

Linux w systemach wbudowanych

Laboratorium L2 G1

Sprawozdanie

Adam Grącikowski, 327350

Warszawa, 8 kwietnia 2025

Spis treści

1	Cel ćwiczenia laboratoryjnego	2
2	Wymagania	2
3	Opis aplikacji	2
3.1	Funkcjonalności przycisków	2
3.2	Działanie wyświetlania	2
4	Struktura pakietu morse	3
5	Opis modyfikacji i konfiguracji Buildroota	3
6	Opis implementacji logiki aplikacji	4

1 Cel ćwiczenia laboratoryjnego

Celem ćwiczenia laboratoryjnego jest zapoznanie się z:

- procesem implementacji własnego pakietu środowiska Buildroot,
- obsługą przycisków i diod LED.

2 Wymagania

1. Implementacja aplikacji w języku kompilowanym, obsługującej przyciski i diody LED.
2. Przetestowanie korzystania z debugger'a przy uruchamianiu aplikacji.
3. Przekształcenie aplikacji w pakiet Buildroot'a.

3 Opis aplikacji

Aplikacja realizuje prosty symulator alfabetu Morse'a z interfejsem opartym o trzy przyciski i jedną diodę LED.

3.1 Funkcjonalności przycisków

Funkcjonalności przycisków są następujące:

- Przycisk DOT - po jego naciśnięciu do sekwencji sygnałów dodawany jest krótki sygnał (kropka),
- Przycisk DASH - po jego naciśnięciu do sekwencji sygnałów dodawany jest długi sygnał (pauza, czyli myślnik),
- Przycisk ACCEPT - po jego naciśnięciu, użytkownik zatwierdza sekwencję, która następnie zostaje odtworzona przy użyciu diody LED.

Program kończy działanie w sytuacji, gdy wciśnięty zostanie przycisk ACCEPT, a bieżąca sekwencja sygnałów będzie pusta.

3.2 Działanie wyświetlania

- Po zatwierdzeniu sekwencji, system iteruje po wprowadzonych sygnałach.
- Dla kropki dioda zapala się na krótki czas ($400ms$).
- Dla myślnika dioda zapala się na dłuższy czas ($800ms$).
- Pomiedzy kolejnymi sygnałami następuje krótkie wyłączenie ($200ms$).

4 Struktura pakietu morse

W celu utworzenia pakietu środowiska Buildroot utworzono w folderze `/package` nowy folder `/morse`. W folderze znalazły się pliki zgodnie ze strukturą przedstawioną na rysunku 1:

```
/morse
├── src/
│   ├── Makefile
│   ├── morse.h
│   ├── morse.c
│   └── main.c
├── morse.mk
└── Config.in
```

Rysunek 1: Struktura plików w pakiecie

Plik `Config.in` zawiera informacje o nazwie pakietu, jego krótki opis oraz informację o tym, że zależy on od pakietu `c-periphery`.

Plik `morse.mk` określa w jaki sposób mają być pobierane źródła do pakietu oraz gdzie mają być umieszczane w docelowym systemie.

W folderze `/src` znajduje się kod źródłowy pakietu wraz ze standardowym plikiem `Makefile`.

Aby pakiet był widoczny z poziomu narzędzia `nconfig` w pliku `package/Config.in` należy dodać następujący wpis:

```
menu "LINSW custom packages"
    source "package/morse/Config.in"
endmenu
```

który powoduje, że pakiet `morse` pojawia się w osobnym menu o nazwie `LINSW custom packages`. Zawartość plików pokazanych na rysunku 1 została przesłana wraz ze sprawozdaniem.

5 Opis modyfikacji i konfiguracji Buildroota

1. Pobranie środowiska Buildroot, rozpakowanie oraz dokonanie wstępnej konfiguracji poprzez:
`make raspberrypi4_64_defconfig`.
2. System configuration > Run a getty (login prompt) after boot > TTY port ustawione na console.
3. Build options > Mirrors and Download locations > Primary download site ustawione na `http://192.168.137.24/dl`.
4. Toolchain > Toolchain type ustawione na External toolchain.
5. Build options > Enable compiler cache zaznaczone i Compiler cache location ustawione na `/home/adam/linsw/ccache-br`.
6. Filesystem images zaznaczona opcja initial RAM filesystem linked into linux kernel oraz Compression method ustawione na gzip.

7. W pliku `board/raspberrypi4-64/genimage.cfg.in` parametr `size` ustawiony na 128M.
8. `System configuration > Enable root login and password > Root password` ustawione na przykładowe hasło `root,123`.
9. `System configuration > System banner` ustawione na `Welcome Adam Gracikowski`.
10. `System configuration > Root filesystem overlay directories` ustawione na `board/overlay`.
11. `Host utilities` zaznaczona opcja `host environment-setup`.
12. `LINSW custom packages` zaznaczona opcja `morse`.

6 Opis implementacji logiki aplikacji

Punktem startowym aplikacji jest funkcja `main` znajdująca się w pliku `main.c`. Inicjalizacja obiektów GPIO (diody oraz trzech przycisków) została zrealizowana przy pomocy funkcji `gpio_new()`, `gpio_open()` oraz w przypadku przycisków dodatkowo `gpio_set_edge()`, które ustawia nasłuchiwanie na zdarzenia określone flagą `GPIO_EDGE_FALLING`.

Przechowywanie sekwencji sygnałów wprowadzonej przez użytkownika zostało zrealizowane przy pomocy prostej listy powiązanej, aby zapewnić nieograniczoną z góry długość sekwencji.

Następnie program wchodzi w główną pętlę, w której oczekiwanie na zdarzenia przycisków zrealizowane zostało przy pomocy funkcji `gpio_pool_multiple()`. Funkcja ta przyjmuje jako parametry wywołania tablicę uchwytów do obiektów GPIO, długość tej tablicy, maksymalny czas oczekiwania na zdarzenie (w tym przypadku ustawiony na wartość `-1`, oznaczającą nieskończone oczekiwanie), a także tablicę typu `bool`, do której zostanie zapisana informacja, dla których przycisków wystąpiło zdarzenie.

W dalszej części pętli wykonywana jest warunkowo logika odpowiedzi na wciśnięcie poszczególnych przycisków.

Ważnym elementem pętli jest wyjmowanie z kolejki zdarzeń przy pomocy `gpio_read_event()`. Po wywołaniu tej funkcji następuje sprawdzenie, czy rozważane zdarzenie nie jest przypadkiem efektem zjawiska drgania styków (ang. *switch bounce*). Jeżeli zdarzenie wystąpiło zbyt szybko w porównaniu do ostatnio zarejestrowanego, to jest ono ignorowane. Sprawdzenie to odbywa się przy pomocy prostej funkcji `has_miliseconds_passed()` oraz tablicy `timestamps` typu `uint64_t`, w której przechowywane są czasy ostatnio zarejestrowanych zdarzeń.

Po wyjściu z pętli następuje sekcja `cleanup`, odpowiedzialna za zwalnianie oraz zamykanie zasobów GPIO, a także zwalnianie zaalokowanej pamięci.

Literatura

- [1] Adding Custom Packages to Buildroot - <https://embeddedinn.com/articles/tutorial/Adding-Custom-Packages-to-Buildroot>, dostęp: 8 kwietnia 2025
- [2] Adding New Packages to Buildroot - <https://buildroot.org/downloads/manual/manual.html>, dostęp: 8 kwietnia 2025