qt framework*, gdzyż pozwala on w bardzo wygodny sposób tworzyć interfejsy graficzne użytkonika. 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 działanie elementów interfejsu, kontenerów i innych narzędzi bibliotecznych. Kolejną zaletą użycia framework'u *Qt* jest możliwość stworzenia wwieloplatformowej aplikacji graficznej bez konieczności utrzymania kilku wersji kodu dla różnych systemów operacyjnych.

2.3 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*. Syste, jest dostepny 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 pereferyjnymi przez interfejs *SPI* odbywa się za pomocą sterownika *SPI*, wbudowanego w jądro systemu operacyjnego. Przed rozpoczęciem komunikacji między *Raspberry Pi* i urządzeniem pereferyjnym 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 2.1 Zmodyfikowany plik */etc/modprobe.d/raspi-blacklist.conf*

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

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.
#
# sound devices
snd-bcm2835
# SPI devices
spi-dev
# I2C devices
# i2c-dev
# 1-Wire devices
# w1-gpio
# 1-Wire thermometer devices
# w1-therm
```

Listing 2.2 Zmodyfikowany plik */etc/modules*

Po wprowadzeniu zmian należy zrestartować *Raspberry Pi* (np. za pomocą wykonania polecenia *reboot* w linii komend *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
```

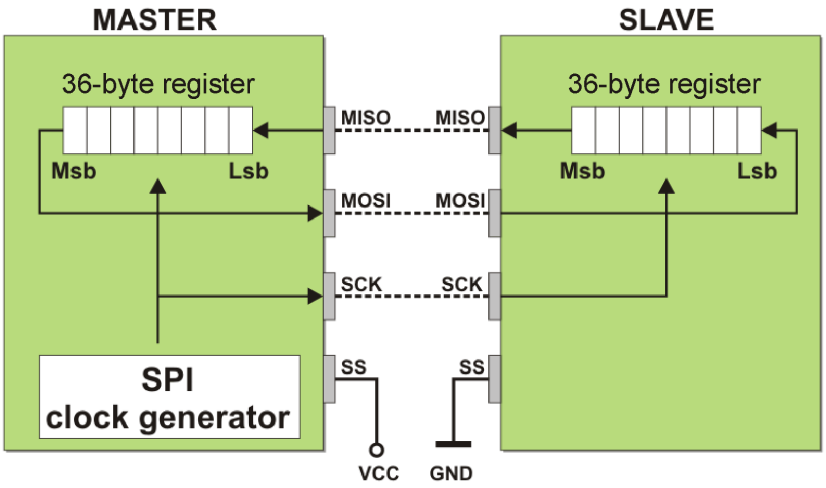
Listing 2.3 Wynik wykonania polecenia *lsmod*

2.4 Interfejs komunikacyjny *SPI*

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

- ***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 pereferyjnego odbywa się za pomocą dodatkowej linii ***SS*** *(Slave Select)*.



Rysunek 2.1 Schemat komunikacji przez interfejs *SPI*

Na rysunku 2.1 jest umieszczony schemat komunikacji między urządzeniem nadrzędnym i pojedyńczym urządzeniem podporządkowanym.

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

Rozdział 3

Implementacja

3.1 Biblioteka do obsługi układu *Texas Intruments tlc5947*

Biblioteka do sterowania układem składa się z trzech plików:

- *tlc5947_ controller.h*
- *tlc5947_ controller.c*
- *RGB.h*

Plik *RGB.h* zawiera structurę *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 cela dalszej możliwości korzystania z nej 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 pllik *tlc5947_ controller.h* zawiera krótki komentarz, który wyjaśnia zasadę indeksowania diod w rejestrze przesuwnym 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 nastepująco:

- **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.

3.1.1 cleanRGB

Funkcja służy do czyszczenia obiektu struktury *RGB*.

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

3.1.2 getLedIndex

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

Nazwa funkcji	*getLedIndex*
Funkcja statyczna	tak
Parametry wejściowe	***uint8_t*** *ledNumber* – numer diody
Parametry wyjściowe	***uint64_t*** *ledAddress* – indeks początku bloku pamięci diody w rejestrze przesuwnym

3.1.3 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 wejsciowy ***RGB*** *rgbSet* ma nastepujace wartosci:

- 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 fragmnet jest później wkładany do tablicy *txdata*, która jest przesyłana do rejestru przesuwnego. Z tego powodu, że otzymany 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 pomoća 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 3.1 Fragment kodu, w którym przedstawiony jest komentarz ilustrujący efekt wykonywania operacji na pamięci

Nazwa funkcji	*compileEvenLedPattern*
Funkcja statyczna	tak
Parametry wejściowe	***RGB*** *rgbSet* – obiek struktury *RGB*, który zawiera wartości dla odpowiednich kolorów
Parametry wyjściowe	***uint64_t*** *pattern* – spreparowany fragment pamięci, który następnie zostanie włożony do rejestru presuwnego

3.1.4 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 wejsciowy ***RGB*** *rgbSet* ma nastepujace wartosci:

- 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 fragmnet jest później wkładany do tablicy *txdata*, która jest przesyłana do rejestru przesuwnego. Z tego powodu, że otzymany 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 pomoća 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 3.2 Fragment kodu, w którym przedstawiony jest komentarz ilustrujący efekt wykonywania operacji na pamięci

Nazwa funkcji	*compileEvenLedPattern*
Funkcja statyczna	tak
Parametry wejściowe	***RGB*** *rgbSet* – obiek struktury *RGB*, który zawiera wartości dla odpowiednich kolorów
Parametry wyjściowe	***uint64_t*** *pattern* – spreparowany fragment pamięci, który następnie zostanie włożony do rejestru presuwnego

3.1.5 insertLedRgb

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

Nazwa funkcji	*insertLedRgb*
Funkcja statyczna	tak
Parametry wejściowe	• ***uint32_t*** *ledNumber* – number diody, której dotyczy działanie • ***uint64_t*** *RGBpattern* – spreparowany 64-bitowy fragment pamięci, zawierający odpowiednie wartości kolorów • ***uint8_t**** *tab* – wskaźnik na tablicę *txdata*, która zostanie później przesłana do rejestru przesuwnego
Parametry wyjściowe	***int*** – wartość *0* mówi o skutecznym wykonaniu operacji; w przypadku błędu zwracana jest wartość *-1*

3.1.6 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 3.3 Fragment kodu, w którym przedstawiony jest komentarz ilustrujący efekt wykonywania operacji na pamięci

Nazwa funkcji	*getLedRGB*
Funkcja statyczna	nie
Parametry wejściowe	• ***uint32_t*** *ledNumber* – number diody, której dotyczy działanie • ***uint64_t*** *RGBpattern* – spreparowany 64-bitowy fragment pamięci, zawierający odpowiednie wartości kolorów • ***uint8_t**** *tab* – wskaźnik na tablicę *txdata*, która zostanie później przesłana do rejestru przesuwnego
Parametry wyjściowe	***RGB*** *ledRGB* – obiekt struktury ***RGB***, który zawiera odpowiednie wartości kolorów czerwonego, zielonego i niebieskiego wybranej diody

3.1.7 setLedRGB

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

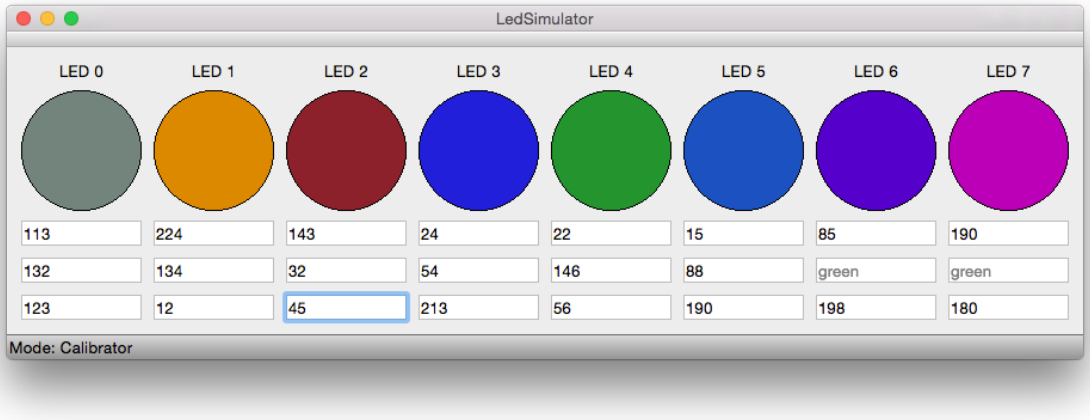
Nazwa funkcji	*setLedRGB*
Funkcja statyczna	nie
Parametry wejściowe	• ***uint32_t*** *ledNumber* – number diody, której dotyczy działanie • ***RGB*** *rgbSet* – obiekt struktury ***RGB***, który zawiera odpowiednie wartości kolorów czerwonego, zielonego i niebieskiego • ***uint8_t**** *tab* – wskaźnik na tablicę *txdata*, która zostanie później przesłana do rejestru przesuwnego
Parametry wyjściowe	***int*** *result* – wartość *0* mówi o skutecznym wykonaniu operacji; w przypadku błędu zwracana jest wartość *-1*

3.1.8 setLedRGB

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

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

3.2 Aplikacja graficzna *LedSimulator*



Rysunek 3.1 Aplikacja graficzna *LedSimulator*: przykładowe kolory

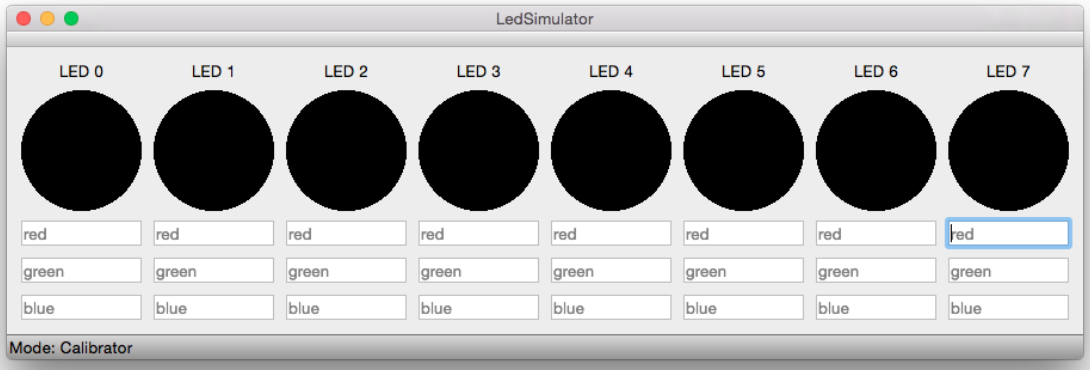
Na podanym zrzucie ekranu (rys. 3.1) jest przedstawione okno główne alikacji graficznej *LedSimulator*. Aplikacja znajduje się w trybie *Color calibration*. Pokazany jest układ zawierający 8 diod RGB. Diody są ponumerowane od *0*. Każdej diodzie przysługują 3 pola tekstowe:

- *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. 3.2).

Aplikacja może pracować w dwóch trybach (rys. 3.3):

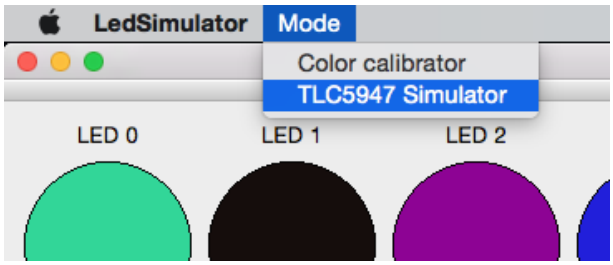


Rysunek 3.2 Aplikacja graficzna *LedSimulator*: domyślny stan diod

- *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 funckcji *runLedScenario*.



Rysunek 3.3 Wybór trybu działania aplikacji *LedSimulator*

Spis rysunków

Literatura

1. A. Janiak, *Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów*, Warszawa: Akademicka Oficyna Wydawnicza PLJ 1999
2. Strona internetowa: http://en.wikipedia.org/wiki/Job-shop_scheduling, 08.10.2014
3. M. Sobolewski, http://www.ioz.pwr.wroc.pl/pracownicy/kuchta/Marek%20Sobolewski_FlowShop.pdf, 08.10.2014
4. C. Smutnicki, *Algorytmy szeregowania zadań*, http://www.kierunkizamawiane.pwr.wroc.pl/materialy/smut.pdf, 08.10.2014
5. Lekcja *The Knapsack Problem*, http://www.es.ele.tue.nl/education/5MC10/Solutions/knapsack.pdf, 08.10.2014
6. Strona internetowa: http://en.wikipedia.org/wiki/Knapsack_problem, 08.10.2014
7. D. Pisinger, *Algorithms For Knapsack Problems*, http://www.diku.dk/~pisinger/95-1.pdf, 08.10.2014
8. Strona internetowa: http://en.wikipedia.org/wiki/Travelling_salesman_problem, 08.10.2014
9. Shen Lin, *Computer Solutions of the Traveling Salesman Problem*, http://alcatel-lucent.com/bstj/vol44-1965/articles/bstj44-10-2245.pdf, 08.10.2014
10. John D. C. Little, Katta G. Murty, Dura W. Sweeney, Caroline Karel, *An algorithm for the traveling salesman problem*, http://dspace.mit.edu/bitstream/handle/1721.1/46828/algorithmfortrav00litt.pdf, 08.10.2014
11. Strona internetowa: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Travelling_salesman_problem.html, 08.10.2014
12. Strona internetowa: http://qt-project.org/doc/, 08.10.2014