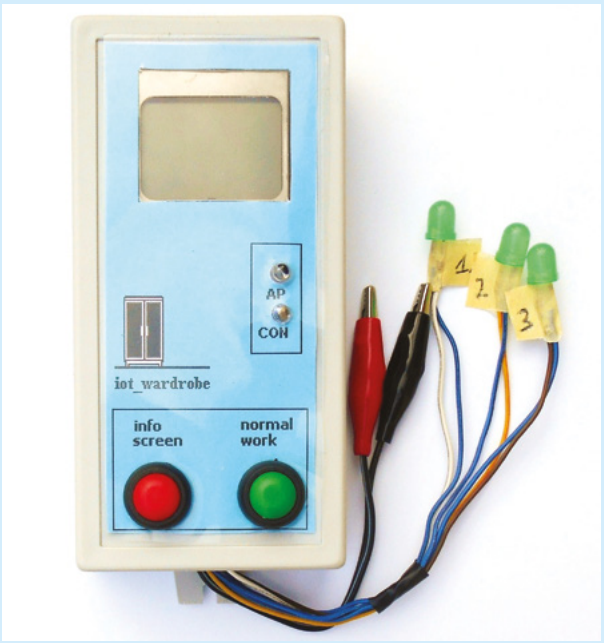
Internet Rzeczy w przykładach (3)

Sterownik inteligentnej szafy na ubrania. Połączenie CC3200 z siecią WWW

W tym artykule zaprezentujemy projekt sterownika inteligentnej szafy na ubrania. Sterownik połączy się z siecią WWW i z serwisu meteorologicznego odczyta prognozę pogody. Następnie na podstawie odczytanych danych zaproponuje użytkownikowi ubrania dopasowane do warunków pogodowych. Do budowy sterownika użyjemy omawianego podczas kursu modułu startowego CC3200 LaunchPad.

Do poprawnej pracy sterownik inteligentnej szafy na ubrania potrzebuje dostępu do sieci Wi-Fi (2.4 GHz standard IEEE 802.11 b/g/n). Należy zapewnić „widoczność" urządzenia z Access Point. Punktem dostępu do sieci może być router, komputer PC, telefon komórkowy. Parametry transmisji (nazwa SSID dla Access Point, szyfrowanie transmisji, ew. hasło dostępu do Access Point) ustawiamy w plikach konfiguracyjnych projektu.

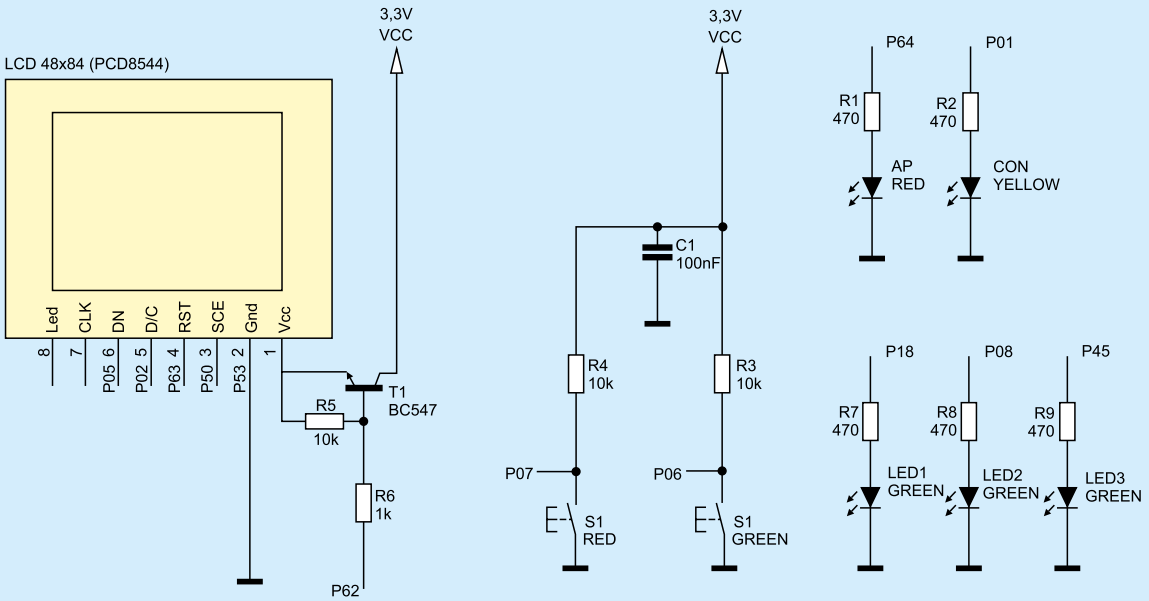


Fotografia 1. Sterownik inteligentnej szafy na ubrania

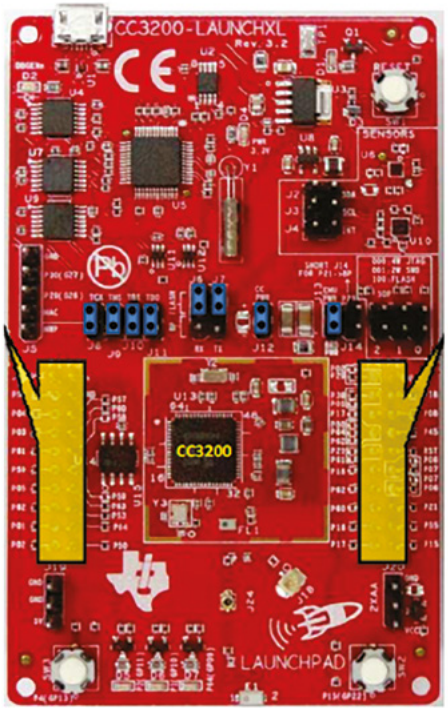
Budowa

Podstawowym elementem sterownika inteligentnej szafy na ubrania jest moduł startowy CC3200 LaunchPad. Poza modułem w sterowniku zamontowany został wyświetlacz monochromatyczny o rozdzielczości 48×84 pikseli (driver pcd8544), dwa przyciski funkcyjne (czerwony i zielony), dwie diody informacyjne (czerwona i żółta), trzy diody sygnalizacyjne (koloru zielonego). Elementy sterownika umieszczono w obudowie Z-52 o wymiarach 74 mm×146 mm×40 mm. Wygląd urządzenia pokazano na **fotografii 1**.

Pracą wyświetlacza, przycisków oraz diod steruje moduł CC3200 Launchpad. Schemat elektryczny



Rysunek 2. Schemat elektryczny urządzeń peryferyjnych sterownika



	P1		P3	
Ref	Signal	Dev Pin#	Dev Pin#	Signal
1	3.3V			5V
2	ADC_CH1	58		GND
3	UART0_RX	4	57	ADC_CH0
4	UART0_TX	3	60	ADC_CH3
5	GPIO	61	58*	ADC_CH1
6	ADC_CH2	59	59*	ADC_CH2
7	SPI_CLK	5	63	AUD_SYNC
8	GPIO	62	53	AUD_CLK
9	I2C_SCL	1	64	AUD_DOUT
10	I2C_SDA	2	50	AUD_DIN

P4		P2		
Signal	Dev Pin#	Dev Pin#	Signal	Ref
PWM	2*		GND	1
PWM	1*	18	GPIO	2
PWM	17*	8	SPI_CS	3
PWM	64*	45	GPIO	4
CCAP/GPIO	21*		RESET_OUT	5
CCAP/GPIO	18*	7	SPI_DOUT	6
GPIO	62*	6	SPI_DIN	7
GPIO	60*	21	GPIO	8
GPIO	16	55	GPIO	9
GPIO	17	15	GPIO	10

Rysunek 3. Modułu CC3200 LaunchPad. Gwiazdką * oznaczono linie sterujące wykorzystane w sterowniku

układów peryferyjnych sterownika pokazano na **rysunku 2**. Budowę sterownika rozpoczynamy od wyfrezowania w obudowie otworów na wyświetlacz, diody oraz przyciski. Następnie przygotowujemy front panel urządzenia i przyklejamy na obudowę (projekt w materiałach dodatkowych dołączonych do artykułu). W kolejnym kroku montujemy wyświetlacz (użyty został układ startowy modLCD1 z sygnałami sterującymi wyprowadzonymi na złącze goldpin). Montujemy diody oraz przyciski. Urządzenia peryferyjne z liniami sygnałowymi mikrokontrolera CC3200 łączymy przy pomocy przewodów połączeniowych. Jest to możliwe, ponieważ piny mikrokontrolera CC3200 zostały wyprowadzone na złącza 2x20 modułu LaunchPad. Opis wyprowadzeń pokazano na **rysunku 3**.

Moduł LaunchPad może być zasilany z portu USB komputera PC (debugowanie/emulowanie/transmisja UART) lub z zewnętrznego źródła zasilania (bateria / zasilacz). Na etapie rozwoju oprogramowania należy korzystać z zasilania z portu USB. W wersji produkcyjnej sterownik należy podłączyć do zasilacza (tryby oszczędzania energii i praca na baterii zostaną omówione w kolejnym artykule). Zasilacz dołączamy do złącza J20. Wartość napięcia zasilania ustawiamy na 3,3 V.

Funkcjonalność

Zadaniem sterownika inteligentnej szafy na ubrania jest zaproponować użytkownikowi elementy garderoby dopasowane do warunków atmosferycznych. W prezentowanej wersji sterownika oprogramowane zostały trzy elementy garderoby: parasol, szalik, okulary przeciwsłoneczne. Funkcjonalność urządzenia można rozbudować o kolejne elementy (kurtka, czapka, kalosze, itp.). Dodatkowo, sterownik może być używany jako stacja meteorologiczna.

Sterownik inteligentnej szafy na ubrania może pracować w jednym z trzech trybów: *normal work, info screen, check access point*. Podstawowym trybem pracy urządzenia jest tryb: *normal work* (aktywacja po naciśnięciu zielonego przycisku). W trybie tym sterownik łączy się z punktem dostępu do sieci (Access Point). Po uzyskaniu dostępu do sieci sterownik łączy się z serwerem meteorologicznym *openweathermap.org* i wysyła zapytanie o dane z prognozą pogody (usługa o nazwie: 5 day / 3h forecast). Serwer zwraca dane w formacie XML (nagłówek oraz prognoza pogody na 5 dni podawana z rozdzielczością 3 godzin). Jeśli wystąpi błąd połączenia, to jest włączana dioda funkcyjna CON (dioda koloru żółtego). W wypadku, gdy dane zostaną odczytane poprawnie praca urządzenia jest kontynuowane. Mikrokontroler CC3200 analizuje dane pogodowe. Z nagłówka odczytywana jest godzina wschodu i zachodu Słońca. Następnie odczytywana jest prognoza pogody na najbliższe 9 godzin (3 razy po 3 godziny). Każda 3 godzinna prognoza pogody poddawana jest szczegółowej analizie. Dla każdej 3 godzinnej prognozy pogody na ekranie wyświetlacza prezentowane są dwa ekrany z danymi pogodowymi (zjawiska atmosferyczne, siła wiatru, temperatura, ciśnienie, wilgotność, opady, zachmurzenie). Następnie uruchamiany jest algorytm wyboru garderoby. W przypadku gdy spełnione zostaną warunki wyboru garderoby (parasol – opady deszczu, szalik – temperatura poniżej 5°C, okulary przeciwsłoneczne – zachmurzenie poniżej

Oprogramowanie sterownika inteligentnej szafy na ubrania zostało stworzone w środowisku programistycznym CCSv6. Utworzony został projekt o nazwie *iot_wardrobe*. W projekcie jest obsługiwany system czasu rzeczywistego freeRTOS, framework SimpleLink, drivery dla CC3200. Konfiguracja projektu jest identyczna jak w przypadku projektu „zero" omawianego w poprzedniej części kursu. Dodatkowo zdefiniowany został symbol ccs (*Build-> ARM Compiler -> Advanced Opions -> Predefined Symbols*) wykorzystywany przy obsłudze uDMA, oraz dołączony został plik *startup_css.c* zawierający procedury startowe dla mikrokontrolera CC3200 (plik z lokalizacji \ *cc3200-sdk\example\common*).

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 64311, pass: 877yqakt

W sterowniku zainstalowano monochromatyczny wyświetlacza LCD o rozdzielczości 84×48 pikseli. Jest to wyświetlacza oparty o sterownik PCD8544 stosowany w popularnych telefonach Nokia 3310/5110. Zaletą wyświetlacza jest niska cena oraz łatwość obsługi. Komunikacja mikrokontrolera z wyświetlaczem odbywa się za pomocą programowego interfejsu SPI. Do sterowania zasilaniem wyświetlacza użyty został klucz TTL. Wyświetlacz może pracować w trybie tekstowym oraz graficznym. Oba tryby pracy wyświetlacza są wykorzystywane w projekcie. Grafiki użyte w projekcie wykonane zostały przy użyciu darmowego oprogramowania MicroLCD by ABXYZ.

10 %), to na 60 sekund włączane są diody informacyjne LED (diody koloru zielonego). Ustawienie diod informuje użytkownika, które elementy garderoby należy wybrać (np.: opady deszczu – weź parasol).

Tryb pracy *info screen* (aktywacja po naciśnięciu czerwonego przycisku) różni się od trybu pracy *normal work* tym, że nie jest „zestawiane" połączenie z Acccess Point i nie są pobierane dane z serwisu pogodowego. Mikrokontroler korzysta z danych odczytanych w trybie *normal work*. Prezentacja danych na wyświetlaczu oraz algorytm wyboru garderoby są identyczne jak w trybie *normal work*.

W trybie pracy *check access point* (aktywacja po starcie mikrokontrolera) sprawdzane jest połączenie sterownika z punktem dostępu do sieci Access Point. W przypadku braku połączenia z Access Point włączana jest dioda funkcyjna AP (dioda koloru czerwonego).

Przykładowe działanie sterownika w każdym z trzech trybów pracy pokazano na fotografii 3. W trybach pracy *normal work* i *info screen* ekrany z danymi meteorologicznymi wyświetlane są trzykrotnie (prognoza na 9 godzin - 3 razy po 3 godziny).

Oprogramowanie

Oprogramowania sterownika inteligentnej szafy na ubrania zostało stworzone w środowisku CCSv6. Projekt napisany został w języku programowania C. W projekcie użyty został system czasu rzeczywistego freeRTOS, framework SimpleLink, drivery dla CC3200. Dodatkowo wykorzystano przygotowane przez Texas Instruments interfejsy obsługi urządzeń peryferyjnych i sieci (uart, udma, timer, network). Pliki z kodem źródłowym interfejsów zostały dołączone do projektu (*cc3200-sdk\example\common*). W katalogu *source* umieszczono pliki z konfiguracją linii wejścia-wyjścia mikrokontrolera CC3200. Konfiguracja linii wejścia-wyjścia została wygenerowana przy użyciu oprogramowania *Pin Mux Tool* (opis

Listing 1. Tryb pracy normal work

```
void SystemNormalWorkTask()
{
    int     iSocketDesc;
    sInt16  apConnection;
    long ulStatus;
    unsigned long ulDestinationIP;

    ScreenWeatherForecast();
    g_wfBlink = 1;          // on
    g_wfParserStatus = 0;
    while(1)
    {
        // Connect to specific AP
        apConnection = Connect2AccessPoint();
        if(apConnection < 0)
        {
            g_wfBlink = 0;
            LedYellowOn();
            DBG_PRINT(„can't connect to %s AP",SSID_NAME);
            ScreenApError(SSID_NAME);
            break;
        }
        else
        {
            LedYellowOff();
            DBG_PRINT(„connected to %s AP",SSID_NAME);
        }
        // Get the serverhost IP address using the DNS lookup
        ulStatus = Network_IF_GetHostIP(WF_SERVER_NAME,&ulDestinationIP);
        if(ulStatus < 0)
        {
            LedYellowOn();
            DBG_PRINT(„DNS lookup failed. \n\r");
            break;
        }
        // Create a TCP connection to the server
        iSocketDesc = CreateConnection(ulDestinationIP);
        if(iSocketDesc < 0)
        {
            LedYellowOn();
            DBG_PRINT(„Socket creation failed.\n\r");
            break ;
        }
        WeatherForecastGet(iSocketDesc,acSendBuff,acRecvbuff);
        g_wfParserStatus = WeatherForecastParse(acRecvbuff);
        // Close the socket
        LedRedOff();
        LedYellowOff();
        close(iSocketDesc);
        DBG_PRINT(„\n\rSocket closed\n\r");
        break;
    }
    g_wfBlink = 0; // blink off
    // Stop the driver
    Network_IF_DeInitDriver();
    DBG_PRINT(„GET_WEATHER_FORECAST Completed\n\r");
    if(g_wfParserStatus)
       WardrobeWork(acRecvbuff);
}
```

Listing 2. Połącz CC3200 z Access Point

```
static sInt16 Connect2AccessPoint()
{
    // Reset The state of the machine
    Network_IF_ResetMCUStateMachine();
    //
    // Start the driver
    Network_IF_InitDriver(ROLE_STA);
    // Initialize AP security params
    SecurityParams.Key = (signed char *)SECURITY_KEY;
    SecurityParams.KeyLen = strlen(SECURITY_KEY);
    SecurityParams.Type = SECURITY_TYPE;
    // Connect to the Access Point
    return(Network_IF_ConnectAP(SSID_NAME,SecurityParams));
}
```

programu w poprzedniej części kursu). Plik projektu *Pin Mux Tool* (*project.pinmux*) dostępny jest w materiałach dodatkowych dołączonych do artykułu. W podkatalogach *device, hardware, system* umieszczone zostały pliki źródłowe oprogramowania. W katalogu *device* umieszczone zostały pliki do obsługi urządzeń peryferyjnych (wyświetlacz, diody, przyciski). W katalogu *hardware* pliki do obsługi modułów sprzętowych mikrokontrolera CC3200 (uart, timer). W katalogu *system* pliki do obsługi logiki pracy sterownika (konfiguracja urządzenia, obsługa sieci, analiza danych z prognozy pogody, wybór elementów garderoby itp.).

W oprogramowaniu uruchomiony został system czasu rzeczywistego freeRTOS. Utworzony został wątek o nazwie *system*, który monitoruje czy należy aktywować jeden z 3 trybów pracy urządzenia. Procedury do obsługi trybów pracy urządzenia oraz systemu operacyjnego umieszczono w pliku *system.c*. Na największą uwagę zasługuje sposób implementacji procedury do obsługi trybu pracy *normal work* (procedura *SystemNormalWorkTask*). Kod źródłowy procedury pokazano na **listingu 1**.

Na początku procedury wyświetlany jest ekran informacyjny *Weather Forecast*. (ikona prezentowana na ekranie pulsuje do momentu odczytu i analizy prognozy pogody). Następnie jest uruchamiana procedura podłączenia sterownika do *Access Point* (procedura *Connect2AccessPoint*). Kod źródłowy procedury pokazano na **listingu 2**. W przypadku błędu połączenia włączana jest dioda koloru żółtego.

Po podłączeniu do sieci pobierany jest adres IP serwera pogodowego (procedura *Network_IF_GetHost_IP*). Adres pobierany jest przy użyciu usługi *DNS lookup* (adres IP na podstawie nazwy serwera). W przypadku błędu podczas pobierania adresu IP jest włączana dioda koloru żółtego. Następnie jest tworzone połączenie TCP z serwerem meteorologicznym (procedura *CreateConnection*). W procedurze *CreateConnection* jest tworzone gniazdko TCP, a następnie jest zestawiane połączenie z serwerem. Kod źródłowy procedury pokazano na **listingu 3**. W przypadku błędu

Fotografia 4. Sterownik podczas pracy w trybach: a) check access point, b) normal work, c) info screen

Listing 3. Utwórz gniazdko TCP IPv4. Połącz z serwerem o podanym adresie IP

```
static int CreateConnection(unsigned long ulDestinationIP)
{
    int iLenorError;
    SlSockAddrIn_t  sAddr;
    int iAddrSize;
    int iSockIDorError = 0;
    sAddr.sin_family = SL_AF_INET;
    sAddr.sin_port = sl_Htons(80);
    //Change the DestinationIP endianity , to big endian
    sAddr.sin_addr.s_addr = sl_Htonl(ulDestinationIP);
    iAddrSize = sizeof(SlSockAddrIn_t);
    //Create TCP socket, IPv4
    iSockIDorError = sl_Socket(SL_AF_INET,SL_SOCK_STREAM, 0);
    if( iSockIDorError < 0 )
    {
        DBG_PRINT(„Error: Error Number = %d .\n\r”, iSockIDorError );
        return iSockIDorError;
    }
    //Connect with server
    iLenorError = sl_Connect(iSockIDorError, ( SlSockAddr_t *)&sAddr, iAddrSize);
    if( iLenorError < 0 )
    {
        // error
        DBG_PRINT(„Error: Error Number = %d.  \n\r”, iLenorError );
        return iLenorError;
    }
    DBG_PRINT(„Socket Id: %d was created.”,iSockIDorError);
    return iSockIDorError;//success, connection created
}
```

Podczas pracy sterownik inteligentnej szafy na ubrania wysyła komunikaty serwisowe (procedura DBG_PRINT). Komunikaty wysyłane są za pomocą transmisji UART. Żeby odebrać informacje wysyłane przez sterownik należy podłączyć moduł LaunchPad do portu USB komputera PC, a zworki JP6, JP7 ustawić w pozycji FLASH. Wówczas w systemie operacyjnym Windows pod nazwą CC3200LP Dual Port aktywowany zostanie port COM do obsługi modułu LaunchPad. Parametry transmisji UART to: 115200, 8, n, 1.

połączenie z serwerem włączana jest dioda koloru żółtego. W momencie, gdy sterownik zostanie podłączony do sieci i ma zestawione połączenie z serwerem meteorologicznym jest uruchamiana procedura odczytu prognozy pogody (*WeatherForecastGet*). Dane z serwera pogodowego *openweathermap.org* pobierane są zapytaniem HTTP metodą GET. Po odczytaniu prognozy pogody jest uruchamiana procedura analizy odebranych danych (*WeatherForecastParse*). Obie procedury do obsługi serwisu pogodowego umieszczono w pliku *weather.c*. Po odczytaniu prognozy pogody jest zrywane połączenie z serwerem meteorologicznym (zamknięcie gniazdka TCP procedura *close*). Następnie, sterownik jest odłączany od punktu dostępu do Internetu *Access Point* (procedura *Network_IF_DeInitDriver*). Na zakończenie działania trybu pracy *normal work* uruchamiana jest procedura wyboru garderoby (*WardrobeWork*). Kod źródłowy procedury umieszczono w pliku *wardrobe.c*.

Uruchomienie

Projekt sterownika inteligentnej szafy na ubrania dostępny jest w materiałach dodatkowych dołączonych do artykułu (folder *iot_wardrobe*). Kopiujemy katalog z projektem do lokalizacji *c:/ti/ep/*. Następnie, uruchamiamy oprogramowanie *Code Composer Studio* i importujemy projekt (*Project Import CCS Projects*). W kolejnym kroku zmieniamy ustawienia oprogramowania sterownika. W pliku konfiguracyjnym *configure.h* ustawiamy nazwę SSID dla Access Point, hasło dostępu do Access Point oraz algorytm szyfrowania transmisji danych. Dodatkowo wprowadzamy nazwę miasta, w którym pracuje sterownik. Jest to domyślna metoda lokalizacji sterownika. Inną metodą ustalenia lokalizacji sterownika jest podanie współrzędnych GPS. Żeby zmienić metodę lokalizacji sterownika z nazwy miasta na współrzędne GPS należy w opcjach projektu w zakładce *Predefined Symbols* z pola *Undefine NAME* usunąć wpis *LOCATION_GPS* a następnie do pola *Predefine NAME* dodać wpis *LOCATION_GPS*. Podanie lokalizacji sterownika jest niezbędne do poprawnego działania urządzenia (prognoza pogody jest odczytywana dla miejsca, w którym pracuje sterownik).

Żeby uruchomić projekt w trybie debugowania zworkę w złączu SOP ustawiamy w pozycji numer 1. Zworki J6, J7 ustawiamy w pozycji *FLASH* (komunikaty serwisowe UART). Następnie włączamy moduł LaunchPad do portu USB komputera PC. Kompilujemy projekt (*Project Build All*) i uruchamiamy debuger (*Run Debug*). Po wgraniu aplikacji uruchamiamy oprogramowanie (*Run Resume*). W trybie debugowania kod programu jest wgrywany do pamięci RAM mikrokontrolera CC3200. Żeby „trwale” zaprogramować sterownik, program należy wgrać do zewnętrznej pamięć S-Flash zamontowanej na płycie modułu LaunchPad.

Do zaprogramowania sterownika potrzebujemy pliku binarnego z oprogramowaniem. Wsad do zaprogramowania pamięci możemy wygenerować korzystając z środowiska CCSv6. W opcjach projektu w kategorii *Build* w zakładce *Steps* w polu *Post-build steps* dodajemy wypis: *„${CCE_INSTALL_ROOT}/utils/tiobj2bin/tiobj2bin” „${BuildArtifactFileName}” „${BuildArtifactFileBaseName}.bin” „${CG_TOOL_ROOT}/bin/armofd” „${CG_TOOL_ROOT}/bin/armhex” „${CCE_INSTALL_ROOT}/utils/tiobj2bin/mkhex4bin”*

Następnie kompilujemy projekt (*Project Build All*). W folderze *Debug* utworzony zostanie plik z oprogramowaniem (plik *iot_wardrobe.bin*). Konfigurujemy moduł LaunchPad w tryb programowania zewnętrznej pamięci S-Flash. Zworkę w złączu SOP ustawiamy w pozycji numer 2. Zworki J6, J7 ustawiamy w pozycji Flash. Dołączamy moduł LaunchPad do portu USB komputera PC. Żeby wgrać oprogramowanie uruchamiamy aplikację *CCS UniFlash* i postępujemy zgodnie z opisem podanym w poprzedniej części kursu. Po zakończeniu programowania usuwamy zworkę z złącza SOP. Teraz po włączeniu zasilania sterownika mikrokontroler CC3200 przekopiuje kod programu z zewnętrznej pamięci S-FLASH do wewnętrznej pamięci RAM i rozpocznie wykonywanie aplikacji.

Na zakończenie montujemy sterownik w szafie na ubrania. Diody informacyjne (parasol, szalik, okulary przeciwsłoneczne) umieszczamy przy odpowiadającym im częściach garderoby. Dołączamy sterownik do zasilania.

Podsumowanie

W kolejnym odcinku kursu zaprezentujemy projekt urządzenia pomiarowego pracującego w technologii Internet of Things. Sterownik wykona pomiary a wyniki będą przechowywane w „chmurze”. Do wizualizacji danych pomiarowych wykorzystany zostanie serwis plot.ly (wykresy liniowe, słupkowe, kołowe, itp.). Sterownik będzie zasilany z akumulatorów ładowanych z paneli słonecznych. Działanie urządzenia zostanie zoptymalizowane pod kątem poboru mocy.

Łukasz Krysiewicz, EP

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 64311, pass: 877yqakt