ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

6. Neopixels driver

Lead Partner: Warsaw University of Technology

Authors: Daniel Krol

University of Applied Sciences in Tarnow

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

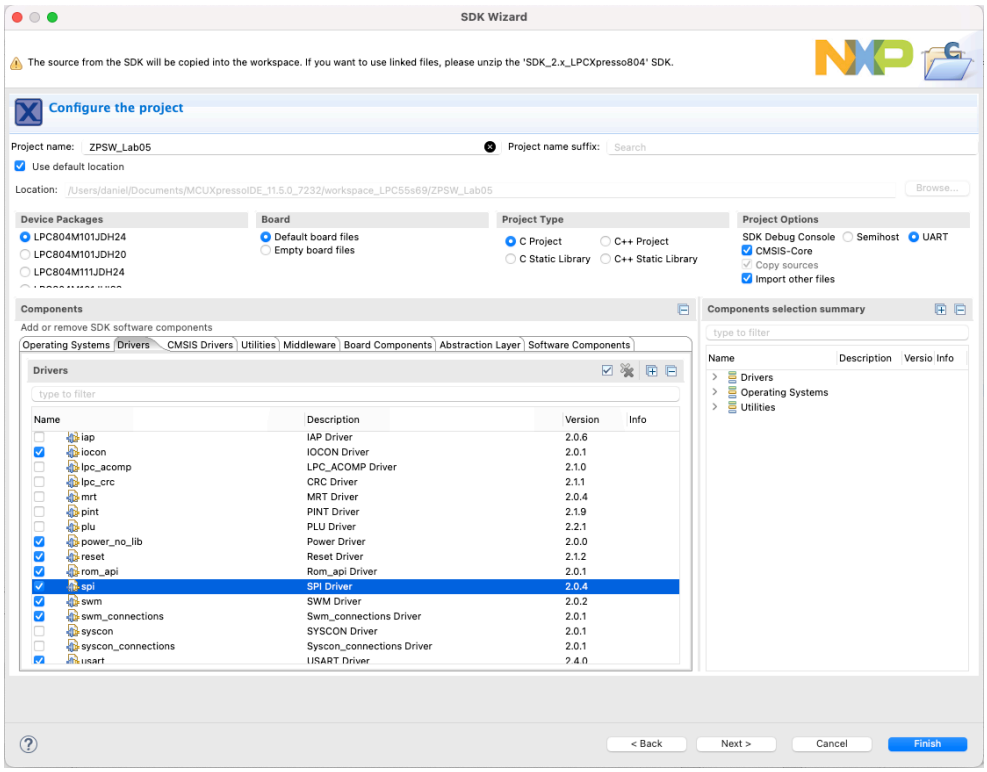
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems

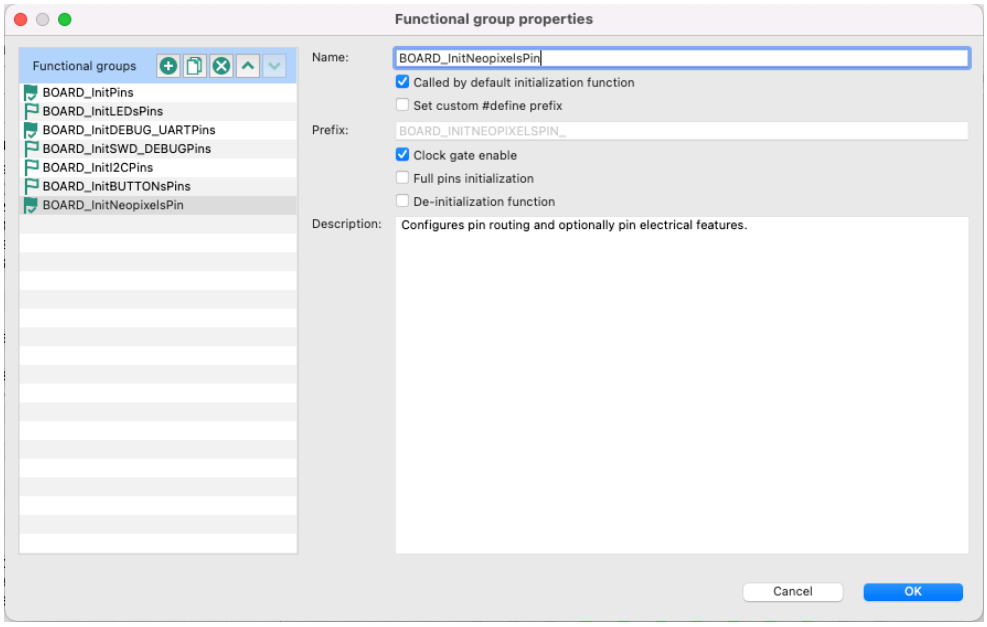
6. Neopixels driver

I. Wykorzystanie interfejsu SPI

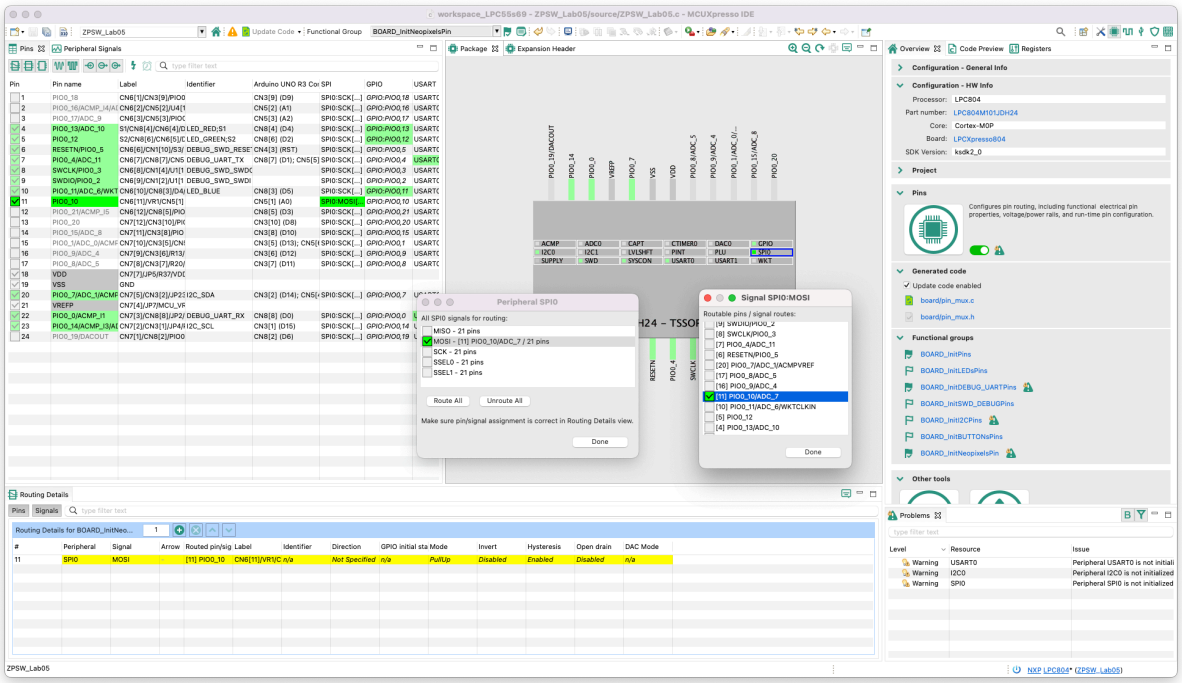
1. Stwórz nowy projekt dla płyty *LPCXpresso804*, tak jak na poprzednich zajęciach i nazwij go *Lab06*.
2. Dodaj sterownik interfejsu *SPI*:



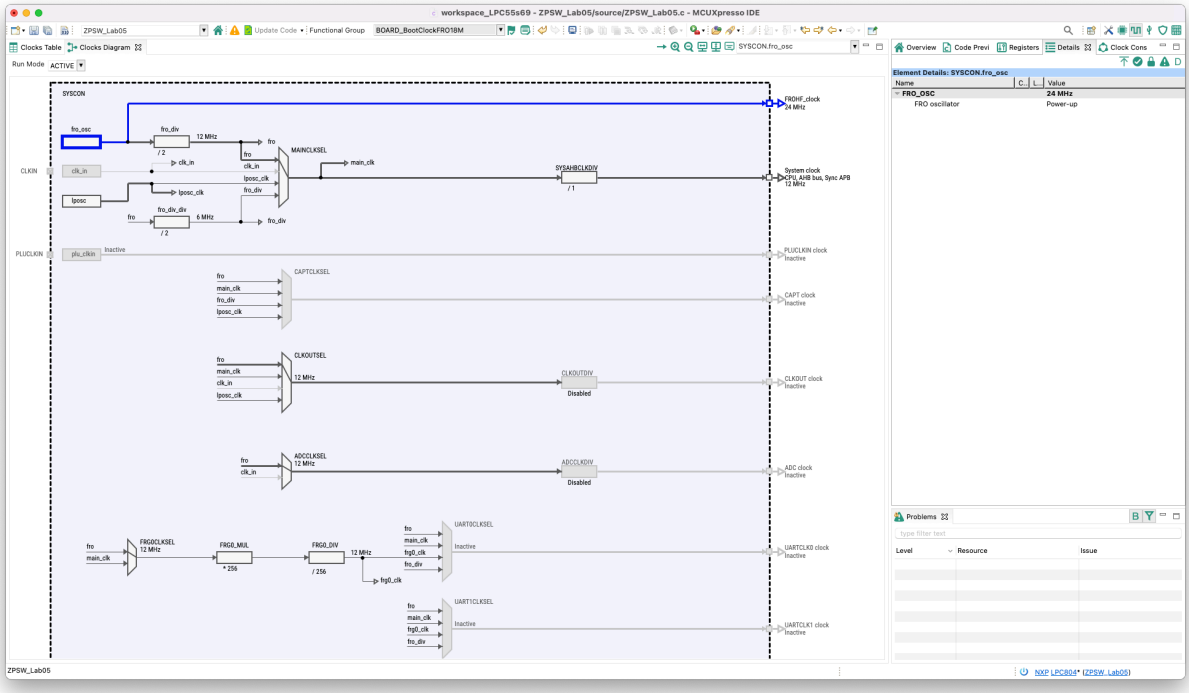
3. Przejdź do *Config Tool ->Pins* a następnie w *Functional group* stwórz nowy preset *BOARD_InitNeopixelsPin*:



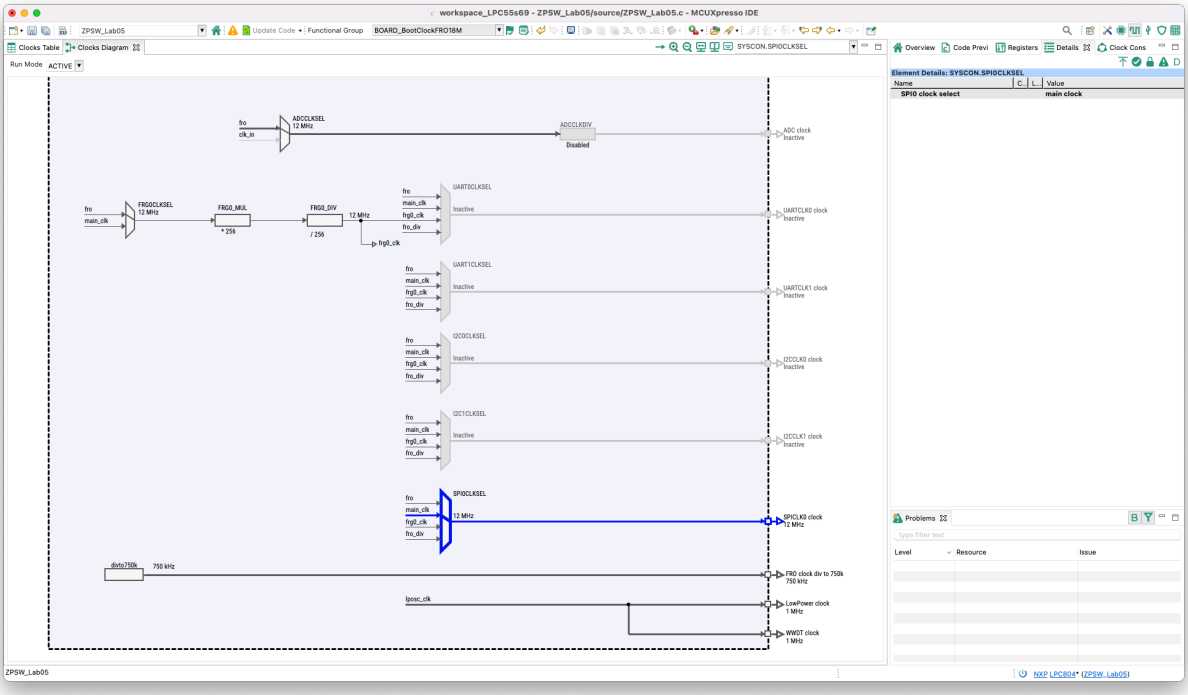
4. Podłącz linię *MOSI* interfejsu *SPI0* do wyprowadzenia *PIO_010* mikrokontrolera:



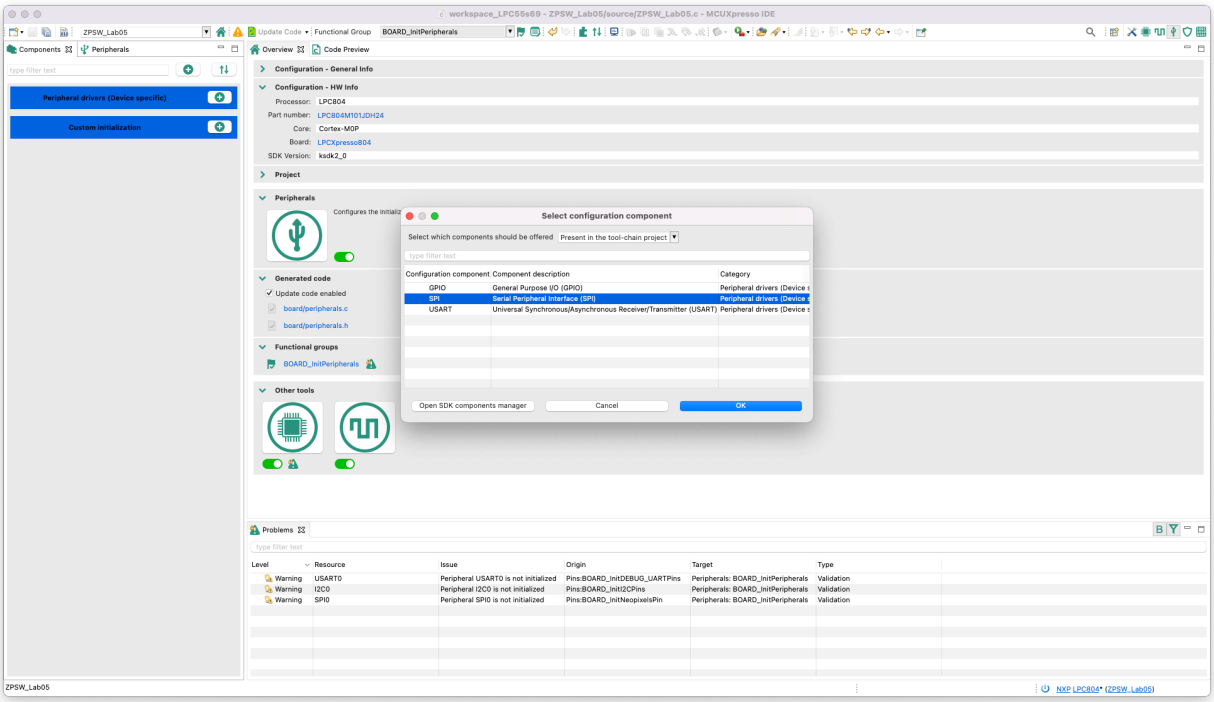
5. W zakładce *Clocks* kliknij na blok oscylatora *FRO_OSC* i zmień jego częstotliwość na 24 MHz:



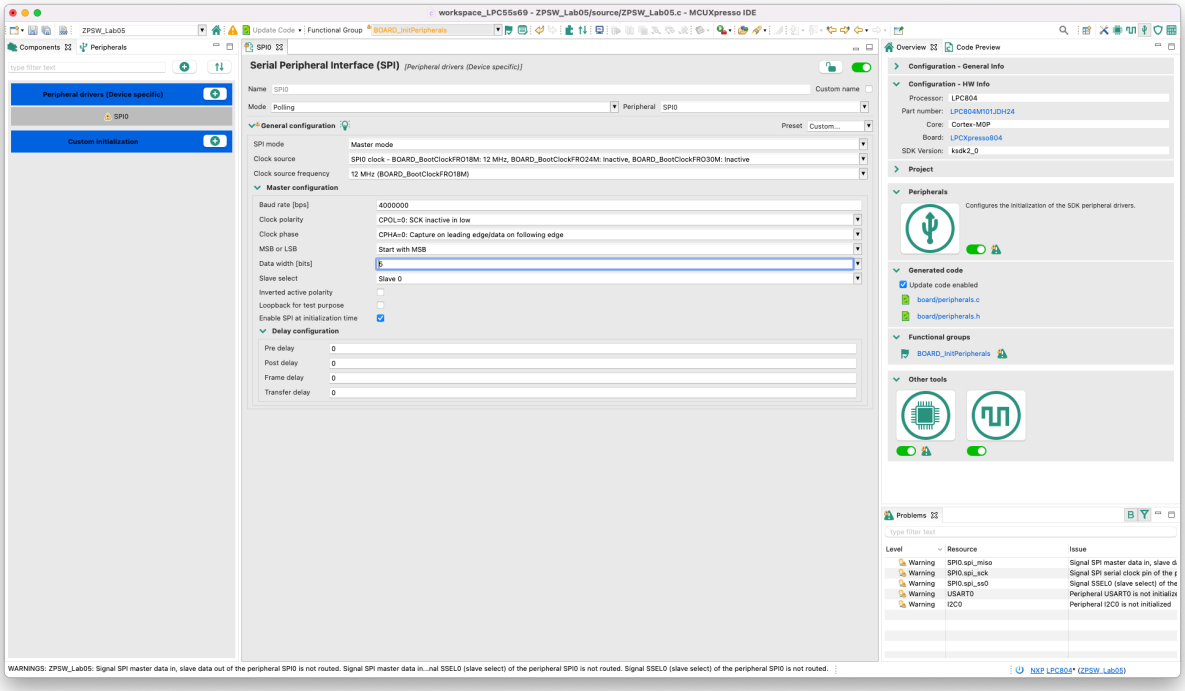
6. Następnie doprowadź sygnał zegara 12 MHz do interfejsu *SPI0* przez wybór *main_clk* w bloku *SPI0CLKSEL*:



7. W zakładce *Peripherals* wybierz sterownik *SPI*:



8. Skonfiguruj parametry transmisji *SPI* w trybie *Polling* dla interfejsu *SPI0*:



9. Przejdź do głównego pliku projektu i zmodyfikuj kod jak poniżej:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
```

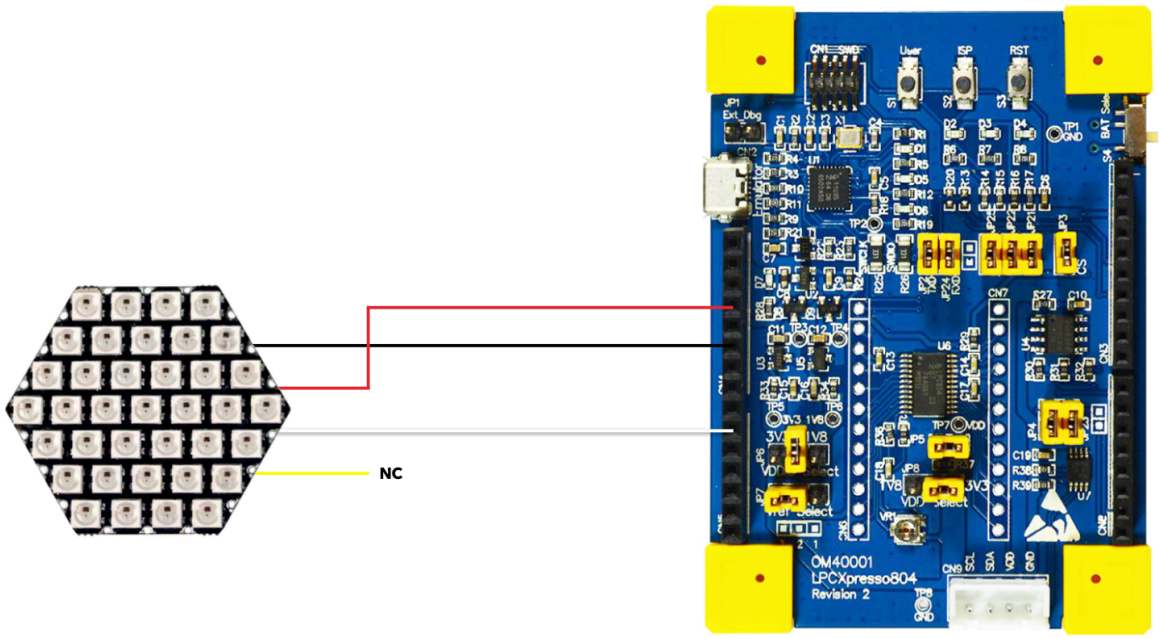
```
        SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

        while(1) {

                colors[18] = 0x00000F;
                Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);

        }
        return 0 ;
}
```

10. Podłącz moduł LED do płytki prototypowej według poniższego schematu:



11. Zbuduj projekt w konfiguracji ***Release***, zaprogramuj układ i sprawdź działanie programu.

UWAGA! Nie wolno włączać **wszystkich diod z pełną jasnością** bez dodatkowego zasilania modułu - grozi uszkodzeniem stabilizatora. Płyta prototypowa jest w stanie zasilić cały moduł LED z ok. **1/16 jasności** - maksymalna wartość dla koloru białego: HEX **0x0F0F0F**.
DEC R:**15,** G:**15,** B:**15**

II. Funkcja setRGB

1. Napisz funkcję ustawiająca kolor w formacie RGB 888
2. Dodaj timer systemowy i sprawdź regulację jasności poszczególnych składowych RGB:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;


void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}


void SysTick_Handler(void) {

          colors[18] = setRGB(k++, 0,   0);
          //colors[18] = setRGB(0,   k++, 0);
          //colors[18] = setRGB(0,   0,   k++);
          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 100); // 100 Hz

          while(1) {

          }
          return 0 ;
}
```

3. Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

II. Funkcja setBrightness

1. Napisz funkcję ustawiająca jasność zadanego koloru:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;


void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

uint32_t setBrightness(uint32_t color, uint8_t level) {

          uint8_t b = level *  (color & 0x0000FF)        / 255;
          uint8_t r = level * ((color & 0x00FF00) >> 8)  / 255;
          uint8_t g = level * ((color & 0xFF0000) >> 16) / 255;

          return ((g<<16) | (r<<8) | b);
}

void SysTick_Handler(void) {

          colors[18] = setRGB(255, 0, 128);
          colors[18] = setBrightness(colors[18], k++);

          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 100); // 100 Hz

          while(1) {

          }
          return 0 ;
}
```

2. Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu dla różnych kolorów RGB.

III. Prosta animacja

1. Napisz funkcję „przesuwającą” świecący punkt po matrycy:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;
uint32_t rgbColor=0;

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

void Animate(uint32_t color)
{
          for(int j=0;j<LEDS;j++)
                    colors[j]=0x000000;

          colors[k]=color;

          k++;
          if(k>=LEDS)
                    k=0;
}

void SysTick_Handler(void) {

          rgbColor = setRGB(0, 0, 15); // MAX RGB: (15, 15, 15)
          Animate(rgbColor);

          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 10); // 10 Hz

          while(1) {

          }
          return 0 ;
}
```

2. Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

IV. Animacja LUT (*lookup table*)

1. Napisz animację wykorzystującą tablicę LUT. Nie przekraczaj wartości 15 na poszczególnych składowych RGB!

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;
uint32_t rgbColor=0;


const bool pic1[4][LEDS] = {{

                        0,0,0,0,
                      0,0,0,0,0,
                     0,0,0,0,0,0,
                    0,0,0,1,0,0,0,
                     0,0,0,0,0,0,
                      0,0,0,0,0,
                       0,0,0,0,
},{
                       0,0,0,0,
                      0,0,0,0,0,
                     0,0,1,1,0,0,
                    0,0,1,0,1,0,0,
                     0,0,1,1,0,0,
                      0,0,0,0,0,
                       0,0,0,0,
},{
                       0,0,0,0,
                      0,1,1,1,0,
                     0,1,0,0,1,0,
                    0,1,0,0,0,1,0,
                     0,1,0,0,1,0,
                      0,1,1,1,0,
                       0,0,0,0,
},{
                       1,1,1,1,
                      1,0,0,0,1,
                     1,0,0,0,0,1,
                    1,0,0,0,0,0,1,
                     1,0,0,0,0,1,
                      1,0,0,0,1,
                       1,1,1,1,
}};

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

void setImage(const bool *image,  uint32_t color) {

          for(int i=0; i<LEDS; i++) {

                    colors[i] = image[i] * color;
          }
}

void SysTick_Handler(void) {
```

```
          rgbColor = setRGB(0, 0, 15); // MAX RGB: (15, 15, 15)
          setImage(pic1[k++ % 4], rgbColor);
          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 5); // 5 Hz

          while(1) {

          }
          return 0 ;
}
```

2. Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.
3. Dodaj kolejną tablicę z animacją:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                      0b10000
#define CODE_1                                      0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;
uint32_t rgbColor=0;


const bool pic1[4][LEDS] = {{

                        0,0,0,0,
                      0,0,0,0,0,
                    0,0,0,0,0,0,0,
                  0,0,0,1,0,0,0,
                    0,0,0,0,0,0,0,
                      0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                      0,0,0,0,0,
                    0,0,1,1,0,0,
                  0,0,1,0,1,0,0,
                    0,0,1,1,0,0,
                      0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                      0,1,1,1,0,
                    0,1,0,0,1,0,
                  0,1,0,0,0,1,0,
                    0,1,0,0,1,0,
                      0,1,1,1,0,
                        0,0,0,0,
},{
                        1,1,1,1,
                      1,0,0,0,1,
                    1,0,0,0,0,0,1,
                  1,0,0,0,0,0,1,
                    1,0,0,0,0,1,
                      1,0,0,0,1,
                        1,1,1,1,
}};

const bool pic2[6][LEDS] = {{

                        1,1,1,1,
                      0,1,0,1,0,
                    0,0,1,1,0,0,
                  0,0,0,1,0,0,0,
                    0,0,0,0,0,0,0,
                      0,0,0,0,0,
                        0,0,0,0,
```

```
},{
                        0,0,0,1,
                       0,0,0,1,1,
                      0,0,0,1,0,1,
                     0,0,0,1,1,1,1,
                      0,0,0,0,0,0,
                       0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                       0,0,0,0,0,
                      0,0,0,0,0,0,
                     0,0,0,1,1,1,1,
                      0,0,0,1,0,1,
                       0,0,0,1,1,
                        0,0,0,1,
},{
                        0,0,0,0,
                       0,0,0,0,0,
                      0,0,0,0,0,0,
                     0,0,0,1,0,0,0,
                      0,0,1,1,0,0,
                       0,1,0,1,0,
                        1,1,1,1,
},{
                        0,0,0,0,
                       0,0,0,0,0,
                      0,0,0,0,0,0,
                     1,1,1,1,0,0,0,
                      1,0,1,0,0,0,
                       1,1,0,0,0,
                        1,0,0,0,
},{
                        1,0,0,0,
                       1,1,0,0,0,
                      1,0,1,0,0,0,
                     1,1,1,1,0,0,0,
                      0,0,0,0,0,0,
                       0,0,0,0,0,
                        0,0,0,0,
}};

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

void setImage(const bool *image,  uint32_t color) {

          for(int i=0; i<LEDS; i++) {

                    colors[i] = image[i] * color;
          }
}

void SysTick_Handler(void) {

          rgbColor = setRGB(0, 0, 15); // MAX RGB: (15, 15, 15)
          setImage(pic2[k++ % 6], rgbColor);
          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);
```

```
        SysTick_Config(SystemCoreClock / 5); // 5 Hz

        while(1) {

        }
        return 0 ;
}
```

4. Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

V. Zadania

1. Stwórz swoje własne animacje LUT
2. Stwórz animację kolorową (pamiętaj żeby nie przekraczać wartości RGB: 15, 15, 15 na wielu diodach)