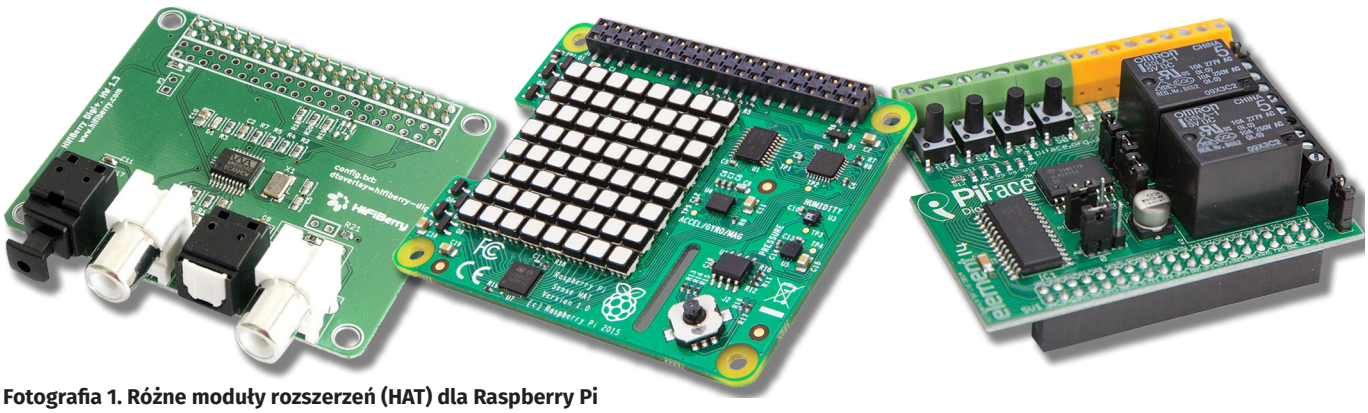
# Modularyzacja projektowania systemów elektronicznych

*W dziedzinie projektowania sprzętu elektronicznego nastąpiła cicha rewolucja. Wraz z postępem w układach krzemowych, które integrują coraz więcej funkcji w pojedynczej obudowie (zgodnie z prawem Moore'a), inżynierowie stopniowo odchodzą od tworzenia projektów opartych na pojedynczych komponentach i obwodach na rzecz korzystania z gotowych płytek, modułów i podsystemów. Dzięki temu udało się w ogromnym stopniu zwiększyć efektywność projektowania elektroniki. Obecnie podobną zmianę możemy zaobserwować w oprogramowaniu, gdzie programiści coraz chętniej korzystają z gotowych modułów, zamiast polegać głównie na samodzielnie napisanym kodzie.*

Z przejściem na modułową architekturę wiąże się wiele korzyści. Jedną z nich jest możliwość łatwiejszego wykorzystania efektu skali, który staje się dostępny dzięki użyciu platform popularnych wśród wielu klientów. Użytkownicy systemów przemysłowych mają bogate doświadczenia ze sprzętem modułowym. Standardy takie jak VME (*Versa Module Eurocard*) i CompactPCI pozwoliły tak integratorom, jak i producentom OEM (*Original Equipment Manufacturer*), skorzystać z wysoce wydajnych systemów obliczeniowych, pomimo działania na rynku o małych woluminach produkcyjnych. Inżynierzy mogli w większym stopniu dostosowywać funkcje komputera do potrzeb klienta, bez konieczności inwestowania czasu i wysiłków w projektowanie bardzo zaawansowanych płytek drukowanych. Od tego czasu, zgodnie z prawem Moore'a, funkcje komputerów znacząco rozrosły się, a jednocześnie zmalał koszt pojedynczych części i modułów. Kluczowym przykładem jest tu komputer jednopłytkowy Raspberry Pi.

## Niedrogi, gotowy sprzęt

Dzięki wykorzystaniu ekonomii skali, jaka wynika z użycia platformy SoC (System on a Chip), przygotowanej z myślą o smartfonach, konsorcjum, które stoi za Raspberry Pi, było w stanie dostarczyć znacznie bardziej wydajny produkt, niż gdyby bazować na projekcie opracowanym pod kątem zastosowań edukacyjnych. Jednorazowe koszty inżynieryjne, które wiążą się z produkcją układu krzemowego, zostały



**Fotografia 1. Różne moduły rozszerzeń (HAT) dla Raspberry Pi**

poniesione przez pierwotny, docelowy rynek, dzięki czemu to, co uzyskali użytkownicy Raspberry Pi, miało znacznie większą wartość, w stosunku do końcowej ceny produktu. Przewaga kosztowa została następnie przeniesiona na rynek przemysłowy. Integratorzy i producenci OEM skorzystali z modułowości platformy Raspberry Pi, używając szyny rozszerzeń HAT, by dobudowywać swoje własne moduły interfejsowe.

Zastosowanie modułów do Raspberry Pi pozwala uwolnić zespoły inżynierów od konieczności osobnego zamawiania podobnych komponentów i samodzielnego uwzględniania ich w projektach własnych płytek drukowanych. Ma to duże znaczenie, gdyż tego typu czynności zazwyczaj wymagają bardziej czasochłonnych testów pod kątem integralności sygnałów i poprawności realizowanych funkcji, niż gdy buduje się jedynie moduł HAT. Bardzo często okazuje się, że w takim własnym module wystarczy zastosować względnie proste, dwu- lub czterowarstwowe płytki PCB.

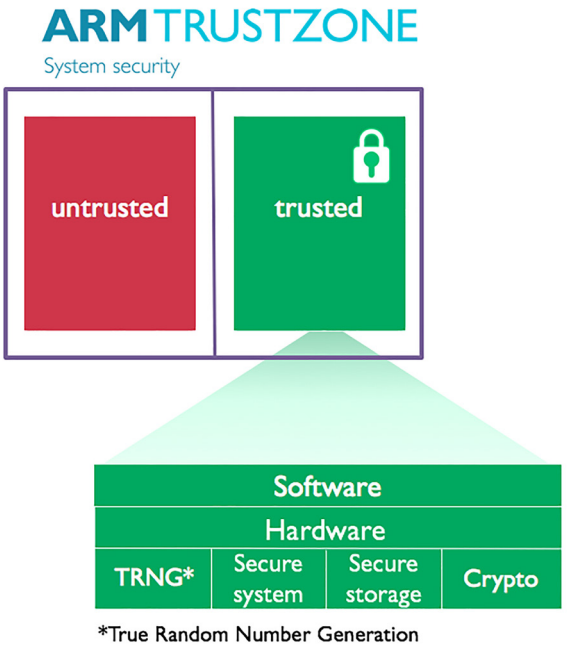
## Gotowe moduły programowe

W ostatnim czasie pojawił się podobny trend, ale w odniesieniu do modularyzacji oprogramowania. Inżynierowie mogą się teraz skoncentrować wyłącznie na właściwych elementach aplikacji, w której dodają własną wartość. Trend ten jest napędzany nie tylko ekonomią skali i możliwością amortyzacji jednorazowych kosztów inżynieryjnych przez niektórych dostawców, ale również rosnącą popularnością integracji sieciowej i opierania modeli biznesowych na usługach. System wbudowany trudno w dzisiejszych czasach nazwać kompletnym, jeśli nie stanowi części większego systemu, chociażby w ramach Internetu Rzeczy (IoT – *Internet of Things*). W takim otoczeniu urządzenie może być użyte, by pomóc dostarczać jedną lub więcej usług, z czego wiele funkcji będzie można zmieniać w trakcie życia sprzętu, na którym usługi te są realizowane. Takie połączenie IoT i chmury skutkuje pojawieniem się nowych modeli biznesowych, które opierają się na opisanych możliwościach – za przykłady mogą posłużyć firmy oferujące oprogramowanie jako usługę (SaaS – *Software As A Service*) i modele biznesowe, w którym płaci się za użytkowanie urządzenia, w trakcie jego działania, zamiast nabywać urządzenie na własność. Elastyczność stała się kluczowym kryterium w tym komercyjnym środowisku i popycha inżynierów w kierunku poszukiwań bardziej modularnych struktur.

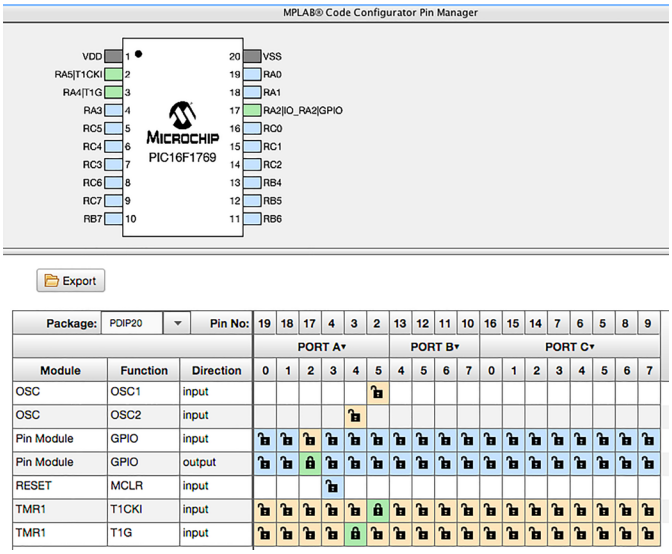
Modularność zaczyna się już w systemie operacyjnym. To właśnie system operacyjny zawiera warstwy abstrakcji, które są niezbędne, by budować elastyczne, modułowe środowiska. Zazwyczaj system operacyjny dostarcza szereg usług, które rozpoczynają się na prostym zapisie czy odczycie, a kończą na pełnych stosach protokołów, do których dostęp odbywa się z użyciem dobrze udokumentowanych interfejsów programistycznych (API – *Application Programming Interface*). Tak długo, jak realizowane usługi wspierają zdefiniowane API, kod, który za nie odpowiada, może zmieniać się dowolnie, bez wpływu na aplikacje, które z tego API korzystają. Sprawdza się to zarówno w prostych systemach czasu rzeczywistego, takich jak FreeRTOS


**Fotografia 2. SmartEdge Agile firmy Avnet to certyfikowane rozwiązanie sprzętowe zapewniające łatwą realizację sztucznej inteligencji i innych złożonych procesów programistycznych**

[1], dostarczanych z różnymi narzędziami deweloperskimi dla mikrokontrolerów, jak też w komercyjnych i bardziej złożonych implementacjach systemów czasu rzeczywistego, takich jak np. VxWorks [2] firmy Wind River. To właśnie VxWorks stanowi obecnie przemysłowy standard wbudowanych systemów operacyjnych, kontrolując pracę niektórych z najbardziej krytycznych elementów infrastruktury i innych urządzeń.




**Rysunek 1. W dużej liczbie układów z rdzeniem ARM jest dostępny mechanizm TrustZone**




**Rysunek 2. Code Configurator firmy Microchip pozwala w łatwy sposób generować gotowe bloki programu**

Linux i inne systemy operacyjne mogą uczynić zarządzanie pamięcią jeszcze bardziej zaawansowanym, pozwalając na izolowanie od siebie poszczególnych zadań. Jednym z potencjalnych problemów z prostymi systemami RTOS (systemami operacyjnymi czasu rzeczywistego – *Real Time Operating System*) jest fakt, że pracują one w całkowicie niepodzielonej przestrzeni pamięci. Błędy lub szkodliwe działanie jednego z zadań może prowadzić do przypadkowego lub złośliwego nadpisania danych czy kodu innego zadania, skutkując zawieszeniem się systemu albo innymi niepożądanymi rezultatami. Linux korzysta z wirtualnego adresowania, w którym pośredniczy sprzętowa jednostka zarządzania pamięcią, by uniemożliwić zadaniom sięganie do obcych obszarów pamięci. Procesy mogą ze sobą wchodzić w interakcje wyłącznie poprzez API systemu operacyjnego lub z użyciem innych protokołów komunikacyjnych, opracowanych na bazie tych API.

Wirtualne adresowanie pamięci nie jest wymaganiem koniecznym do zapewnienia izolacji realizowanych zadań. Niektóre architektury mikrokontrolerów, a w tym część układów z rodzin ARM Cortex-M i Cortex-R, mogą wymusić ochronę pamięci, przy zachowaniu jej płaskiej struktury. ARM, w niemałej liczbie swoich układów, udostępnia mechanizm TrustZone, który wprowadza oprogramowanie do trybu bezpiecznej pracy, umożliwiając odizolowanie wrażliwych programów od zadań uruchamianych przez użytkownika. Stosując tę ochronę, łatwiej jest połączyć własny kod z rosnącą liczbą gotowych modułów programowych, opracowanych, by obsługiwać poszczególne, często realizowane zadania.

## Integracja otwartego kodu źródłowego z funkcjami własnościowymi

Obecnie inżynierowie mają dostęp do szerokiego wyboru bezpłatnych, otwartych modułów i stosów protokołów, publikowanych w serwisach GitHub, SourceForge [3] i innych. Można również znaleźć komercyjne stosy protokołów, oferowane z lepszym wsparciem i dodatkowymi funkcjami lub z certyfikatami, cennymi z uwagi na aplikacje krytyczne, głównie z punktu widzenia bezpieczeństwa [5]. Projekty referencyjne, złożone przez producentów układów scalonych, często łączą funkcje dostępne na licencjach *open source* z funkcjami własnościowymi, by ułatwić klientom pracę, począwszy od budowania prototypów, aż po ostateczne, pełne implementacje produktów. W niektórych przypadkach projekty referencyjne to tak naprawdę już pełne aplikacje, które użytkownik może sobie dostosować do własnych potrzeb.

Niektórzy projektanci systemów korzystają z rosnącej modułowości oprogramowania, aby budować środowiska rozwojowe o precyzyjnie dobranych parametrach i automatycznie generują kod. Narzędzia te nierzadko korzystają z blokowych reprezentacji oprogramowania, łączonych ze sobą przez programistę, w ramach graficznego interfejsu użytkownika. Jednym z takich przykładów jest MPLAB Code Configurator firmy Microchip, przeznaczony do pracy z mikrokontrolerami rodzin PIC8, PIC16 i PIC32.

Zaawansowane aplikacje, takie jak uczenie maszynowe i przetwarzanie obrazów, to dobre przykłady obszarów, w których użytkownicy mogą skorzystać z jednorazowych inwestycji, jakie ponieśli specjaliści, i w ten sposób uniknąć lat prac rozwojowych, które normalnie byłyby potrzebne na przygotowanie danego oprogramowania od zera. Caffe, PyTorch i opracowany przez Google Tensorflow umożliwiają budowanie, uczenie i strojenie złożonych modeli sztucznej inteligencji, które następnie można z łatwością uwzględnić w procesach przetwarzania danych w systemach wbudowanych. Wraz ze wzrostem popularności uczenia maszynowego rośnie też powszechność stosowania bibliotek OpenCV do wstępnego przetworzenia danych obrazów, zanim zostaną one przekazane do modelu AI, zbudowanego z użyciem środowisk Caffe lub Tensorflow. W aplikacjach tego typu kod napisany przez użytkownika jest używany przede wszystkim do generowania odpowiedzi na zdarzenia, wykrywane przez modele.

## Jak to wszystko połączyć ze sobą?

Projektanci mają teraz dostęp do modułów programowych i narzędzi opartych na wykorzystaniu chmury, które z łatwością można zintegrować z popularnymi stosami protokołów i implementacjami systemów operacyjnych czasu rzeczywistego. Pozwala to na tworzenie systemów wbudowanych o różnych stopniach złożoności i integrowanie ich z systemami IoT. Za przykład może posłużyć platforma IoT Connect [5] firmy Avnet, która udostępnia przetwarzanie danych w chmurze na potrzeby takich złożonych zadań jak np. algorytmy sztucznej inteligencji. Cechy całego systemu są określane zarówno przez infrastrukturę chmurową, jak i przez usługi programowe, realizowane wewnątrz urządzenia wbudowanego. Z tego względu dostawcy usług chmurowych, tacy jak Amazon Web Services i Microsoft Azure, zapewniają szereg rozwiązań, które pozwalają połączyć oba światy – a wszystko to bazuje na modularności używanych komponentów programowych.

Modularyzacja zmienia niezbędny zestaw umiejętności, potrzebnych inżynierom zajmującym się programowaniem systemów wbudowanych. Ich zakres obowiązków odchodzi od pisania kodu w stronę zdolności do tworzenia elastycznych architektur, bazujących na istniejących już modułach, co prowadzi do łatwego dobudowywania własnego kodu i konfiguracji aplikacji wraz z wprowadzaniem nowych usług. Okiełznanie tej modularności pozwoli integratorom i producentom OEM na nadążenie za popytem ze strony klientów, którego nie dałoby się zaspokoić, stosując tradycyjne sposoby pracy.

**Fotografia 3. Cliff Ortmeyer**

**Cliff Ortmeyer**
**Global Head of Technical Marketing, Farnell**

Przypisy:
[1] https://bit.ly/37haNyo
[2] https://bit.ly/375OwDy
[3] https://bit.ly/37aeeah
[4] https://bit.ly/3690H3s
[5] https://bit.ly/37bByo6