



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Instrukcja współfinansowana przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
w projekcie

*„Innowacyjna dydaktyka bez ograniczeń
– zintegrowany rozwój Politechniki Łódzkiej – zarządzanie Uczelnią,
nowoczesna oferta edukacyjna i wzmacniania zdolności
do zatrudniania osób niepełnosprawnych”*

Instrukcja jest dystrybuowana bezpłatnie.

Instrukcja do laboratorium

Mariusz Czarnecki

Wybrane zagadnienia projektowania i programowania systemów bezprzewodowych

Zadanie nr 14 – Studia podyplomowe „Bezprzewodowe systemy nadzoru i monitorowania”



Politechnika Łódzka
Instytut Elektroniki

90-924 Łódź, ul. Żeromskiego 116,
tel. 042 631 28 83
www.kapitalludzki.p.lodz.pl



Zawartość

Zawartość	2
1 Instalacja oprogramowania	3
1.1 Instalacja kompilatora.....	3
1.2 Instalacja emulatora	3
1.3 Konfiguracja środowiska	6
2 Ćwiczenia.....	8
2.1 Ćwiczenia podstawowe	8
Prosta aplikacja z menu i jednym oknem	8
Tworzenie własnych elementów interfejsu użytkownika	8
Aplikacja realizująca dostęp do lokalnej bazy danych	8
2.2 Ćwiczenia dodatkowe.....	10
Aplikacja wyświetlająca wybrane dane z GPS (*)	10
Aplikacja wyświetlająca prognozy meteorologiczne dla wybranej lokalizacji (*)	10
Komunikacja między procesami – Mini system: GUI z procesem pracującym w tle (*)	11





1 Instalacja oprogramowania

1.1 Instalacja kompilatora

W czasie ćwiczeń będziemy posługiwać się pakietem CeGCC, który zawiera pełny zestaw narzędzi do tworzenia oprogramowania dla platformy ARM Windows CE / Windows Mobile w systemie Linux.

Aby zainstalować pakiet przejdź do katalogu \$HOME/mobile/linux i jako root uruchom polecenie rozpakowania pakietu **cegcc-mingw32ce-0.51.0.tar.gz**.

```
$ cd ~/mobile/linux
$ su
# tar -xzf cegcc-mingw32ce-0.51.0.tar.gz -C /
# exit
```

Aby sprawdzić czy kompilator został poprawnie zainstalowany wykonaj próbę kompilacji prostego programu, hello.c który znajdziesz w katalogu \$HOME/mobile/lab/hello

```
$ cd ~/mobile/lab/hello
$ /opt/mingw32ce/arm-wince-mingw32ce/bin/gcc hello.c -o hello.exe
```

Po wykonaniu powyższego polecenia w bieżącym katalogu powinien zostać utworzony plik hello.exe.

1.2 Instalacja emulatora

Do testowania stworzonych przez siebie programów będziemy używali Microsoft Device Emulator - emulatora urządzenia opartego o 32-bitowy procesor ARM z systemem Windows Mobile 5. Emulator jest aplikacją działającą w systemie Windows, dlatego do jego uruchomienia posłużymy się pakietem **wine**.

Do poprawnego działania instalatora i emulatora konieczne jest odpowiednie skonfigurowanie pakietu wine. Odpowiedni zestaw narzędzi znajduje się w katalogu \$HOME/mobile/wine.





```
$ ./winetricks volnum  
$ wine cmd  
Z:\...\wine> vs_emulator_3
```

Po zakończeniu pracy instalatora można sprawdzić efekt jego działania:

```
Z:\...\wine> C:  
C:\> cd "C:\Program Files\Microsoft Device Emulator\1.0"  
C:\Program Files\Microsoft Device Emulator\1.0> DeviceEmulator
```

Emulator nie jest jeszcze w stanie działać prawidłowo, ale powinien przynajmniej wyświetlić okno z dostępnymi opcjami. Zwróć uwagę na opcje dotyczące określania wielkości pamięci, emulacji portów szeregowych i wymiany danych z hostem.

Po zamknięciu okna wyjdź ze środowiska wine poleceniem exit.

```
C:\Program Files\Microsoft Device Emulator\1.0> exit
```

Do prawidłowego działania emulatora potrzebne są obrazy pamięci systemu operacyjnego, które instalujemy z katalogu \$HOME/mobile/wine.

```
$ cd ~/mobile/wine  
$ tar -xzf msemulimg_wm5.tar.gz -C ~/.wine/drive_c/
```

Po instalacji można uruchomić emulator przy pomocy przygotowanego skryptu **wmemul.sh**.



Gotowy do pracy program emulatora powinien wyglądać następująco:



W celu sprawdzenia działania programu testowego należy przenieść go do wirtualnego urządzenia. Emulator mapuje podany w opcji /sharedfolder katalog jako kartę SD. Skrypt uruchamiający emulator tworzy odpowiedni katalog w \$HOME/.wine/drive_c/sdcard. Po skopiowaniu pliku hello.exe z katalogu \$HOME/mobile/examples/1 do \$HOME/.wine/drive_c/sdcard będziemy mogli uruchomić testowy program w emulatorze.

```
$ cd ~/mobile/lab/hello
$ cp hello.exe ~/.wine/drive_c/sdcard/
```

Po uruchomieniu programu hello zobaczymy na ekranie efekt jego działania.



Wszystkie narzędzia zostały zainstalowane prawidłowo. Możemy przystąpić do trudniejszych zadań.

1.3 Konfiguracja środowiska

Przy kompilacji programów i bibliotek przy pomocy narzędzia *make* wykorzystywane są zmienne środowiskowe zawierające domyślny kompilator i jego opcje. Dużym ułatwieniem w pracy programistów czy integratorów systemów są skrypty ułatwiające codzienne czynności. W katalogu \$HOME/mobile/config jest umieszczony przykładowy skrypt **arm-wince.inc**. Zawiera on definicje zmiennych rozpoznawanych przez *make* i zmiennych pomocniczych.

```
CCDIR = /opt/mingw32ce/arm-wince-mingw32ce/  
CC = $(CCDIR)/bin/gcc  
CXX = $(CCDIR)/bin/g++  
LD = $(CXX)  
RANLIB = $(CCDIR)/bin/ranlib  
WINDRES = /opt/mingw32ce/bin/arm-wince-mingw32ce-windres  
  
CFLAGS = -D _MSC_VER=1200 -D _WIN32_WCE  
CXXFLAGS = -D _MSC_VER=1200 -D _WIN32_WCE  
PLATFORM = arm-wince
```



Program *make* korzysta ze zmiennych *CC* i *CXX* definiujących domyślny kompilator języka C i C++. Zmienna *LD* oznacza domyślny konsolidator, *RANLIB* – narzędzie do tworzenia bibliotek statycznych. Zmienne *CFLAGS* i *CXXFLAGS* oznaczają domyślne opcje kompilatora C i C++.

Pozostałe zmienne zdefiniowane w przykładowym pliku mogą być przydatne przy budowaniu późniejszych projektów.

Plik konfiguracyjny jest włączany do właściwego pliku *Makefile* przy pomocy dyrektywy **include**.

Przykład: Prosty plik *Makefile* znajduje się w *\$HOME/mobile/examples/1*. Po utworzeniu takiego pliku, program który wcześniej musieliśmy kompilować przy pomocy nieco długiego polecenie zbudujemy teraz poleceniem *make*. Zawartość pliku informuje program *make* w jaki sposób ma być zbudowany moduł, którego dotyczy.

```
include ../../config/arm-wince.inc

all: hello.exe

hello.exe: hello.o
        $(CC) $< -o $@

install: hello.exe
        cp hello.exe ~/.wine/drive_c/sdcard
```

Pierwsza linia włącza plik konfiguracyjny do projektu, druga opisuje jakie moduły mają być zbudowane. Jeśli program *make* uruchomimy bez parametrów, zostaną zbudowane wszystkie moduły wymienione po słowie „all:”. Trzecia sekcja informuje *make*, że program *hello.exe* ma być zbudowany z obiektu *hello.o* (powstaje on przez kompilację pliku źródłowego *hello.c*, która odbywa się automatycznie). Czwarta, ostatnia sekcja określa sposób instalacji gotowego modułu. W tym wypadku, po wydaniu polecenia *make install* zostanie on skopiowany do katalogu, który mapujemy w emulatorze jako kartę pamięci. Jeśli plik *hello.exe* jeszcze nie istnieje, zostanie automatycznie zbudowany.





2 Ćwiczenia

Każda grupa realizuje 4 ćwiczenia - 3 obowiązkowe i jedno wybrane ćwiczenie oznaczone (*). Ćwiczenia oparte są o moduły programowe przygotowane na użytek kursu oraz biblioteki Open Source. Zewnętrzne komponenty przydatne do realizacji ćwiczeń znajdziesz w katalogu **\$HOME/mobile/ext**.

2.1 Ćwiczenia podstawowe

Prosta aplikacja z menu i jednym oknem

Celem ćwiczenia jest zapoznanie się ze sposobem budowania standardowego interfejsu użytkownika w Windows Mobile, tworzeniem zasobów, widoków i obsługą podstawowych zdarzeń.

Przebieg ćwiczenia:

Utwórz klasy reprezentujące aplikację, główne okno programu i osadzony w nim widok (np. listę). Zaprojektuj menu i pasek narzędzi programu. Zaimplementuj obsługę zdarzeń (rysowanie w oknie, obsługa poleceń menu).

Do realizacji ćwiczenia wykorzystaj bibliotekę Win32++ i przykład podany w katalogu **\$HOME/mobile/lab/simpleapp**.

Tworzenie własnych elementów interfejsu użytkownika

Celem ćwiczenia jest zapoznanie się z metodami tworzenia własnych komponentów graficznych i budowania niestandardowych interfejsów użytkownika.

Przebieg ćwiczenia:

Utwórz klasy reprezentujące aplikację i główne okno programu. Okno musi być ustawione w tryb pełnoekranowy, nie zawierać standardowych dekoracji (menu, pasek narzędzi) i przykrywać wszystkie elementy interfejsu systemowego. Zaprojektuj i dodaj do okna własne elementy interfejsu.

Do realizacji ćwiczenia wykorzystaj bibliotekę Win32++ i przykład podany w katalogu **\$HOME/mobile/lab/customgui**. Zwróć uwagę na implementację klasy realizującej operacje graficzne na mapach bitowych (pobieranie pikseli w formacie niezależnym od sprzętu, mieszanie obrazów z wykorzystaniem kanału alfa, itp.).

Aplikacja realizująca dostęp do lokalnej bazy danych

Celem ćwiczenia jest zapoznanie się z podstawami budowania relacyjnych baz danych i tworzenia oprogramowania służącego do wyświetlania i modyfikowania zawartości bazy.

Przebieg ćwiczenia:



Zaprojektuj schemat bazy danych zawierający 2 tabele powiązane relacją. Zaprojektuj klasy obsługujące pobieranie danych do obiektów interfejsu Windows – listy i listy rozwijanej. Utwórz aplikację zawierającą okno główne ze standardowymi elementami, zawierające widoki przełączane komendami menu.

Do utworzenia bazy danych wykorzystaj narzędzie sqlite3. Dostęp do bazy zapewnia biblioteka sqlite oraz napisana w C++ biblioteka sharplite „opakowująca” natywny interfejs.

Interfejs użytkownika zbuduj przy użyciu biblioteki Win32++. Wykorzystaj przykład podany w katalogu **\$HOME/mobile/lab/database**.



2.2 Ćwiczenia dodatkowe

Aplikacja wyświetlająca wybrane dane z GPS (*)

Celem ćwiczenia jest opanowanie metody komunikacji programów z portem szeregowym, uzyskiwania danych z odbiornika GPS i prezentacji wyników w postaci graficznej.

Przebieg ćwiczenia:

Zbuduj aplikację odbierającą dane z portu szeregowego. Przeanalizuj ramki protokołu NMEA i wybierz właściwe ramki konieczne do uzyskania położenia, prędkości i kierunku ruchu oraz jakości sygnałów odbieranych z poszczególnych satelitów. Zaprezentuj wyniki na ekranie w czytelnej formie.

Wykorzystaj komponenty do komunikacji z portem szeregowym i analizy ramek NMEA umieszczone w katalogu **\$HOME/mobile/lab/gps** oraz znane z poprzednich ćwiczeń komponenty do tworzenia interfejsu użytkownika.

Aplikacja wyświetlająca prognozy meteorologiczne dla wybranej lokalizacji (*)

Celem ćwiczenia jest zapoznanie się z podstawami komunikacji klient – serwer za pośrednictwem protokołu TCP/IP, podstawami protokołu HTTP i analizą plików XML. Jako przykład posłuży aplikacja wysyłająca żądanie do serwera HTTP (Google Weather), analizująca odpowiedź i wyświetlająca wyniki.

Przebieg ćwiczenia:

Aplikacja powinna oferować możliwość wybrania jednej ze zdefiniowanych miejscowości (można wykorzystać bazę danych, ale nie jest to konieczne). Dla wybranej miejscowości program będzie cyklicznie (np. co 15 minut) wysyłał odpowiednie żądanie do serwera Google Weather. Format URL takiego żądania jest następujący:

`www.google.com/iq/api?weather=[Nazwa_miejscowości]`

Zbuduj aplikację wysyłającą odpowiednio sformatowane żądanie do serwera. Przeanalizuj odpowiedź serwera – plik XML. Zidentyfikuj tagi zawierające pożądane informacje i użyj parsera XML do ich wydobywania. Zaprezentuj wyniki w postaci graficznej (warunki pogodowe) i tekstowej (temperatura, wilgotność powietrza itp). Możesz wykorzystać piktogramy proponowane przez serwer lub zaprojektować własne.

Przy realizacji ćwiczenia wykorzystaj komponenty do komunikacji sieciowej umieszczone w katalogu **\$HOME/mobile/net**, bibliotekę tinyparser do analizy plików XML oraz przykłady zamieszczone w katalogu **\$HOME/mobile/lab/weather**. Wykorzystaj umiejętności nabyte w poprzednich ćwiczeniach do budowy interfejsu użytkownika.



Komunikacja między procesami – Mini system: GUI z procesem pracującym w tle (*)

Celem ćwiczenia jest zapoznanie się, na prostym przykładzie, ze sposobami realizacji komunikacji i synchronizacji między procesami w złożonych systemach. Ćwiczenie polega na zaprojektowaniu systemu składającego się z dwóch procesów: z procesu komunikującego się z urządzeniem zewnętrznym i gromadzącego dane w pamięci współdzielonej oraz z aplikacji pobierającej dane i prezentującej je w formie czytelnej dla użytkownika.

Przebieg ćwiczenia:

Zbuduj prostą aplikację realizującą interfejs użytkownika. Zadaniem tej aplikacji będzie prezentacja informacji oraz zarządzanie procesem potomnym. Przed utworzeniem procesu potomnego aplikacja utworzy blok pamięci współdzielonej służący do przekazywania informacji między procesem aplikacji i procesem potomnym. Aplikacja może również kontrolować pracę procesu zależnego.

Proces potomny będzie programem, który nie tworzy własnego okna i nie wyświetla żadnych informacji. Jego zadaniem będzie komunikowanie się z wybranym urządzeniem (może to być GPS, inne urządzenie dostępne przez port szeregowy lub interfejs sieciowy), zbieranie informacji i przesyłanie jej do aplikacji zarządzającej za pośrednictwem pamięci współdzielonej i komunikatów systemu Windows. Wszystkie operacje wykonywane przez proces potomny będą również rejestrowane w pliku logu.

Komponenty realizujące funkcje systemowe (pamięć współdzielona, synchronizacja) znajdziesz w katalogu **\$HOME/mobile/system**. Przykłady przydatne przy realizacji ćwiczenia znajdują się w katalogu **\$HOME/mobile/lab/multiproc**.

