# Podstawy programowania STM32F746G-DISCO

## Jak zbudować oscyloskop z FFT z użyciem STM32F746G-DISCO

**Zaczynając pisać oprogramowanie dla systemu wbudowanego zawsze warto zwrócić uwagę na trzy czynniki mające znaczący wpływ na jakość i czas powstawania projektu. Są to: dostępność sprzętowych zestawów deweloperskich, bibliotek, oraz narzędzi programistycznych. Zestawy deweloperskie zawierające dodatkowe peryferia umożliwiają szybkie tworzenie prototypów bez konieczności projektowania dodatkowego sprzętu. Zwykle dostarczane są wraz z oprogramowaniem, zawierającym konfigurację i sterowniki (BSP Board Support Package). Dostępne biblioteki pozwalające m. in, na tworzenie interfejsów graficznych, lub implementujące protokoły komunikacyjne umożliwiają skupienie się na właściwej aplikacji, która dzięki systemom operacyjnym czasu rzeczywistego może stać się bardziej stabilna i łatwiejsza w utrzymaniu – zwłaszcza, gdy pracuje nad nią cały zespół. Na koniec, narzędzia programistyczne, zintegrowane w całe środowiska ułatwiają budowanie projektów bez potrzeby żmudnej konfiguracji wszystkich elementów systemu.**

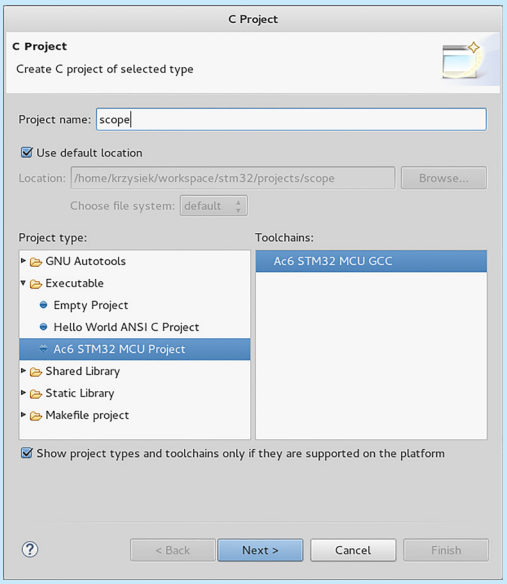
W artykule zostanie zaprezentowane środowisko łączące w sobie wszystkie wymienione powyżej elementy. Jako platforma sprzętowa posłuży zestaw STM32F746G-DISCOVERY zawierający m. in. mikrokontroler STM32F746NGH6 oraz wyświetlacz o rozdzielczości 480×272 pikseli z panelem dotykowym. Do stworzenia aplikacji wykorzystane zostaną biblioteki STemWin i CMSIS DSP oraz system operacyjny FreeRTOS. Środowiskiem programistycznym do utworzenia projektu będzie zbudowane na programie Eclipse, SW4STM32. Środowisko to ma w sobie wsparcie zarówno dla wszystkich zestawów Discovery oraz Nucleo, jak i wielu bibliotek programistycznych, przez co umożliwia błyskawiczne przygotowanie projektu i rozpoczęcie pracy nad docelową aplikacją.

Jako przykład posłuży aplikacja pozwalająca na odbiór sygnału z liniowego wejścia audio oraz wyświetlenie go na ekranie wraz z jego widmem amplitudowym. Użytkownik będzie mógł przesuwać oraz skalować oba wykresy za pośrednictwem panelu dotykowego. Kod źródłowy aplikacji został zbudowany na bazie dwóch przykładów znajdujących się w bibliotece STM32CubeF7: **FreeRTOS_DelayUntil** oraz **StemWin_HelloWorld**.

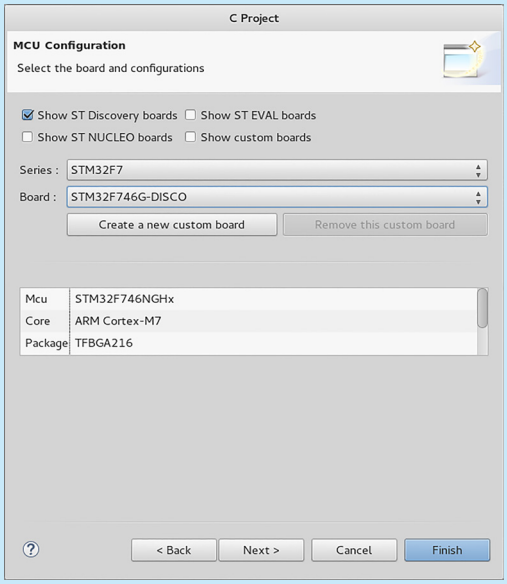
### Środowisko SW4STM32

Za wyborem SW4STM32 jako środowiska programistycznego dla mikrokontrolerów z rodziny STM32 przemawia jego kilka niewątpliwych zalet. Pierwszą z nich jest obsługa wszystkich dostępnych na rynku zestawów serii Discovery i Nucleo oraz kilku innych. Konfiguracja projektu obejmuje też wybór podstawowej biblioteki peryferiów (starszej StdPeriph Library lub nowej biblioteki Cube) oraz dodatkowych bibliotek zewnętrznych, takich jak STemWin, FreeRTOS, lwIP i innych, które po utworzeniu projektu są automatycznie pobierane i dołączane do niego. Pozwala to pominąć żmudny proces dodawania ścieżek do zewnętrznych źródeł. Niestety, nie wszystkie biblioteki są ze sobą kompatybilne od razu po ich importowaniu i czasem wymagają drobnych modyfikacji, np. podczas łączenia ich z systemem operacyjnym. Jest jednak szansa, że w przyszłości sytuacja ta ulegnie poprawie. SW4STM32 jest oparty na darmowym i powszechnie używanym środowisku Eclipse, przez co jest dostępny zarówno dla systemów Windows, jak i Linux. Do kompilacji wykorzystywany jest toolchain GCC ARM Embedded. SW4STM32 nie ma żadnych limitów kodu i jest całkowicie darmowy. Do przygotowania opisywanego projektu użyto SW4STM32 w wersji 1.7 oraz systemu operacyjnego Ubuntu 14.04 LTS (x86_64). Z uwagi na to, że SW4STM32 po instalacji nie wymaga żadnej dodatkowej konfiguracji, projekt można w analogiczny sposób przygotować na innych systemach operacyjnych.

Na początku trzeba pobrać środowisko ze strony *www.openstm32.org*. Niestety jest to możliwe dopiero po rejestracji. W menu System Workbench for STM32 można wybrać jedną z dwóch opcji instalacji: przez zainstalowanego wcześniej Eclipse'a, lub za pomocą pełnego instalatora. Oba sposoby instalacji zostały opisane na stronie. Po pierwszym uruchomieniu środowiska warto upewnić się czy nie ma dostępnych aktualizacji: *Help → Check for Updates*. W starszych wersjach oprogramowania może nie być wsparcia dla najnowszych zestawów ewaluacyjnych.




**Rysunek 1. Wprowadzenie nazwy projektu**




**Rysunek 2. Wybranie docelowego zestawu ewaluacyjnego**

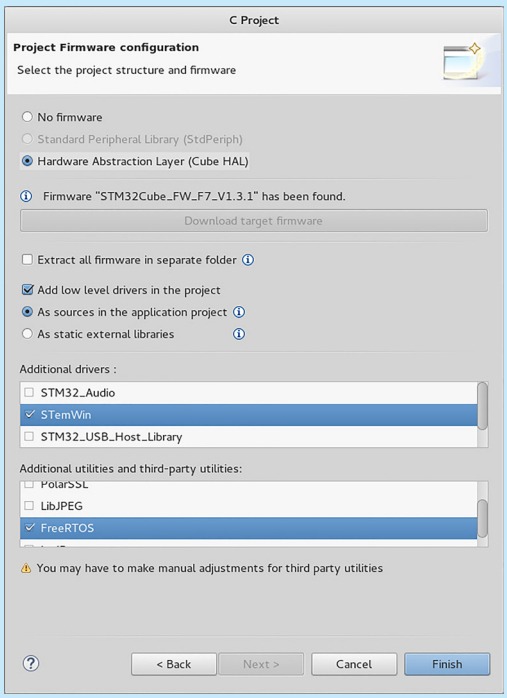
W przypadku instalacji na komputerach z 64-bitowym systemem Linux, podczas kompilacji może wystąpić błąd *arm-none-eabi-gcc file not found*. Jednym z możliwych rozwiązań tego problemu jest zainstalowanie 32-bitowych wersji bibliotek C oraz ncurses (libc-dev:i386 i lib32ncurses5, pakiety mogą się różnić w zależności od dystrybucji).

## Tworzenie i konfigurowanie nowego projektu

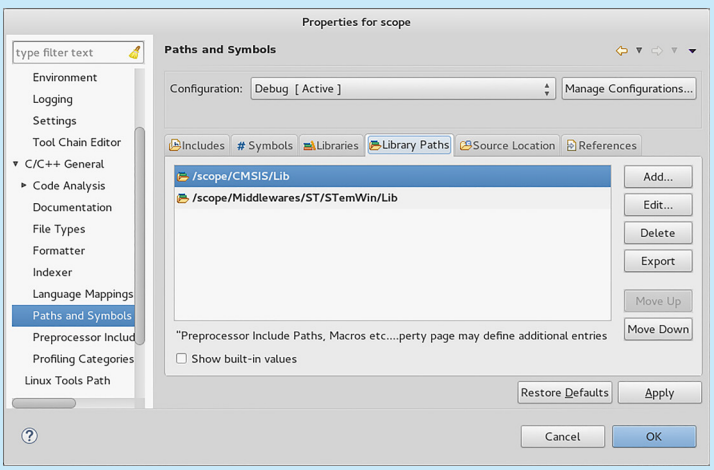
W tym rozdziale zostanie opisane utworzenie nowego projektu krok po kroku oraz konfigurowanie bibliotek i kompilowanie przykładowej aplikacji. Etap ten można pominąć importując bezpośrednio projekt z archiwum dostępnego do pobrania na stronie (*File → Import..., General → Existing Project into Workspace, Select archive file*).

Do utworzenia nowego projektu służy opcja File →New→C Project. W wyświetlonym oknie należy wybrać *Executable → AC6 STM32 MCU Project → AC6 STM32 MCU GCC* i podać nazwę projektu (**rysunek 1**). W dalszej konfiguracji można wybrać docelowy zestaw deweloperski (**rysunek 2**) oraz pobrać potrzebne sterowniki i biblioteki (**rysunek 3**). Ze względu na zależności między bibliotekami, w ostatnim kroku konfiguracji najlepiej wybrać opcję *Add low level drivers in the project → As sources in the application project*. Dzięki temu wszystkie źródła znajdą się wewnątrz jednego projektu. Z listy *Additional Drivers* należy wybrać bibliotekę graficzną STemWin, a z listy *Additional utilities and third party utilities* – system *FreeRTOS*. Po kliknięciu przycisku *Finish*, nowo utworzony projekt będzie widoczny po lewej stronie w oknie Project Explorer.

Wygenerowane źródła wymagają jeszcze kilku zmian przed pierwszą kompilacją. Po pierwsze, z katalogu *Middlewares/ST/STemWin* trzeba usunąć katalogi *Simulation* i *Config*. Pierwszy z nich zawiera źródła umożliwiające kompilację interfejsu graficznego pod systemem

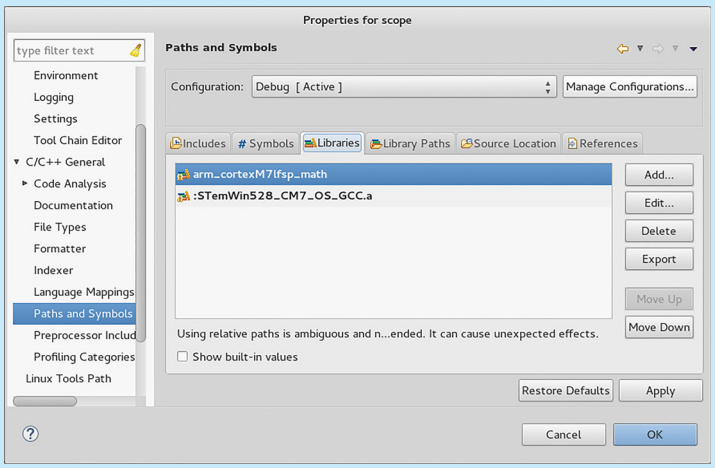



**Rysunek 3. Instalowanie potrzebnych bibliotek i sterowników**

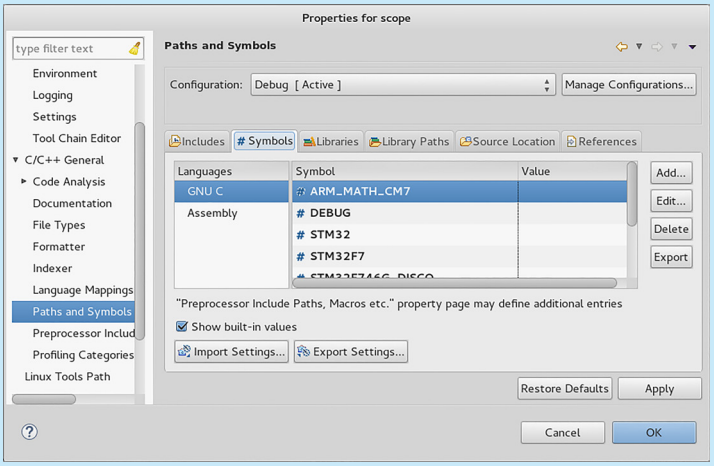



**Rysunek 4. Wprowadzenie nazw katalogów z bibliotekami**

Windows i uruchomienie symulacji. W drugim katalogu znajdują się szablony konfiguracji wyświetlacza LCD i interfejsu graficznego wymagane przez bibliotekę STemWin. Pliki konfiguracyjne zostaną później utworzone w źródłach projektu.



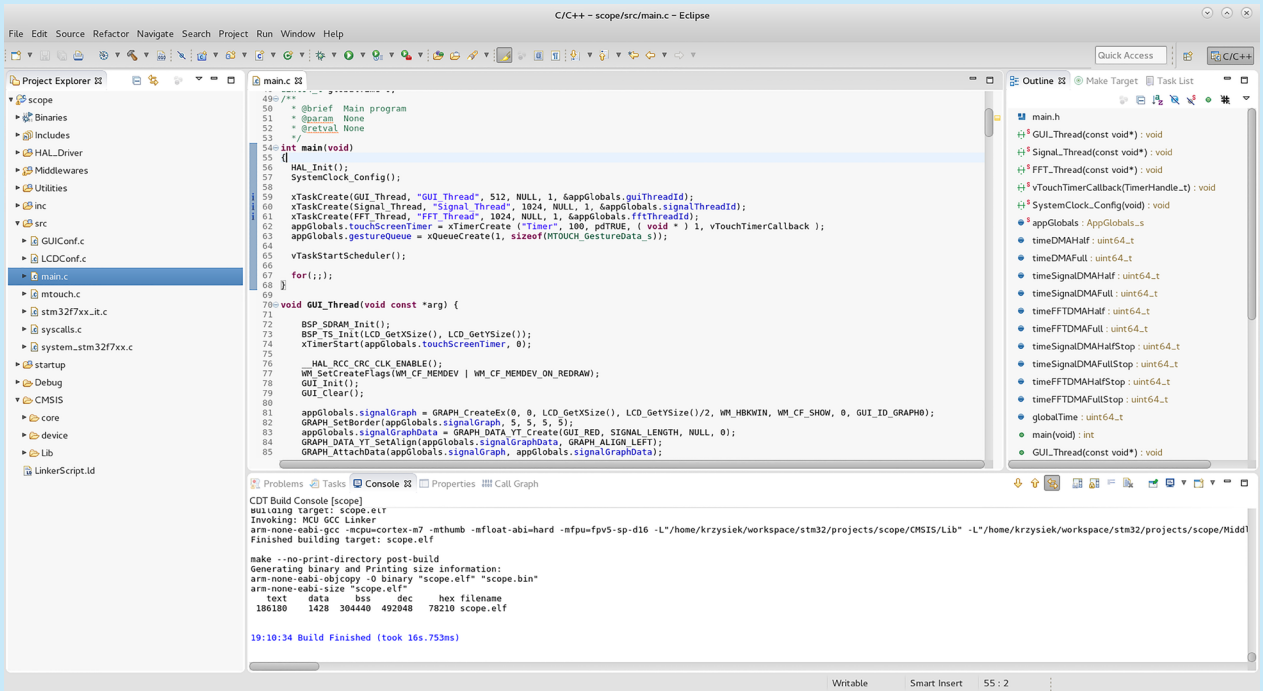

**Rysunek 5. Podanie nazw wybranych bibliotek**



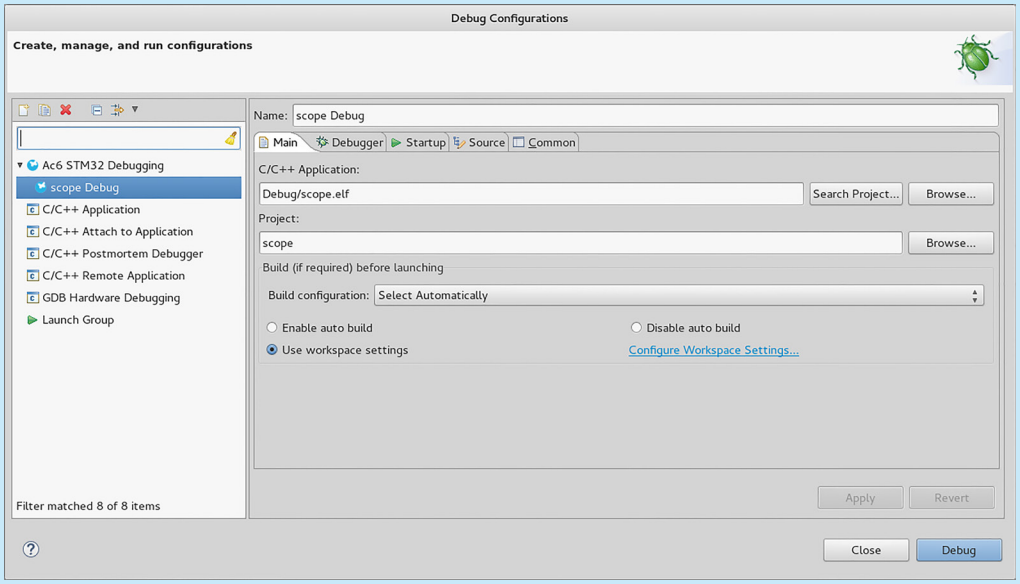

**Rysunek 6. Dodanie symbolu ARM_MATH_CM7**

Kolejną zmianą jest usunięcie pliku **Middlewares/ST/StemWin/OS/GUI_X.c**. Zawiera on m.in. funkcję *GUI_X_Delay*, w wersji dla projektu bez systemu operacyjnego. W przykładzie będzie używany drugi z plików w tym katalogu: **GUI_X_OS.c**, zawierający wszystkie funkcje niezbędne do pracy pod kontrolą systemu operacyjnego. Kolejnym plikiem, który należy usunąć jest **FreeRTOSConfig.h** znajdujący się w katalogu */Middlewares/Third_Party/FreeRTOS/Source/include*. Zawiera on domyślną konfigurację systemu operacyjnego, która zostanie zastąpiona przez konfigurację w źródłach projektu. Ostatnim plikiem do usunięcia jest **HAL_Driver/Inc/stm32f7xx_hal_conf.h**, który z kolei zawiera definicje wszystkich modułów dołączanych do projektu z biblioteki peryferiów. Zamiast niego w projekcie będzie znajdowała się lista modułów, które są faktycznie używane.

Kolejnym krokiem jest dołączenie do projektu odpowiednich wersji bibliotek STemWin oraz matematycznej ARM CMSIS DSP. W katalogu *Middlewares/ST/StemWin/Lib* znajdują się różne warianty biblioteki STemWin dla różnych kompilatorów. W projekcie będzie używana biblioteka *STemWin528_CM7_OS_GCC.a* (kompilator GCC ze wsparciem dla systemu operacyjnego). Niestety, biblioteki matematycznej nie ma domyślnie w wygenerowanych źródłach projektu i należy ją pobrać osobno ze strony ST razem z platformą STM32CubeF7 – *http://goo.gl/azAcHQ*. Biblioteka *libarm_cortexM7lfsp_math.a* (*little endian*, zmiennoprzecinkowa pojedynczej precyzji) znajduje się

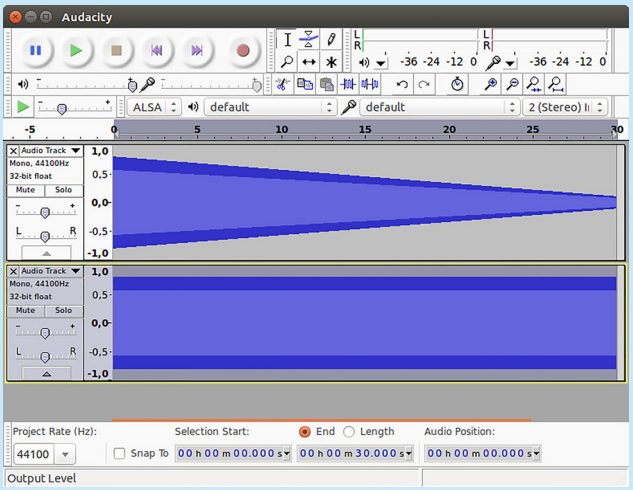



**Rysunek 7. Wynik poprawnego skompilowania projektu**




**Rysunek 8. Domyślna konfiguracja uruchamianego programu**

w katalogu *STM32Cube_FW_F7_V1.3.0/Drivers/CMSIS/Lib/GCC*. Można ją skopiować do katalogu projektu. Wybrane biblioteki trzeba dodać do kompilacji w ustawieniach projektu. W tym celu należy kliknąć prawym przyciskiem myszy na nazwę projektu i wybrać Properties. W opcji *C/C++ General → Paths and Symbols* zakładce *Library Paths* trzeba dodać katalogi zawierające wspomniane biblioteki (**rysunek 4**), natomiast w zakładce Libraries podać nazwy wybranych bibliotek (**rysunek 5**). Warto przy tym zwrócić uwagę na konwencje podawania nazw plików bibliotek. W przypadku biblioteki STemWin podawana jest pełna nazwa, razem z rozszerzeniem. Wymaga to wpisania znaku dwukropka przez nazwą pliku. Biblioteka matematyczna ma z kolei w nazwie pliku przedrostek lib, którego nie wpisuje się w nazwie biblioteki, pomijając także rozszerzenie „.a". Obie te konwencje są akceptowane przez program linkera. W konfiguracji projektu należy również dodać symbol ARM_MATH_CM7, wymagany przez plik nagłówkowy **arm_math.h**. Można go dodać w zakładce *Symbols* zaznaczając opcje *Add to all configurations* i *Add to all languages* (**rysunek 6**).




**Rysunek 9. Oprogramowanie Audacity**

Po ukończeniu powyższych kroków projekt jest już w pełni skonfigurowany. Teraz wystarczy do katalogów *inc* oraz *src* skopiować pliki źródłowe projektu z archiwum zamieszczonego na stronie, nadpisując te, które się tam już znajdują. W razie błędów związanych z nieznanymi symbolami, pokazywanych w edytorze kodu, można wymusić indeksowanie źródeł projektu klikając prawym przyciskiem myszy na projekt i wybierając *Index → Rebuild* oraz *Index → Freshen All Files*. Po tym etapie projekt powinien zostać poprawnie skompilowany (**rysunek 7**).

## Uruchomienie projektu

Projekt można uruchomić klikając ponownie prawym przyciskiem myszy na projekcie i wybierając *Debug As → Debug Configurations*. W otwartym oknie należy dwukrotnie kliknąć na opcję *Ac6 STM32 Debugging*. Ustawienia powinny zostać wybrane automatycznie (**rysunek 8**). Po przyłączeniu płytki Discovery i kliknięciu przycisku *Debug* projekt zostanie uruchomiony w trybie debugowania, w którym dostępne są opcje pracy krokowej, podglądu pamięci itp.

W systemach Linux, przy pierwszej próbie uruchomienia debugowania można natrafić na problem z uprawnieniami. Jego rozwiązaniem może być skopiowanie pliku **plugins/fr.ac6.mcu.externaltools.openocd.linux64_1.7.0.201602121841/tools/openocd/share/openocd/contrib/99-openocd.rules** z katalogu, w którym zainstalowane zostało środowisko SW4STM32, do katalogu systemowego **/etc/udev/rules.d**. Aktualny użytkownik musi także znajdować się w grupie **plugdev**.

Aby obejrzeć sygnał i jego widmo na wyświetlaczu, można podłączyć wejście liniowe płytki Discovery do wyjścia słuchawkowego komputera. Do generowania przebiegów testowych dobrze nadaje się program **Audacity** dostępny dla systemu Windows i Linux (**rysunek 9**). Umożliwia on m. in. generowanie podstawowych przebiegów (sinus, prostąkąt) o stałych lub zmiennych częstotliwościach za pomocą opcji *Generate → Tone* i *Generate → Chirp*.

W kolejnych częściach artykułu zostaną bliżej przedstawione poszczególne biblioteki i elementy aplikacji.

**Krzysztof Chojnowski**