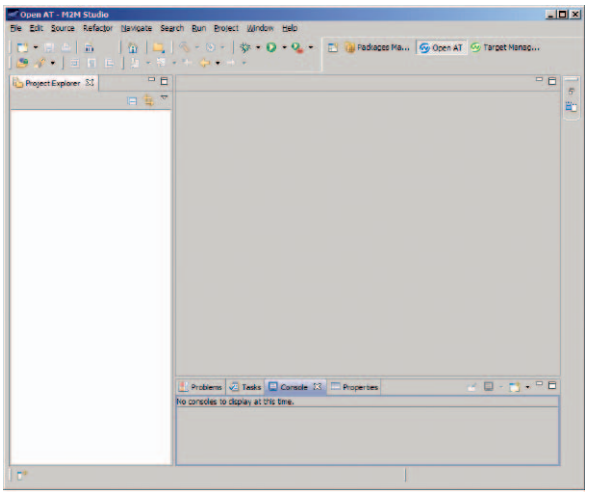
Technologia GSM w elektronice (3)

Hello World – pierwsze kroki w Open AT

W trzecim odcinku kursu programowania modemów GSM nadszedł czas na bliższe zapoznanie się ze środowiskiem M2M Studio oraz napisanie własnej aplikacji, którą następnie wgramy do modułu Sierra Wireless AirPrime Q26.

Zagadnienie instalacji zostało poruszone w pierwszym odcinku naszego cyklu, kiedy opisywano możliwości *Espresso* – jednej z aplikacji znajdujących się w pakiecie. Dla przypomnienia dodam, że środowisko *Sierra Wireless Software Suite* jest dostępne bezpłatnie pod adresem strony internetowej www.sierrawireless.com.

W tym odcinku pokażemy krok po kroku, w jaki sposób napisać i wgrać własną aplikację do modułu AirPrime Q26 oraz zaprezentujemy pomocne narzędzia.



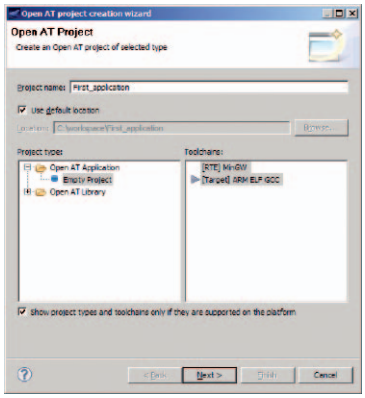
Rys. 9. Okno główne aplikacji M2M Studio

Na początek należy podłączyć do komputera zestaw startowy modułu *Sierra Wireless AirPrime Q26* opisywany w poprzednim artykule (EP 4/2010), włączyć jego zasilanie a następnie uruchomić *M2M Studio*. Okno główne aplikacji pokazano na **rys. 9**.

Zanim aplikacja się uruchomi, użytkownik zostanie zapytany o lokalizację katalogu *workspace*, czyli miejsca, gdzie będą przechowywane wykonywane projekty.

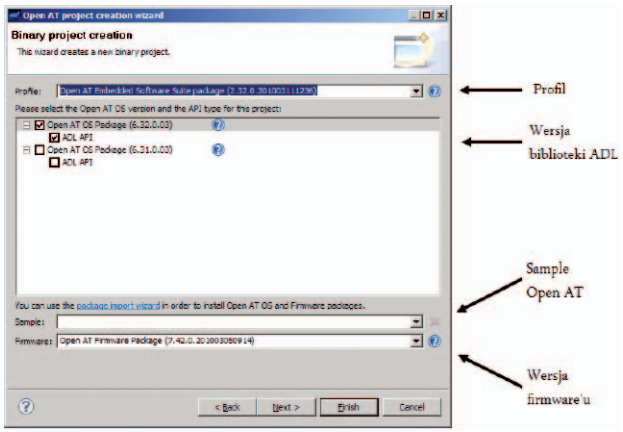
Aby napisać własną aplikację, z górnego menu należy wybrać *New*, a następnie *Open AT Project*. Pojawi się okno kreatora, w którym trzeba wpisać nazwę tworzonego projektu (**rys. 10**).

Oprócz zwykłej aplikacji użytkownik ma również możliwość tworzenia własnych bibliotek funkcji, których można później używać w kolejnych wykonywanych projektach. W tym celu w lewym oknie, pod wyborem nazwy projektu, zamiast *Open AT Application* należy wybrać *Open AT library*. Po wpisaniu nazwy naciskamy przycisk *Next* do pojawienia się okienka jak na **rys. 11**.

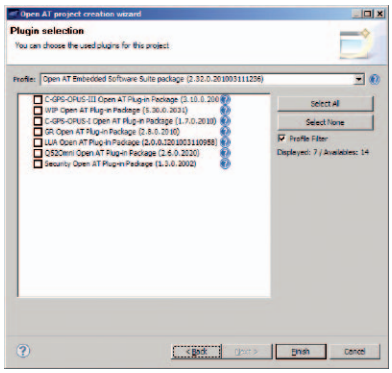


Rys. 10. Okno kreatora projektu

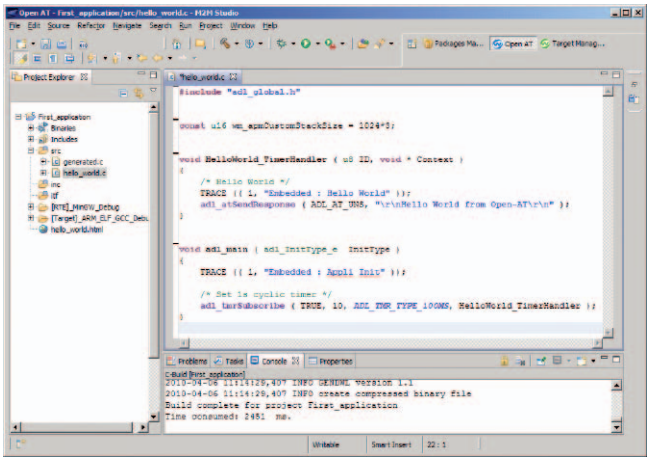
Zawartość okna wymaga nieco szerszego komentarza, ponieważ zawiera ono kilka pól wyboru parametrów kluczowych do poprawnego działania aplikacji. Chociaż programowanie w *Open AT* odbywa się w języku C, jednak sam język C nie zawiera funkcji pozwalających np. na wysłanie SMS-ów. Do tego celu producent stworzył specjalną bibliotekę o nazwie ADL (*Application Development Layer*), która udostępnia programiście funkcje API pozwalające na kontrolowanie pracy modułu GSM. Wersja biblioteki ADL jest ściśle powiązana z wersją *firmware*, pod kontrolą którego pracuje moduł, dlatego należy zwrócić szczególną uwagę na wybór odpowiedniej. Wbrew pozorom nie jest to takie trudne, ponieważ zarówno biblioteka, jak i *firmware* są dostarczane w kompletnym pakiecie. Po zaimpor-



Rys. 11. Kreator projektu – konfiguracja wersji biblioteki ADL



Rys. 12. Kreator projektu – wybór dodatkowych Plug-inów



Rys. 13. Okno edytora – nasza aplikacja „Hello World"

towaniu pakietu, w górnym menu wyboru pojawi się profil odpowiadający nazwie importowanego pakietu. Przy wyborze odpowiedniego profilu automatycznie zostanie wybrana biblioteka i *firmware*. Dla przypomnienia – wersję *firmware* modułu można sprawdzić komendą ATI3.

Producent zadbał również o dostarczenie wraz ze środowiskiem dużego zbioru aplikacji przykładowych, które demonstrują możliwości modułu. Wybierając odpowiednią aplikację, mamy możliwość obejrzenia kodu źródłowego oraz skompilowania go i wgrania do modułu. Na początek wybierzmy kod najprostszej aplikacji, czyli „*Hello World*". Po jej wybraniu w polu *Sample*, naciskamy klawisz *Next*. Pojawia się okno zatytułowane „*Plugin Selection*" (**rys. 12**). W nim jest możliwość dołączenia do aplikacji kolejnych bibliotek, które co prawda nie są obowiązkowe, tak jak biblioteka ADL, ale oferują dodatkowe funkcje i możliwości, które mogą przydać się w aplikacji. Mamy do wyboru m.in. dość często stosowaną bibliotekę WIP (*Wavecom IP*) pozwalającą w łatwy sposób korzystać z protokołów internetowych (TCP, UDP i inne) czy bibliotekę LUA, umożliwiającą wgranie do modułu interpretera języka skryptowego LUA.

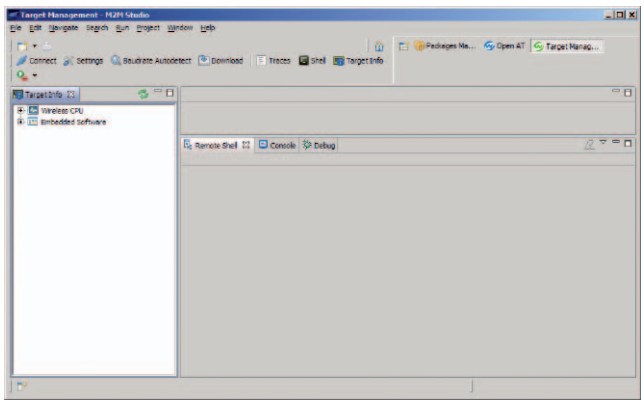
Biblioteka *Security* pozwala na obsługę zagadnień związanych z bezpieczeństwem, takich jak wykrywanie zagłuszania GSM czy bezpieczna transmisja z wykorzystaniem protokołu SSL. Pozostałe biblioteki to biblioteki specjalistyczne.

Bibliotekę z sekcji *plugins* możemy dołączyć na dowolnym etapie tworzenia projektu, kiedy zajdzie potrzeba skorzystania z funkcji oferowanych przez bibliotekę.

Dokumentację do bibliotek znajdziemy w zakładce pomocy (*Help –> Help Contents –> Open AT Plug-ins documentation*).

Naciśnięcie przycisku *Finish* kończy pracę kreatora i powoduje utworzenie przestrzeni projektu i przejście do okna edytora kodu (**rys. 13**).

Po tej czynności należy zwrócić uwagę na okno *Project Explorer* po lewej stronie. Zawiera ona katalogi wszystkich projektów znajdujących się w lokalizacji *workspace*. Po rozwinięciu projektu zauważymy, że składa się on z wielu katalogów. Dla nas istotne są: katalog „*src*", w którym znajdują się pliki źródłowe aplikacji oraz katalog „*inc*", zawierający pliki nagłówkowe. Przy tworzeniu kolejnego pliku źródłowego lub nagłówkowego należy pamiętać o tym, żeby znalazł się we właściwym katalogu.



Rys. 14. Okno aplikacji – widok Target Managment

Każda aplikacja z katalogu przykładów (*samples*) ma dodatkowo plik html, w którym znajduje się opis jej działania. Plik ten jest widoczny w katalogu projektu. Warto się z nim zapoznać.

Po kliknięciu na nazwie pliku *hello_world.c* z katalogu „*src*", w oknie edytora pojawi się kod źródłowy aplikacji. Opisywany przykład składa się z polecenia *include*, które dołącza bibliotekę ADL, deklaracji stosu oraz dwóch funkcji: *HelloWorld_TimerHandler()* i *adl_main()*. Punktem startu aplikacji jest funkcja *adl_main()*. W odróżnieniu od standardowej funkcji *main()* pełni ona rolę funkcji konfiguracji i ustawienia zdarzenia. Ma zmienną wejściową *InitType* typu *adl_InitType_e*. Gdy zaznaczymy typ zmiennej, np. klikając dwukrotnie na *adl_InitType_e*, a następnie naciśniemy F3, wyświetli się definicja typu jak niżej:

```
typedef enum _adl_InitType_e
{
    ADL_INIT_POWER_ON,  ///< Normal power-on
    ADL_INIT_REBOOT_FROM_EXCEPTION, ///< Reboot
after an exception
    ADL_INIT_DOWNLOAD_SUCCESS,  ///< Reboot after
a successful install process (cf. adl_adInstall()
API)
    ADL_INIT_DOWNLOAD_ERROR,  ///< Reboot after an
error in install process (cf. adl_adInstall() API)
    ADL_INIT_RTC  ///< Power-on due to an RTC alarm
                  //(cf. the <b>AT+CALA</b> command
                  // documentation for more
information)
} adl_InitType_e;
```

Wartość zmiennej wskazuje na przyczynę startu modułu. Gdy moduł został zresetowany lub włączony, wtedy zmienna ta ma wartość *ADL_INIT_POWER_ON*. W przypadku, gdy przeprowadzany był proces zdalnej aktualizacji aplikacji, zmienna (zależnie od wyniku) przyjmie wartość *ADL_INIT_DOWNLOAD_SUCCESS* lub *ADL_INIT_DOWNLOAD_ERROR*. Jeśli moduł został załączony przez alarm, to zmienna przyjmie wartość *ADL_INIT_RTC*.

Głównym problemem, z którym często borykają się początkujący programiści, jest zrozumienie zastosowanej koncepcji programowania z użyciem zdarzeń. W dużej mierze jest ono oparte na rejestrowaniu się (subskrypcji) do zdarzeń i usług, które chcemy zastosować. Rejestracja polega zazwyczaj na wskazaniu funkcji, która zostanie wykonana przez system operacyjny w momencie wystąpienia zdarzenia. Nie ma więc potrzeby cyklicznego sprawdzania, czy wystąpiło dane zdarzenie. Zawsze w momencie jego wystąpienia system operacyjny wykona wskazaną funkcję. Jeśli jakiś serwis nie zostanie zarejestrowany w aplikacji, to zostanie wykonany tak, jakby to miało miejsce w przypadku, gdyby aplikacji w ogóle nie było, np. jeśli nie zarejestrujemy obsługi SMS-ów, to będą one obsługiwane, tak jak w przypadku tradycyjnego modemu GSM. W przykładowej aplikacji *Hello World* mamy następującą funkcję:

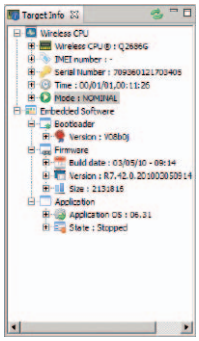
```
adl_tmrSubscribe ( TRUE, 10, ADL_TMR_TYPE_100MS,
HelloWorld_TimerHandler );
```

Powoduje ona powołanie timera o kroku 1 s (10×100 ms) wywołującego cyklicznie (pierwszy parametr ma wartość TRUE) funkcję *HelloWorld_TimerHandler*. Natomiast w funkcji *HelloWorld_TimerHandler* wykonywane jest polecenie wysłanie tekstu „*Hello World from Open-AT*" przez port szeregowy: `adl_atSendResponse (ADL_AT_UNS, "\r\nHello World from Open-AT\r\n" );`

Funkcje *adl_tmrSubscribe* oraz *adl_atSendResponse* zostały dokładnie opisane w dokumencie o nazwie *Open AT ADL Development Guide*. Dokument ten można znaleźć w menu *Help –> Help Contents*

–> *Open AT Embedded Software Suite package version 6.32*. Krótką dokumentację funkcji można również wyświetlić poprzez zaznaczenie jej nazwy i naciśnięcie F3.

Warto również zwrócić uwagę na dwie linie w kodzie zaczynające się od słowa kluczowego *TRACE*. Są to makra, które pozwalają na debugowanie aplikacji. Za ich pomocą można wysyłać przez port szeregowy tekst lub wartości zmiennych służące do analizy pracy aplikacji. Wartość cyfrowa będąca argumentem określa poziom *TRACE*. Można zdefiniować do 32 poziomów, nadając poleceniom TRACE różne wartości w zależności od funkcji, jakie pełnią. Pozwala to filtrować informacje wysyłanie przez debuger zależnie od potrzeb. Informacje te są wysyłane wyłącznie przez port szeregowy, przez który moduł jest połączony z aplikacją *M2M Studio*. Użytkownik nie musi zatem obawiać się, że w jakiś sposób będą one zakłócać komunikację z zewnętrznym urządzeniem podłączonym do portu szeregowego.

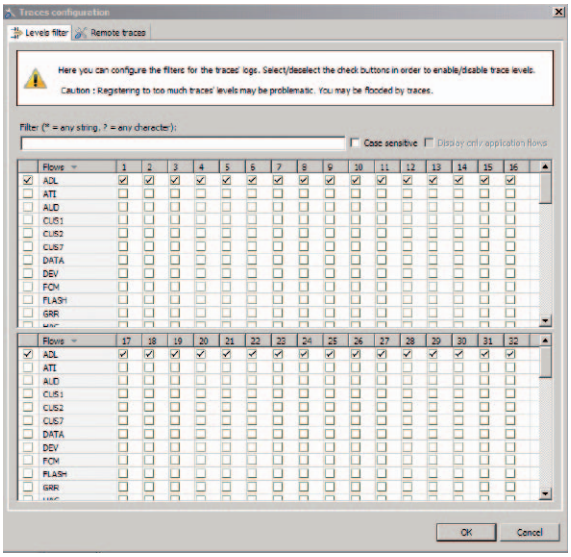


Rys. 15. Target Info – informacje o module

Kompilację aplikacji wykonuje się, wybierając z menu *Project –> Build All* lub klikając na symbol młotka znajdującego się na górnej listwie paska narzędzi. W celu przesłania skompilowanej już aplikacji do modułu należy przejść do trybu *Target Management* (**rys.14**, zakładka w górnej prawej części okienka edytora).

Klikając zakładkę *Settings*, ustawiamy numer i parametry portu szeregowego PC, do którego podłączony jest moduł *AirPrime Q26*, a następnie naciskamy *Connect*. Poprawność połączenia z modułem sprawdzamy, naciskając dwie zielone strzałki w okienku *Target Info*. Powinniśmy otrzymać informacje o podłączonym module, jak na **rys. 15**. Teraz klikamy na zakładkę *Download* i wskazujemy ścieżkę do projektu (na przykład: C:\workspace\First_application\[Target]_ARM_ELF_GCC_Debug). Wskazujemy plik z rozszerzeniem .dwl i czekamy, aż zakończy się proces ładowania do pamięci. Po jego zakończeniu naciskamy zakładkę *Shell* i *Traces*. Po otworzeniu obu okien klikamy w prawym oknie o nazwie *Target Info* na pozycję *Application –> State* i wybieramy pozycję *Start Application* lub wpisujemy w linii komend (*Command*) polecenie AT+WOPEN=1. Jeśli wszystko poszło jak trzeba, powinniśmy zobaczyć rezultat jak na **rys. 16**.

Może się zdarzyć, że w oknie *Traces View* nic się nie pojawi, podczas gdy w oknie *Target* będziemy co sekundę otrzymywali komunikat „*Hello World from Open-AT*". Należy wtedy kliknąć na ikonkę przypominającą skrzyżowane klucz i śrubokręt *Remote Traces Configuration* umieszczoną w oknie *Traces View*. Pojawi się okno z **rys. 17**.



Rys. 17. Okno filtra dla makra TRACE

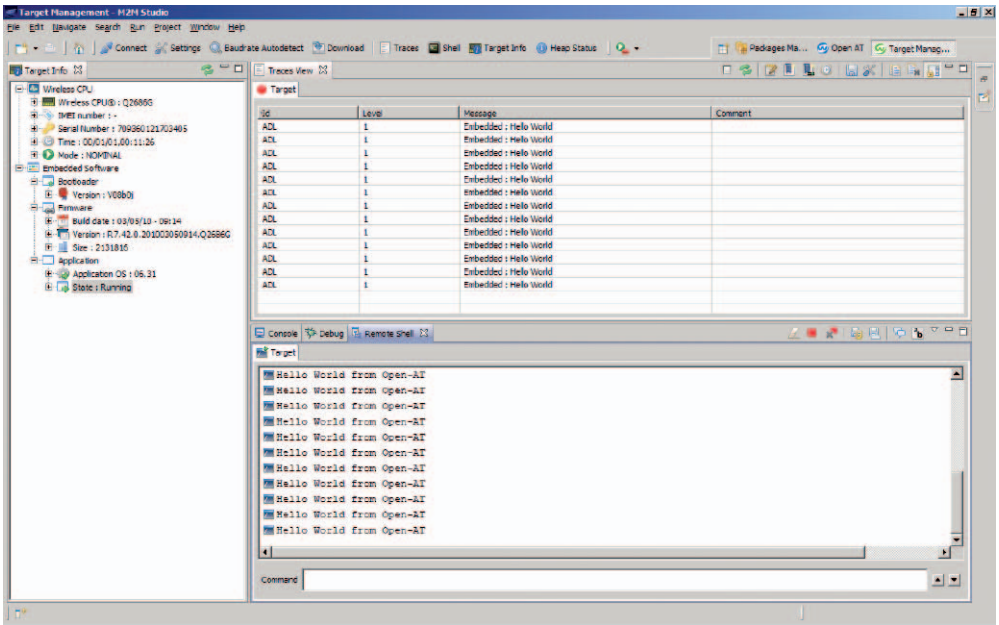
Zaznaczamy wszystkie 32 poziomy dla wartości ADL lub tylko te, które są potrzebne. Właśnie tu możemy filtrować informacje wyświetlane w oknie *Traces View*, a wysyłane poprzez makra *TRACE*.

Pozostałe poziomy logowania to logi systemowe. Po zaznaczeniu odpowiednich poziomów logowania i naciśnięciu *OK* pozostaje jeszcze kliknąć na *Active logging* - pierwszą ikonę od lewej w oknie *Traces View*.

Powróćmy teraz do edytora klikając, na zakładkę *Open AT* (górna, prawa część okna głównego) i nieco zmodyfikujmy funkcję *HelloWorld_TimerHandler*:

```
void HelloWorld_TimerHandler ( u8 ID, void * Context )
{
    /* Hello World */
    static int a=0;
    TRACE (( 1, "Embedded : Hello World: %d",a++ ));
    adl_atSendResponse ( ADL_AT_UNS, "\r\nHello World from Open-AT \r\n" );
}
```

Następnie skompilujmy, wgrajmy do modułu i obejrzyjmy rezultat w oknie *Traces View*. Po zakończeniu eksperymentów z aplikacją, warto również uruchomić i przetestować działanie innych przykładów przygotowanych przez producenta.



Rys. 16. Rezultat działania naszej aplikacji „Hello World"

Pierwsze kroki w Open AT już za nami. W kolejnych odcinkach cyklu zajmiemy się bardziej zaawansowanymi aplikacjami. Tymczasem polecam lekturę dokumentacji biblioteki ADL - *Open AT ADL Development Guide*.

Więcej informacji na temat produktów Sierra Wireless można znaleźć na stronach producenta *www.sierrawireless.com* lub kontaktując się z firmą ACTE Sp. z o.o., która jest oficjalnym dystrybutorem opisywanych produktów oraz zapewnia pełne wsparcie techniczne.

Adrian Chrzanowski
Acte Sp. z o.o.