Linux w systemach wbudowanych

Laboratorium L4 G1

Sprawozdanie

Adam Grącikowski, 327350

Warszawa, 7 maja 2025

Spis treści

1 Cel ćwiczenia laboratoryjnego

Celem ćwiczenia jest zrealizowanie z wykorzystaniem płytki Raspberry Pi 4 z płytką rozszerzającą (i ewentulanie innego modułu I/O) urządzenia wyposażonego w złożony interfejs użytkownika.

2 Wymagania

- Przyciski i diody LED powinny być użyte do podstawowej obsługi urządzenia.
- Interfejs WWW lub inny interfejs sieciowy powinien być użyty do bardziej zaawansowanych funkcji.

3 Opis projektu i przyjętych założeń

3.1 System Kolejkowania Zleceń Produkcyjnych

System składa się z dwóch modułów:

- Webowego interfejsu planisty,
- Hardware'owego modułu pracownika linii produkcyjnej.

W przyjętym modelu każde zamówienie tworzone przez planistę składa się z nazwy oraz określonej liczby sztuk trzech rodzajów elementów (produktów):

- Kategorii A,
- Kategorii B,
- Kategorii C.

3.2 Webowy interfejs planisty

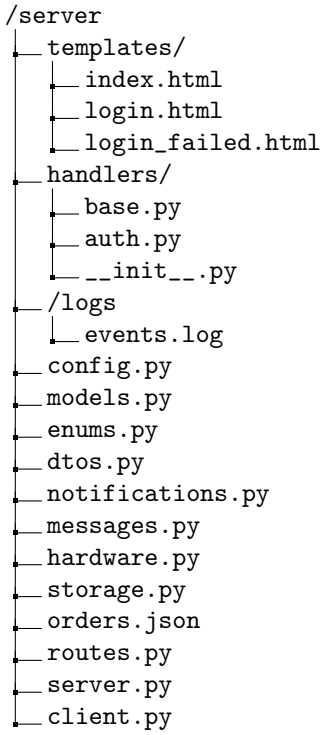
- Dostępny tylko po zalogowaniu.
- Pozwala wyświetlić listy zamówień z podziałem na ich aktualny stan:
  - W oczekiwaniu na realizację (`pending`),
  - W trakcie realizacji (`in_progress`),
  - Ukończone (`completed`).
- Umożliwia dodawanie nowych zamówień, które są dodawane do kolejki zamówień oczekujących.
- Wyświetla postęp realizacji aktualnie montowanego zamówienia.
- Pozwala zresetować linię produkcyjną w przypadku zgłoszenia awarii przez pracownika realizującego zamówienie.
- Po zresetowaniu linii produkcyjnej wznawiane jest zamówienie realizowane przed występnieniem awarii.

3.3 Moduł pracownika linii produkcyjnej

- Po uruchomieniu łączy się z serwerem i pobiera z kolejki nowe zamówienie.
- W trakcie realizacji zamówienia świeci się zielona dioda.
- Pracownik ma dostęp do następujących przycisków:
  - Zmontowano element kategorii A,
  - Zmontowano element kategorii B,
  - Zmontowano element kategorii C,
  - Awaria na linii produkcyjnej.
- Wciśnięcie przycisku o zmontowaniu elementu powoduję aktualizację progresu zamówienia w interfejsie webowym planisty.
- Po zrealizowaniu zamówienia z kolejki pobierane jest następne zamówienie.
- W przypadku braku dostępnych zamówień świeci się żółta dioda.
- Po wciśnięciu przycisku awarii zapala się czerwona dioda, odświeża się interfejs webowy oraz jest wysyłane powiadomienie mailowe do planisty, z informacją o wystąpieniu awarii.

4 Opis struktury projektu oraz implementacji

Rysunek 1 przedstawia strukturę folderów i plików w projekcie `server` dołączonym do archiwum. Najważniejszymi elementami projektu są pliki `server.py` oraz `client.py`, które odpowiedzialne są odpowiednio za działanie serwera i klienta (pracownika linii produkcyjnej).



Rysunek 1: Struktura plików w projekcie `server`

Folder `templates` zawiera sparametryzowane szablony `.html`, które serwer Tornado, zaimplementowany w języku Python przesyła użytkownikowi końcowemu. W folderze `handlers` znajduje się klasa bazowa dla wszystkich handler-ów oraz logika odpowiedzialna za logowanie użytkowników.

W pliku `config.py` umieszczone zostały stałe dla całego projektu, czyli m.in. numery pinów dla diód i przycisków, a także domyślne wartości adresu i portu serwera.

Do dwukierunkowej komunikacji pomiędzy serwerem oraz użytkownikiem interfejsu webowego, a także komunikacji pomiędzy serwerem i pracownikiem linii produkcyjnej, wykorzystano gniazda sieciowe (ang. *websockets*).

Powiadomienia mailowe zostały zrealizowane w pliku `notifications.py` przy pomocy wbudowanych funkcjonalności języka Python, nie wymagających konfiguracji żadnych zależności zewnętrznych.

W pliku `hardware.py` znajduje się implementacja logiki odpowiedzialnej za inicjalizację, przetwarzanie i zwalnianie zasobów związanych z GPIO (diód i przycisków). Plik ten zawiera abstrakcyjną klasę `HardwareModule` oraz jej dwie implementacje:

- `KeyboardModule`, która służyła do testowania (ang. *mockowania*) rzeczywistego GPIO przy pomocy wczytywania i wypisywania informacji w konsoli.
- `GPIOModule`, która realizuje komunikację z rzeczywistym GPIO przy pomocy biblioteki pigpio.

Aby uruchomić wersję skryptu `client.py`, używającą implementacji `GPIOModule` należy podać argument pozycyjny `"gpio"`, w następujący sposób:

```
./client.py gpio
```

W każdym innym przypadku zostanie uruchomiona wersja wykorzystująca `KeyboardModule`. Skrypt `server.py` przyjmuje natomiast dwa opcjonalne argumenty pozycyjne - `port` oraz `address`. W przypadku braku argumentów, ustawione zostaną wartości domyślne określone w pliku `config.py`.

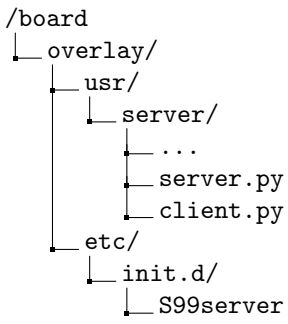
5 Opis modyfikacji i konfiguracji Buildroota

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 `/malina/gracikowskia/ccache-br`.
6. `Filesystem images` zaznaczona opcja `initial RAM filesystem linked into linux kernel` oraz `Compression method` ustawione na `gzip`.
7. `System configuration > Enable root login and password > Root password` ustawione na przykładowe hasło `root,123`.
8. `System configuration > System banner` ustawione na `Welcome to Production Line System`.
9. W pliku `board/raspberrypi4-64/genimage.cfg.in` parametr `size` ustawiony na `256M`.
10. `System configuration > Root filesystem overlay directories` ustawione na `board/overlay`.
11. `Target packages > Networking applications` zaznaczona opcja `dhcp (ISC)` oraz `dhcp client`, a także `netplug`, `ntp` i `ntpd`.
12. `Target packages > Interpreter languages and scripting` zaznaczona opcja `python3` oraz w `External python modules` zaznaczone opcje `python-tornado`, `python-websocket-client`, `python-websockets`, `python-pigpio` oraz `python-pyopenssl`.
13. `Target packagex > Hardware handling` zaznaczona opcja `pigpio` (w celu automatycznego uruchamiania daemona).
14. `Target packages > Libraries > Crypto` zaznaczona opcja `CA Certificates`.

6 Opis plików konfiguracyjnych i skryptów dołączonych do archiwum

6.1 *Overlay* na wygenerowany obraz systemu

Rysunek 2 przedstawia strukturę plików dodanych w ramach nakładki (ang. *overlay*) na wygenerowany obraz systemu.



Rysunek 2: Struktura plików nakładce `overlay`

Folder `server` zostanie omówiony w rozdziale 4. Najważniejszym jego elementami są jednak pliki `server.py`, zaznaczony na rysunku 2, do którego odwołuje się skrypt `S99server` oraz `client.py`, uruchamiany ręcznie po zalogowaniu do systemu.

Skrypt `S99server` jest odpowiedzialny za zarządzanie uruchamianiem i zatrzymywaniem serwera.

Literatura

[1] dr hab. inż. Wojciech Zabołotny, prezentacja do *Wykładu 6.*

[2] Biblioteka `Tornado` - *https://www.tornadoweb.org/en/stable/*, dostęp 7 maja 2025

[3] Biblioteka `pigpio` - *https://abyz.me.uk/rpi/pigpio/*, dostęp 7 maja 2025

[4] Biblioteka `websockets` - *https://pypi.org/project/websockets/*, dostęp 7 maja 2025

[5] WebSocket API Javascript - *https://developer.mozilla.org/en-US/docs/Web/API/WebSocket*, dostęp 7 maja 2025