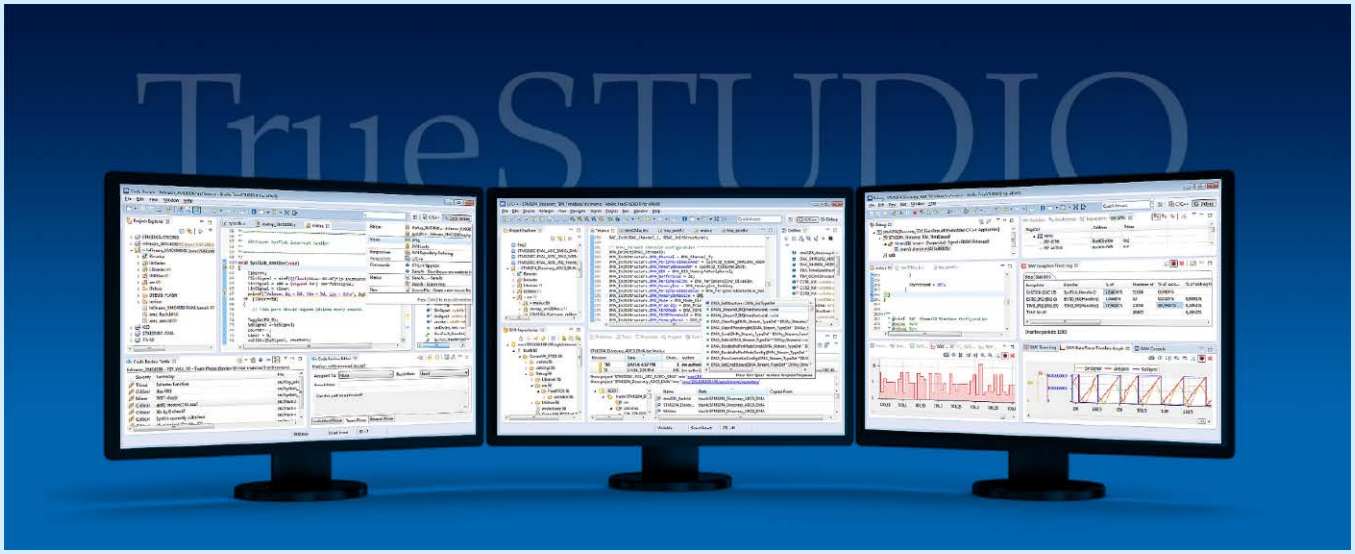


Atollic TRUEStudio 9.x

bezpłatny zestaw narzędzi dla programistów STM32

Każdy znaczący gracz na rynku mikrokontrolerów ma swojej ofercie środowisko projektowe IDE – Integrated Development Environment. Są to rozbudowane pakiety programowe pozwalające na tworzenie projektów z zaawansowaną edycją plików źródłowych ich kompilacją w języku C lub opcjonalnie C++, sterowaniem programatorem/debugerem i możliwością sprzętowego debugowania programu w docelowym środowisku. Oprócz tego dostarczane są zaawansowane firmowe biblioteki obsługujące układy peryferyjne mikrokontrolerów ale też współpracujące z układami zewnętrznymi typu moduły sterowników graficznych, moduły łączności Wi-Fi, Bluetooth, GPS itp. Standardem stają się programowe konfiguratory układów peryferyjnych z przyjaznym interfejsem użytkownika. Jeszcze do nie dawna takie narzędzia programistyczne dla mikrokontrolerów STM32 były oferowane przez firmy zewnętrzne i ich pełne wersje kosztowały dość sporo. Teraz jest do dyspozycji całkowicie bezpłatny doskonały pakiet Atollic True Studio for STM32.

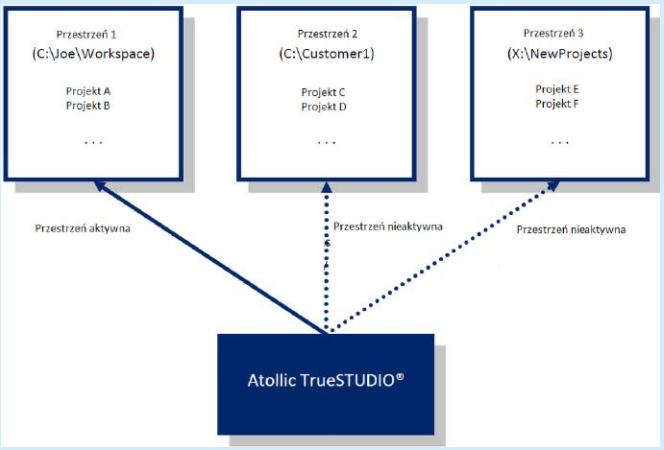
IDE Atollic TRUEStudio wykorzystuje otwarty standard framework Eclipse , kompilator GCC (język C/C++), oraz debuger oparty o standard GDB. Eclipse IDE był początkowo przeznaczony dla języka Java, ale stosunkowo łatwo go przystosować do pracy z innymi językami na przykład C/C++, PHP itp. Cechą charakterystyczną jest możliwość uzupełniania IDE tworzonego w oparciu o framework Eclipse o własne wtyczki narzędziowe.

Konsekwencją stosowania Eclipse jest organizowanie pracy w oparciu o przestrzeń roboczą (workspace) i projekty (projects):

- Przestrzeń robocza może zawierać jeden lub wiele projektów.
- Projekty zawierają wszystkie niezbędne pliki potrzebne do pracy.
- Każda przestrzeń robocza może pomieścić wiele projektów w dowolnej lokalizacji komputera.
- Użytkownik może swobodnie wybierać przestrzenie robocze, ale w danym momencie aktywna może być tylko jedna.
- Wybieranie przestrzeni roboczych jest wygodnym i szybkim mechanizmem zarządzania aktywnym projektem, lub zestawem projektów.

W praktyce przestrzenie robocze i projekty są zestawami hierarchicznych katalogów: katalog przestrzeni roboczej zawiera katalogi projektów, a projekty zawierają pliki – **rysunek 1**.

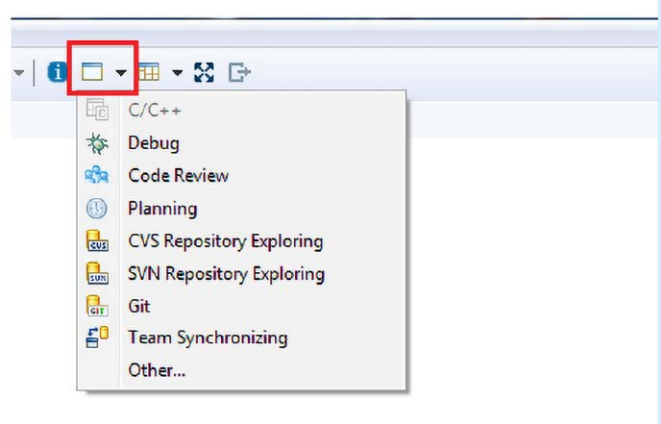
Atollic TrueStudio ma bardzo wiele możliwości pracy nad projektem, od edycji plików źródłowych po ich kompilację i debugowanie.



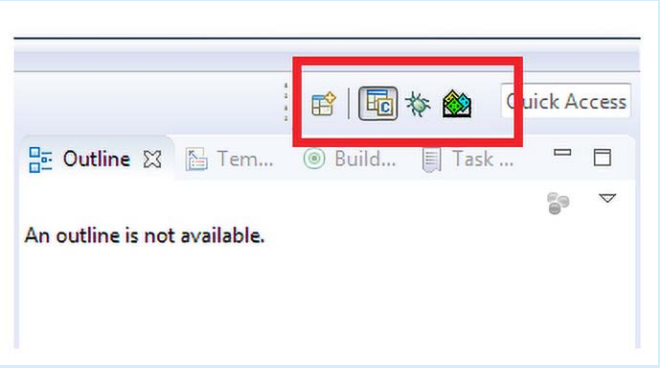
Rysunek 1. Organizacja pracy – przestrzenie robocze i projekty

Każde z wykonywanych zadań jest powiązane z określoną liczbą okien, które muszą się znaleźć na ekranie. Jednoczesne otwarcie wielu okien powoduje, że w dużej ilości informacji wyświetlanych na ekranie łatwo się pogubić i znacznie spada przez to komfort pracy. Żeby temu zaradzić można wyświetlane informacje pogrupować tak, żeby odpowiadały wykonywaniu określonego zadania. Takie grupy nazwano perspektywami (Perspectives). W TrueStudio zdefiniowano

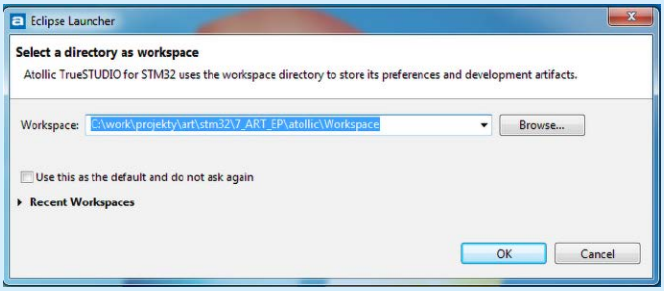
Poprzednie części kursu i dodatkowe materiały dostępne są na stronie www.media.avt.pl



Rysunek 2. Wybór aktywnej perspektywy z paska narzędziowego

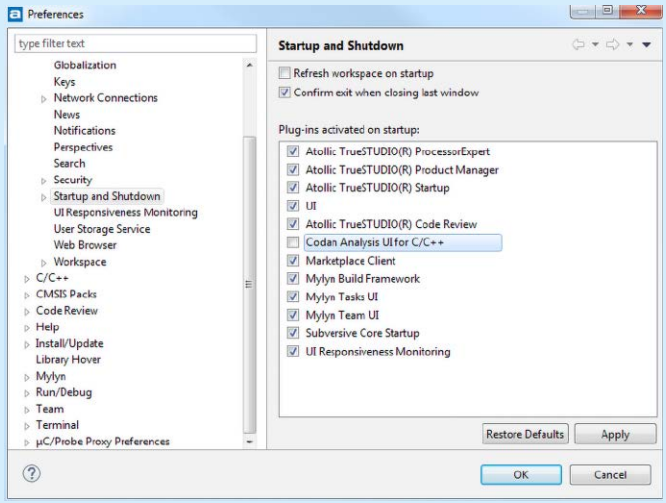


Rysunek 3. Alternatywny wybór aktywnej perspektywy



Rysunek 4. Okno z wyborem katalogu przestrzeni roboczej

Rysunek 5. Okno Information Center



Rysunek 6. Okno personalizacji ładowanych wtyczek

kilka predefiniowanych perspektyw na przykład edytowanie kodu źródłowego C/C++, czy jego debugowanie. Prawidłowo zdefiniowana perspektywa powinna być użyta do wykonania jednego zadania i nie zawiera elementów z innych perspektyw. Zakłada się też, że główną uruchamianą zawsze na początku programu jest ta przeznaczona do edytowania plików źródłowych C/C++, bo przy edycji programista spędza najwięcej czasu.

Perspektywy mona wybierać (przełączać) klikając na przyciski Open Perspective umieszczony w pasku narzędziowym okna programu – **rysunek 2**. Alternatywnie można do tego użyć przycisków umieszczonych w prawym górnym rogu głównego okna programu – **rysunek 3**.

Pierwsze kroki

Program instalacyjny Atollic TRUEStudio for STM32 można pobrać ze strony producenta *https://atollic.com/truestudio/*. Instalacja jest typowa i nie wymaga specjalnego komentarza. Po uruchomieniu programu pojawia się okno wyboru katalogu z przestrzenią roboczą. Można zaakceptować domyślną ścieżkę zaproponowaną przez program, lub wybrać sobie własny katalog – **rysunek 4**.

Przy pierwszym uruchomieniu programu otwiera się okno Information Center. Są tam zebrane w jednym miejscu przydatne informacje (w formie hyperlinków) zarówno dla poczatkujących, jak i dla bardziej zaawansowanych użytkowników programu. W każdej chwili można mieć dostęp do tego okna po kliknięciu na Help → Information Center – **rysunek 5**.

W czasie uruchamiania programu jest ładowanych szereg wtyczek. Jeżeli części z nich nie zamierzamy nigdy używać to warto je wyłączyć. Redukuje to ilość potrzebnej pamięci, ale tez przyspiesza działanie programu. Można to sobie spersonalizować w oknie Windows → Preferences → Startup and Shutdown – **rysunek 6**.

Tworzenie nowego projektu

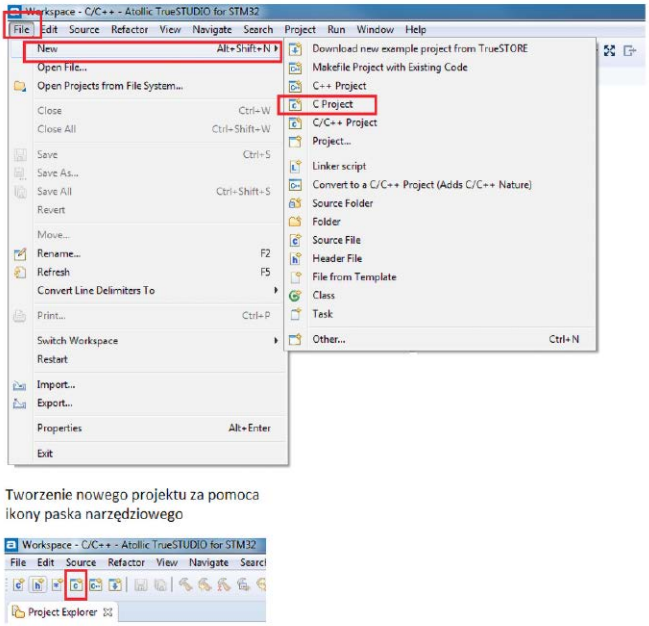
Tworzenie nowego projektu można rozpocząć z menu File → New → C Project lub przez kliknięcie na ikonkę Create New C Project w pasku narzędziowym – zostało to pokazane na **rysunku 7**.

Pierwsze okno konfiguracji projektu zostało pokazane na **rysunku 8**. W pierwszej kolejności musimy nadać nazwę tworzonemu projektowi w oknie Project name. Potem wybieramy zestaw narzędzi. Atollic TRUE Studio oferuje dwa rodzaje zestawów narzędzi programowych: Atollic ARM Toolchains i PC Tollchains. Dla pracy z systemami mikroprocesorowymi wybieramy oczywiście ARM Toolchains. I na koniec pozostał do wyboru typ projektu. Tu na początek wybieramy Embedded C Project.

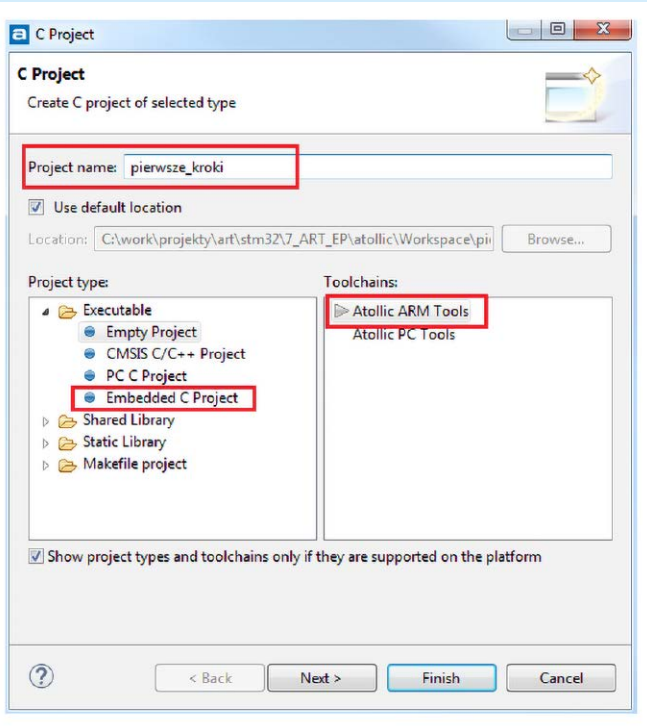
Po kliknięcie na przycisk Next konfigurator projektu przechodzi do kolejnego okna konfiguracji sprzętu (hardware configurator). Z głównej listy wybieramy rodzinę mikrokontrolerów STM32F4.

Potem z rozwijanej listy można wybrać konkretny typ mikrokontrolera rodziny , lub wspierany moduł ewaluacyjny. Ponieważ w czasie testów będziemy używać modułu NUCLEO-F401, to taki został wybrany – **rysunek 9**. Wybranie modułu powoduje automatyczne wypełnienie pól z typem mikrokontrolera (STM32F401RE), miejscem ładowania kodu (pamięć Flash mikrokontrolera), zestaw instrukcji (Thumb2) i dosunięcie danych. Ponieważ mikrokontroler użyty w tym module ma sprzętową implementację arytmetyki zmiennoprzecinkowej to została ona wybrana domyślnie (szybsze obliczenia).

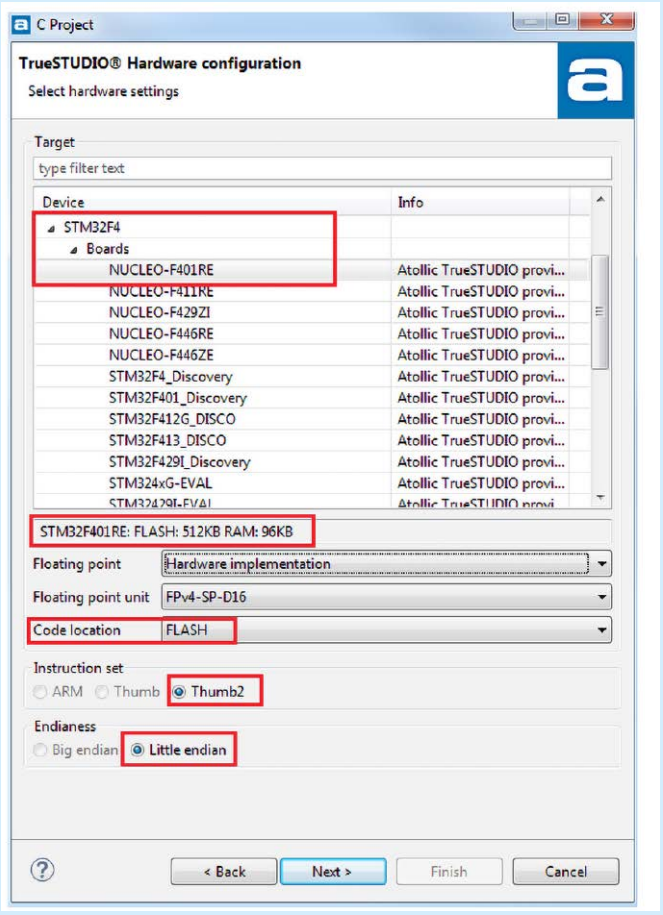
Kolejne okno, pokazane na **rysunku 10**, jest przeznaczone do konfiguracji biblioteki Runtime i ustawień optymalizacji kompilatora. Kiedy zasoby mikrokontrolera są ograniczone wybieramy opcje Newlib-nano i wersje tiny funkcji bibliotecznych printf. Ustawienia optymalizacji powinny zostać domyślne.



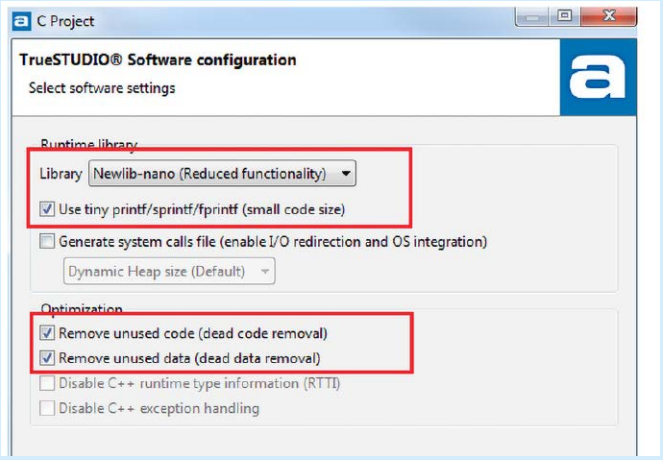
Rysunek 7. Dwie możliwości tworzenia nowego projektu



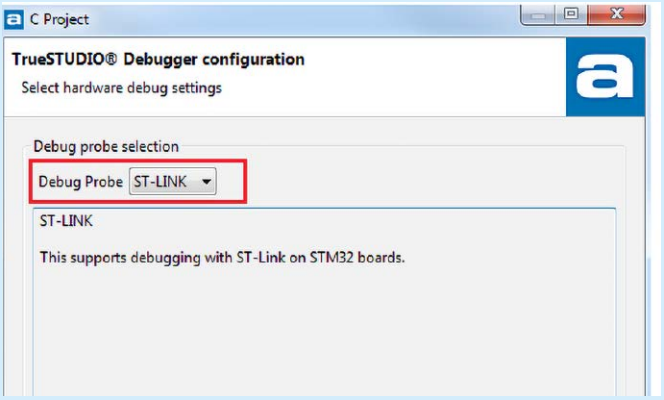
Rysunek 8. Okno konfiguracji projektu



Rysunek 9. Konfiguracja sprzętowa projektu

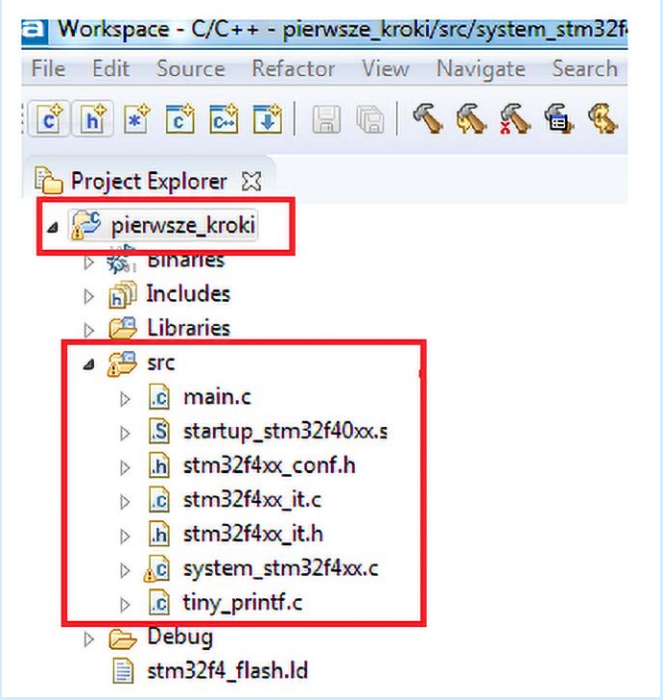


Rysunek 10. Ustawienia konfiguracji biblioteki runtime i optymalizacji kodu

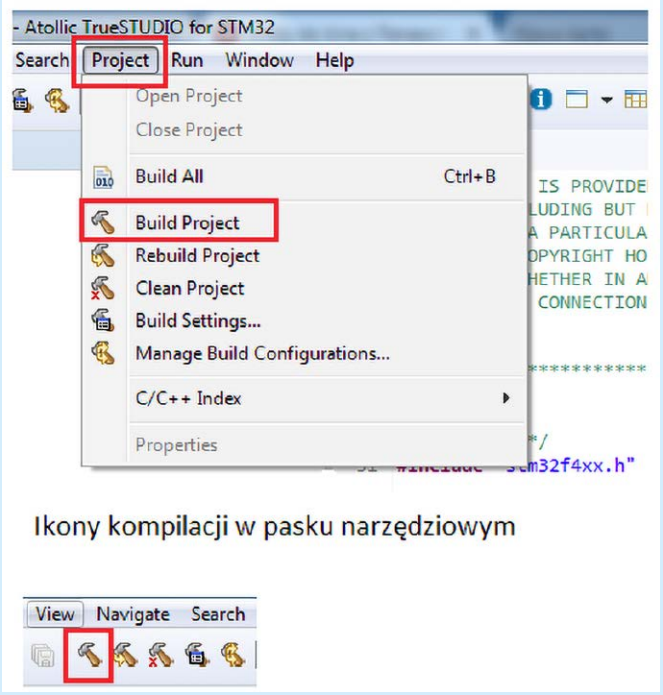


Rysunek 11. Wybór programatora/debuggera

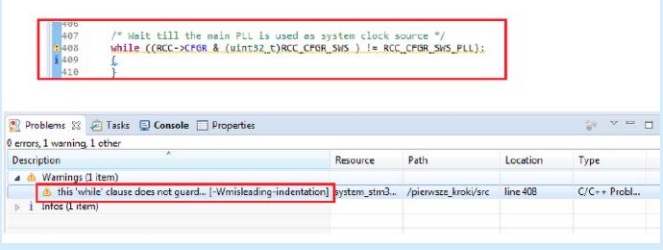
W ostatnim oknie konfiguratora projektu można wybrać typ programatora/debuggera. Dla modułu Nucleo jedyną opcją jest ST-Link (**rysunek 11**).



Rysunek 12. Utworzony szkielet projektu z plikami źródłowymi



Rysunek 13. Kompilacja projektu



Rysunek 14. Ostrzeżenie kompilatora po skompilowaniu szkieletu utworzonego przez kreator projektu

Po kliknięciu na przycisk Finish kreator projektu utworzy szkielet projektu z niezbędnymi plikami tak jak to pokazano na **rysunku 12**. Automatycznie jest też ustawiana perspektywa.

Pliki szkieletu projektu kompilujemy, przy czym przez kompilację należy rozumieć kompilowanie plików przez kompilator języka C, ich łączenie w procesie linkowania i generowanie pliku wykonywalnego w jednej z opcji Debug, lub Release. Kompilacja jest uruchamiana po kliknięciu na ikonę Projekt → Build Project. Ikony kompilacji są również dostępne bezpośrednio na pasku narzędziowym – **rysunek 13**.

Alokacja programów pamięci Flash i pamięci RAM mikrokontrolera jest określana przez plik skryptu linkera stm32f4_flash.ld. Zaawansowani użytkownicy mogą ten plik edytować i zmieniać działanie linkera.

Kompilowanie Build Project kompiluje tylko pliki, które były zmieniane od dostatniej kompilacji, wybranie Rebuild Project powoduje kompilowanie wszystkich plików projektu. Clean Project usuwa wszystkie skompilowane pliki. Opcja Build All powoduje kompilowanie wszystkich projektów z aktywnej przestrzeni roboczej (workspace).

Wykonywanie kompilowania programu jest kontrolowane przez szereg parametrów zapisywanych w skrypcie. Domyślne ustawienia będą dobre dla większości projektów, a na pewno dla prostych programów testowych. Jeżeli jednak zajdzie potrzeba ich korekty, to wszystkie niezbędne informacje można znaleźć w dokumentacji pakietu Atollic TRUEStudio for STM32.

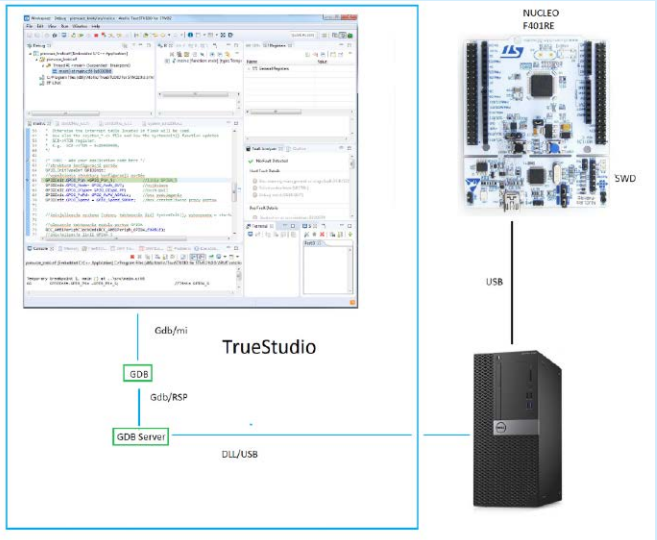
Co ciekawe, po skompilowaniu wygenerowanego szkieletu zostaje wyświetlone ostrzeżenie o tym, że jedna z pętli w pliku system_stm32f4xx.c nie jest chroniona przed całkowitym zablokowaniem programu, kiedy coś pójdzie nie tak. Rzeczywiście nie jest to przykład dobrej praktyki programowania.

Debugowanie

Debugowanie programu jest bardzo istotnym elementem pracy nad projektem. Żeby było możliwe, potrzebny jest odpowiedni program uruchomiony na komputerze i współpracujący z nim sprzętowy połączony zazwyczaj interfejsem USB układ debugera sprzężony z układami programatora/debugera z mikrokontrolerze.

Debuger pakietu TRUEStudio jest oparty o standard GDB (GNU debugger). Serwer GDB łączy środowisko graficzne TRUEStuido ze sprzętowym debugerem przez interfejs USB – **rysunek 15**.

Proces debugowania pokażemy na prostym przykładzie. Szkielet programu wygenerowany przez kreatora projektu uzupełnimy o prostą funkcjonalność polegającą na cyklicznym zapalaniu i gaszeniu



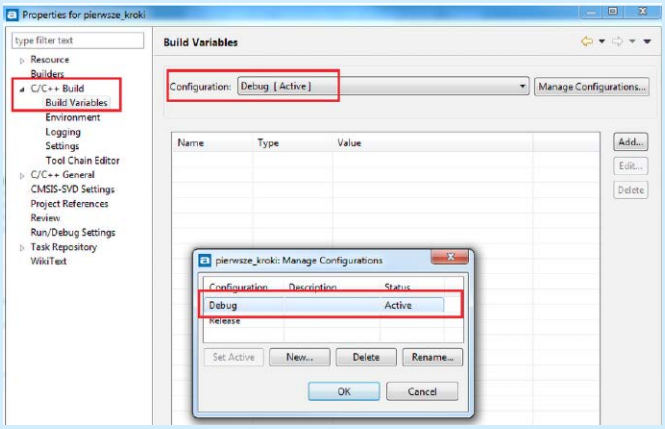
Rysunek 15. Idea działania debugera Atollic TRUEStudio

Listing 1. Konfiguracja linii PA5 i pętla cyklicznego sterowania diodą LED2

```
int main(void)
{
    /* TODO - Add your application code here */
    //struktura konfiguracji portów
    GPIO_InitTypeDef GPIOInit;
    //wypełnienie struktury konfiguracji portów
    GPIOInit.GPIO_Pin =GPIO_Pin_5;  //linia GPIOA_5
    GPIOInit.GPIO_Mode= GPIO_Mode_OUT;  //wyjściowa
    GPIOInit.GPIO_OType= GPIO_OType_PP;  //push-pull
    GPIOInit.GPIO_PuPd= GPIO_PuPd_NOPULL;  //bez podciągania
    GPIOInit.GPIO_Speed = GPIO_Speed_50MHz;  //maks. częstotliwość pracy portów
    //inicjalizacja systemu (rdzeń, taktowanie itd) SystemInit(); wykonywana w startup przed funkcją main()
    //włączenie taktowania modułu portów GPIOA
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA,ENABLE);
    //inicjalizacja linii GPIOA_5
    GPIO_Init(GPIOA, &GPIOInit);
    /* Infinite loop */
    while (1)
    {
        //poziom wysoki na linii GPIO_5 - LED2 świeci się
        GPIO_SetBits(GPIOA, GPIO_Pin_5);
        //opoznienie
        Delay(2000000);
        //poziom niski na linii GPIO_5 - LED2 zgaszona
        GPIO_ResetBits(GPIOA, GPIO_Pin_5);
        //opoznienie
        Delay(2000000);
    }
}
```

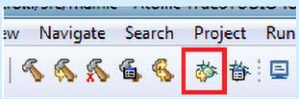
Listing 2. Opóźnienie programowe

```
void Delay(uint32_t del)
{
    uint32_t del1, i;
    for(i=0; i<del; i++)
    {
        for(del1=0; del1<10000; del1++) del1=del;
    }
}
```

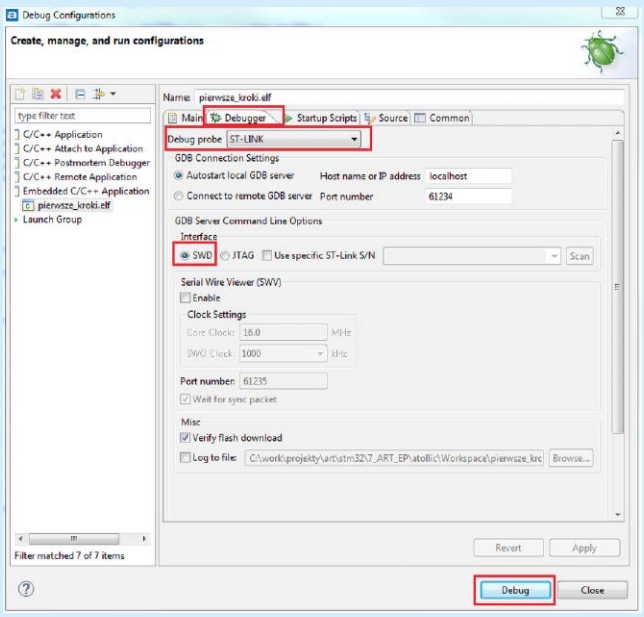


Rysunek 16. Opcja kompilacji Debug

diody LED2 zabudowanej w module NUCLEO-F401RE. Dioda jest połączona anodą z linią portu PORTA5 (PA5). Żeby sterowanie linią portu było możliwe to trzeba wykonać kilka prostych czynności konfiguracyjnych:

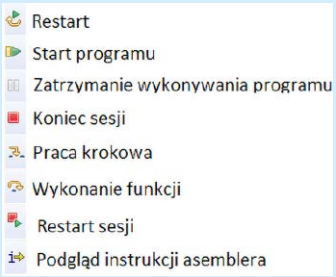


Rysunek 17. Uruchomienie sesji debugowania



Rysunek 18. Konfiguracja debugera

- Włączyć taktowanie modułu portów
- Ustawić kierunek linii PA5 jako wyjściową
- Ustawić typ portu jako push-pull
- Określić maksymalną częstotliwość pracy portu

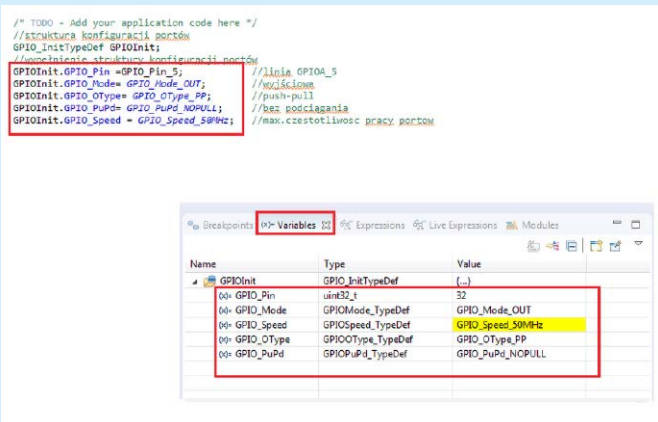


Rysunek 19. Podstawowe komendy debuggera

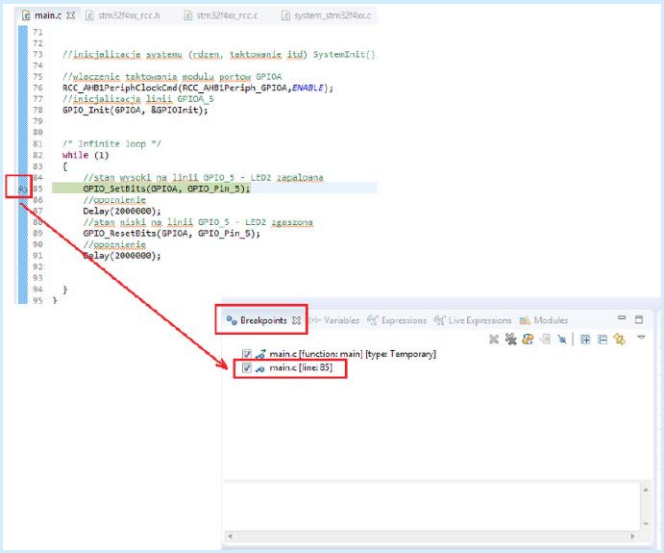
Konfiguracja portu i nieskończona pętla, w której jest cyklicznie zmieniany stan linii GPA5 zostały pokazane na **listingu 1**.

Przed uruchomieniem sesji debugowania trzeba podłączyć moduł NUCLEO – F401RE do portu USB komputera, na którym jest uruchomiony Atollic TRUEStudio. Testowy projekt trzeba skompilować z opcją kompilacji Debug ustawiana w opcjach kompilatora – **rysunek 16**. Sesję debugowania uruchamia się klikając na ikonę Debug umieszczoną w pasku narzędziowym – **rysunek 17**. Przy pierwszym uruchomieniu sesji pojawi się okno konfiguracji, w którym najważniejsza jest zakładka Debugger – **rysunek 18**. Dla modułu Nucleo musi być wybrany ST-LINK i interfejs SWD. Konfigurowanie debugowania jest dość rozbudowane i dokładnie opisane w dokumentacji TRUEStudio. My skorzystamy z domyślnych ustawień oferowanych przez program.

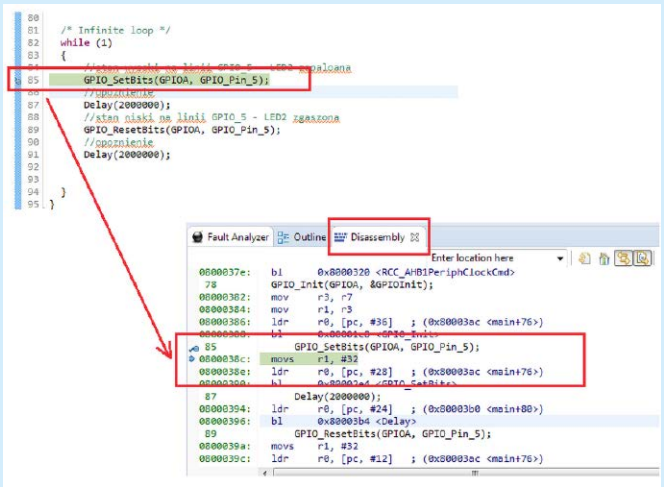
Po kliknięciu na Debug uruchomi się sesja debugowania z automatycznie otwieraną perspektywą Debug. Łączy się to z automatycznym załadowaniem pliku wynikowego kompilacji do pamięci



Rysunek 20. Podgląd struktury konfiguracyjnej GPIOInit w oknie Variables



Rysunek 21. Ustawianie punktów zatrzymań i zarzadzanie nimi w zakładce Breakpoints



Rysunek 22. Podgląd wykonywanego kodu w asemblerze

mikrokontrolera. W pasku narzędziowym są umieszczone podstawowe ikony wykorzystywane w pracy z debugerem – **rysunek 19**.

Na początku wskaźnik ustawia się na pierwszą instrukcję funkcji main(). Kliknięcie na ikonę „Start Programu" program zaczyna się wykonywać do momentu kliknięcia na „zatrzymanie wykonywania programu", lub do natrafienia na ustawiony punkt zatrzymań Breakpoint. Kolejne instrukcje w języku C są wykonywane po każdym kliknięciu na ikonę „praca krokowa" (alternatywnie klawisz F5).

W domyślnych ustawieniach perspektywy Debug jest otwierane okno z zakładkami, w których umieszczono:

- Punkty zatrzymań – Breakpoints
- Podgląd zmiennych globalnych – Variables
- Podgląd zmiennych Expressions i Live Expressions

Na **rysunku 20** pokazano podgląd w oknie Variables struktury konfiguracyjnej GPIOInit po krokowym wykonaniu instrukcji zapisywania jej składowych z funkcji main().

Punkty zatrzymań można ustawiać w bardzo wygodny sposób przez klikanie obok instrukcji w oknie z wykonywanym plikiem źródłowym. Na **rysunku 21** pokazano ustawienie punktu zatrzymania na linii 85 w pliku źródłowym main.c, oraz okno z zakładka Breakpoints.

W czasie wykonywania programu można też podglądać wykonywanie kodu w w asemblerze. Okno Disassembly jest otwierane po kliknięciu na ikonkę Podgląd instrukcji asemblera (**rysunek 22**).

Tomasz Jabłoński, EP