1. Wprowadzenie

1.1. Wstęp

We współczesnym świecie trudno wyobrazić sobie dowolne urządzenie, które byłoby pozbawione bogatego interfejsu użytkownika, a często również graficznego wyświetlacza z panelem dotykowym. Możliwe jest to dzięki zastosowaniu mikroprocesorów lub systemów mikroprocesorowych kontrolujących pracę danego urządzania umieszczonych w jego wnętrzu, stąd takie systemy są często nazywane systemami wbudowanymi (ang. *embedded system*). Takie rozwiązanie spotyka się obecnie prawie w każdym sprzęcie elektronicznym lub urządzeniach zawierających sterownik elektroniczny, np. urządzeniach RTV/AGD, inteligentnych domach, sterownikach przemysłowych, samochodach czy sprzęcie medycznym. Pierwsze mikroprocesory powstały w latach 70. XX wieku, umożliwiały wykonywanie tylko najprostszych zadań i zawierały małą ilość zasobów, a ich programowanie było możliwe jedynie w asemblerze. Wraz z rozwojem techniki mikroprocesorowej coraz popularniejszy stawał się język C częściowo uniezależniający program od zastosowanego rodzaju mikroukładu. Dalszy rozwój mikroprocesorów i mikrokontrolerów spowodował coraz częstsze stosowanie systemów operacyjnych czy instalowanie systemu Linux. Tworzenie oprogramowania pod systemem Linux ma wiele zalet: wygodną obsługę zaawansowanych aplikacji sieciowych i graficznych, ale jednocześnie wymaga dużych zasobów sprzętowych, a najprostsze przykłady wykorzystania linii GPIO są stosunkowo trudne i potrzebują specjalnych sterowników do kernela. Równolegle w świecie informatyki ogromną popularność zyskiwała platforma .NET Framework dzięki swojej uniwersalności oraz niezależności od sprzętu a nawet języka programowania. Rozwój ten spowodował, że obecnie znajomość technologii .NET i języka C# jest obowiązkiem wśród programistów, którzy tworzą aplikacje wykorzystywane w komputerach, serwerach, tabletach czy na stronach internetowych. Sporym postępem było stworzenie przez firmę Microsoft uproszczonej wersji technologii .NET nazwanej .NET Micro Framework przeznaczonej dla systemów wbudowanych, co pozwala coraz wygodniej sprostać zaawansowanym aplikacjom graficznym. Główną ideą stosowania platformy .NET Micro Framework w systemach wbudowanych jest tworzenie zaawansowanych aplikacji, a jednocześnie wydajnych i bezpiecznych przy zaoszczędzeniu włożonej pracy przez programistę. Idea ta pozwala porównać programy tworzone dla systemów wbudowanych opartych na mikroprocesorach lub mikrokontrolerach z programami tworzonymi dla komputerów przy jednoczesnym pozostawieniu pełnej funkcjonalności mikroukładów. Wbudowane biblioteki .NETMF mają zintegrowaną obsługę peryferii wewnętrznych i wielu peryferii zewnętrznych. Przykładowo obsługa kolorowego, graficznego wyświetlacza TFT z panelem dotykowym sprowadza się do prostej aplikacji okienkowej (analogia do *Windows Application* w przypadku programów komputerowych) wraz z zastosowaniem pełnej techniki obiektowej. W takim przypadku programista jest zwolniony z konieczności uruchamiania całego procesu komunikacji mikrokontrolera z wyświetlaczem i panelem dotykowym jak również ręcznego implementowania jego obsługi, co zapewnia platforma .NETMF z pakietem WPF. Jednocześnie tworzona aplikacja jest bezpieczna dzięki wbudowanej obsłudze wyjątków oraz kontroli procesora i pamięci przez platformę jak też wielu zabezpieczeniom w przypadku oprogramowywania modelu TCP/IP. Tworzone aplikacje mogą

być w wygodny sposób debugowane. Omawiana technika programowania systemów wbudowanych w .NET Micro Framework z jednej strony umożliwia szybkie tworzenie zaawansowanych aplikacji a z drugiej strony daje szybki i wygodny dostęp do części sprzętowej (np. linii GPIO), dlatego można powiedzieć, że jest rozwiązaniem pośrednim pomiędzy ręcznym tworzeniem aplikacji w języku C a instalacją systemu Linux Embedded.

Niniejsza książka poświęcona jest tematyce tworzenia aplikacji dla systemów wbudowanych opierając się na mikrokontrolerach STM32 z rdzeniem Cortex-M4, za pośrednictwem .NET Micro Framework w obiektowym języku C#. Głównym założeniem książki jest łagodne a zarazem solidne wprowadzenie Czytelnika w poruszaną tematykę poprzez dwadzieścia sześć praktycznych przykładów zróżnicowanych od najprostszych zagadnień po nieco bardziej zaawansowane. Niniejsza książka jest skierowana do profesjonalistów elektroników i informatyków jak również szerokiego grona hobbystów chcących poznać najnowszą technikę programowania mikrokontrolerów STM32. Dla zawodowych programistów znających technologię .NET oraz język C# książka może stanowić zestawienie możliwości .NET Micro Framework, będącej okrojoną i zoptymalizowaną wersją macierzystej platformy .NET. Dzięki solidnemu przedstawieniu podstaw oraz na bazie wielu praktycznych przykładów książka może stanowić dobry wstęp do techniki programowania systemów wbudowanych w C# .NET Micro Framework.

1.2. Dlaczego STM32F4?

Mikrokontrolery STM32 produkowane przez firmę STMicroelectronics w ostatnim czasie zyskują coraz większą popularność na rynku światowym i polskim. Szybki rozwój nowoczesnej serii mikrokontrolerów z rdzeniem Cortex opartych na architekturze ARM sprawia, że znajomość tej rodziny staje się nieodzownym obowiązkiem wśród konstruktorów oraz programistów. Dzięki coraz nowocześniejszej architekturze i bardzo bogatemu wyposażeniu w wewnętrzne układy peryferyjne mikrokontrolery te zyskują coraz szersze spektrum zastosowań. Oprócz popularnej rodziny mikrokontrolerów STM32 z rdzeniem Cortex-M3, najprostszej STM32F0 z rdzeniem Cortex-M0, powstała najnowsza seria STM32F4, w której STMicroelectronics wprowadził pierwszy raz najbardziej zaawansowany rdzeń Cortex-M4. Największymi zaletami nowej serii jest duża częstotliwość taktowania sięgająca nawet 180 MHz, duża ilość pamięci (do 2 MB pamięci Flash oraz do 256 kB pamięci RAM) i bardzo bogate wyposażenie w układy peryferyjne oraz interfejsy komunikacyjne. Dzięki zastosowaniu sprzętowego modułu FPU (*Floating Point Unit*) dokonującego szybkich obliczeń na liczbach zmiennoprzecinkowych oraz wielu inklinacjom do procesorów sygnałowych DSP nowa rodzina świetnie nadaje się do cyfrowego przetwarzania sygnałów. Wbudowane rozwiązania do obsługi graficznych wyświetlaczy TFT pozwalają na tworzenie coraz bardziej zaawansowanych interfejsów graficznych z panelem dotykowym będących coraz popularniejszymi wśród systemów wbudowanych. Również w aspekcie pracy z siecią Ethernet mikrokontrolery STM32F4 nie pozostają w tyle dzięki wprowadzeniu interfejsu MAC Ethernet obsługującego protokół synchronizacji czasu IEEE1588 w nowej

wersji v2. Najnowsza seria mikrokontrolerów STM32 znajduje zastosowanie w automatyce przemysłowej, medycynie oraz wielu innych dziedzinach.

1.3. Dlaczego .NET Micro Framework?

Wraz z rozwojem nowych mikrokontrolerów rośnie zapotrzebowanie na coraz bardziej zaawansowane interfejsy graficzne GUI zawierające okna, menu czy ikony, które w istocie są oparte na dość zawiłych aplikacjach. W świecie prężnie rozwijającej się techniki obiektowej popularne kiedyś asemblery nie są już wykorzystywane w najnowszych mikrokontrolerach. W ich miejsce wykorzystuje się języki wysokiego poziomu, coraz częściej będące językami obiektowymi oraz systemy operacyjne takie jak FreeRTOS oraz ISIX-RTOS. Rozwijana przez wiele lat pod systemem Windows technika programowania komputerów spowodowała wprowadzenie platformy .NET Framework, której zadaniem jest zarządzanie systemem i odciążanie programisty od schodzenia do operacji niskopoziomowych. Dodatkowymi zaletami .NET jest możliwość tworzenia oprogramowania w wielu różnych językach, gdyż wszystkie są tłumaczone na język kodu pośredniego, jakim jest CIL. W czasie pisania tej książki znajomość platformy .NET jest obowiązkiem wśród programistów, którzy tworzą aplikacje dla komputerów, serwerów, stron internetowych i urządzeń mobilnych. Błyskawiczny rozwój nowej techniki opracowanej przez Microsoft musiał znaleźć odzwierciedlenie w programowaniu systemów wbudowanych, co miało miejsce wraz z przedstawieniem .NET Micro Framework, przystosowanym do programowania systemów wbudowanych. Platforma .NETMF jest silnie okrojoną wersją popularnej platformy .NET, przy czym nie wymaga systemu operacyjnego i może samodzielnie zarządzać mikrokontrolerem. Zapewnia bezpieczeństwo podczas uruchamiania programów, zarządza pamięcią i układami peryferyjnymi. Dodatkowo biblioteki .NETMF zawierają przygotowane przez producenta biblioteki do obsługi praktycznie wszystkich peryferii wewnętrznych oraz wielu interfejsów.

1.4. Dlaczego język C#?

Chociaż platforma .NET Framework jest wielojęzyczna i wszystkie programy są tłumaczone na język kodu pośredniego CIL, Microsoft przygotował specjalny język C# stworzony z myślą o programistach .NET. Jako że język C# jest hybrydą wielu języków, jest składniowo najczystszym językiem, jednocześnie prawie tak samo prostym jak Visual Basic i zapewnia te same możliwości co C++, bez jego zawiłych konstrukcji. Dodatkowo C# jest językiem w pełni obiektowym. Programiści C/C++ mogą w swobodny sposób poruszać się w C#, ponieważ ma on bardzo podobną składnię. Sympatycy Javy oraz Visual Basica znajdą w C# również wiele podobnych rozwiązań.

Podczas korzystania z C# nie ma konieczności stosowania wskaźników, chociaż można zejść do takiego poziomu. Hybryda Microsoftu nie zawiera słowa kluczowego *delete*, ponieważ automatycznie zarządza pamięcią za pomocą odśmiecania. Język C# jest w pełni zintegrowany z platformą .NET i jest najlepszym rozwiązaniem do programowania z jej wykorzystaniem.