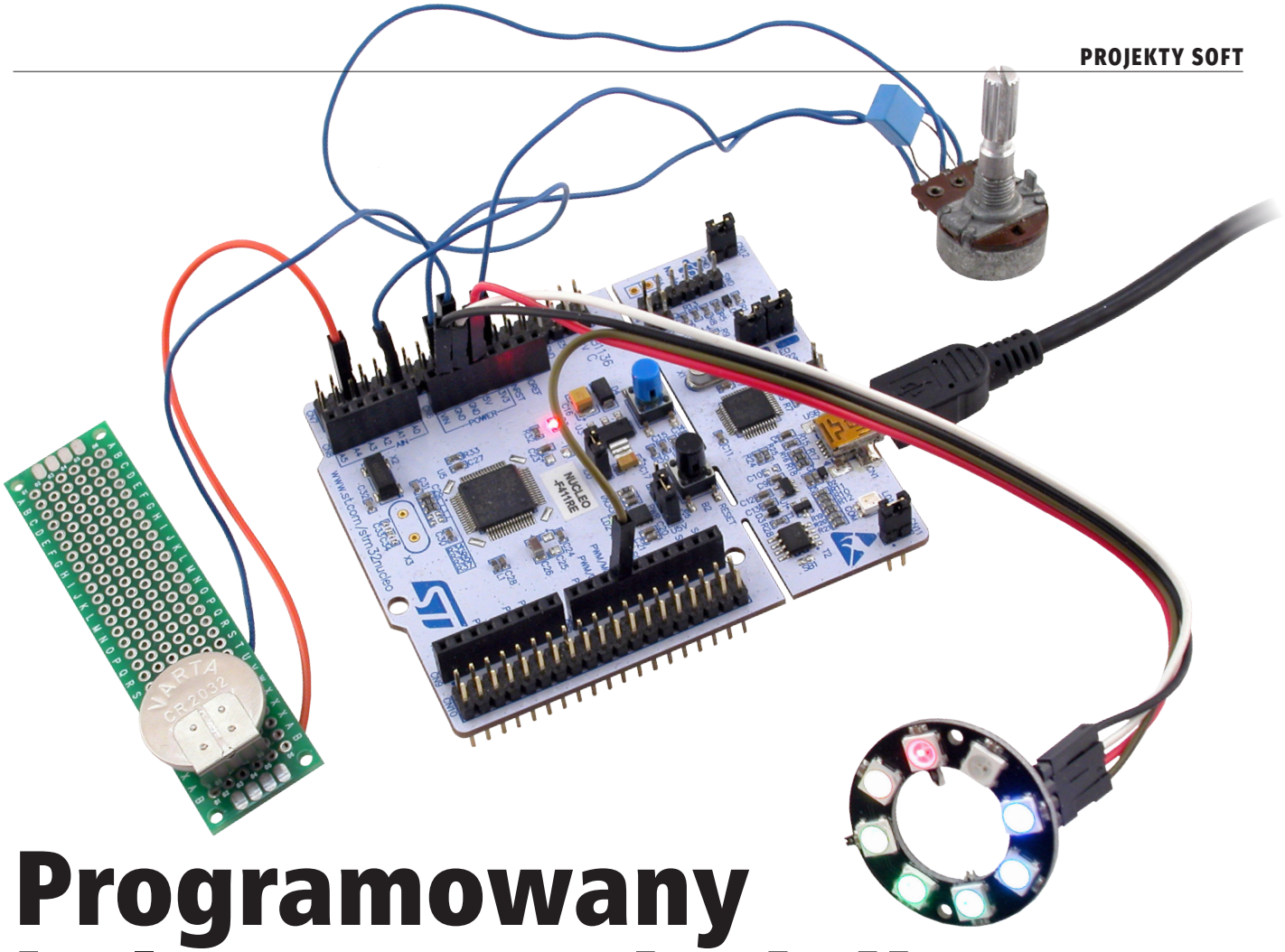


Programowany kolorowy wskaźnik poziomu napięcia

Wskaźniki służą do sygnalizacji stanu obserwowanej wielkości fizycznej. Zamiast szczegółowego wyniku pomiaru, wskaźnik przekazuje informację ogólną, co często jest dużo praktyczniejsze. Ten projekt pokazuje, jak można zastosować płytkę NUCLEO do budowy wskaźnika poziomu napięcia wykorzystującego do sygnalizacji diody WS2812B. Urządzenie jest proste w budowie, z dowolnie programowalnymi progami, których przekroczenie będzie sygnalizowane.

Założenia konstrukcyjne wskaźnika

Wskaźnik ma być przystosowany do sygnalizacji poziomu napięcia w zakresie od 0 V do +3,3 V (po użyciu dzielnika oporowego zakres będzie mógł być zwiększony). Powinna istnieć możliwość dostosowania wskaźnika do sygnalizacji stanu innych wielkości, takich jak temperatura, liczba zliczonych impulsów, czas. Progi zadziałania wskaźnika powinny być w szerokim zakresie konfigurowalne tak jak ich liczba. Powinna istnieć możliwość komunikacji z komputerem w celu programowania wskaźnika i przesyłania komunikatów o jego aktualnym stanie.

Diody LED WS2812

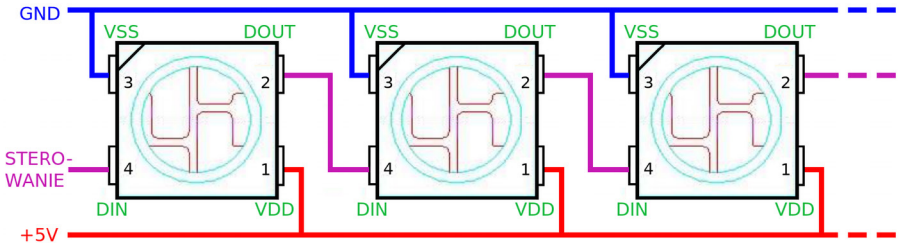
Najprostszym sposobem przekazywania stanu wskaźnika jest sygnalizacja świetlna. Dobrze się do tego nadają trójkolorowe diody LED WS2812B. Są to elementy znane i używane od lat. Ich najważniejsze cechy to:

- zintegrowane we wspólnej obudowie trzy diody LED: czerwona, zielona i niebieska
- duża optyczna jasność świecenia przy relatywnie niskim poborze prądu
- standardowe napięcie zasilania +5 V
- połączona z LED-ami w tej samej obudowie logika sterująca, pozwalająca regulować natężenie każdego z trzech kolorów
- jednoprzewodowa magistrala sterująca
- możliwość szeregowego połączenia WS2812B w wielosegmentowe łańcuchy sterowane tylko jednym wyprowadzeniem

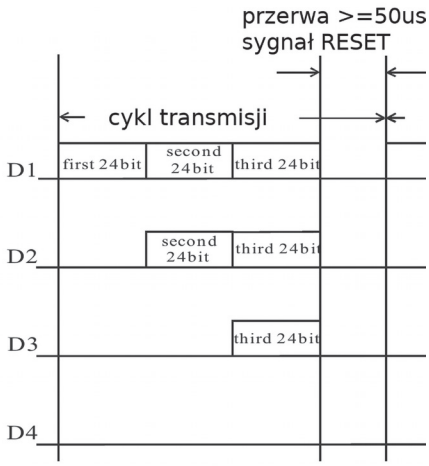
Elementy WS2812B mogą być wstępnie montowane na różnego kształtu i wielkości płytkach drukowanych. Ja miałem dostęp do pierścienia z zamontowanymi 8 diodami WS2812B. Tak więc wskaźnik będzie sygnalizował poziom badanego napięcia świeceniem ośmioma kolorowymi diodami.

WS2812B zasada działania

Dla tych, którzy do tej pory nie zetknęli się jeszcze z LED-ami WS2812B, kilka słów o ich parametrach i sposobie sterowania. Obudowa każdej z diod ma cztery wyprowadzenia. Do dwóch dołącza się zasilanie o standardowym poziomie +5 V. Pozostałe dwa wyprowadzenia to wejście i wyjście magistrali sterującej. WS2812B można łączyć w łańcuchy, tak jak to pokazano na **rysunku 1**.



Rysunek 1.



Rysunek 2.

Ponieważ magistrala sterująca składa się z 1 przewodu, oznacza to, że przesyłane dane są wysyłane szeregowo, bit za bitem. Całkowita długość transmisji zależy od długości kaskady LED. Do sterowania świeceniem pojedynczego elementu WS2812B potrzeba 24 bitów. Składa się na to 8 bitów określających jasność składowej zielonej (G), 8 bitów składowej czerwonej (R) i 8 bitów składowej niebieskiej (B). Cała sekwencja sterująca wygląda tak:

G7 G6 G5 G4 G3 G2 G1 G0 R7 R6 R5 R4 R3 R2 R1 R0 B7 B6 B5 B4 B3 B2 B1 B0

Jako pierwszy wysyłany jest najstarszy bit koloru zielonego, czyli G7, jako ostatni najmłodszy bit koloru niebieskiego B0. Po danych sterujących pracą pierwszego elementu w łańcuchu od razu wysyłane są dane kolejnego i następnego. Wystąpienie przerwy w transmisji oznacza, że po wznowieniu dane będą transmitowane od początku dla pierwszego elementu i kolejnych.

Każdy element odbiera sygnały z magistrali na wejściu DIN, regeneruje je i wystawia na swoim wyjściu DOUT. Z przekazywanej dalej transmisji element wycina pierwsze 24 bity. Pozwala to kolejnym WS2812B odnaleźć część transmisji przeznaczoną dla siebie. Ilustruje to **rysunek 2**.

Przy cyklu transmisji składającym się z 3×24 bitów, w kaskadzie 4 elementów WS2812B, do ostatniego transmisja nie dotrze i nie będzie sterowany.

Ostatnią rzeczą, o której trzeba wspomnieć, jest sposób kodowania bitów. Na każdy bit składają się 2 stany magistrali: wysoki i niski, tak jak to pokazano na **rysunku 3**.

Dla bitu '0' czas trwania impulsu wysokiego T0H wynosi 0,35 µs, a impulsu niskiego T0L 0,9 µs.

Dla bitu '1' czas trwania impulsu wysokiego T1H wynosi 0,9 µs, a impulsu niskiego T1L 0,35 µs.

W przypadku obu bitów '0' i '1' czas transmisji jest taki sam i wynosi 1,25 µs z dopuszczalną tolerancją ±150 ns. Jeżeli czas trwania stanu niskiego przekroczy 50 µs, układy WS2812B zinterpretują to jako sygnał RESET kończący bieżący cykl transmisji.

Platforma konstrukcyjna wskaźnika

Ze względu na sposób sterowania elementami WS2812B najprościej skorzystać z mikrokontrolera i gotowej biblioteki. Ponieważ wskaźnik ma kontrolować poziom napięcia, potrzebny będzie przetwornik ADC i pamięć nieulotna do przechowywania nastaw. Potrzeba także interfejsu UART do komunikacji z komputerem najlepiej poprzez port USB.

Najwygodniej skorzystać z gotowej płytki ewaluacyjnej, takiej, która jest pod ręką. Ja wybrałem NUCLEO-F411, ale nada się do tego większości STM-ów.

Biblioteka sterująca dla WS2812B

Przeszukując zasoby Internetu, znalazłem odpowiadającą mi bibliotekę pod tym adresem ***http://bit.ly/2Wwsof7***.

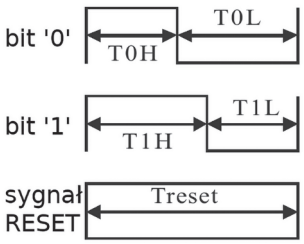
Oprócz linku do plików biblioteki jest tam także opis jej używania. Biblioteka dostosowana jest co prawda do kontrolera STM32F103C8, ale przystosowanie dla STM32F411 wymaga niewiele pracy.

Do przesyłania danych procedury biblioteki wykorzystują interfejs SPI kontrolera oraz DMA. Ale zanim zainstalujemy bibliotekę, mikrokontroler należy zainicjować. W przypadku STM-ów można to bardzo łatwo, zrobić korzystając z graficznego narzędzia STM32CubeMX.

STM32CubeMX – szkielet kodu wskaźnika

W kolejnych krokach pokażę, jak wygenerować szkielet kodu programowalnego wskaźnika poziomu dla płytki NUCLEO-F411. Taki sam sposób postępowania można zastosować w przypadku innych płytek STM.

Po uruchomieniu STM32CubeMX wybieramy nowy projekt dla płytki NUCLEO-F411,



Rysunek 3.

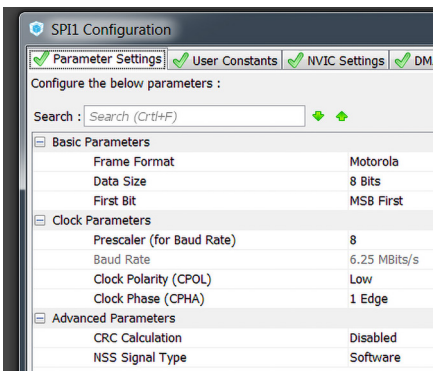
zezwalając na wstępną jej konfigurację. Przechodzimy do zakładki *Pinout*.

- Usuwamy zaznaczenie PA5 LD2 [Green Led], przestawiając port PA5 w tryb *Reset_State*.
- Jako źródło sygnałów taktujących wybieramy sygnał z zamontowanego na płytce programatora *Peripherals → RCC → High Speed Clock (HSC): BYPAS Clock Source*
- Ustawiamy SPI1 w trybie Master *Peripherals → SPI1 → Mode: Transmit Only Master*
- Do komunikacji z zewnętrznym komputerem wybieramy USART2 *Peripherals → USART2: Asynchronous*
- Wybieramy port PA0 jako wejście przetwornika ADC *Peripherals → ADC1: IN0*
- Aktywujemy domenę rejestrów BACKUP, które będą służyć do przechowywania nastaw wskaźnika, gdy nie będzie zasilany *Peripherals → RTC: Activate Clock Source*
- Przechodzimy do zakładki *Clock Configuration*
- Zaznaczamy *PLL Source Mux: HSE*
- Zaznaczamy *System Clock Mux: PLLCLK*
- W polu *HCLK (MHz)* wpisujemy '50' i pozwalamy programowi samodzielnie dobrać podzielniki
- Przechodzimy do zakładki *Configuration*. Wybieramy do ustawienia *Connectivity → SPI1* i w polu *Parameter Settings* ustawiamy parametry SPI1:

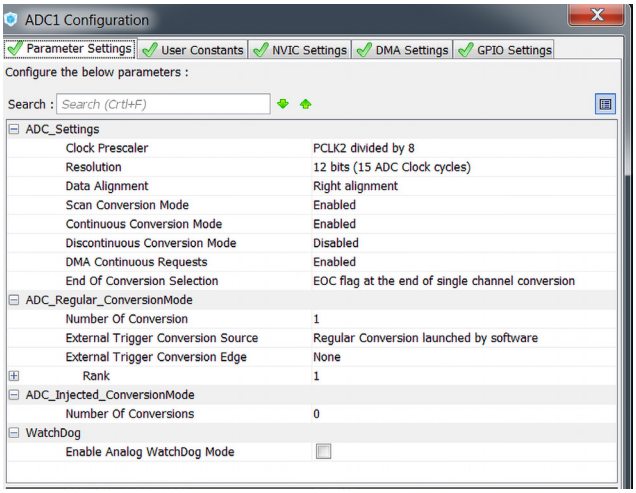
  Prescaler: 8 (Baud Rate 6,25 MHz)
  Data Size: 8 bits
  First Bit: MSB First

Po zmianach pole *Parameter Settings* powinno wyglądać jak na **rysunku 4**.

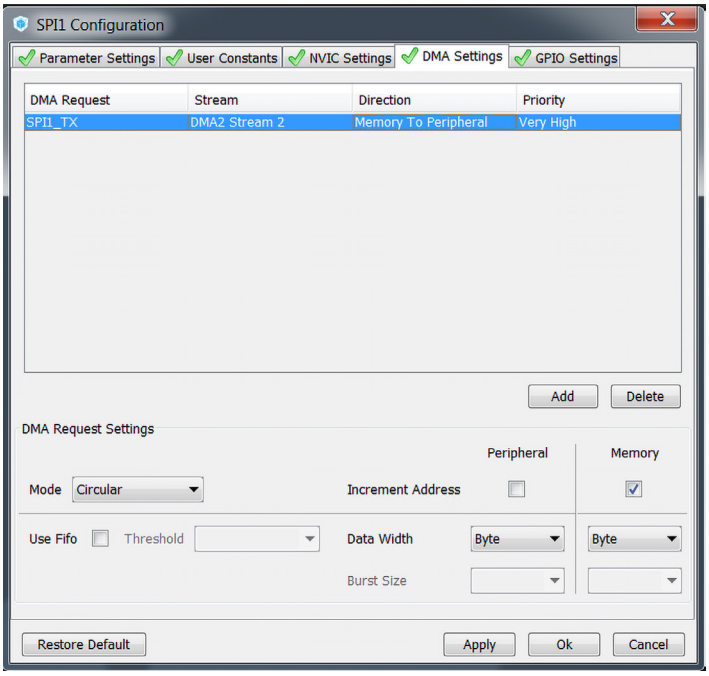
Konfigurujemy DMA do współpracy z SPI1. Wybieramy SPI1 *Configuraion → DMA Settings* i ustawiamy:



Rysunek 4.



Rysunek 6.



Rysunek 5.

Add: SPI1_TX Dma2 STREAM 2 Memory To Peripheral Prority: Very High

Mode Circular, Increment Address Memory, Data Width Byte Byte

Po zmianach pole *DMA Settings* powinno wyglądać jak na **rysunku 5**.

W polu *Connectivity* → *USART 2* ustawiamy parametry USART-a:

Baud Rate: 115200, 8bit, Parity None

Włączamy globalne przerwanie USART2 *NVIC Settings: USART2 global interrupt Enabled.*

W polu *Configuration* → *Analog* → *ADC1* ustawiamy parametry przetwornika ADC, do którego będzie podłączone napięcie badane:

- Parameter Settings

Clock Prescaler PCLK Divided by 8

Resolution 12 bits

Data Alignment Right alignment

Scan Conversion Mode Enabled

- ADC_Settings

Continuous Conversion Mode Enabled

Discontinuous Conversion Mode Disabled

DMA Continuous Request Enabled

- ADC_RegularConversionMode

Number Of Conversion 1 (liczba kanałów wykorzystanych do konwersji ADC)

- Rank 1

Channel: Channel 0

Sampling Time: 15 cycles

Po zmianach pole *ADC1 Configuration* → *Parameter Settings* powinno wyglądać jak na **rysunku 6**.

W sposób podobny jak w przypadku SPI ustawiamy parametry DMA do współpracy z przetwornikiem *ADC1 Analog* → *ADC1* → *DMA Settings*:

Add: ADC1 Dma2 STREAM 0 Peripheral To Memory Prority: Low

Mode Circular, Increment Address Memory, Data Width Half Word

To prawie koniec pracy z STM32CubeMX. Jeszcze tylko w zakładce Project Settings należy podać nazwę projektu, ścieżkę, wybrać tolchain /IDE (w moim przypadku to SW4STM32).

Po wybraniu opcji Project → Generate Code zostanie w podanym przez nas katalogu wygenerowany szkielet kodu.

Uzupełnienie kodu projektu programowalnego wskaźnika poziomu

Teraz można uzupełnić projekt pozostałą częścią kodu.

Biblioteka WS2812B.

Plik źródłowy biblioteki ws2812b.c można pobrać spod tego adresu: ***http://bit.ly/2Hbkyma***.

Plik nagłówkowy ws2812b.h spod tego: ***http://bit.ly/2LtSaja***.

Pliki należy dodać do projektu. Ja najpierw utworzyłem w drzewie projektu folder WS2812_LIB (w środowisku SW4STM32 robi się to tak: File → New → Folder: nazwa folderu). Potem przekopiowałem do utworzonego folderu pobrane pliki File → Import → File System → Next: wskazanie położenia plików do przekopiowania.

Uwaga: jeśli tworzymy w projekcie nowy folder należy podać kompilatorowi jego ścieżkę dostępu, tak żeby wiedział, gdzie znaleźć dodane pliki. Robi się to tak: Project Explorer → nazwa projektu → Properties → C/C++ Build → Settings → Includes: (+) Add…

Workspace: wskazać na liście dodany podkatalog i zatwierdzić.

Wreszcie poprawki kosmetyczne dostosowujące bibliotekę do współpracy z STM32F411 (stary tekst zakomentowałem).

W pliku ws2812b.c

```
#include „stm32f4xx_hal.h”
//#include „stm32f1xx_hal.h”
//#include „spi.h”
//#include „dma.h”
//#include „gpio.h”
```

W pliku ws2812b.h

```
#define WS2812B_LEDS 8
//#define WS2812B_LEDS 35
```

Biblioteka powinna zostać zainicjowana w pliku *main.c* wywołaniem funkcji z parametrem, który jest wskaźnikiem do portu SPI1

```
WS2812B_Init(&hspi1);
```

Procedury przetwornika ADC

W folderze Procedury_ADC_DMA umieściłem krótki kod związany z obsługą przetwornika ADC. Najpierw w *main.c* należy zainicjować nieprzerwany cykl konwersji, których rezultaty DMA automatycznie będzie umieszczać w zadeklarowanej 16-bitowej zmiennej (jako że zakres konwersji ADC mieści się w przedziale 0…4095, zmienna typu uint16_t wystarczy do przechowywania rezultatu pracy przetwornika).

```
//start konwersji
void ADC_DMA_Start(void)
{
        HAL_ADC_Start_DMA(&hadc1,
        &pomiar_ADC, 1);
}
```

Jako argumenty przekazuje się do funkcji wskaźnik (uchwyt) do wcześniej automatycznie zainicjowanego przetwornika ADC, wskaźnik do zadeklarowanej 16-bitowej zmiennej i liczbę konwersji w grupie, w tym przypadku aktywne jest tylko 1 wejście przetwornika.

Funkcja odczytu bieżącego pomiaru (konwersji) przetwornika jest równie nieskomplikowana:

```
//odczyt wyniku konwersji
uint16_t
ADC_DMA_OdczytWyniku(void)
{
        return pomiar_ADC;
}
```

Procedury Backupu

Podtrzymywane bateryjnie 32-bitowe rejestry kontrolera przechowują wartości przełączania

progów wskaźnika w czasie, gdy normalne zasilanie płytki jest odłączone. Progi to 16-bitowe wartości porównywane z bieżącym wynikiem konwersji przetwornika ADC. Jeżeli wynik konwersji jest równy lub większy od wartości progu, odpowiednia dioda jest zapalana. W przeciwnym wypadku dioda gaśnie.

Każdym razem gdy są programowane nowe wartości progów i przesyłane z dołączonego komputera, są one zapisywane do rejestrów Backup. Procedura zapisu do pojedynczego rejestru – **listing 1**.

Dla oszczędności miejsca w jednym 32 bitowym rejestrze Backup zapisuje się dwie 16 bitowe wartości kolejnych progów. Ponieważ do dyspozycji jest 8 diod potrzeba zapamiętać 8 wartości progów co oznacza użycie 4 rejestrów Backup.

Po włączeniu zasilania zawartość rejestrów Backup jest odczytywana, a procedura odczytu pojedynczego rejestru wygląda jak na **lisingu 2**.

Przesyłany jako pierwszy wskaźnik &hrtc to uchwyt do zainicjowanej wcześniej domeny zegara RTC i rejestrów Backup.

Procedury UART-a

Żeby móc odbierać transmisje z przesyłanymi wartościami progów, układ musi być wyposażony w obsługę portu szeregowego UART2. Każdy odebrany portem bajt uruchamia przerwanie, którego obsługa w sekcji użytkownika, umieszczona jest w pliku *main.c* i wygląda jak na **listingu 3**.

Bajty po odebraniu umieszczane są w buforze kołowym. Zmienne za dyrektywą *extern* związane są właśnie z tym buforem. Osobna procedura służy do przepisania odebranych bajtów transmisji z buforu kołowego do innego wskazanego miejsca.

Z kolei do wysyłania transmisji portem UART2 (np. żeby wysłać potwierdzenie odebranej transmisji) służy taka prosta procedura z **listingu 4**.

Najpierw dane do wysłania przepisywane są z miejsca wskazywanego przez wskaźnik **p_bufor* do buforu transmisji *bufor_Tx_UART2[]*. Następnie wywoływana jest nisko poziomowa procedura HAL. Wskaźnik &huart2 to uchwyt do zainicjowanego wcześniej portu UART2.

Procedury protokołu

W tym folderze zgromadzone zostały procedury obsługujące transmisje odbierane z komputera. Sprawdzana jest kompletność odebranej transmisji po czym nowe ustawienia progów są dekodowane i zapisywane do rejestrów Backup. Na koniec do komputera odsyłane jest potwierdzenie odebranej transmisji.

Progowanie

To główna procedura animująca pracę programowalnego wskaźnika poziomu (**listing 5**).

Najpierw sprawdza się czy od ostatniej obsługi stanu diod upłynął czas 500 ms. Chodzi o to żeby zapalanie i gaszenie diod nie przebiegało zbyt nerwowo.

Potem do zmiennej *adc_wynik_konwersji* przepisywany jest wynik bieżącej konwersji przetwornika ADC. Następnie w pętli wynik konwersji porównywany jest z progiem przełączania dla każdej z diod, zapisanym w tabeli *progi_FORMAT_DEC_ADC_tab[]*.

Jeżeli wynik konwersji ma wartość mniejszą od wartości progu zadziałania diody, zostanie ona wyłączona. W przeciwnym wypadku zostanie zapalona rozkazem *WS2812B_SetDiodeRGB()*. Jego pierwszym parametrem jest numer diody w łańcuchu a pozostałymi trzema wartości odpowiadające natężeniu świecenia składowych RGB. Ja kolory zapalanych diod dobrałem od niebieskiego (dioda 0), przez odcienie zielonego, żółtego, czerwonego po jasnofioletowy (dioda 7). Ponadto natężenie świecenia diod zostało tak ustawione, żeby łączny średni prąd nie przekraczał 10 mA.

Na koniec, jeżeli wystąpiła zmiana, włączenie lub zgaszenie którejś z diod, procedura *KomunikatZmianaStanudDiody()* wysyła poprzez UART2 komunikat o tym, które progi zadziałały.

Główna pętla

Główna pętla programu w pliku *main.c* przedstawiona jest na **listingu 6**.

W nieskończonej pętli wywoływane są 3 podprogramy:

- transmisja odświeżająca stan diod
- sprawdzenie, czy zadziałał któryś z progów
- obsługa odbioru ewentualnej transmisji z nowymi ustawieniami progów

Schemat podłączeń

Na **rysunku 7** pokazany został schemat podłączeń do płytki NUCLEO działającej jako programowany wskaźnik poziomu napięcia wejściowego.

- Wyprowadzenie VBAT połączyć z '+' baterii podtrzymującej, '–' baterii połączyć z dowolnym wyprowadzeniem GND. UWAGA: przed dołączeniem baterii należy usunąć zworę SB45 normalnie zamontowaną na spodniej części płytki.
- Wyprowadzenie PA7 połączyć z wejściem magistrali DI płytki z diodami WS2812B. Wejście 5 V połączyć

Listing 1.

```
#define BKP_DR_NUMBER                    4
//zapis do rejestru BACKUP
void WriteToBackupReg(uint8_t num_rejestru, uint32_t dana)
{
          if (num_rejestru >=BKP_DR_NUMBER) return;//błąd rejestr o takim numerze nie istnieje
          HAL_RTCEx_BKUPWrite(&hrtc, num_rejestru, dana);
}
```

Listing 2.

```
//odczyt z rejestru BACKUP
uint32_t ReadFromBackupReg(uint8_t num_rejestru)
{
uint32_t dana;

          dana =HAL_RTCEx_BKUPRead(&hrtc, num_rejestru);
          return dana;
}
```

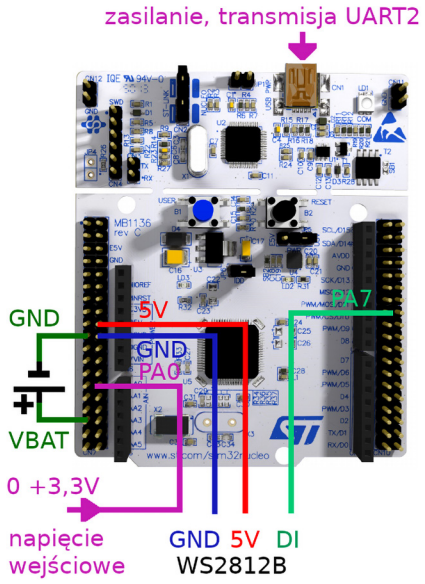
Listing 3.

```
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *UartHandle)
{
extern uint8_t *p_zapis_buf_kolowy_Rx_UART2, bajt_odczytany_z_UART2, bufor_kolowy_Rx_UART2[];

          if (UartHandle->Instance == USART2)
          {
                    //zapis odczytanego bajtu do bufora kołowego
                    *p_zapis_buf_kolowy_Rx_UART2 =bajt_odczytany_z_UART2;
                    //inicjacja przerwania odczytu UART-a
                    HAL_UART_Receive_IT(&huart2, &bajt_odczytany_z_UART2, 1);
                    //inkrementacja wskaźnika zapisu do bufora kołowego
                    p_zapis_buf_kolowy_Rx_UART2++;
                    if (p_zapis_buf_kolowy_Rx_UART2 >=&bufor_kolowy_Rx_UART2[BUF_KOLOWY_ROZMIAR_RX_UART])
                    {
                              //przewiniecie wskaźnika zapisu bufora na początek
                              p_zapis_buf_kolowy_Rx_UART2 =&bufor_kolowy_Rx_UART2[0];
                    }
          }
}
/* USER CODE END 4 */
```

Listing 4.

```
//procedura inicjacji zapisu portem
void ZapisPortemUART2(uint8_t *p_bufor, uint16_t ile)
{

                    memmove ((char *)&bufor_Tx_UART2[0], (char *)p_bufor, ile);
                    while (HAL_UART_Transmit_IT(&huart2, &bufor_Tx_UART2[0], ile) != HAL_OK);
}
```

```
Listing 5.
uint8_t stan_diod_tab[FORMAT_DEC_ADC_ILE_PROGOW] ={0,0,0,0,0,0,0,0};
//sterowanie przełączaniem LED
void Progowanie(void)
{
uint16_t adc_wynik_konwersji;
static uint32_t sysTick_back;
extern uint16_t progi_FORMAT_DEC_ADC_tab[FORMAT_DEC_ADC_ILE_PROGOW];
uint8_t stan_diod_tab_tmp[FORMAT_DEC_ADC_ILE_PROGOW];

          //zmiana stanu LED co 500ms
          if ((HAL_GetTick() -sysTick_back) <500) return;

          sysTick_back =HAL_GetTick();
          adc_wynik_konwersji =ADC_DMA_OdczytWyniku();

          for (uint8_t x=0;x<FORMAT_DEC_ADC_ILE_PROGOW;x++)
          {
                    if (adc_wynik_konwersji <progi_FORMAT_DEC_ADC_tab[x])
                    {
                              //segment wygaszany
                              WS2812B_SetDiodeRGB(x, 0, 0, 0);
                              stan_diod_tab_tmp[x] =0;
                              continue;
                    }
                    switch (x)
                    {
                              case 0:
                              {
                                        WS2812B_SetDiodeRGB(0, 0, 0, 10);
                                        stan_diod_tab_tmp[x] =1;
                                        break;
                              }
                              ***
                              w tym miejscu
                              obsługa zapalania pozostałych diod czyli
                              case 1:
                              …
                              case 2:
                              …
                              case 3:
                              …
                              case 4:
                              …
                              case 5:
                              …
                              case 6:
                              …
                              ***
                              case 7:
                              {
                                        WS2812B_SetDiodeRGB(7, 4, 1, 1);
                                        stan_diod_tab_tmp[x] =1;
                                        break;
                              }
                    }
          }
          KomunikatZmianaStanuDiody(stan_diod_tab_tmp);
}
```



Rysunek 7.

```
Listing 6.
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
            WS2812B_Refresh();
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
            Progowanie();
            ProcProtok_OdbiorTransmisji();
  }
  /* USER CODE END 3 */
```

z wyprowadzeniem +5 V płytki NUCLEO a wejście GND z dowolnym wyprowadzeniem GND.

- Wyprowadzenie PA0 płytki NUCLEO podłączyć do badanego napięcia wejściowego.
- Gniazdo USB zamontowane na płytce NUCLEO połączyć z portem USB komputera lub po zaprogramowaniu progów do gniazda podłączyć wtyk zasilacza stabilizowanego napięcia +5 V.

Dokonując modyfikacji programu, czy to przez ustawienie większej siły świecenia wybranych diod, czy przez zwiększenie ich liczby w kaskadzie, należy zadbać, aby sumaryczny prąd pobierany przez elementy WS2812B nie przekroczył 15 mA. Zbyt duże obciążenie spowoduje przegrzanie i uszkodzenie stabilizatora na płytce, w przypadku tego opisu NUCLEO-F411.

Praca wskaźnika

Po zaprogramowaniu kontrolera powinna się zapalić co najmniej jedna dioda, pierwsza w kaskadzie. W mojej wersji oprogramowania ma na stałe przypisany próg o wartości '0' i jej świecenie ma sygnalizować prawidłową pracę urządzenia.

Żeby przesłać ustawienia progów, płytka po podłączeniu do portu USB komputera powinna zostać wykryta w menedżerze urządzeń jako nowy port wirtualny. Do przesłania ustawień można użyć dowolnego programu terminalowego, ja posługuję się Brajem. Trzeba ustawić szybkość transmisji na 115200, 8 bitów danych, bez bitów parzystości, 1 bit stopu. Cała transmisja jest czysto tekstowa i może wyglądać tak:

```
a;1=100;2=250;3=400;4=560;5=599;
6=3000;7=4095\r\n
```

gdzie:

- 'a' jest prefiksem otwierającym transmisję
- ';' znaki rozdzielające poszczególne pozycje transmisji. Z wyjątkiem '.' i '=' może to być dowolny znak przestankowy
- '1=100' próg zadziałania diody 1. Można ustawić progi dla diod o numerach od 1 do 7. Wartość progu to liczba dziesiętna do porównywania z wynikiem konwersji przetwornika ADC. Gdy próg ma wartość 0, dioda będzie zapalona cały czas, ustawienie progu większego lub równego 4096 spowoduje, że bez względu na poziom napięcia wejściowego dioda się nie zapali
- '\r\n' kody powrotu i nowej linii CR LF kończą transmisję

Po odebraniu transmisji wskaźnik odsyła potwierdzenie będące komunikatem tekstowym zawierającym ustawienia wszystkich progów i zakończonym znakami 'OK'.

Jeżeli chcemy ustawić próg na dokładną wartość napięcia w woltach, korzystamy ze wzoru:

ustawienie progu =
(poziom napięcia [V] / 3,3) * 4095

gdzie liczba 3,3 jest poziomem napięcia referencyjnego dla przetwornika ADC płytki NUCLEO.

Kolejność podawania w komunikacie wartości progów jest dowolna. Jeżeli w komunikacie nie podamy wartości progu dla którejś z diod, zostanie ona zgaszona a wartość progu ustawiona na 65535. Szczególnym przypadkiem jest wysłanie samego prefiksu 'a', co spowoduje zgaszenie wszystkich diod.

Pliki projektu dla IDE SW4STM32 (System Workbench for STM32) powinny być dostępne na serwerach EP. W zamyśle ma to być nie tylko projekt działającego urządzenia, ale i materiał do eksperymentów. Szczególnie dla konstruktorów, którzy dopiero poznają STM-y. Chciałem pokazać, jak bardzo ułatwia i przyśpiesza pracę nad nowym projektem konfigurator graficzny STMCubeMX.

Wykorzystując jeden z wewnętrznych timerów kontrolera i modyfikując kod, można stworzyć wskaźnik pokazujący poziom zliczonych impulsów.

Ryszard Szymaniak