Sterowniki robotów

Laboratorium – Wprowadzenie

Wykorzystanie narzędzi STM32CubeIDE oraz STM32CubeMx do budowy programu mrugającej diody z obsługą przycisku

Wojciech Domski

Dokument sr_lab01.pdf, Wersja 2.1.0

Spis treści

1	Wprowadzenie	2
2	Opis ćwiczenia	3
3	Tworzenie projektu w STM32CubeIDE 3.1 Wgrywanie programu	$egin{array}{c} 4 \\ 6 \\ 7 \end{array}$
4	Praca z narzędziem STM32CubeMX 4.1 Przygotowanie projektu w programie STM32CubeMX 4.1.1 Konfiguracja zegara 4.2 Automatyczne generowanie kodu	9 9 11 12
5	GPIO – cyfrowe wejścia/wyjścia ogólnego przeznaczenia 5.1 GPIO w bibliotece HAL	16 17
6	 Zadania do wykonania 6.1 Uzupełnienie programu o kod odpowiedzialny za miganie diodą LED 6.2 Wgranie i testowanie programu na mikrokontrolerze 6.3 Modyfikacja projektu i programu w celu obsługi przycisku pracującego w roli przełącznika dla diody LED 6.4 Wgranie i testowanie zmodyfikowanego programu na mikrokontrolerze 6.5 Uporządkowanie stanowiska 	 19 19 19 22 22 23
7	Podsumowanie	24
Li	teratura	25

1 Wprowadzenie

Ćwiczenie prezentuje podstawowe narzędzia do pracy z mikrokontrolerem firmy ST. Jest to STM32CubeMX [3], [8] – aplikacja pozwalająca w wygodny sposób na przygotowanie podstawowej konfiguracji peryferiów mikrokontrolera poczynając od GPIO, a kończąc na middleware firm trzecich, jak FreeRTOS. Jest ono integralną częścią programu STM32CubeIDE lub występuje osobno jako samodzielna aplikacja. Ponadto, oprogramowanie pozwala na konfigurację częstotliwości taktowania rdzenia w trybie ręcznym lub półautomatycznym. Do rozwoju oprogramowania dla mikrokontrolera wykorzystywane będzie środowisko STM32CubeIDE. Środowisko to bazuje na zmodyfikowanej wersji oprogramowania Atollic TrueStudio for STM32. Jest to środowisko oparte o popularne IDE – Eclipse, jednakże zostało ono odpowiednio skonfigurowane do pracy z układami firmy ST.

Jako zestaw ewaluacyjny wykorzystana zostanie płytka deweloperska NUCLEO-L476RG [4], [5]. Posiada ona mikrokontroler STM32L476RGT6 [1], szerszy opis tej rodziny mikrokontrolerów można znaleźć w [9].

Zdjęcia płytki ewaluacyjnej zostały przedstawione poniżej.



Rysunek 1: Zestaw ewaluacyjny NUCLEO-L476RG



Rysunek 2: Zestaw ewaluacyjny NUCLEO-L476RG w opakowaniu

2 Opis ćwiczenia

Ćwiczenie ma za zadanie zaznajomić studenta z obsługą oraz wykorzystaniem przedstawionego oprogramowania: konfiguracja peryferiów, dobór częstotliwości pracy układu, automatyczne generowanie kodu. Ponadto ćwiczenie, obejmuje zakres związany z obsługą GPIO na przykładzie diody LED (cyfrowe wyjście) oraz przycisku (cyfrowe wejście).

3 Tworzenie projektu w STM32CubeIDE

Nowy projekt można utworzyć na kilka sposobów. Jednym z nich jest wybranie odpowiedniej pozycji z głównego menu ($File \rightarrow New \rightarrow STM32 \ Project$). Wówczas pojawi się okno, które pozwala na wybór jednostki, czy wybór płytki testowej. Konfiguracja projektu od tego momentu jest bardzo podobna do konfiguracji projektu w STM32CubeMX z kilkoma różnicami. Po wybraniu układu, bądź płytki deweloperskiej należy nadać nazwę projektu, tak jak to pokazano na rysunku 3. Pozostałe opcje powinny być bez zmian, to jest wybrany język ustawiony na C, docelowy plik binarny ustawiony an wykonywalny, a typ projektu to STM32Cube.

DE STM32 Projec	t			×
Setup STM32 pi	oject		IC)E
Project				
Project Name:	lab01			
🗹 Use default	ocation			
Location:	C:/Users/Wojciech/STM32CubeIDE/workspace_1.4	.0	Brows	se
Targeted La C C + Targeted Bir Executate Targeted Pro STM32Co	nguage + hary Type le O Static Library bject Type ube O Empty			
? <	Back Next > Finish		Cancel	

Rysunek 3: Wybór nazwy projektu

Po konfiguracji podstawowych ustawień projektu należy przejść do następnego okna (Rys. 4). Na tym etapie należy zaakceptować domyślną konfigurację.

DE STM32 Project		×
Firmware Library Package Setup		DE
Setup STM32 target's firmware		DE
Target and Firmware Package		
Target Reference: NUCLEO-L476RG		
Firmware Package Name and Version: STM32Cube FW_L4 V1.16.0 ~		
Firmware package Repository		
Location:		
C:\Users\Wojciech\STM32Cube\Repository		
See <u>'Firmware Updater'</u> for settings related to firmware package installat	ion	
Code Generator Ontions		
\bigcirc Add necessary library files as reference in the toolchain project configu	iration fi	ile
O Copy all used libraries into the project folder		
• Copy only the necessary library files		
<u>A Back</u> Next > Einish	Canc	el

Rysunek 4: Konfiguracja bibliotek projektu

Po przejściu dalej, pojawi się komunikat z pytaniem, czy zainicjalizować dostępne peryferia (Rys. 5). Należy zaakceptować akcję i oczekiwać na zakończenie działań przez środowisko. W tym momencie program może wymagać pobrania dodatkowych bibliotek z Internetu, których rozmiar to około 700MB.

DE Board	Project Options:			×
2	Initialize all peripherals with their default Mode ?			
		Yes	No	
-				

Rysunek 5: Inicjalizacja wybranych peryferiów na płytce deweloperskiej

W odróżnieniu do aplikacji STM32CubeMX generowanie kodu odbywa się z poziomu programu STM32CubeIDE. W tym celu należy wybrać $Project \rightarrow Generate \ Code$, tak jak to zostało pokazane na rysunku 6.



Rysunek 6: Generowanie kodu w aplikacji STM32CubeIDE

Budowanie projektu odbywa się poprzez wybranie z menu opcji $Project \rightarrow Build All$. Po zbudowaniu projektu, jeśli skompiluje się on bez ostrzeżeń, w katalogu projektu pojawi się folder Debug. To w nim zostaną umieszczone zbudowane obrazy firmware'u (*.bin, *.hex).

Uwaga! Podczas regeneracji kodu może zajść konieczność wyczyszczenia oraz przebudowania projektu. Aby wyczyścić projekt należy z menu wybrać projekt, a następnie wyczyść (*Project* \rightarrow *Clean*)

3.1 Wgrywanie programu

Wgranie programu do mikrokontrolera jest możliwe bezpośrednio z poziomu środowiska programistycznego. W tym celu należy wybrać z menu $Run \rightarrow Run$. Pierwsze wywołanie akcji uruchomienia spowoduje otwarcie okna z konfiguracją (rys. 7). Domyślnie wybrane ustawienia są wystarczające do prawidłowego zaprogramowania układu.

DE Edit Configuration		×	
Edit launch configuration properties			
Name: lab01 Debug			
📄 Main 🕸 Debugger 🐌 Startup 🧤 Source 🔲 Common			
C/C++ Application:			
Debug\lab01.elf	Search Project	Browse	
Project:			
lab01		Browse	
Build (if required) before launching			
Build Configuration: Select Automatically			~
○ Enable auto build ○ Disable auto	build		
Use workspace settings <u>Configure Work</u>	kspace Settings		
	Revert	Apply	
	OK	Cancel	

Rysunek 7: Wgrywanie aplikacji w programie STM32CubeIDE

Każde kolejne wywołanie tej akcji spowoduje wybranie ostatniej konfiguracji. Dostęp do listy dostępnych konfiguracji jest również dostępny z poziomu menu $Run \rightarrow Run \ Configurations \ldots$ Uruchamianie sesji debugowania jest niemal identyczne, różny się ono jedynie wyborem odpowiedniej akcji ($Run \rightarrow Debug$).

Uwaga! Aby możliwe było poprawne uruchomienie aplikacji muszą być spełnione następujące warunki.

- 1. Aplikacja została zbudowana bez błędów.
- 2. Do komputera został podłączony programator (Płytka testowa posiada już wbudowany programator).
- 3. Wszystkie zworki na płytce testowej muszą być we właściwej pozycji.

W przeciwnym wypadku mogą pojawić się błędy.

- 1. Konfiguracja uruchomieniowa nie może odnaleźć zbudowanego programu.
- 2. Programator nie jest podłączony do komputera.
- 3. Mikrokontroler docelowy nie jest odnaleziony najprawdopodobniej przez niewłaściwą konfigurację zworek na płytce testowej.

3.2 Przydatne informacje

STM32CubeMX dostarcza kilku przydatnych opcji podczas pisania oprogramowania. Pierwszą z nich jest automatyczne podpowiadanie nazw funkcji, zmiennych, itp. W tym celu należy wpisać początek nazwy funkcji Ctrl + Spacja. Wówczas wyświetlą się dopasowania, które odpowiadają aktualnie wprowadzonemu fragmentowi funkcji, zmiennej (Rys. 8).



Rysunek 8: Podpowiadanie nazw

Innym przydatnym narzędziem jest podpowiadanie listy argumentów funkcji. Przykład okazano na rysunku 9). Aby wyświetlić podpowiedź należy znaleźć się w liście argumentów funkcji (pomiędzy okrągłymi nawiasami) i wcisnąć kombinację klawiszy **Ctrl** + **Shift** + **Spacja**. Aktualnie aktywny argument jest podświetlany.

Rysunek 9: Podpowiadanie nazw

Podczas gdy chcemy przełączyć się z pliku źródłowego na odpowiadający plik nagłówkowy wystarczy wykorzystać kombinację klawiszy $\mathbf{Ctrl} + \mathbf{Tab}$. Skrót ten umożliwia naprzemienne przełączanie się pomiędzy powiązanymi plikiem nagłówkowym oraz plikiem źródłowym.

Jeśli zajdzie potrzeba przejścia do definicji funkcji można wykorzystać przycisk funkcyjny **F3**. Po najechaniu kursorem na funkcję, której definicja jest interesująca wystarczy wcisnąć klawisz **F3**, a środowisko programistyczne automatycznie odnajdzie definicję. Możliwy jest powrót do poprzedniego miejsca w programie przez wciśnięcie kombinacji klawiszy **Alt** $+ \leftarrow$

4 Praca z narzędziem STM32CubeMX

4.1 Przygotowanie projektu w programie STM32CubeMX

Należy uruchomić program STM32CubeMX. Następnie wybieramy nowy projekt (*New Project*). Operacja ta może chwilę potrwać, gdyż aplikacja sprawdza dostępność nowych produktów. W oknie nowego projektu wybieramy odpowiednią płytkę ewaluacyjną, aby to zrobić należy wybrać zakładkę wyboru płytki (Board Selector), a następnie wybrać typ płytki (*Type*) oraz serię mikrokontrolera (*MCU Series*), Nucleo64 oraz STM32L4, odpowiednio. Po zredukowaniu liczby dostępnych zestawów deweloperskich wybieramy NUCLEO-L476RG z listy (Rys. 10).

New Project										
MCU/MPU Selector Board Selector Board Fitters	Example Select	or Cross	Selector Features	Large Picture	Docs & Resources	Datasheet	Buy	Start Project		
Commercial Part Number	~ + -	*			网络马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马马					
PRODUCT INFO Type Supplier	> > >			STM32 Cube						
MCU / MPU Series > Marketing Status >										
Price	<u> </u>	STM32U5 ultra-low-power MCU series								
Ext. Flash From 0 to 384 (MBit)	384	Boards I	.ist: 3 items					📩 Export		
Ext. EEPROM From 0 to 24 (kBytes)			Overview	Commercial Part No 🎈	Туре	X Marketing Status	Unit Price (US\$)	× Mounted Device		
0 Ext. RAM From 0 to 16 (MBit) 0	24	☆		NUCLEO-L476RG	Nucleo-64	Active	14.0	STM32L476RGT3		
FEATURES	~	☆		STM32L476G-DISCO	Discovery Kit	Obsolete	NA	STM32L478VGT3		
Liser Button	>									
Camera	>			STM321 476G-EV/AI	Evaluation Roard	Active	289.0	STM901 4787079		
CAN	>	ਸ		OTMOZE#/00-EVAL	Evaluation Board	Active	203.0	51W32L4702G13		
Connector	>									
Memory Card	>									

Rysunek 10: Okno z wyborem zestawu ewaluacyjnego NUCLEO-L476RG

Po kliknięciu przycisku *Start Project* zostaniemy zapytani, czy chcemy zainicjalizować peryferia domyślnymi ustawieniami. Wybór przycisku Tak *Yes* spowoduje, że peryferia takie, jak zegar *LSI*, czy USART zostaną skonfigurowane domyślnie. Wybranie opcji Nie *No* nie spowoduje przeprowadzenia domyślnej konfiguracji. W obu przypadkach wyjścia dla diody LED oraz dla przycisku będą odpowiednio skonfigurowane. Po tym etapie powinien pojawić się czysty projekt z wstępnie skonfigurowanymi wyjściami mikrokontrolera, jak na rysunku 11.



Rysunek 11: Okno przedstawiające początkową konfigurację mikrokontrolera dla płyty ewaluacyjnej NUCLEO-L476RG

Piny podświetlone na kolor zielony (PC13 i PA5) oznaczają, że te porty mikrokontrolera zostały skonfigurowane poprawnie. Piny podświetlone na kolor pomarańczowy oznaczają, że funkcjonalność portu została wybrana lecz nie został on jeszcze w pełni skonfigurowany, jak np. PA2 oraz PA3, które wykorzystywane są do komunikacji za pomocą portu szeregowego, jednakże sam port szeregowy (USART) nie został jeszcze skonfigurowany (chyba, że wybrano automatyczną inicjalizację peryferiów domyślnymi ustawieniami).

Uwaga! Warto zwrócić uwagę na nazwy poszczególnych pinów (ich etykiety), jakie są widoczne w widoku *Pinout*. Przykładowo dla pinu PA5, do którego podłączona jest dioda LED została ustawiona etykieta *LD2 [green Led]*. Podczas automatycznego generowania kodu opisanego w rozdziale 4.2 zostaną wykorzystane etykiety, które zostały nadane przez użytkownika do stworzenia przyjaznych definicji, którymi można się później posługiwać. Definicje te znajdują się zwyczajowo w pliku *Inc/main.h.* Dla wyżej wspomnianego pinu PA5 zostaną utworzone poniższe definicje:

#define LD2_Pin GPIO_PIN_5 #define LD2_GPIO_Port GPIOA

Mogą one później zostać wykorzystane do wywołań funkcji z biblioteki HAL, jak np. przełączanie stanu cyfrowego wyjścia. Wówczas programista nie musi pamiętać, do którego portu i numeru pinu przynależy dana nóżka mikrokontrolera, a jedynie wystarczy wiedza na temat etykiety, jaka została do niej przypisana. Jak widać na przykładzie napis znajdujący się w nawiasach kwadratowych jest pomijany w czasie generacji definicji.

Do poprawnego działania oprogramowanie STM32CubeMX wymaga połączenia z Internetem. Jest to spowodowane tym, że pobierane są aktualizacje produktów dostępnych w aplikacji. Jeśli uruchamianie narzędzia trwa zbyt długo można wyłączyć sprawdzanie aktualizacji w menu programu. Wystarczy wybrać $Help \rightarrow Updater\ settings\ ...,$ a następnie w zakładce $Updater\ Settings\ zaznaczyć\ opcję\ Manual\ checks\ oraz\ No\ Auto-Refresh\ at\ Application\ start.$ Zostało to zaprezentowane na Rys. 12.

Updater Settings	×
Updater Settings Connection Parameters	
Firmware Repository	
Repository Folder	
C:/Users/Wojciech Domski/STM32Cube/Repository/	Browse
Check and Update Settings	
Manual Check	
O Automatic Check Interval between two Checks (days) 5	
Data Auto-Refresh	
No Auto-Refresh at Application start	
O Auto-Refresh Data-only at Application start	
O Auto-Refresh Data and Docs at Application start	
Interval between two data-refreshs (days) 3	
ОК	Cancel

Rysunek 12: Wyłaczenie automatycznego sprawdzania aktualizacji

4.1.1 Konfiguracja zegara

Ważnym aspektem pracy mikrokontrolera jest konfiguracja częstotliwości pracy zegara. W ramach tego ćwiczenia należy zapoznać się z informacjami, jakie są dostępne w zakładce konfiguracji zegara (Clock Configuration) (Rys. 13).



Rysunek 13: Zakładka z konfiguracją zegara dla całego układu

Z rysunku 13 można wywnioskować, jaka jest częstotliwość pracy rdzenia oraz jakie jest jego źródło – zidentyfikuj je i prześledź ścieżkę generowania zegara. Czy wykorzystywany jest do tego jakiś układ, jeśli tak, to jaka jest jego rola.

4.2 Automatyczne generowanie kodu

STM32CubeMX nie tylko pozwala na łatwą konfigurację peryferiów mikrokontrolera, ale przede wszystkim jest narzędziem do automatycznego generowania kodu. Przed przystąpieniem do generowania kodu z projektu należy upewnić się, że zostałą zainstalowana biblioteka sterowników HAL dla rodziny mikrokontrolerów STM32L4. W tym celu należy wybrać z menu $Help \rightarrow Manage \ embedded \ software \ packages$ oraz upewnić się, że odpowiedni pakiet został zainstalowany (będzie posiadał ikonę małego zielonego kwadratu). Na potrzeby laboratoriów nie jest wymagana instalacja najnowszej wersji biblioteki, jeśli jest taka dostępna (Rys. 14).

Embedded Software Packages Manager STH32Cube MCU Packages and embedded software packs releases Releases Information was last refreshed less than one hour ago. STM32Cube MCU Packages Description Inst STM32Cube MCU Package for STM32L4 Series STM32Cube MCU Package for STM32L4 Series (Size : 1790 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) Details STM32Cube L4 Firmware Package V1.11.0 / 22-December-2017 Main Changes			>
STH32Cube MCU Packages and embedded software packs releases Releases Information was last refreshed less than one hour ago. STM32Cube MCU Packages Description Ins ▼ STM32L4 STM32Cube MCU Package for STM32L4 Series STM32Cube MCU Package for STM32L4 Series (Size : 1790 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) Details STM32Cube L4 Firmware Package V1.11.0 / 22-December-2017 Main Changes			
Description Ins ▼ 5TH32L4 STM32Cube MCU Package for STM32L4 Series □ STM32Cube MCU Package for STM32L4 Series (Size : 1790 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) □ STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) □ STM32Cube L4 Firmware Package V1.11.0 / 22-December-2017 Main Changes		+	-
V STM32L4 STM32Cube MCU Package for STM32L4 Series STM32Cube MCU Package for STM32L4 Series (Size : 1790 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) Details STM32CubeL4 Firmware Package V1.11.0 / 22-December-2017 Main Changes	L-II- d Mi	Augusta Manaian	
STM32Cube MCU Package for STM32L4 Series STM32Cube MCU Package for STM32L4 Series (Size : 1790 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) Details STM32CubeL4 Firmware Package V1.11.0 / 22-December-2017. Main Changes		Available version	^
STM32Cube MCU Package for STM32L4 Series (Size : 1790 MB) STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) Details STM32CubeL4 Firmware Package V1.11.0 / 22-December-2017 Main Changes	1.11.0	1.11.0	
STM32Cube MCU Package for STM32L4 Series (Size : 692 MB) STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) tetails STM32CubeL4 Firmware Package V1.11.0 / 22-December-2017 Jain Changes		1.10.0	
STM32Cube MCU Package for STM32L4 Series (Size : 1086.0 MB) etails STM32CubeL4 Firmware Package V1.11.0 / 22-December-2017 Jain Changes		1.9.0	1
etails <u>TM32CubeL4 Firmware Package V1.11.0 / 22-December-2017</u> <u>Iain Changes</u>		1.8.1	
TM32CubeL4 Firmware Package V1.11.0 / 22-December-2017 Jain Changes			
 Maintenance release of HAL and Low Layer drivers(new HAL CAN driver APIs). Reduce package size by gathering all media files used by Demonstrations under Utiliti Add PWR examples with external SMPS for NUCLEO-L4R5ZI-P board. Rename project using board product names. 	ies\Media.		<

Rysunek 14: Zainstalowane biblioteki

Aby skonfigurować sposób w jaki kod zostanie wygenerowany należy użyć kombinacji kombinacji klawiszy Alt+P lub wybrać *Project* \rightarrow *Settings*. W nowym oknie należy wpisać potrzebne dane 15.

Pinout & Configuration	on Clock	Configuration	Project Manager		Tools
	Project Settings Project Name Project Location			Browse	
	Application Structure Toolchain Folder Location Toolchain / IDE	Advanced	Generate Under Root	UD not generate the main()	
	/ Linker Settings Minimum Heap Size Minimum Stack Size	0x200 0x400			
	Thread-safe Settings Cortex-M4NS Enable multi-threaded support Thread-safe Locking Strategy	Default - Mapping suitable strategy de	vending on RTOS selection.		
	Mcu and Firmware Package Mcu Reference Firmware Package Name and Version I Use Default Firmware Location Firmware Relative Path	STM32L476RGTx STM32Cube FW_L4 V1.18.0 C/Users/Wolpiech/STM32Cube/Repo		Browse	

Rysunek 15: Konfiguracja projektu – ustawienia ogólne

Konfigurację należy rozpocząć od ustawienia zestawu narzędzi, IDE (Toolchain / IDE). Należy wybrać STM32CubeIDE. Dodatkowo opcja generowania projektu w głównym drzewie katalogu (Generate Under Root) powinna być zaznaczona. Po wyborze IDE ustalamy nazwę projektu (Project Name), w tym przypadku jest to lab01 oraz wybieramy lokalizację. Kolejnym etapem jest ustalenie, w jaki sposób będzie generowany sam kod. Przykładowa konfiguracja została pokazana na rysunku 16.

Pinout & Configuration		Clock Configuration	Project Manager	Tools					
	STM32Cube M0	CU packages and embedded software packs							
	Copy all used libraries into the project folder								
Project	Copy only the	e necessary library files							
	O Add necessa	rry library files as reference in the toolchain project configuratio	n file						
	Generated files								
Code Generator	Generated files								
	Generate pe	ripheral initialization as a pair of ".c/.h' files per peripheral							
	Backup prev	ackup previously generated files when re-generating							
	Keep User C	eep User Code when re-generating							
	L Delete previo	usly generated files when not re-generated							
Advanced Settings									
	HAL Settings								
	Set all free p	ins as analog (to optimize the power consumption)							
	Enable Full A	ssert							
	User Actions								
	Before Code G	eneration		Browse					
	After Code Gen	eration		Browse					
	Template Softing								
	Select a templat	*-	Ings						
	o or o o to o to to to to più	a to Ballarate and an and a contraction of the							

Rysunek 16: Konfiguracja projektu – ustawienia generowania plików

Szczególną uwagę należy zwrócić na grupę generowanych plików (Generated files). Znajdują się tam następujące opcje:

- 1. Generate peripheral initialization as a pair of '.c/.h' files per peripheral
- 2. Backup previously generated files when re-generating
- 3. Keep User Code when re-generating
- 4. Delete previously generated files when not re-gnerated

Pierwsza z nich powoduje, że tworzona jest para plików (kod źródłowy i nagłówek) dla każdego z peryferium. Opcja ta powinna być wybrana. Jest ona szczególnie przydatna podczas pracy zespołowej ponieważ pozwala na fragmentację kodu i ułatwia jego utrzymanie oraz zarządzanie. Drugie pole dotyczy tworzenia kopii zapasowej plików, które będą powtórnie generowane. Wówczas tworzony jest katalog BACKUP do wnętrza, którego przenoszone są pliki, do których nazwy dodawany jest człon ".old". Owy katalog jest tworzony zarówno dla kartotek Src oraz Inc, w których przechowywane są pliki źródłowe projektu oraz pliki nagłówkowe, odpowiednio. Wybór tego pola nie jest wymagany. Trzecia opcja pozwala na zachowanie modyfikacji kodu, jakie wprowadził już programista do kodu programu. Zachowywane są one jedynie, gdy zostały one zapisane wewnątrz odpowiednich znaczników w postaci komentarzy (jest to jedynie przykład, gdyż liczba takich znaczników jest większa):

```
/* USER CODE BEGIN 2 */
//tutaj powinien znaleźć się kod programisty
/* USER CODE END 2 */
```

Ostatnia z pól dotyczy usuwania plików, które w przypadku regeneracji nie ulegną zmianie. Owa opcja powinna być odznaczona, w przypadku, gdy jest aktywna może prowadzić do nieoczekiwanych zachowań podczas kompilacji wynikających z braku plików.

Po zaakceptowaniu ustawień przystępujemy do generowania kodu. Podobnie, jak poprzednim razem można posłużyć się skrótem klawiszowym Ctrl+Shift+G lub wybrać $Project \rightarrow Generate \ Code$. Operacja ta nie powinna trwać dłużej niż kilkanaście sekund w zależności od konfiguracji sprzętowej stacji roboczej. W efekcie powinien się pojawić komunikat o pomyślnym wygenerowaniu kodu źródłowego. Komunikat ten należy zamknąć wybierając przycisk Zamknij (*Close*). Zawartość katalogu i jego struktura po operacji została pokazana na rysunku 17.



Rysunek 17: Struktura i zawartość drzewa dla przykładowego projektu

5 GPIO – cyfrowe wejścia/wyjścia ogólnego przeznaczenia

Jednym z podstawowych peryferiów mikrokontrolera są cyfrowe wejścia/wyjścia popularnie nazywane GPIO (*General Purpose Input Output*). Zazwyczaj wykorzystywane są one do sterowania zewnętrznymi układami, które nie wymagają wyrafinowanych szeregowych protokołów komunikacyjnych. Każdy pin może działać w jednym z poniższych trybów:

- wejście pływające,
- wejście podciągnięte do góry $(pull-\!up),$
- wejście ściągnięte w dół (pull-down),
- wejście analogowe,
- wyjście typu otwarty dren (*open-drain*) z możliwości podciągnięcia do góry (*pull-up*) lub ściągnięcia na dół (*pull-down*),
- wyjście typu *push–pull* z możliwości podciągnięcia do góry (*pull–up*) lub ściągnięcia na dół (*pull–down*),
- funkcja alternatywna typu otwarty dren z możliwości podciągnięcia do góry (pull-up) lub ściągnięcia na dół (pull-down),
- funkcja alternatywna typu *push-pull* z możliwości podciągnięcia do góry (*pull-up*) lub ściągnięcia na dół (*pull-down*).

Na szczególną uwagę zasługuje tryb analogowy. Należ tutaj podkreślić, że nie wszystkie wejścia GPIO mogą być podłączone do przetwornika analogowo–cyfrowego ADC. Jednakże konfiguracja danego pinu mikrokontrolera w trybie analogowym pozwala na ograniczenie pasożytniczych prądów. Osiąga się to poprzez wyłączenie:

- buforów wyjściowych,
- bramki Schitta,
- rezystorów podciągających/ściągających (pull-up/pull-down).



Rysunek 18: Wybór trybu pracy pinu mikrokontrolera

Na rysunku 19 została przedstawiona przykładowa konfiguracja siedmiu wybranych pinów mikrokontrolera (PA5, PA8, PA9, PA10, PA11, PC9, PC13). Aplikacja STM32CubeMX pozwala na łatwą i przejrzystą

Nazwa pinu	Funkcja
PA5	Wyjście (dioda LED)
PA8	Wejście przerwania zewnętrznego
PA9	Wyjście zdarzenia
PA10	Tryb analogowy
PA11	Wejście
PC9	Wyjście
PC13	Wejście przerwania zewnętrznego (przycisk)

Tabela 1: Przykładowa konfiguracja pinów mikrokontrolera

konfigurację peryferiów układu. Konfiguracja wybranego peryferium odbywa się na jeden z dwóch sposobów. Pierwszym z nich jest wybranie funkcji pinu mikrokontrolera przez kliknięcie na niego lewym przyciskiem myszy i wybraniu zadanej funkcjonalności z menu kontekstowego (rys. 18). Taka konfiguracja ma sens w przypadku, gdy będzie wykorzystywany pojedynczy pin, tak jak to ma miejsce w wypadku konfiguracji GPIO. Druga ze ścieżek wymaga w pierwszej kolejności wybrania odpowiedniego peryferium np. SPI, a wówczas wymagane piny zostaną automatycznie przypisane do domyślnych portów.

Pin Configuration ×									
GPIO Single Ma	oped Signals								
Search Signals Search (Crtl+F	-)						Show onl	y Modified Pins	
Pin Name	Signal on Pin	GPIO output I	GPIO mode	GPIO Pull-up/	Maximum out	Fast Mode	User Label	Modified	
PA5	n/a	Low	Output Push Pull	No pull-up and	Low	n/a	LD2 [green Led]	\checkmark	
PA8	n/a	n/a	External Interr	No pull-up and	n/a	n/a			
PA9	EVENTOUT	n/a	Alternate Funct	No pull-up and	Low	n/a			
PA10	n/a	n/a	Analog mode	No pull-up and	n/a	n/a			
PA11	n/a	n/a	Input mode	No pull-up and	n/a	n/a			
PC9	n/a	Low	Output Push Pull	No pull-up and	Low	n/a			
PC13	n/a	n/a	External Interr	No pull-up and	n/a	n/a	B1 [Blue PushB	\leq	
PC9 Configurati GPIO output lev	on : vel			Low				~	
GPIO mode				Output Pus	h Pull			~	
GPIO Pull-up/Pu	ull-down			No pull-up a	No pull-up and no pull-down				
Maximum outpu	it speed			Low	Low				
User Label									
Group By Pe	Group By Peripherals Apply Ok Cancel								

Rysunek 19: Przykładowa konfiguracja wybranych trybów

Jak można zauważyć na rysunku 19 zostało skonfigurowanych aż siedem piny mikrokontrolera. Dwa z nich dostępne są w podstawowej konfiguracji dla płytki deweloperskiej. Są nimi piny PA5, który pełni funkcję wyjścia, do którego podłączona jest dioda LED oraz PC13, który z kolei wykorzystywany jest jako wejście, do którego podłączony jest przycisk. Pozostałe piny zostały skonfigurowane w taki sposób, aby pokazać pełen wachlarz możliwości układu scalonego. Po kliknięciu w jeden z wybranych pinów GPIO rozwija się menu konfiguracyjne, gdzie możliwe jest wybranie dodatkowych ustawień dla aktywnego portu. Są one dostosowywane kontekstowo w zależności, czy nóżka mikrokontrolera pracuje, jako wejście, wyjście, czy w trybie analogowym.

5.1 GPIO w bibliotece HAL

Do podstawowych funkcji, które pozwalają na ingerowanie w pracę portów GPIO należą [2]:

- GPI0_PinState HAL_GPI0_ReadPin(GPI0_TypeDef * GPI0x, uint16_t GPI0_Pin),
- void HAL_GPIO_WritePin(GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState),
- void HAL_GPI0_TogglePin(GPI0_TypeDef * GPI0x, uint16_t GPI0_Pin),

- void HAL_GPIO_LockPin(GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin),
- void HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin),
- void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin).

HAL_GPIO_ReadPin() pozwala na odczytanie aktualnej wartości rejestru wejściowego przypisanego do danego portu. Funkcja przyjmuje dwa parametry. Pierwszym z nich jest port, którego pin ma zostać odczytany natomiast drugim parametrem jest numer pinu, którego wywołanie dotyczy. Wartością zwracaną jest stan portu, która może przyjąć dwa stany GPIO_PIN_RESET (0), bądź GPIO_PIN_SET (1).

Modyfikację stanu wyjścia portu można wykonać poprzez wywołanie funkcji HAL_GPIO_WritePin(). Przyjmuje ona trzy parametry, gdzie pierwsze dwa są identyczne, jak w przypadku procedury odczytującej, natomiast trzeci z parametrów określa żądany stan cyfrowego wyjścia.

W przypadku, gdy wymagane jest przełączenie wyjścia na przeciwny stan można wykorzystać funkcję HAL_GPI0_TogglePin(). Przełącza ona stan pinu na przeciwny, to jest, z stanu wysokiego na niski i analogicznie z niskiego na wysoki. Parametry wywołania są identyczne, jak w przypadku HAL_GPI0_ReadPin().

Kolejną funkcją, która jest wykorzystywana rzadziej niż poprzednie jest HAL_GPIO_LockPin(). Zadaniem funkcji jest zablokowanie rejestrów należących do danego pinu mikrokontrolera. W ten sposób po wywołaniu operacji zapisu, czy przełączenia stanu na zadanym porcie nie przyniesie ona żadnego efektu. Modyfikacja stanu zablokowanego portu będzie dopiero możliwa pod resecie mikrokontrolera.

Kolejne dwa wywołania HAL_GPIO_EXTI_IRQHandler() oraz HAL_GPIO_EXTI_Callback() dotyczą one obsługi przerwań zewnętrznych, zostaną one opisane szczegółowo później.

6 Zadania do wykonania

6.1 Uzupełnienie programu o kod odpowiedzialny za miganie diodą LED

Biblioteka HAL oferuje szeroką gamę funkcji właściwych dla każdego z peryferium. W przypadku cyfrowych portów wejściowo–wyjściowych ogólnego przeznaczenia (GPIO) można wyróżnić trzy podstawowe funkcje, których prototypy zostały podane poniżej [2].

GPIO_PinState HAL_GPIO_ReadPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin);

void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);

void HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin);

Funkcje służą odpowiednio do: odczytu stanu portu, zapisu stanu portu oraz przełączenia aktywnego stanu portu na przeciwny. Szczegółowy ich opis, a także typy wyliczeniowe, jakie są wykorzystywane w owych funkcjach można znaleźć w [2]. Ponadto, STM32CubeMX dostarcza definicji nazw portów oraz pinów, jakie zostały nadane przez użytkownika w projekcie STM32CubeMX. Mają one postać XXX_Port oraz XXX_Pin, odpowiednio dla portów i dla pinów. Definicje te znajdują się w pliku nagłówkowym main.h, bądź mxconstants.h

Wykorzystując podane wyżej API napisz prosty program, który będzie zapalał, a następnie gasił diodę LED podłączoną do pinu PA5.

Funkcją wprowadzającą aktywne opóźnienie (tzw. busy-wait) jest

1 void HAL_Delay (uint32_t Delay);

Przyjmuje ona jako parametr wartość wyrażoną w milisekundach [ms].

Uwaga! W przypadku, kiedy funkcja HAL_Delay() nie przynosi efektu należy stworzyć własną funkcję o nazwie delay_ms(). Jej prototyp został podany poniżej:

```
1 void delay_ms (uint32_t ms){
2 uint32_t i, delay;
3 delay = ms * 80000;
4 for(i = 0; i < delay;){
5 ++i;
6 }
7 }</pre>
```

Wartość opóźnienia należy dostosować na zasadzie obserwacji. Jaki jest powód, że funkcja HAL_Delay() nie przynosi oczekiwanego efektu?

Uwaga! Numer portu, jak i pinu, do którego podłączona jest dioda LED można odczytać z projektu w programie STM32CubeMX.

6.2 Wgranie i testowanie programu na mikrokontrolerze

Kolejnym etapem po zaimporotwaniu projektu jest jego zbudowanie. Można to zrobić poprzez wybranie polecenia z menu (*Project* \rightarrow *Build All*) lub użyć kombinacji klawiszy **Ctrl**+**B**. Budowa projektu jest czasochłonnym i może zająć kilka minut w zależności od konfiguracji projektu, jaki i od parametrów stacji roboczej.

Aby wgrać oprogramowanie na mikrokontroler należy podłączyć zestaw deweloperski do komputera za pomocą kabla USB-mini USB. Kolejnym krokiem jest uruchomienie aplikacji STM32 ST-Link Utility [6], [7]. Widok aplikacji ST-Link zaraz po uruchomieniu został zaprezentowany na rysunku 20.

STM32 ST-LINK Utility				-		×
File Edit View Target ST-LINK Exte	ernal Loader Help					
🖴 🖥 🖕 🥋 🖉 📓 🔜 -						
Memory display		Device				
Address: 0x08000000 ~ Size: 0x47F4	Data Width: 32 bits 🗸	Device ID Revision ID				
Douiso Momory Diana Eta		Flash size				
Device Memory					LiveUp	odate
period memory				_	_	
Disconnected	Device ID :		Core State : No Memory Loa	ded		

Rysunek 20: Aplikacja STM32 ST-Link Utility

Otwórz plik binarny (*.bin lub *.hex) za pomocą polecenia z menu ($File \rightarrow Open \ file \ldots$). Plik binarny znajduje się wewnątrz katalogu Debug w głównej kartotece projektu. Po otwarciu pliku z programem aplikacja powinna zmienić wygląd na podobny do prezentowanego na rysunku 21.

🖏 STM32 ST-L	INK Utility				- 0	\times
File Edit Vie	w Target !	ST-LINK Ex	ternal Loade	r Help		
🖴 🖥 🖕	Ç 🧳 🖗	, 🧝 🔜				
Memory display					Device	
Address: 0x080	00000 🗸 Size	: 0x47F4	Data Widt	h: 32 bits 🗸	Device ID Revision ID	
Device Memory Fi	le : lab01.bin				Flash size	
[lab01.bin], File size:	6440 Bytes	1				
Address	0	4	8	С	ASCII	^
0x00000000	20018000	08001811	08001861	08001861	.€aa	
0x00000010	08001861	08001861	08001861	00000000	aaa	
0x0000020	00000000	0000000	00000000	08001861	a	
0x0000030	08001861	0000000	08001861	08001795	aa•	
0x00000040	08001861	08001861	08001861	08001861	aaa	
0x00000050	08001861	08001861	08001861	08001861	aaaa	
0×00000060	08001861	08001861	08001861	08001861	a a a a	>
15:18:59 : [lab01.bit	n] opened succes	sfully.				
Disconnected			Device ID :		Core State : No Memory Grid Selected	

Rysunek 21: ST-Link Utility z otwartym kodem programu

Aby wgrać program należy wykonać trzy operacje:

- 1. Połączyć się z zestawem edukacyjnym wybierając z menu opcję Połąc
z ($Target \rightarrow Connect$), Rys. 22.
- 2. Wgrać program za pomocą polecenia Programowania i weryfikacji (Target \rightarrow Program & Verify ...), Rys. 23.
- 3. Odłączyć zestaw ewaluacyjny wybierając polecenie Rozłącz ($Target \rightarrow Disconnect$), Rys. 24.

Uwaga! W przypadku problemów z wgraniem programu na płytkę deweloperską należy upewnić się, czy system Windows zainstalował do niej sterowniki. Instalacja oprogramowania może potrwać około minuty. Postęp instalacji można śledzić w zasobniku systemowym.

🖷 STM32 ST-LI	NK Utility							-		×
File Edit View	w Target S	ST-LINK Ex	ternal Loade	r Help						
🖴 🖥 🖕 🖗	🤹 🥔 🗭	. 👰 🔜								
Memory display Address: 0x080	00000 🗸 Size	: 0x1988	Data Widt	h: 32 bits ∽		Device Device ID Revision ID Flash size	STM32L4xx 0x415 Rev 1 1MBytes			
Target memory, Addr	ress range: [0x08	le : lab01.bin 3000000 0x0800	1988]						Livel	Jpdate
Address	0	4	8	С	ASCI					^
0x08000000	20018000	0800022D	0800027D	0800027D	.€.	} }				
0x08000010	0800027D	0800027D	0800027D	00000000	}}	}				
0x08000020	00000000	00000000	00000000	0800027D		}				
0x08000030	0800027D	0000000	0800027D	08001045	}	}E				
0x08000040	0800027D	0800027D	0800027D	0800027D	}}	}}				
0x08000050	0800027D	0800027D	0800027D	0800027D	}}	}}				
0v08000060 <	08000270	0800027D	0800027D	08000270	1 1	3 3				>
22:59:59 : ST-LINK S 22:59:59 : ST-LINK F 22:59:59 : ST-LINK F 22:59:59 : SWD Free 22:59:59 : SWD Free 22:59:59 : Debug in 22:59:59 : Debug in 22:59:59 : Debug in 22:59:59 : Device fla 22:59:59 : Device fla	IN: 066DFF5049 Firmware version Id via SWD, quency = 1,8 MH; on mode : Normal Low Power mode :0x415 ssh Size : 1MByte mily :STM32L4xx	5570726724281: : V2J27M15 z. : enabled. s	}							
Debug in Low Power	mode enabled.		Device ID:0x41	5			Core State : Live Up	date Disabled		

Rysunek 22: Widok aplikacji ST-Link po udanej próbie połączenia z programatorem

👼 STM32 ST-LI	NK Utility						_		\times
File Edit View	w Target S	ST-LINK Exte	ernal Loader Help						
🖴 🖥 🖕 🖗	Ç 🧳 🖉	. 🤌 🔜							
Memory display				Device	STM32L4xx				
Address Du020		01000	Data Width, 22 kits	Device ID	0x415				
Address: 0x0800	512e	: 0x1966	Data Width: 32 bits V	Revision ID	Rev 1				
et	Larle			Flash size	1MBytes				
Device Memory	e:labu1.bin								
(Iabu1.binj, File size:	6536 Bytes	Download []	lab01 bin 1		~				
Address	0	Download	labo1.bin j		^				<u>^</u>
0x0000000	20018000	Start address :	0x08000000						
0x0000010	080018C1	File path : Verification	D:\lab01\Debug\lab01.bin		Browse				
0x00000020	00000000	٩٧	/erify while programming OVe	rify after programmin	9				
0x0000030	080018C1	Click "Start" to p	program target.						
0x00000040	080018C1								
0x00000050	080018C1	I Heset after	programming						
0×0000060	08001801		otait Cario	e					~
<									>
23:00:20 : ST-LINK S	N : 066DFF 5049	55707267242813							^
23:00:20 : ST-LINK F	irmware version	: V2J27M15							
23:00:20 : Connecte 23:00:20 : SWD Free	uencv = 1.8 MH	z.							
23:00:20 : Connectio	n mode : Normal	L							
23:00:20 : Debug in I 23:00:20 : Device ID	Low Power mode	enabled.							
23:00:20 : Device fla	ish Size : 1MByte	s							
23:00:20 : Device fai	mily:STM32L4xx								~
Debug in Low Power r	mode enabled		Device ID:0v415		Core State : No	Momory Grid	Solorta	4	-

Rysunek 23: Widok aplikacji po wczytaniu pliku oraz wybraniu operacji zapisu programu z weryfikacją

🖷 STM32 ST-LI	NK Utility							-		×
File Edit Viev	w Target S	ST-LINK Ext	ternal Loade	r Help						
🖴 🖥 🖕 🖗	Ç 🥠 🖗	🧼 駴								
Memory display						Device	STM32L4xx			
Address: 0x08000000 V Size: 0x1988 Data Width: 32 bits V Device ID 0x415										
Revision ID Rev 1										
Device Memory @ 0:	x08000000 : Fil	le : lab01.bin				T IDDIT DIEC	1. by ccs		LiveU	pdate
Target memory, Addr	ess range: [0x08	8000000 0x0800	1988]							
Address	0	4	8	С	ASCII					^
0x08000000	20018000	08001871	080018C1	080018C1	.€. (q Á Á				
0x08000010	080018C1	080018C1	080018C1	00000000	Á	ÁÁ	••			
0x08000020	00000000	00000000	00000000	080018C1		Á				
0x08000030	080018C1	00000000	080018C1	080017F5	Á	Áő.				
0x08000040	080018C1	080018C1	080018C1	080018C1	Á	ÁÁÁ				
0x08000050	080018C1	080018C1	080018C1	080018C1	Á	ÁÁÁ				
0v0800060	08001801	08001801	08001801	08001801	Δ	<u>ά ά</u>				× .
23.00.20 - 31 - 114K 1		. V232/1113								-
23:00:20 : Connecte 23:00:20 : SWD Free	d via SWD. Juency = 1.8 MH;	z,								~
23:00:20 : Connectio	on mode : Normal	, enabled								
23:00:20 : Device ID	:0x415	chabica.								
23:00:20 : Device fla 23:00:20 : Device fa	ish Size:1MByte: mily:STM32L4xx	s								
23:00:36 : Memory p	rogrammed in Os	and 500ms.								
25.00.00 : Verificado										~
Debug in Low Power r	mode enabled.		Device ID:0x41	.5			Core State : Live Upda	ate Disabled		_

Rysunek 24: Widok aplikacji po pomyślnym wgraniu nowego oprogramowania na mikrokontroler

Po pomyślnym wgraniu programu dioda użytkownika powinna migać.

Uwaga! W przypadku, gdy operacja połączenia się nie powiodła to najprawdopodobniej układ nie został połączony z komputerem za pomocą dostarczonego przewodu USB lub zwory na złączu CN2 nie są zwarte [5]. Innym powodem może być brak zainstalowanych sterowników w systemie. W takim przypadku należy zweryfikować, czy sterowniki te są instalowane, a w tym czasie nie wolno odłączać urządzenia (płytki) od komputera.

6.3 Modyfikacja projektu i programu w celu obsługi przycisku pracującego w roli przełącznika dla diody LED

Jako modyfikację programu należy wykorzystać przycisk znajdujący się na płycie do zmiany stanu diody (włączona/wyłączona). Innymi słowy, pojedyncze naciśnięcie przycisku powinno zmieniać stan diody. Zmiana stanu diody powinna nastąpić tylko podczas wciskania przycisku, a nie jego puszczania. Dodatkowo, w przypadku, gdy przycisk jest trzymany cały czas stan diody nie powinien się zmieniać. Ponadto, w programie nie można wykorzystać opóźnienia większego niż 10 milisekund.

Zaimplementuj procedurę do niwelowania zjawiska drgań styków (tzw. debouncing). Do tego celu można wykorzystać przerwania, ale nie jest to wymagane. Procedura powinna być tak napisana, aby nie zatrzymywać wykonywania programu niepotrzebnie. Do tego celu można wykorzystać mały jednostanowy automat.

Czy modyfikacja projektu w programie STM32CubeMX jest konieczna? Jeśli tak to co należy w nim zmienić. Powtórz kroki związane z automatycznym generowaniem kodu jeśli zajdzie taka potrzeba.

Przydatną funkcją może okazać się

1 uint32_t HAL_GetTick (void);

zwraca ona liczbę tików zegara SysTick. Można przyjąć, że jeden tick to 1 milisekunda. Funkcję tą warto wykorzystać podczas implementowania debouncingu.

Uwaga! Numer portu, jak i pinu, do którego podłączony jest przycisk można odczytać z projektu w programie STM32CubeMX.

6.4 Wgranie i testowanie zmodyfikowanego programu na mikrokontrolerze

Postępując podobnie, jak to zostało opisane w sekcji 6.2 nanieś modyfikacje, przebuduj program (Ctrl+B) oraz wgraj go na płytkę za pomocą narzędzia ST-Link Utility.

6.5 Uporządkowanie stanowiska

Odłóż płytkę i kabel na miejsce. Usuń projekt z Atollic TrueSTUDIO. Można to zrobić przez kliknięcie prawym przyciskiem myszki na projekt i wybranie opcji Usuń (Delete) z menu kontekstowego.

7 Podsumowanie

Poprzez wykonanie ćwiczeń z tej instrukcji zaznajomiłeś(aś) się z podstawowymi narzędziami do generowania i rozwijania oprogramowania dla mikrokontrolera serii STM32L4. Przedstawiono tutaj narzędzie STM32CubeMX, które pozwala na wstępne skonfigurowanie peryferiów układu oraz dobór taktowania dla poszczególnych zespołów funkcjonalnych mikrokontrolera. Ponadto, STM32CubeMX pozwala na automatyczne generowanie kodu, który może być następnie modyfikowany. Tak wygenerowane pliki źródłowe wraz z całym projektem mogą być bez przeszkód zaimportowane do darmowego środowiska programistycznego Atollic TrueSTUDIO for STM32, w którym to wykonuje się edycję kodu źródłowego oraz kompilację. W końcu możliwe jest wgranie pliku binarnego na mikrokontroler umieszczony na płytce deweloperskiej za pośrednictwem STM32 ST-LINK Utility.

Literatura

- ST. STM32L476xx, Ultra-low-power ARM® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB Flash, 128 KB SRAM, USB OTG FS, LCD, analog, audio., Grudzień, 2015.
- [2] ST. Description of STM32L4 HAL and Low-layer drivers., Luty, 2016.
- [3] ST. STM32 configuration and initialization C code generation., Kwiecień, 2016.
- [4] ST. STM32 Nucleo-64 board., Listopad, 2016.
- [5] ST. STM32 Nucleo-64 board, User manual., Listopad, 2016.
- [6] ST. STM32 ST-LINK Utility for STM32 MCUs., Listopad, 2016.
- [7] ST. STM32 ST-LINK utility software description, User manual., Sierpień, 2016.
- [8] ST. STM32CubeMX for STM32 configuration and initialization C code generation, User manual., Marzec, 2017.
- [9] ST. STM32L4x5 and STM32L4x6 advanced ARM@-based 32-bit MCUs, Reference Manual., Marzec, 2017.