Płytka ewaluacyjna Nucleo-F411RE

Płytka Nucleo-F411RE należy do rodziny zestawów ewaluacyjnych umożliwiających testowanie możliwości zamontowanego procesora. Oprócz tego, czasami taka płytka świetnie nadaje do roli jednostki centralnej budowanego urządzenia (zwłaszcza przy produkcji małoseryjnej) lub jego prototypu. Zwalnia to konstruktora z konieczności wykonania, uruchomienia i przetestowania „najbliższego otoczenia" procesora – wtedy może on po prostu zająć się oprogramowaniem, nie musi tracić czasu na uruchomienie sprzętu. Na wspomnianej płytce zamontowano mikrokontroler STM32F411RE w obudowie z 64 wyprowadzeniami. Przyjrzyjmy się jej możliwościom.

Nowo projektowane urządzenie składa się zazwyczaj z kilku bloków funkcjonalnych. Obok procesora może to być klawiatura, wyświetlacz, bloki interfejsów do komunikacji z otoczeniem, czujników i inne. Płytka Nucleo-F411RE nawiązuje do filozofii Arduino, więc została wyposażona w standardowe złącza, do których można dołączać kolejne płytki z potrzebnymi układami bądź interfejsami kompatybilnymi z Arduino.

Wyposażenie płytki prototypowej

Nucleo kieruje się inną filozofią, niż rodzina płytek prototypowych Discovery. Tamte płytki oprócz kontrolera mają zamontowany zestaw komponentów np. wyświetlacz, interfejsy, czujniki, pamięci, które są na stałe przyłączone do określonych wyprowadzeń.

Na płytce Nucleo oprócz kontrolera, przycisku zerowania, stabilizatora i diody sygnalizacyjnej zamontowane są tylko dwa dodatkowe elementy: przycisk i dioda LED. Integralną częścią jest także programator zgodny z ST-Link. Pozostałe układy na dodatkowych płytkach można dobudowywać korzystając ze standardowych złącz Arduino Uno. Dodatkowo, wszystkie wyprowadzenia mikrokontrolera połączone są z osobnymi złączami szpilkowymi.

Nucleo-F411RE może być zasilany z jednego z 3 źródeł napięcia:

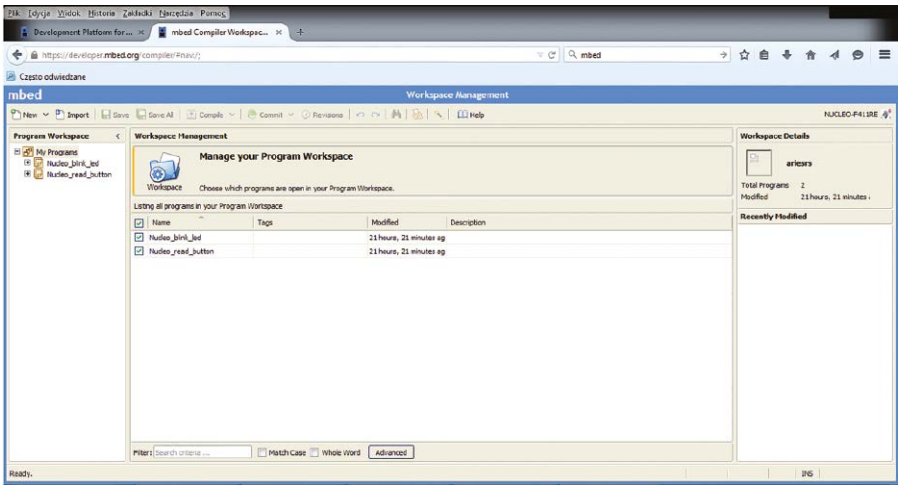
- CN1 – złącza mini USB programatora. W tym wypadku maksymalne obciążenia wnoszone przez układ mikrokontrolera i dołączonych rozszerzeń mogą być rzędu 100 mA.
- VIN – napięcia zewnętrznego, stałego 7…12 V, doprowadzanego do złącz CN6-8 lub CN7-24. W tym wypadku obciążenia wnoszone przez dołączane płytki nie mogą przekraczać 800 mA (dla 7 V) i 250 mA (dla 12 V).
- E5V – zewnętrzne napięcie stabilizowane +5 V o maksymalnej wydajności 500 mA.

W standardowej konfiguracji przy wykorzystaniu do zasilania złącza USB programatora powinny na płytce być założone zwory: CN2 (2 zwory), JP6, JP5 (w pozycji U5V).

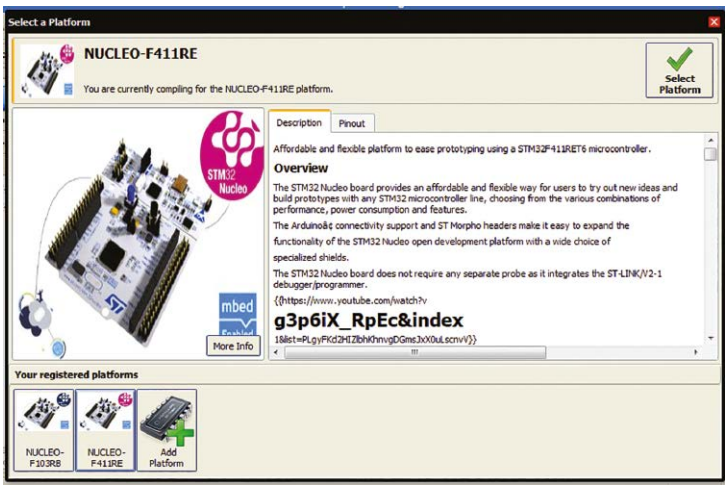
Sterowniki

Do uruchomienia komunikacji z zintegrowanym z płytką Nucleo-F411RE programatorem trzeba zainstalować odpowiednie sterowniki. W tym celu należy wejść na stronę *http://goo.gl/RMrk6y* i pobrać plik *STSW-LINK007*. Po rozpakowaniu i standardowej instalacji programator powinien być widziany w systemie i obsługiwany chociażby przez program ST-Link Utility.

Oprócz wyposażenia sprzętowego, o sile i funkcjonalności każdego urządzenia mikroprocesorowego decyduje jego oprogramowanie. Do pisania i testowania oprogramowania potrzebne są narzędzia programistyczne, w tym kompilator. Dla kontrolera STM32F411 zamontowanego na płytce



Rysunek 1. Okno robocze środowiska mBed



Rysunek 2. Okno wyboru platformy w środowisku mBed

Nucleo-F411RE istnieje sporo narzędzi programistycznych, jednak najbardziej interesujące są te najbardziej dostępne, czyli częściowo lub całkowicie bezpłatne. Dla porównania przedstawione zostaną trzy z nich, środowiska programistyczne: mBed, Keil, CooCox CoIDE.

Środowisko programistyczne mBed

mBed jest czymś więcej niż środowiskiem programistycznym. Jest to portal internetowy, który zawiera narzędzia do szybkiego tworzenia oprogramowania dla wybranych typów płytek ewaluacyjnych z 32-bitowymi mikrokontrolerami rodziny ARM Cortex-M. Inicjatywa jest wspierana przez wiele firm produkujących mikrokontrolery ARM i urządzenia na nich oparte.

Warunkiem koniecznym do korzystania z portalu są zarejestrowanie się i założenie swojego konta. Można wejść na stronę główną portalu *https://mbed.org/* i klikając link *Developer Site* przejść do strony logowania bądź rejestracji. Po rejestracji i zalogowaniu się można przejść do strony środowiska programistycznego klikając na link *Compiler*. Powinien wyświetlić się pulpit zbliżony do tego z **rysunku 1**. W prawym górnym rogu pulpitu znajduje się przycisk z nazwą platformy programistycznej, czyli typem płytki ewaluacyjnej, dla której będzie tworzone oprogramowanie. Klikając na niego można otworzyć okno wyboru platformy, jak na **rysunku 2**. W tym oknie można dodać do menu nowy rodzaj płytki lub z wcześniej wybranych wskazać tę, z którą aktualnie chcemy pracować.

Mocną stroną platformy jest duża ilość programów demonstracyjnych dostępnych „od ręki". Wystarczy użyć przycisku *Import* a na pulpicie wyświetlone zostaną dziesiątki kolejnych stron prostych programów pobieranych z bazy danych portalu. Dodatkowo możemy zawęzić ilość proponowanych programów tylko do tych związanych z interesującym nas problemem np. sterowaniem diody LED zainstalowanej na Nucleo-F411RE. Należy w polu *Search criteria* wpisać słowo *led* i przycisnąć *Search*. Wyświetlone zostaną te programy, w których nazwie lub opisie znajdzie się wyszukiwane słowo.

Jednym z najprostszych może być *Nucleo_blink_led* demonstrujący sposób, w który można zamigotać diodą LD2. Cały kod zawarty w pliku *main.c* zamieszczono na **listingu 1**. Użytkownik może dowolnie modyfikować kod dostosowując przykład do własnych potrzeb. Potem wystarczy kliknąć na *Compile* i po chwili potrzebnej na kompilację zostanie wyświetlony komunikat z zapytaniem o miejsce zapisania otrzymanego pliku wynikowego. Po podaniu katalogu docelowego, na dysku twardym użytkownika z portalu zostanie przesłany

Listing 1. Przykład – zaświecanie/gaszenie diody LED

```
#include „mbed.h"

DigitalOut myled(LED1);
int main() {
    while(1) {
        myled = 1; // LED is ON
        wait(0.2); // 200 ms
        myled = 0; // LED is OFF
        wait(1.0); // 1 sec
    }
}
```

STM32CubeMX – przygotowanie do generowania pakietu plików dla Keil-a:

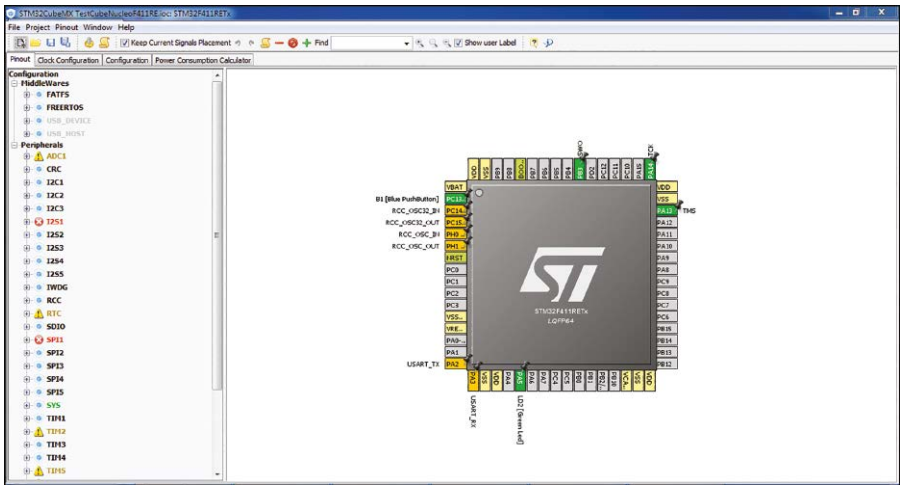
- Otworzyć opcję Project->Settings...
- Wpisać nazwę projektu, który zostanie utworzony.
- Wskazać katalog, w którym mają być zapisane wygenerowane pliki.
- Jako środowisko (Tolchain/IDE) należy wybrać MDK-ARM (rolę MCU Reference powinien pełnić STM32F411RETx).

Po wybraniu opcji Project → Generate Code zostaną wygenerowane i zapisane we wskazanym katalogu pliki programu dla środowiska Keil.

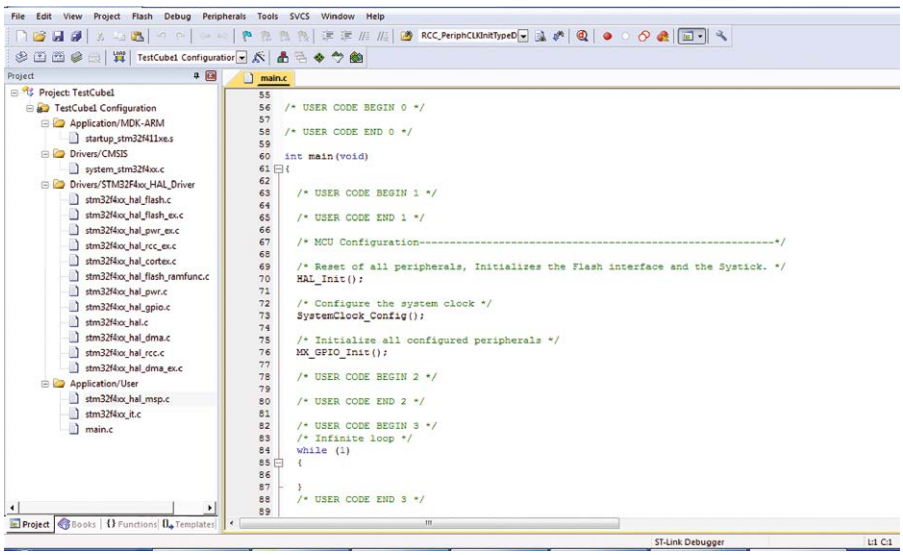
Listing 2. Przykład – obsługa przycisku zamontowanego na płytce

```
#include „mbed.h"

DigitalIn mybutton(USER_BUTTON);
DigitalOut myled(LED1);
int main() {
    while(1) {
        if (mybutton == 0) { // Button is pressed
            myled = !myled; // Toggle the LED state
            wait(0.2); // 200 ms
        }
    }
}
```



Rysunek 3. Okno robocze kompilatora firmy Keil



Rysunek 4. Automatycznie wygenerowany szkielet programu w środowisku Keil

i zapisany plik *Nucleo_blink_led_NUCLEO-F411RE.bin*. Taki plik binarny nadaje się do zaprogramowania pamięci Flash kontrolera za pośrednictwem zintegrowanego z płytką programatora. Do obsługi programatora można użyć firmowego programu firmy ST *STM32 ST-Link Utility*. Innym nieskomplikowanym programem demonstrującym obsługę przycisku B1 na płytce Nucleo-F411RE jest *Nucleo_read_button*. Cały kod źródłowy zawarty w pliku *main.c* zamieszczono na **listingu 2**.

Tworzenie od podstaw własnych programów wymaga znajomości bibliotek *mbed.h*. Biblioteki przeznaczone są dla konkretnych typów płytek ewaluacyjnych, więc trudno zastosować je do pisania programów dla procesorów we własnych urządzeniach. Bardzo jednak przyśpieszają testowanie poszczególnych rozwiązań i układów stosowanych na płytkach ewaluacyjnych.

Środowisko programistyczne Keil

Zintegrowane środowisko programistyczne Keil jest płatnym pakietem programów dla procesorów z rdzeniem ARM, w tym STM32F. W bezpłatnej wersji demo można generować kod wynikowy o objętości do 32 kB. Niby nie jest to dużo, ale w zupełności wystarcza do tworzenia nieskomplikowanych programów. Wielkim atutem środowiska Keil jest duża liczba przykładów w formie gotowych pakietów przygotowanych dla różnych mikrokontrolerów. Dodatkową zaletą jest możliwość generowania szkieletów programów w formacie Keil za pomocą kreatora graficznego STM32CubeMX.

Kreator STM32CubeMX jest narzędziem pozwalającym między innymi na wygenerowanie szkieletu programu źródłowego w formie pakietu gotowego do skompilowania w środowisku Keil. Po wybraniu obiektu (typu procesora lub płytki ewaluacyjnej) oraz pożądanych ustawień np. określeniu, które porty będą używane jako wejścia, a które jako wyjścia, jakie interfejsy i zasoby sprzętowe będą używane (np. UART, timery itp.) automatycznie tworzone są pliki źródłowe dla środowiska Keil. W plikach zawarte są kompletne procedury inicjujące: procedura inicjacji mikrokontrolera, dołączenie wewnętrznych impulsów zegarowych do wybranych interfejsów, skonfigurowanie portów i interfejsów w wybranym trybie pracy. Resztę programu związanego z logiką jego działania użytkownik tworzy samodzielnie.

Pliki instalacyjne kreatora można pobrać ze strony *www.st.com/stm32cube*. Po zainstalowaniu programu i uruchomieniu wybieramy opcję nowego projektu. Teraz należy wskazać obiekt, czyli typ procesora lub płytki. W (STM32CubeMX wersja 4.6.0 lub nowsza) można wybrać z listy płytkę ewaluacyjną Nucleo-F411RE. Po zaakceptowaniu wyboru wyświetli się pulpit, jak na rysunku 3. Na rysunku obudowy kontrolera, który jest zamontowany na płytce, kolorem zaznaczone są wyprowadzenia użyte do doprowadzenia zasilania, podłączenia oscylatorów oraz wyprowadzenia portów (kolor zielony) podłączonych do programatora oraz elementów oddanych do dyspozycji użytkownika: diody LED i przycisku. Widoczna z lewej lista dostępnych interfejsów kontrolera pozwala je aktywować i ustawić tryb ich pracy.

Po otwarciu zakładki *STM32CubeMX zakładka Clock Configuration* możemy wybrać źródła wewnętrznych przebiegów zegarowych mikrokontrolera płytki ewaluacyjnej. Należy zwrócić uwagę, że domyślnie głównym źródłem przebiegów taktujących jest wewnętrzny oscylator HSI RC kontrolera. Na płytce Nucleo-F411RE źródłem impulsów zegarowych może być również przebieg o częstotliwości 8 MHz podawany ze zintegrowanego z płytką programatora ST-Link na pin oznaczony RCC_OSC_IN (PH0). Wybierając to źródło sygnałów taktujących należy:

- Wrócić na zakładkę *Pinout*, na liście interfejsów wybrać *RCC* oraz *High Speed Clock (HSE)* zaznaczyć *BYPAS Clock Source*.

Listing 3. Wygenerowany przez STM32CubeMX szkielet programu

```
/* Private function prototypes */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
/* MCU Configuration */

/* Reset of all peripherals, Initializes the Flash
   interface and the Systick */
    HAL_Init();
/* Configure the system clock */
    SystemClock_Config();
/* Initialize all configured peripherals */
    MX_GPIO_Init();
/* Infinite loop */
    while (1)
    {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
/* Insert delay 500 ms */
        HAL_Delay(500);
    }
}

void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    __PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = 6;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
    HAL_RCC_OscConfig(&RCC_OscInitStruct);
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_PCLK1;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0);
}

/* Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
     PA2    ------> USART2_TX
     PA3    ------> USART2_RX */
void MX_GPIO_Init(void)
{

    GPIO_InitTypeDef GPIO_InitStruct;
/* GPIO Ports Clock Enable */
    __GPIOC_CLK_ENABLE();
    __GPIOH_CLK_ENABLE();
    __GPIOA_CLK_ENABLE();
    __GPIOB_CLK_ENABLE();
/*Configure GPIO pin : PC13 przycisk B1*/
    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
/*Configure GPIO pins : PA2 PA3 USART2*/
    GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
/*Configure GPIO pin : PA5 LD2*/
    GPIO_InitStruct.Pin = GPIO_PIN_5;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}
```

- Na zakładce *Clock Configuration* zaznaczyć opcję *PLL Source Mux->HSE* lub *System Clock Mux->HSE* albo *PLLClk*.
- W wypadku zaznaczenia na czerwono któregoś z bloków skorygować ustawienia podzielników tak aby częstotliwości zegarów mieściły się w dopuszczalnych granicach.

Na zakładce *STM32CubeMX zakładka Configuration* można podejrzeć i zmienić wstępne ustawienia portów i interfejsów. Na przykład, po kliknięciu przycisku GPIO, zostaną wyświetlone ustawienia portów, do których są doprowadzone dioda LED i przycisk. Jeżeli użytkownik wybrał dodatkowe porty, w tym miejscu może ustawić ich parametry np. sposób podciągania do napięcia zasilania czy szybkość działania.

Wygenerowany automatycznie przez *STM32CubeMX* szkielet oprogramowania po otwarciu w środowisku Keil powinien wyglądać podobnie, jak rysunku 4. W osobnych podkatalogach znajdują się najważniejsze pliki w tym pliki biblioteki HAL.

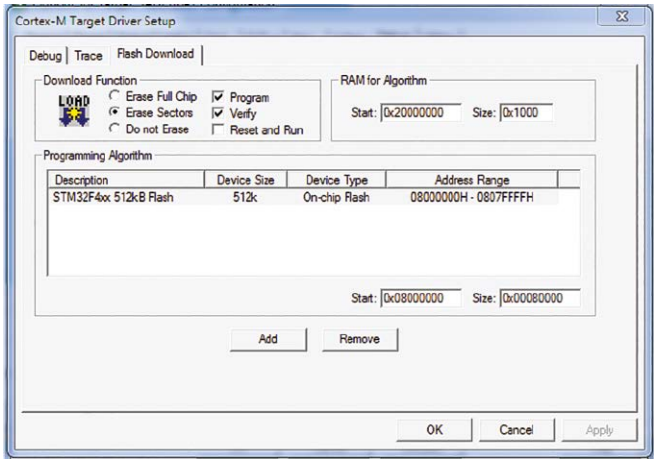
Biblioteka ta (*Hardware Abstraction Layer*) zawiera niskopoziomowe procedury obsługi interfejsów sprzętowych kontrolera. Producent – firma STMicroelectronics – dla każdego typu procesora wykonała odrębne biblioteki, które łączy interfejs API. Dzięki temu ujednolicono nazwy procedur i sposób ich wywołania. Biblioteki nie mogą być wymiennie stosowane dla różnych typów kontrolerów chociażby z powodu różnic w wyposażeniu, liczbie i rodzajach dostępnych interfejsów. Jednak dzięki ujednoliconemu interfejsowi, zostało ułatwione przenoszenie oprogramowania między różnymi typami pokrewnych mikrokontrolerów.

Pliki z oprogramowaniem STM32CubeF4, w tym bibliotekami HAL dla mikrokontrolera STM32F411 zamontowanego na płytce Nucleo-F411RE, można znaleźć pod adresem http://goo.gl/JFHlJH. Oprócz bibliotek wśród plików można też znaleźć przykłady użycia bibliotek HAL, w tym dla Nucleo-F411RE. Na tej samej stronie znajduje się link do pliku „*UM1725: Description of STM32F4xx HAL drivers*" z opisem drajwerów HAL, składnią wywołania funkcji i parametrami.

TestCube1 przykład sterowania portem z dołączoną diodą LED

W przykładowym programie testowym będziemy na powitanie migotali diodą LD2 na płytce NUCLEO-F411RE. Dioda jest podłączona do portu PA5 kontrolera. Wygenerowany przez STM32CubeMX szkielet programu zawiera procedury inicjujące a w podanym przykładzie należało dopisać jedynie 2 linijki kodu. Oczywiście wykorzystane zostały biblioteki HAL. Poniżej zasadnicza część programu zawarta w pliku main.c (**listing 3**).

Na początku procedura *HAL_Init()* inicjuje między innymi systemowy zegar *SysTick Timer* odliczający czas z rozdzielczością 1ms. Zegar wykorzystywany jest przez procedurę pauzy i inne procedury biblioteki HAL. Procedura *SystemClock_Config()* inicjuje wewnętrzny system impulsów taktujących kontrolera. Źródłem jest wewnętrzny oscylator HSI RC kontrolera pracujący z częstotliwością 16 MHz. Procedura *MX_GPIO_Init()* inicjuje wyprowadzenia GPIO. Oprócz portu PA5, do którego dołączono diodę LD2, konfigurowanego jako wyjściowy, konfigurowane są nieużywane w tym przykładzie porty przycisku i interfejsu szeregowego UART2. Jeżeli nie chcemy, aby były inicjowane nieużywane porty, w generowanym szkielecie kodu należy otworzyć projekt dla kreatora STM32CubeMX *TestCube1.ioc* i klikając na symbol graficzny portów skonfigurować je jako nieużywane w trybie *Reset_State*. Następnie, w opisany wcześniej sposób, za pomocą STM32CubeMX należy utworzyć nowy szkielet programu.



Rysunek 5. Okno wyboru opcji STM32F4xx 512kB Flash

W nieskończonej pętli *while(1)* umieszczono dwie dodatkowe linijki kodu. Instrukcja *HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5)* zmienia poziom na wyjściu wyprowadzenia sterującego diodą LED na przeciwny. Jest ona opisana w dokumentacji bibliotek HAL „UM1725: Description of STM32F4xx HAL drivers" w sekcji poświęconej sterowaniu portami GPIO. Kolejna instrukcja, *HAL_Delay(500)*, korzystając z systemowego zegara *SysTick Timer* odmierza 500 ms pauzy. Jest ona opisana w dokumentacji bibliotek HAL w sekcji *HAL System Driver*.

Po skompilowaniu można zapisać plik wynikowy do pamięci Flash kontrolera. Jeżeli zrobimy to z poziomu Keil, najpierw trzeba przygotować kompilator do współpracy z zintegrowanym programatorem na płytce Nucleo-F411RE. W tym celu należy:

- Wybrać opcję: *Flash* → *Configure Flash Tools* → *Debug* i z listy wybrać programator typu *ST-Link Debugger*.
- W opcjach *Settings* → *Flash Download* → *Add* wybrać opcję STM32F4xx 512kB Flash (**rysunek 5**).

Teraz można połączyć kablem port USB komputera z gniazdem mini USB programatora na płytce Nucleo-F411RE i zaprogramować pamięć Flash mikrokontrolera.

TestCube2 przykład na badanie stanu przycisku B1

Drugi przykład jest nieznacznie zmodyfikowaną wersją poprzedniego. Jego działanie polega

Listing 4. Testowanie stanu przycisku B1

```
while (1)
{
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) ==0)
    {
        //przycisk naciśnięty LED świeci
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
    }
    else
    {
        //przycisk zwolniony LED zgaszony
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    }
}
```

Listing 5. Przykładowy program IOToggle po modyfikacji i dostosowaniu do płytki Nucleo-F411RE

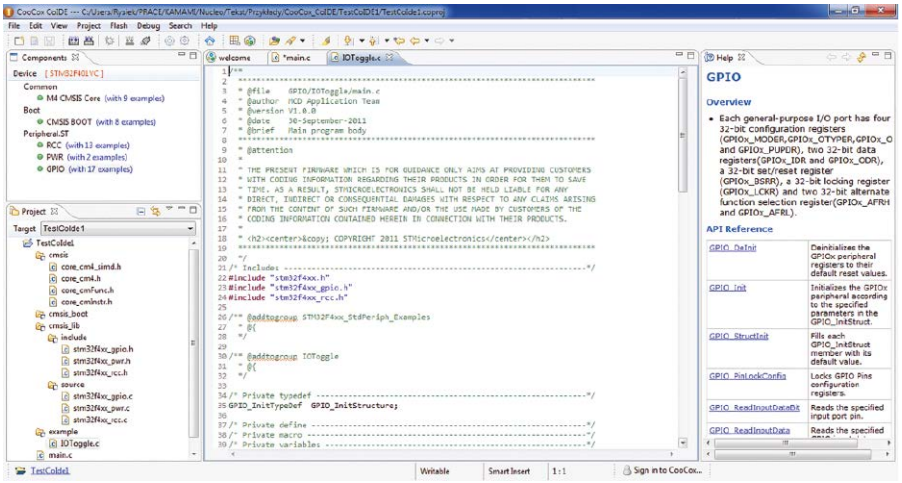
```
/* GPIOA Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
/* Configure PA5 in output pushpull mode */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    while (1)
    {
        /* PA5 to be toggled */
        GPIO_SetBits(GPIOA, GPIO_Pin_5);
        /* Insert delay */
        Delay(0x3FFFFF);
        GPIO_ResetBits(GPIOA, GPIO_Pin_5);
        /* Insert delay */
        Delay(0x3FFFFF);
    }
```

na badaniu stanu przycisku B1. Gdy przycisk jest wciśnięty, dioda LED świeci się, gdy jest zwolniony – gaśnie. Tak jak poprzednio wykorzystano szkielet wygenerowany przez program STM32CubeMX. Kolejne procedury inicjują przebiegi taktujące peryferia mikrokontrolera i ustawienia portów dołączonych do diody oraz przycisku. Zmieniona została treść instrukcji w obrębie pętli *while (1)* – **listing 4**. Instrukcja *HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13)* bada stan przycisku. Jeżeli zwracaną wartością jest „0" oznacza to, że przycisk jest wciśnięty. Instrukcja *HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET)* ustawia wysoki poziom na wyjściu sterującym diodą LED. Jeżeli jest zwracana wartość różna od „0", wyjście zasilające diodę LED jest zerowane. Obie instrukcje opisane są w dokumentacji bibliotek HAL w sekcji poświęconej sterowaniu portami GPIO.

Środowisko programistyczne CooCox CoIDE

Pakiet programistyczny CooCox jest darmowym środowiskiem nienakładającym na użytkownika żadnych ograniczeń odnośnie do sposobu wykorzystania. Przystosowany jest do tworzenia oprogramowania dla procesorów z rdzeniem ARM, w tym dla rodziny układów wytwarzanych przez firmę ST. W dniu pisania artykułu do obsługiwanych typów jeszcze nie dodano obsługi STM32F411, jednak jeśli zadeklarujemy jako docelowy układ STM32D401VC, to większość napisanego oprogramowania po skompilowaniu bez problemu uruchomi się na płytce NUCLEO-F411RE.

Jak w większości współczesnych środowisk IDE, każdy program jest traktowany jako projekt. Dlatego, aby zacząć tworzenie nowego, należy wybrać z menu *Project* → *New Project* i podać nazwę projektu i ścieżkę dostępu do katalogu, w którym ma być zapisany. Następnie wybieramy opcję *Chip* i wskazujemy układ STM32D401VC. Po otwarciu zakładki *Repository* (*View* → *Repository*) zaznaczamy pozycje komponentów, które środowisko automatycznie doda do nowoutworzonego projektu. Dla prawidłowego działania pierwszego programu demonstracyjnego, którym będzie migotanie diodą na płytce należy zaznaczyć następujące komponenty: *M4 CMSIS Core*, *CMSIS BOOT*, *RCC*, *PWR*, *GPIO*. Po zatwierdzeniu jest utworzony szkielet projektu, do którego można dodawać własne pliki. Można sobie ułatwić pracę wczytując jakiś przykładowy projekt dotyczący portów GPIO i przystosować go do wymagań płytki Nucleo-F411RE. Należy w polu Components (lewy górny róg pulpitu) kliknąć na link do przykładów przy nazwie dodanego wcześniej komponentu GPIO. Automatycznie otworzy się zakładka GPIO Examples, gdzie można wybrać np. *IOToggle* i dodać go do projektu klikając *add*.



Rysunek 6. Okno robocze środowiska CooCox CoIDE

Listing 6. Zmodyfikowana wersja programu z list. 5 – dioda LED świeci po naciśnięciu B1

```
#define button GPIOC, GPIO_Pin_13
/* LED GPIOA Periph clock enable */
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
/* LED Configure PA5 in output pushpull mode */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOA, &GPIO_InitStructure);
/* Przycisk GPIOC Periph clock enable */
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
/* Przycisk Configure PC13 in inputt pushpull mode */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

while (1)
{
    if(GPIO_ReadInputDataBit(button) == 0)
    //przycisk naciśnięty, LD2 zapalony
        GPIO_SetBits(GPIOA, GPIO_Pin_5);
        else
    //przycisk zwolniony, LD2 zgaszony
        GPIO_ResetBits(GPIOA, GPIO_Pin_5);
}
```

Widok pulpitu CoIDE z tak utworzonym projektem pokazany został na **rysunku 6**.

Dołączony do projektu przykład *IOToggle* posługuje się funkcjami standardowej biblioteki dla rodziny STM32F4, do której należą kontrolery STM32D401VC i STM32F411. Odpowiednie pliki biblioteki zostały już zainstalowane w projekcie podczas dodawania komponentów. Kompletną bibliotekę można pobrać ze strony *http://goo.gl/alSgGI* w formie spakowanej jako plik *STSW-STM32065*. Po rozpakowaniu dostępne stają się zarówno pliki samej biblioteki, jak i liczne projekty przykładowe. Po uruchomieniu pliku wykonawczego *stm32f4xx_dsp_stdperiph_lib_um.chm* uruchamia się przeglądarka zasobów biblioteki. Wpisanie w pasku indeksu frazy GPIO wyświetla listę dostępnych funkcji związanych z dostępem i sterowaniem portami kontrolera.

Test CoIDE – przykład sterowania wyprowadzeniem z dołączoną diodą LED

Przykładowy program *IOToggle* po modyfikacji i dostosowaniu do płytki NUCLEO-F411RE zamieszczono na **listingu 5**. Najpierw instrukcją *RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE)* jest dołączany sygnał zegarowy do portu PA. Następnie, port PA5 jest konfigurowany do pracy w trybie wyjścia. W nieskończonej pętli *while(1)* naprzemiennie instrukcjami *GPIO_SetBits* i *GPIO_ResetBits* jest ustawiane/zerowane wyjście zasilające diodę LED. Procedura *Delay(0x3FFFFF)* zapewnia około 1-sekundowe opóźnienie pomiędzy przełączeniami stanu diody LED.

TestCoIde – przykład testowania stanu przycisku B1

Teraz nieznacznie zmodyfikujmy program z list. 5, aby dioda LED świeciła się w momencie naciśnięcia przycisku B1. Cały program zamieszczono na **listingu 6**. Najpierw następuje deklaracja etykiety *button* jako portu PC13 do którego na płytce Nucleo-F411RE jest dołączony przycisk B1. Następnie, wyprowadzenie PA5 jest konfigurowane jako wyjście zasilające diodę LED, a PC13 jako wejście używane do odczytania stanu przycisku. Potem, w nieskończonej pętli *while(1)*, jest sprawdzany stan portu PC13. Jeżeli przycisk jest naciśnięty, to port PA5 jest ustawiany W przeciwnym wypadku – zerowany.

Ryszard Szymaniak, EP