# Przemysłowy Internet Rzeczy (3)

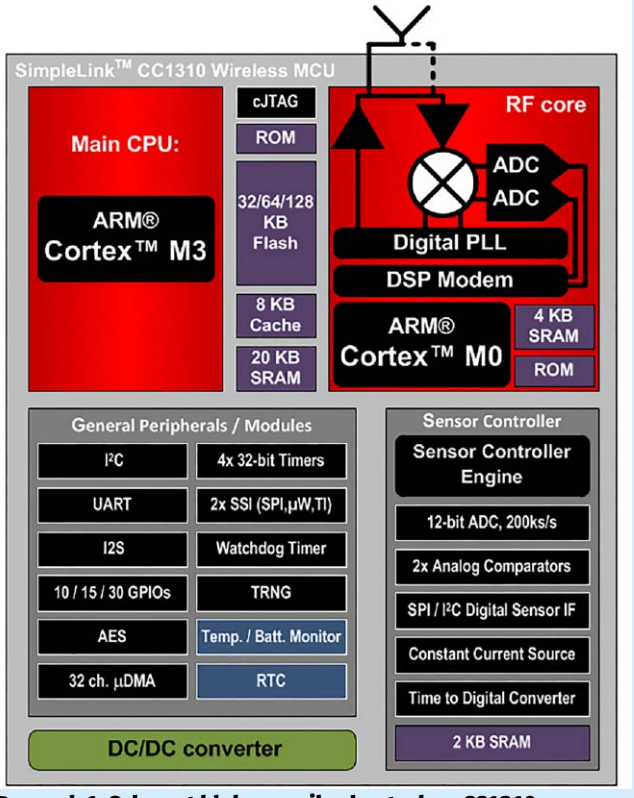
## Mikrokontroler CC1310 – programowanie kontrolera czujników

*W artykule zaprezentujemy obsługę wbudowanego w mikrokontroler CC1310 modułu kontrolera czujników. Wykonamy projekt urządzenia do bezdotykowego pomiaru temperatury. W artykule korzystać będziemy z oprogramowania Code Composer Studio, Sensor Controller Studio oraz z układu do bezdotykowego pomiaru temperatury TMP007.*

W obudowie mikrokontrolera CC1310 umieszczono: procesor użytkownika z rdzeniem ARM Cortex-M3 (wraz z blokami peryferyjnymi), procesor dedykowany do transmisji radiowej w paśmie ISM poniżej 1 GHz z rdzeniem ARM Cortex-M0, przetwornicę DC/DC oraz przedmiot dzisiejszego artykułu *kontroler czujników pomiarowych*. Schemat blokowy mikrokontrolera CC1310 pokazano na **rysunku 1.**

### Sensor Controller

Wbudowany w mikrokontroler CC1310 kontroler czujników jest zarządzany przez 16-bitowy, energooszczędny mikroprocesor. Moduł kontrolera czujników wyposażono w 2 kB pamięci operacyjnej SRAM oraz liczne moduły peryferyjne (12-bitowy przetwornik A/C, dwa komparatory analogowe, jeden licznik Timer0, ***źródło prądowe,*** konwerter czas/cyfra TDC ). Zaimplementowano programową obsługę interfejsów szeregowych SPI, I²C oraz emulator interfejsu UART. Kontroler czujników ma zasilanie oraz taktowanie niezależne od rdzenia CPU mikrokontrolera CC1310.




**Rysunek 1. Schemat blokowy mikrokontrolera CC1310**


**Rysunek 2. Ekran startowy oprogramowania Sensor Controller Studio**

**Tabela 1. Plik źródłowe sterownika Sensor Controller Interface**

| pliki | funkcja |
|---|---|
| scif.c<br>scif.h | Konfiguracja sterownika (dane i kod źródłowy programu, definicje stałych, struktury ze zmiennymi, mapowanie linii I/O). |
| scif_framework.c<br>scif_framework.h | Interfejs API do zarządzania zadaniami (kontrola zadań, wymiana danych). |
| scif_osal_tirtos.c<br>scif_osal_tirtos.h | Implementacja warstwy programowej do obsługi pracy z systemu czasu rzeczywistego TI-RTOS. |
| scif_osal_none.c<br>scif_osal_none.h | Implementacja warstwy programowej do pracy bez obsługi systemu czasu rzeczywistego. |

Programowanie pracy kontrolera czujników polega na budowaniu zadań (tasks). Podstawowe zadania, które można oprogramować to:

- odczyt wartości z czujników analogowych (użycie przetwornika A/C oraz komparatora analogowego),
- odczyt wartości z czujników cyfrowych (użycie interfejsów szeregowych SPI, I²C, UART ),
- obsługa przycisków i klawiatur pojemnościowych (użycie źródła prądowego, komparatora oraz konwertera czas/cyfra).

Kontroler czujników może mieć zaprogramowane aż 8 zadań, przy czym tylko jedno zadanie może mieć blok programu, którego wykonanie jest aktywowane przez zdarzenie (odmierzenie zdefiniowanego odcinka czasu albo zmiana poziomu logicznego na wyjściu komparatora, albo zmiany poziomu logicznego na wyjściu I/O mikrokontrolera).

Zadania dla kontrolera czujników tworzymy w przygotowanym przez Texas Instruments oprogramowaniu *Sensor Controller Studio*. ***Język programowania zadań jest zbliżony do języka C. Każde zadanie składa się z trzech*** podstawowych bloków kodu programu:

1. *Initialization* (inicjalizacja zadania – wykonywana raz na początku zadania).
2. *Execution* (wykonanie zadania – wykonywane cyklicznie zgodnie z harmonogramem zegara RTC ).
3. *Ttermination* (zakończenie zadania – wykonywane raz przy kończeniu zadania).

Zadanie z obsługą zdarzenia ma dodatkowy blok kodu programu *event handling* (obsługa zdarzenia – kod programu jest wykonywany po wystąpieniu zdefiniowanego zdarzenia).

Korzystając z oprogramowania *Sensor Controller Studio* możemy nie tylko tworzyć zadania, ale także testować i emulować ich działanie. Produktem wyjściowym oprogramowania jest sterownik *Sensor Controller Interface* w skrócie *SCIF*. Kod źródłowy sterownika jest zapisywany w plikach o funkcjonalności opisanej w **tabeli 1**. Podczas tworzenia oprogramowania pliki sterownika *SCIF* wygenerowane za pomocą *Sensor Controller Studio* dołączamy do projektu w *Code Composer Studio*.

Wykonując zadania kontroler czujników pracuje w trybie aktywnym. W pozostałym czasie pracy jest wprowadzany w tryb ***uśpienia. W trybie aktywnym rdzeń kontrolera czujników*** jest taktowany przebiegiem zegarowym o częstotliwości maksymalnej 24 MHz. W trybie uśpienia, gdy żadne z zadań nie ma zdefiniowanej obsługi zdarzenia, rdzeń kontrolera jest wyłączony. Jeśli któreś z zadań ma włączoną obsługę zdarzenia, to rdzeń kontrolera jest taktowany przebiegiem o częstotliwości zegarkowej 32,768 kHz. Zmiana trybu pracy z aktywnego na tryb ***uśpienia*** i odwrotnie jest wykonywana automatycznie bez udziału programisty.

Kontroler czujników nie posiada dostępu do pamięci SRAM, FLASH oraz do modułów peryferyjnych i rejestrów mikrokontrolera CC1310. Z drugiej strony, jednostka centralna MCU mikrokontrolera CC1310 ma dostęp do modułów peryferyjnych oraz pamięci SRAM kontrolera czujników.

Programując pracę kontrolera czujników dla każdego zadania możemy zdefiniować zmienne globalne. Chcąc zdefiniować zmienne najpierw musimy jednak utworzyć strukturę, ***w której będą przechowywane. Do dyspozycji mamy cztery rodzaje struktur*** a zmienne umieszczamy w strukturach zgodnie z ich funkcjonalnością. Dostępne struktury danych to: *cfg* (zmienne do konfigurowania zadań), *input* (zmienne z danymi wejściowymi np.: konfiguracja parametrów czujników pomiarowych), *output* (dane wyjściowe np.: wyniki pomiarów), *state* (zmienne wewnętrzne np.: stan wykonania pomiaru). Struktury są tworzone w pamięci SRAM kontrolera czujników i mogą być buforowane pojedynczo lub wielokrotnie.

***Żeby uzyskać dostęp do danych jednostka CPU mikrokontrolera CC1310 może cyklicznie sprawdzać stan kontrolera czujników i w momencie***, gdy zmienne będą dostępne (kontroler nie wykonuje zadania) uzyskać do nich dostęp. Chęć wymiany danych może być również zainicjowana przez zadanie. W takim przypadku to zadanie zgłaszając przerwanie alarmuje jednostkę CPU o chęci wymiany danych (przykładowo wymiana danych ze struktury *output* po zakończeniu pomiaru).

## Sensor Controller Studio

Sposób pobrania i zainstalowania oprogramowania *Sensor Controller Studio* prezentowaliśmy w poprzedniej części kursu (EP11/2016). Zainstalowane zostało oprogramowanie w wersji 1.3.0.

Produkowany przez *Texas Instruments* układ scalony **TMP007** (poprzednia wersja TMP006 ) mierzy temperaturę otoczenia oraz temperaturę obiektu oddalonego od czujnika. Pomiar temperatury otoczenia jest wykonywany przez wbudowany w strukturę czujnika scalony układ półprzewodnikowy. Bezprzewodowy pomiar temperatury obiektu jest wykonywany przy wykorzystaniu zjawiska promieniowania podczerwonego. Czujnik wyposażono w termostos, który mierzy poziom emisji promieniowania podczerwonego obiektu i na podstawie wyniku pomiaru oblicza jego temperaturę. Zakres pomiaru temperatury wynosi od -40 do +125°C. Układ scalony TMP007 jest oferowany w obudowie BGA o wymiarach 1,9 mm×1,9 mm×0,625 mm. Szeroki zakres napięcia zasilania 2,5...5,5 V ułatwia integrację z systemem pomiarowym. Charakterystyczną cechą układu TMP007 jest mały pobór mocy. Podczas pomiaru temperatury wynosi 270 μA, a w trybie uśpienia jest obniżony do 2 μA. Komunikacja z czujnikiem TMP007 odbywa się za pomocą interfejsu I²C. Obsługiwane są dwa tryby transmisji: tryb *fast* (400 kHz) oraz *high-speed* (do 2,5 MHz). Szczegółowe informacje o budowie oraz pracy czujnika TMP007 są dostępne na stronie **https://goo.gl/B5orNP**.

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 97325, pass: 6yfwxr8q

Uruchamiamy aplikację i sprawdzamy czy są dostępne aktualizacje. W tym celu z menu programu z zakładki *Updates* wybieramy *Check for updates*. Jeśli aktualizacje są dostępne to w prawym górnym roku okna aplikacji zostanie wyświetlona informacja o najnowszej wersji programu. W naszym wypadku jest dostępna nowa wersja aplikacji. Ponownie uruchamiamy zakładkę *Updates* i wybierając zakładkę *Manage Updates* uruchamiamy panel zarządzania aktualizacjami. Z dostępnych opcji wybieramy *Get Installer*. Pobieramy i instalujemy najnowszą wersje oprogramowanie *Sensor Controller Studio* (w chwili pisania artykułu jest to wersja o numerze 1.4.0). Procedura pobierania i instalacji oprogramowania jest identyczna jak opisana w poprzedniej części kursu.

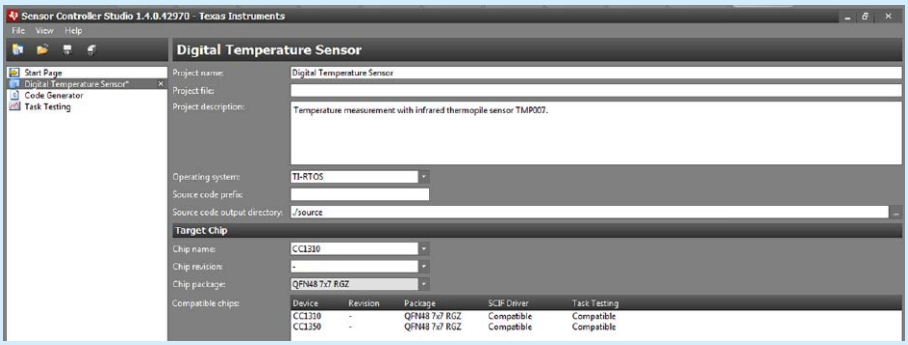
Po zakończonej aktualizacji uruchamiamy oprogramowanie *Sensor Controller Studio*. Prezentowany jest ekran startowy aplikacji, którego wygląd pokazano na **rysunku 2**. Dostępne są okna z historią projektów z przykładami oraz z dokumentacją. Na szczególna uwagę zasługują przygotowane przez *Texas Instruments* przykłady oraz dokumentacja obsługi oprogramowania. Dokumentację *Sensor Controller Studio Help* możemy uruchomić z okna startowego aplikacji, z menu *Help* albo korzystając ze skrótu klawiaturowego (klawisz F1). Pozostałe funkcje oprogramowania *Sensor Controller Studio* są aktywowane w momencie utworzenia projektu. Zostaną omówione w dalszej części artykułu przy okazji tworzenia projektu urządzenia do bezdotykowego pomiaru temperatury obiektu.

## Projekt: Bezdotykowy pomiar temperatury obiektu

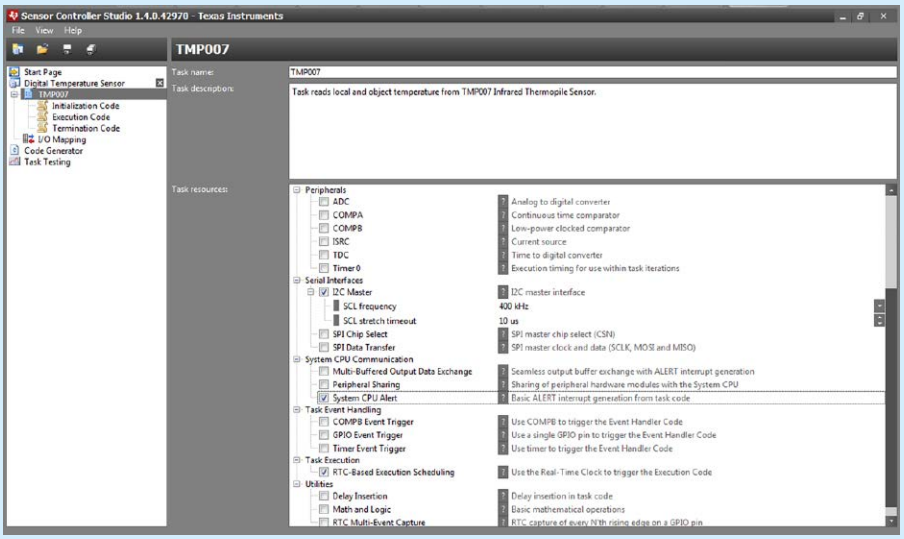
Aby zaprezentować obsługę kontrolera czujników wykonamy projekt urządzenia do bezdotykowego pomiaru temperatury. W praktyce urządzenia takie mają szerokie zastosowanie w przemyśle. Przykładowo mogą mierzyć temperaturę obiektów, które są w ruchu (wyroby na linii produkcyjnej, ruchome i trudno dostępne części maszyn). Do bezprzewodowego pomiaru temperatury wykorzystamy produkowany przez *Texas Instruments* czujnik TMP007. Za bazę sprzętową projektu posłuży nam moduł startowy LauchPad CC1310 oraz płyta rozszerzeń BoosterPack BOOSTXL-SENSORS.

Uruchamiamy oprogramowanie *Sensor Controller Studio* i tworzymy nowy projekt. W tym celu z menu programu z zakładki *File* wybieramy opcję *New Project*. Następnie wprowadzamy nazwę projektu, opis projektu oraz ustawiamy obsługę systemu operacyjnego TI-RTOS. W sekcji *Target Chip* zaznaczamy mikrokontroler CC1310 oraz wybieramy obudowę, w jakiej został zamontowany w module LaunchPad CC1310. Następnie zapisujemy projekt na dysku komputera (*File –> Save Project*). Konfigurację projektu prezentuje **rysunek 3**.

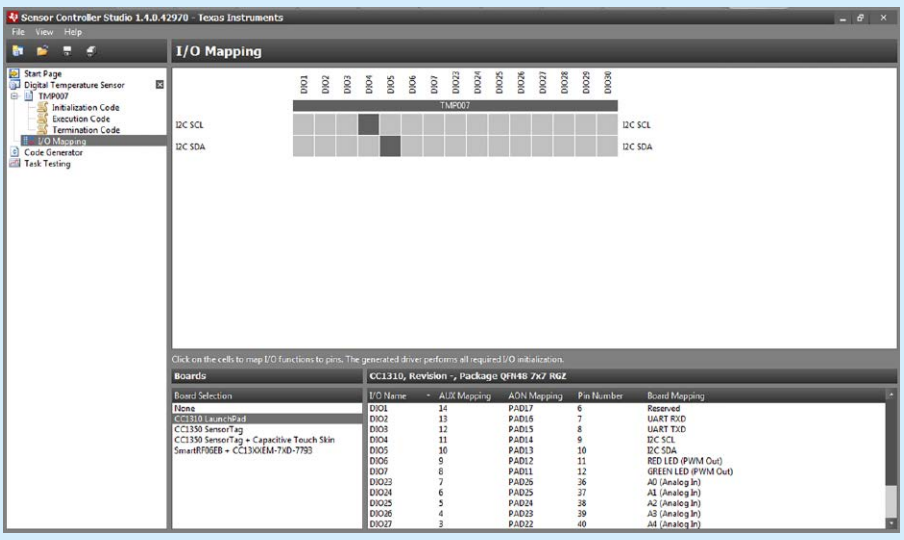
W kolejnym kroku rozpoczynamy konfigurowanie zadania do obsługi czujnika temperatury TMP007. W oknie konfiguracji projektu, w sekcji *Sensor Controller Tasks* wybieramy opcje *Add new*. Aplikacja *Sensor Controller Studio* uruchamia nowe okno programu, w którym definiujemy parametry zadania. Wprowadzamy nazwę zadania, opis zadania oraz co najważniejsze wybieramy komponenty zadania ( obsługa układów peryferyjnych, protokołów komunikacyjnych, sposób komunikacji kontrolera czujników




**Rysunek 3. Sensor Controller Studio – konfiguracja projektu**




**Rysunek 4. Sensor Controller Studio – konfiguracja zadania**




**Rysunek 5. Sensor Controller Studio – mapowanie linii wejścia-wyjścia**

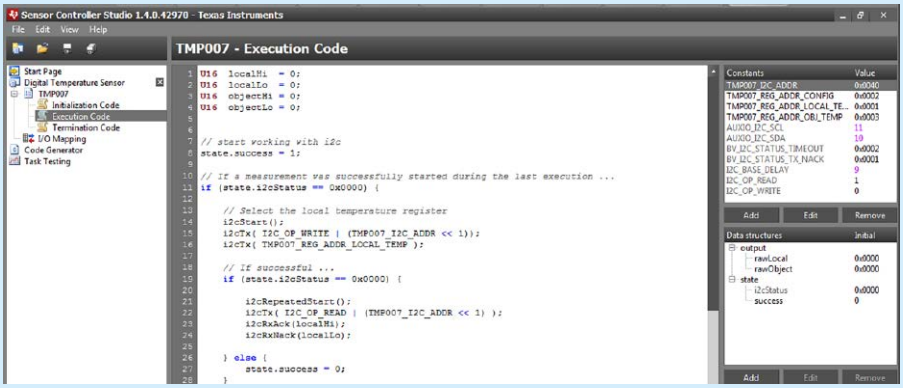
z CPU mikrokontrolera CC1310, konfiguracja linii I/O, itp.). W naszym wypadku zaznaczamy obsługę interfejsu szeregowego I²C, taktowanie wykonania zadania zegarem RTC oraz komunikację zadania z jednostką CPU przy pomocy przerwań zgłaszanych przez kontroler czujników. Po wprowadzeniu ustawień zapisujemy projekt (*File –> Save Project*). Konfigurację zadania pokazano na **rysunku 4**.

W momencie utworzenia zadania automatycznie są tworzone bloki programu: inicjalizacja, wykonanie, zakończenie (blok obsługi zdarzeń nie jest tworzony, bo do zadania nie dodaliśmy obsługi zdarzenia). Dodatkowo, jest aktywowane narzędzie *I/O Mapping* służące do mapowania linii wejścia-wyjścia mikrokontrolera CC1310. W zadaniu do komunikacji z czujnikiem temperatury TMP007 będziemy używali interfejsu szeregowego I²C, zatem musimy skonfigurować linie interfejsu I²C. Uruchamiamy narzędzie *I/O Mapping*. Z okna *Board* wybieramy płytę startową *CC1310 LaunchPad*, a następnie dla sygnału I²C SCL wybieramy linię DIO04, a dla sygnału I²C SDA linię DIO05. Zrzut ekranu z konfiguracji mapowania linii wejścia-wyjścia pokazano na **rysunku 5**.

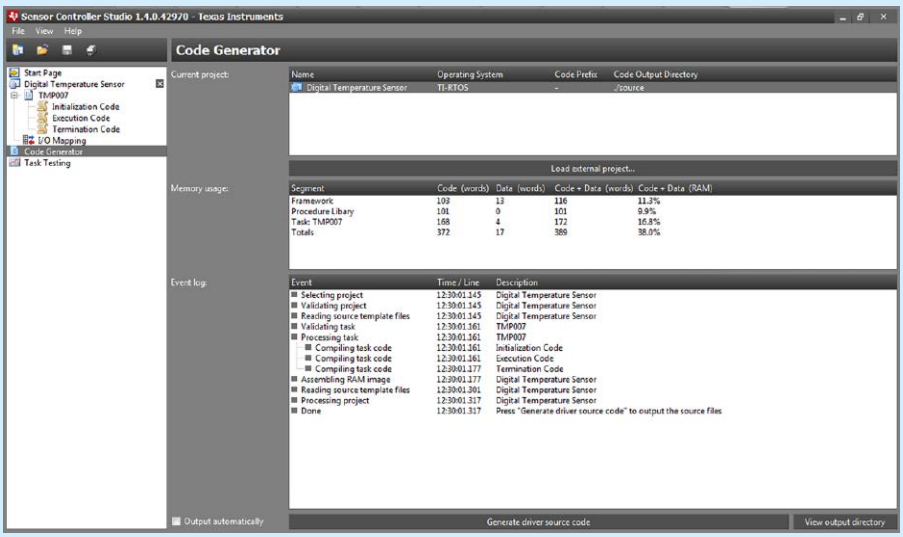
W sekcjach z definicją kodu programu: inicjalizacja, wykonanie, zakończenie wprowadzamy kod źródłowy zadania do obsługi czujnika TMP007. Korzystając z protokołu I²C ustawiamy parametry pomiaru a następnie odczytujemy zmierzoną temperaturę otoczenia oraz temperaturę obiektu. Kod źródłowy sekcji wykonania *Execution* pokazano na **listingu 1**. W kodzie źródłowym zadania tworzymy definicje stałych (adres I²C czujnika TMP007, adresy rejestrów czujnika) oraz definiujemy zmienne globalne umieszczone w strukturach danych (status pomiaru, wynik pomiaru temperatury otoczenia, wynik pomiaru temperatury obiektu). Panele do zarządzania definicjami stałych oraz strukturami danych ze zmiennymi globalnymi (dodawanie, usuwanie, edycja) są dostępne ***w każdym*** z bloków kodu programu po prawej stronie okna aplikacji. Wygląd panelu pokazano na **rysunku 6**. Wprowadzając definicje stałych podajemy nazwę, typ oraz wartość definiowanej stałej. Wprowadzając zmienne globalne dodatkowo definiujemy, ***w której strukturze danych mają być przechowywane.***

Po wprowadzeniu kodu źródłowego zadania do obsługi czujnika TMP007 sprawdzamy czy projekt nie zawiera błędów. W tym celu z panelu sterowania projektu wybieramy opcję *Code Generator*. Uruchamiane jest nowe okno oprogramowania, a projekt jest automatycznie kompilowany. Jeśli projekt nie zawiera błędów, to w oknie *Event Log* brak jest wpisów informujących o błędach. Zrzut ekranu z działania kompilatora pokazano na **rysunku 7**.

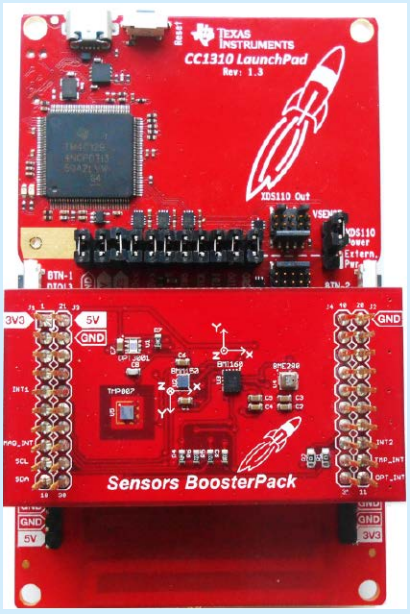
Kolejnym etapem projektu są testy utworzonego zadania. W tym celu moduł startowy LaunchPad CC1310 z zamontowana płytą rozszerzeń BOOSTXL-SENSORS przy***łączamy do złącza USB komputera PC (*fotografia 8**). Następnie, z panelu sterowania projektu wybieramy opcję *Task testing*. Otwierane jest nowe okno aplikacji z zakładkami *Setup* (konfiguracja testów) oraz *Graph* (wizualizacja zmiennych). W zakładce *Setup* definiujemy sekwencję pracy zadania podczas testów. Wybieramy uruchomienie




**Rysunek 6. Sensor Controller Studio – widok na panele do zarządzania definicjami stałych i struktur danych**




**Rysunek 7. Sensor Controller Studio – kompilowanie projektu**



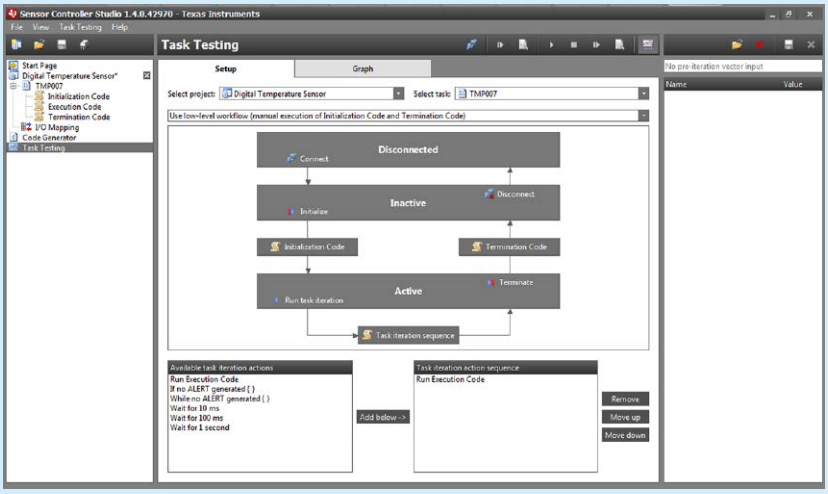

**Fotografia 8. LaunchPad CC1310 z płytą rozszerzeń BOOSTXL-SENSORS**

Płyta rozszerzeń BoosterPack **BOOSTXL-SENSORS** jest produkowana przez Texas Instrument. Na płycie umieszczono 5 czujników cyfrowych MEMS: czujnik bezdotykowego pomiaru temperatury TMP007 (Texas Instruments), czujnik oświetlenia OPT3001 (Texas Instruments), czujnik ciśnienia atmosferycznego, wilgotności i temperatury BME280 (Bosch Sensortec), 6-osiowy (żyroskop, akcelerometr) czujnik ruchu BMI160 (Bosch Sensortec), 3-osiowy kompas BMM150 (Bosch Sensortec). Moduł rozszerzenia BOOSTXL-SENSORS jest kompatybilny z większością układów startowych LaunchPad, w tym z modułem LaunchPad dla mikrokontrolera CC1310. Cena modułu w sklepie Texas Instruments wynosi ok. 25 USD. Szczegółowe informacje o module rozszerzeń BoosterPack są dostępne na stronie ***https://goo.gl/FJCuzO***.

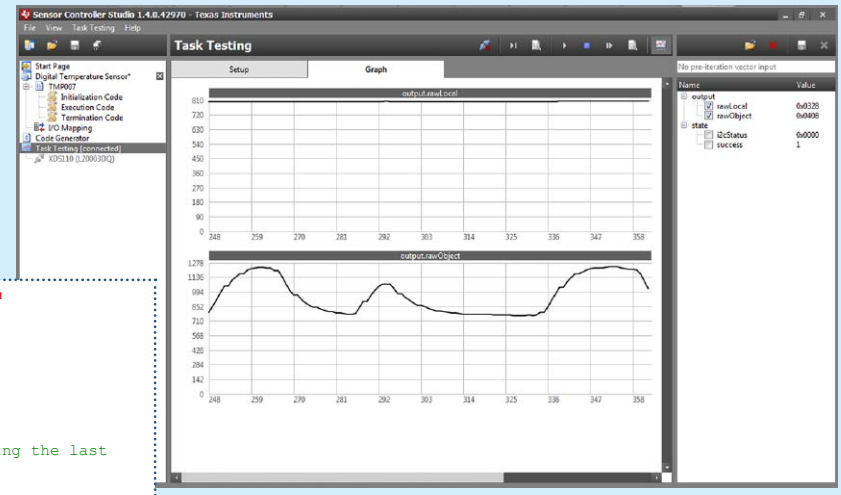
Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **97325**, pass: **6yfwxr8q**

bloku programu z kodem wykonywalnym *Execution* (wybieramy opcję *Run Execution Code*). Ustawienia konfiguracyjne programu pokazano na **rysunku 9**.

W kolejnym kroku, z menu programu z zakładki *Task Testing* wybieramy opcję *Connect*. Oprogramowanie nawiązuje połączenie z modułem startowym LaunchPad CC1310 i automatycznie przenosi nas do zakładki wizualizacji zmiennych *Graph*. W opcjach konfiguracyjnych zakładki *Graph* zaznaczamy zmienne globalne, które chcemy wizualizować. W naszym przypadku będzie to odczytana z czujnika TMP007 temperatura lokalna oraz temperatura obiektu. Następnie, wybieramy z menu programu *Task Testing* opcję *Run Initialization Code* – jest uruchamiane wykonanie kodu programu z bloku *Initialization*. W kolejnym kroku uruchamiamy cykliczne wykonywanie zdefiniowanej w zakładce *Setup* sekwencji testu. W tym celu, z menu programu *Task Testing* wybieramy opcję *Run Task Iteriations Continuously*. Oprogramowanie rozpoczyna cykliczny odczyt danych, a wynik pomiaru temperatury wizualizowany jest na wykresie oraz w oknie zmiennych programu. Zrzut ekranu z działania oprogramowania pokazano na **rysunku 10**.




**Rysunek 9. Sensor Controller Studio. Moduł testowania zadań – konfiguracja testów**




**Rysunek 10. Sensor Controller Studio. Moduł testowania zadań – wizualizacja zmiennych**

**Listing 1. Kod źródłowy programu sekcja *Execution***

```
U16 localHi    = 0;
U16 localLo    = 0;
U16 objectHi   = 0;
U16 objectLo = 0;

// start working with i2c
state.success = 1;

// If a measurement was successfully started during the last
execution ...
if (state.i2cStatus == 0x0000) {
    // Select the local temperature register
    i2cStart();
    i2cTx( I2C_OP_WRITE | (TMP007_I2C_ADDR << 1));
    i2cTx( TMP007_REG_ADDR_LOCAL_TEMP );
    // If successful ... read local temperature
    if (state.i2cStatus == 0x0000) {
        i2cRepeatedStart();
        i2cTx( I2C_OP_READ | (TMP007_I2C_ADDR << 1) );
        i2cRxAck(localHi);
        i2cRxNack(localLo);
    } else {
        state.success = 0;
    }
    i2cStop();
    // Select the object temperature register
    i2cStart();
    i2cTx( I2C_OP_WRITE | (TMP007_I2C_ADDR << 1));
    i2cTx( TMP007_REG_ADDR_OBJ_TEMP );
    // If successful ... read object temperature
    if (state.i2cStatus == 0x0000) {
        i2cRepeatedStart();
        i2cTx( I2C_OP_READ | (TMP007_I2C_ADDR << 1) );
        i2cRxAck(objectHi);
        i2cRxNack(objectLo);
    } else {
        state.success = 0;
    }
    i2cStop();
} else {
    i2cStop();
    state.success = 0;
}

if(state.success == 1) {
    //  scale local and object temperature ( divided by 4 )
    output.rawLocal = (localHi << 6) | (localLo >> 2);
    output.rawObject = (objectHi << 6) | (objectLo >> 2);
    // if result is negative add extra ones
    if ( localHi & 0x80) {
        output.rawLocal |= 0xC000;
    }
    if (objectHi & 0x80) {
        output.rawObject |= 0xC000;
    }
    // Notify the driver
    fwGenAlertInterrupt();
}

// Schedule the next execution
fwScheduleTask(1);
```

Wynik pomiaru temperatury odczytany z czujnika TMP007 jest prezentowany w formie binarnej. Żeby przekonwertować wynik na stopnie Celsjusza, należy odczytaną wartość pomnożyć przez współczynnik 0,03125.

Na zakończenie pracy z oprogramowaniem *Sensor Controller Studio* generujemy pliki sterownika SCIF. W tym celu, aby otworzyć okno kompilatora, z panelu projektu wybieramy opcję *Code Generator*. Jeśli projekt nie zawiera błędów, wybieramy opcję *Generate driver source code*. Oprogramowanie tworzy pliki wynikowe sterownika SCIF, które należy dołączyć do projektu w *Code Composer Studio*. Dodatkową funkcją oprogramowania *Sensor Controller Studio* jest możliwość emulowania pracy zdefiniowanych zadań. Opcje emulatora dostępne są w menu programu w zakładce *Task Testing*.

## Podsumowanie

Omówiliśmy działanie wbudowanego w mikrokontroler CC1310 kontrolera czujników. Korzystając z oprogramowania *Sensor Controller Studio* oprogramowaliśmy zadanie bezdotykowego pomiaru temperatury obiektu. W kolejnym odcinku kursu dołączymy pliki sterownika SCIF do projektu w *Code Composer Studio*, uruchomimy komunikację pomiędzy jednostką centralną mikrokontrolera CC1310, a kontrolerem czujników (odczyt wyniku pomiaru temperatury) oraz rozpoczniemy budowę bezprzewodowej sieci czujników pomiarowych.

**Łukasz Krysiewicz, EP**

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **97325**, pass: **6yfwxr8q**