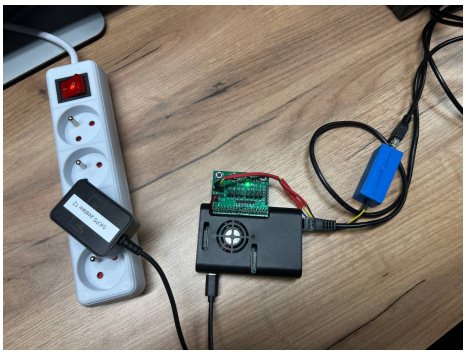
SKPS-LAB1
Sprawozdanie

Matvii Ivashchenko
Katsyaryna Anikevich

1 Złożenie stanowiska laboratoryjnego: zestaw z Raspberry Pi 4B (RPi)

Podłączylśimy Raspberry Pi 4B z adapterem, karty SD, zasilacz do RPi, konwerter USB <-> UART, konwerter USB <-> Ethernet, kabel Ethernet i przedłużacz USB, zgodnie z instrukcją.



2 Pierwsze uruchomienie RPi, sprawdzenie połączenia sieciowego, wykonanie próbnych transferów plików

Podłączenie do terminala UART
Uruchomiliśmy terminal tio na porcie /dev/ttyUSB0 i nawiązano połączenie

```
tistudent-pl@lab-44:~ tio /dev/ttyUSB0
    [tio 16:49:54] tio v1.32
    [tio 16:49:54] Press ctrl-t q to quit
    [tio 16:49:54] Connected
```

Urochomenie DHCP:
Zalogowaliśmy się do systemu ratunkowego Buildroot i uruchomiliśmy udhcpc.

```
Welcome to Buildroot rescue OS
rescue login: root
    [35.804859] cam-dummy-reg: disabling
    [35.808478] cam1-reg: disabling
```

```
# udhcpc
    udhcpc: started, v1.33.1
    udhcpc: sending discover
    udhcpc: sending select for 10.42.0.248
    udhcpc: lease of 10.42.0.248 obtained, lease time 3600
    deleting routers
    adding dns 10.42.0.1
```

Sprawdzenie stanu połączenia na RPi:
Sprawdziliśmy stan połączenia sieciowego na RPi. Polecenie ifconfig potwierdziło poprawne przypisanie adresu IP 10.42.0.248 oraz aktywność interfejsu eth0. Test ping wykazał poprawną komunikację z hostem 10.42.0.1, bez utraty pakietów, co potwierdza prawidłowe działanie połączenia.

- ifconfig:

```
# ifconfig
    eth0      Link encap:Ethernet  HWaddr E4:5F:01:2B:50:DD
              inet addr:10.42.0.248  Bcast:10.42.0.255  Mask:255.255.255.0
              inet6 addr: fe80::e65f:1ff:fe2b:50dd/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:11 errors:0 dropped:0 overruns:0 frame:0
              TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:2072 (2.0 KiB)  TX bytes:2591 (2.5 KiB)

    lo        Link encap:Local Loopback
              inet addr:127.0.0.1  Mask:255.0.0.0
              inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING  MTU:65536  Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 frame:0
              TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- ping:

```
# ping 10.42.0.1
    PING 10.42.0.1 (10.42.0.1): 56 data bytes
    64 bytes from 10.42.0.1: seq=0 ttl=64 time=1.004 ms
    64 bytes from 10.42.0.1: seq=1 ttl=64 time=0.912 ms
    64 bytes from 10.42.0.1: seq=2 ttl=64 time=0.881 ms

    --- 10.42.0.1 ping statistics ---
    3 packets transmitted, 3 packets received, 0% packet loss
    round-trip min/avg/max = 0.881/0.932/1.004 ms
```

Uruchomienie serwera HTTP
Uruchomiliśmy lokalny serwer HTTP na komputerze hosta za pomocą python3 -m http.server. Serwer nasłuchiwał na porcie 8000, umożliwiając pobieranie plików przez sieć.

```
student-pl@lab-44:~$ python3 -m http.server
    Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Pobranie pliku przez wget
Z poziomu Raspberry Pi pobraliśmy testowy plik test.txt z hosta za pomocą wget. Połączenie zostało nawiązane poprawnie, a plik został zapisany na RPi, co potwierdza poprawne działanie transferu plików.

```
# wget http://10.42.0.1:8000/test.txt
    --1970-01-01 00:06:54--  http://10.42.0.1:8000/test.txt
    Connecting to 10.42.0.1:8000... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 0 [text/plain]
    Saving to: 'test.txt'

    test.txt                [ <=>                 ]       0  --.-KB/s    in 0s

    1970-01-01 00:06:54 (0.00 B/s) - 'test.txt' saved [0/0]
```

Ściągnięcie pliku z hosta:

```
# scp student-pl@10.42.0.1:examples.desktop test.txt
    student-pl@10.42.0.1's password:
    examples.desktop                              100% 8980     3.8MB/s   00:00
```

Wysłanie pliku na hosta:

```
# touch sshtest.txt
# scp sshtest.txt student-pl@10.42.0.1:sshtest.txt
    student-pl@10.42.0.1's password:
    sshtest.txt                                   100%    0     0.0KB/s   00:00
```

3 Zbudowanie za pomocą Buildroot obrazu Linuxa dla RPi, z init RAM fs

Wykonaliśmy polecenie make raspberrypi4_64_defconfig, aby uruchomić domyślną konfigurację urządzenia.
Następnie w make menuconfig zaznaczyliśmy opcje:
Toolchain -> Toolchain Type -> External
Filesystem images -> initial RAM filesystem linked into linux kernel [x]
Filesystem images -> ext2/3/4 root filesystem [ ]
System configuration -> System hostname -> Ivashchenko_Anikevich

Po wprowadzeniu zmian rozpoczęliśmy kompilację za pomocą polecenia make.

Zamontowaliśmy pierwszą partycję karty SD w katalogu '/mnt' za pomocą polecenia 'mount /dev/mmcblk0p1 /mnt'.
Następnie przesłaliśmy pliki Image, cmdline.txt, bcm2711-rpi-4-b.dtb korzystając z polecenia wget.
Urządzenie zostało ponownie uruchomione poleceniem 'reboot', przytrzymując przycisk SW4.

Następnie utworzyliśmy plik test2.txt, a jego obecność potwierdzono poleceniem ls. Plik został poprawnie zapisany w bieżącym katalogu.

```
    Welcome to Buildroot
    Ivashchenko_Anikevich login: root
# ls
# ls /
    bin             init        media       root        tmp
    crond.reboot    lib         mnt         run         usr
    dev             lib64       opt         sbin        var
    etc             linuxrc     proc        sys
# touch test2.txt
# ls
test2.txt
```

Po wykonaniu rebootu systemu ponowne wykonanie polecenia ls potwierdziło, że plik test2.txt został usunięty, co jest zgodne z oczekiwanym zachowaniem systemu działającego w pamięci RAM (initramfs).

```
    Welcome to Buildroot
    Ivashchenko_Anikevich login: root
# ls
# ls /
    bin             init        media       root        tmp
    crond.reboot    lib         mnt         run         usr
    dev             lib64       opt         sbin        var
    etc             linuxrc     proc        sys
```

4 Zbudowanie za pomocą Buildroot obrazu Linuxa dla RPi, z systemem plików na trwałym nośniku.

Skasowaliśmy wcześniejszy obraz za pomocą polecenia make linux-dirclean.
Następnie w make menuconfig zaznaczyliśmy opcje:
Toolchain -> Toolchain Type -> External

Filesystem images -> initial RAM filesystem linked into linux kernel [ ]
Filesystem images -> ext2/3/4 root filesystem [x]
System configuration -> System hostname -> Ivashchenko_Anikevich

Wynikowy obraz jest mniejszy niż w poprzednim ćwiczeniu, ponieważ wcześniej zawierał cały system plików, a teraz jest on przechowywany osobno w niezależnej lokalizacji.

Po wprowadzeniu zmian rozpoczęliśmy kompilację za pomocą polecenia make. Następnie przesłaliśmy pliki Image, cmdline.txt, bcm2711-rpi-4-b.dtb, rootfs.ext2 korzystając z polecenia wget.
Dodatkowo zapisaliśmy plik rootfs.ext2 na drugą partycję za pomocą polecenia dd if=rootfs.ext2 of=/dev/mmcblk0p2 bs=4096.

Urządzenie zostało ponownie uruchomione poleceniem ‘reboot‘, przytrzymując przycisk SW4.

Utworzyliśmy plik test.txt, a jego obecność potwierdzono poleceniem ls. Plik został poprawnie zapisany w bieżącym katalogu.

```
    Welcome to Buildroot
    Ivashchenko_Anikevich login: root
# touch test.txt
# ls
    test.txt
# reboot
```

Po wykonaniu rebootu systemu ponowne wykonanie polecenia ls potwierdziło, że plik test.txt nadal istnieje, co jest zgodne z oczekiwanym zachowaniem systemu plików na trwałym nośniku

```
    Welcome to Buildroot
    Ivashchenko_Anikevich login: root
# ls
    test.txt
```