GUI dla mikrokontrolerów

Na przestrzeni ostatniego dziesięciolecia prawie wszystkie platformy oferujące możliwość projektowania aplikacji z graficznym interfejsem użytkownika rozpieściły nas, programistów, do granic możliwości. Niezależnie od tego, czy piszemy aplikację desktopową, webową czy mobilną, możemy przebierać w różnych frameworkach, które dostarczają bogate zestawy gotowych kontrolek i w wielu przypadkach dają nam prawie nieograniczone możliwości w zakresie ich stylowania. Istnieją jednak środowiska, w których sytuacja nie wygląda wcale tak różowo.

ŚWIAT EMBEDDED

Wyjątkiem od wspomnianej reguły jest świat oprogramowania wbudowanego: nie znajdziemy tu dużych, okrzepniętych, rozbudowanych i wygodnych bibliotek na miarę Windows Forms czy WPF. Przyczyna tego stanu rzeczy jest stosunkowo prosta: środowisko to jest po prostu zbyt zróżnicowane.

Po pierwsze, mamy do czynienia z całym wachlarzem różnych mikrokontrolerów. Weźmy na przykład znane pewnie wszystkim Arduino Uno. Sercem tego układu jest 8-bitowy mikrokontroler o nazwie ATmega328 taktowany z częstotliwością 16 Mhz. Programy pisane na tę płytkę muszą zmieścić się w 32 kB pamięci flash, a podczas działania korzystać mogą z 2 kB pamięci operacyjnej SRAM. Do dyspozycji mamy 14 cyfrowych wejść/wyjść oraz 6 wejść wbudowanego przetwornika analogowo-cyfrowego.

Rysunek 1. Arduino Uno

Inną powszechnie dostępną płytką jest STM32F746NG. Tym razem na pokładzie mamy mikrokontroler ARM Cortex M7 taktowany z częstotliwością do 216 Mhz. Na programy – mogące korzystać z 340 kB SRAM – dostajemy cały 1 MB pamięci flash. Przy czym mowa tu o samym chipie, bo na płytce znajdziemy dodatkowe 128 MB pamięci flash i 128 MB SDRAM. Oprócz tego w zestawie dostajemy pojemnościowy, kolorowy ekran LCD o przekątnej 4,3″ i rozdzielczości 480x272 px. Znajdziemy tam również złącza MicroSD, microUSB OTG, ethernet, wejście i wyjście audio, złącze kamery, wyjście głośnika stereo oraz dwa mikrofony.

Rysunek 2. Płytka rozwojowa STM32F746NG

Widać chyba wyraźnie, jak bardzo przytoczone układy różnią się od siebie możliwościami obliczeniowymi, dostępną pamięcią i rodzajami wejść oraz wyjść. Różnice te stanowią jednak tylko część problemu – weźmy bowiem na warsztat same wyświetlacze. W ofercie sklepów elektronicznych znajdziemy na przykład malutki, monochromatyczny ekranik OLED o przekątnej 0,66″ i rozdzielczości 64x48 px. Ale zaraz obok widnieje opcja zakupu dotykowego wyświetlacza 3.5″ o rozdzielczości 320x480 albo 4.3″ modułu E-paper o rozdzielczości 800x600 px.

Idąc dalej, różne platformy w odmienny sposób realizują interakcję z użytkownikiem. W najlepszym przypadku możemy korzystać z ekranu dotykowego, co oczywiście rozwiązuje wiele problemów. Innym razem jednak możemy mieć do dyspozycji tylko urządzenie wskazujące (np. mysz), enkoder (pokrętło z możliwością wciskania) albo zestaw kilku przycisków (w skrajnych przypadkach – jeden). Zbudowanie uniwersalnego rozwiązania, które będzie działało z każdą kombinacją mikrokontrolera, ekranu i fizycznego interfejsu, graniczy z niemożliwością.

EMBEDDED GUI

Najprostszym sposobem jest oczywiście narysowanie interfejsu ręcznie i w wielu przypadkach jest to w zupełności wystarczające. Dla przykładu, jakiś czas temu złożyłem sobie małą, wielofunkcyjną płytkę wyposażoną między innymi w moduł pogodowy. Interfejs graficzny zaprojektowałem tam samodzielnie od zera i nie zajęło mi to wprawdzie dużo czasu, ale był on naprawdę bardzo prosty: składał się zaledwie z kilku histogramów, ramek i przycisków. Ponadto cała