

SKPS laboratorium nr 1

Kinga Świderek (318 734) i Jakub Kowalczyk (318 676)

Złożenie stanowiska laboratoryjnego i pierwsze uruchomienie RPi

Na początku podłączyliśmy urządzenia zgodnie z instrukcją. Następnie podłączyliśmy się do terminala UART za pomocą programu tio, zalogowaliśmy się i przyznaliśmy adres IP protokołem DHCP

Adres IP RPi:

```
udhcpc: sending select for 10.42.0.188
udhcpc: lease of 10.42.0.188 obtained, lease time 3600
deleting routers
adding dns 10.42.0.1
# ifconfig
eth0      Link encap:Ethernet  HWaddr E4:5F:01:2B:50:A1
          inet addr:10.42.0.188  Bcast:10.42.0.255  Mask:255.255.255.0
```

IP hosta:

```
enx00e04c36ff69: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
          inet 10.42.0.1  netmask 255.255.255.0  broadcast 10.42.0.255
          inet6 fe80::e04c:36ff:fe69:1111  prefixlen 64  scopeid 0x20  link-local
```

Stan połączenia sieciowego na Raspberry Pi sprawdziliśmy za pomocą komendy ping, pingując komputer host z płytki oraz na odwrót. Obie operacje powiodły się:

```
# 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=0.974 ms
64 bytes from 10.42.0.1: seq=1 ttl=64 time=0.911 ms
64 bytes from 10.42.0.1: seq=2 ttl=64 time=0.881 ms
64 bytes from 10.42.0.1: seq=3 ttl=64 time=0.870 ms
64 bytes from 10.42.0.1: seq=4 ttl=64 time=0.887 ms
^C
--- 10.42.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.870/0.904/0.974 ms
```

```
user@lab-18:~$ ping 10.42.0.188
PING 10.42.0.188 (10.42.0.188) 56(84) bytes of data.
64 bytes from 10.42.0.188: icmp_seq=1 ttl=64 time=0.785 ms
64 bytes from 10.42.0.188: icmp_seq=2 ttl=64 time=0.825 ms
64 bytes from 10.42.0.188: icmp_seq=3 ttl=64 time=0.827 ms
64 bytes from 10.42.0.188: icmp_seq=4 ttl=64 time=0.768 ms
64 bytes from 10.42.0.188: icmp_seq=5 ttl=64 time=0.804 ms
^C
--- 10.42.0.188 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.768/0.801/0.827/0.042 ms
```

Zbudowanie obrazu Linuxa dla RPi z init RAM fs

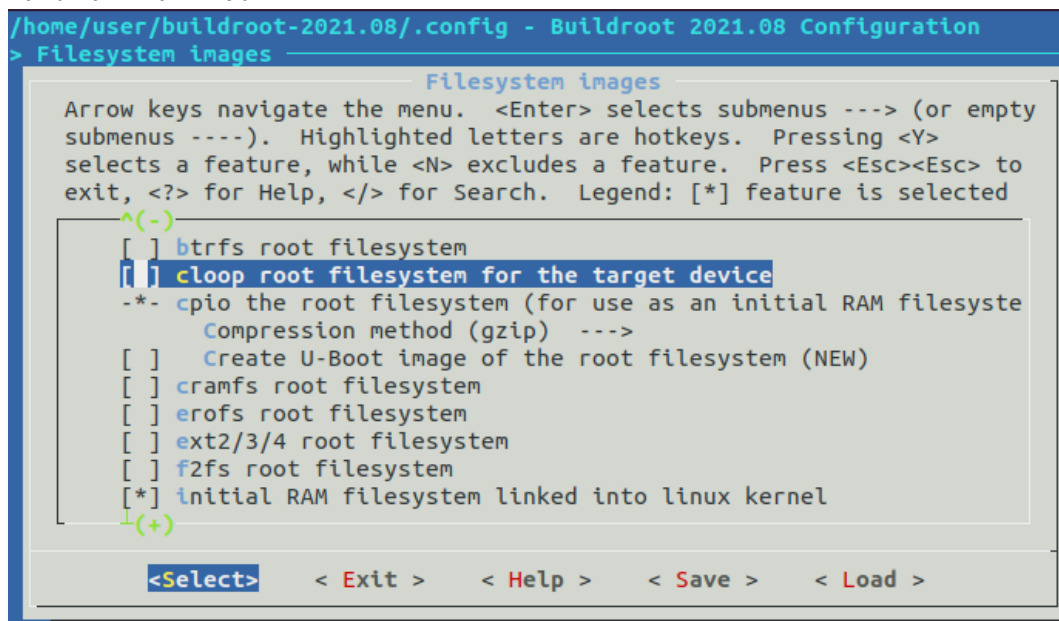
Po pobraniu i rozpakowaniu archiwum zawierającego Buildroot wykonaliśmy dwa polecenia:

```
make raspberrypi4_defconfig
```

```
make menuconfig
```

W menu wybraliśmy opcję *Toolchain* -> *Toolchain type: External toolchain*.

Następnie, aby włączyć initramfs wybraliśmy opcję *Filesystem images* -> *cpio the root filesystem* i włączyliśmy kompresję gzip, oraz *initial RAM filesystem [...]* Ostatecznie wyłączyliśmy opcję *ext2/3/4*



Po przeprowadzeniu tej konfiguracji stworzyliśmy obraz za pomocą komendy *make*.

```
root /home/user/buildroot-2021.08/output/staging
>>> Executing post-image script board/raspberrypi4-64/post-image.sh
Adding 'dtoverlay=miniuart-bt' to config.txt (fixes ttyAMA0 serial console).
board/raspberrypi4-64/genimage-raspberrypi4-64.cfg:31: no sub-section title/index for 'config'
INFO: cmd: "mkdir -p "/home/user/buildroot-2021.08/output/build/genimage.tmp" (
stderr):
INFO: cmd: "rm -rf "/home/user/buildroot-2021.08/output/build/genimage.tmp"/*" (
stderr):
INFO: cmd: "mkdir -p "/home/user/buildroot-2021.08/output/build/genimage.tmp" (
stderr):
INFO: cmd: "cp -a "/tmp/tmp.FAqsXS0pz6" "/home/user/buildroot-2021.08/output/bui
ld/genimage.tmp/root" (stderr):
INFO: cmd: "find '/home/user/buildroot-2021.08/output/build/genimage.tmp/root' -
depth -type d -printf '%P\0' | xargs -0 -I {} touch -r '/tmp/tmp.FAqsXS0pz6/{}'
'/home/user/buildroot-2021.08/output/build/genimage.tmp/root/{}'" (stderr):
ERROR: file(rootfs.ext4): stat(/home/user/buildroot-2021.08/output/images/rootfs
.ext4) failed: No such file or directory
ERROR: hdimage(sdcard.img): could not setup rootfs.ext4
Makefile:821: recipe for target 'target-post-image' failed
make[1]: *** [target-post-image] Error 1
Makefile:84: recipe for target '_all' failed
make: *** [_all] Error 2
user@lab-18:~/buildroot-2021.08$
```

Gdy obraz był gotowy wykorzystaliśmy protokół HTTP do przekopiowania plików z komputera hosta na RPi:

```
# wget -O kernel8.img http://10.42.0.1:8000/output/images/Image
--1970-01-01 01:24:03-- http://10.42.0.1:8000/output/images/Image
Connecting to 10.42.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 47213056 (45M) [application/octet-stream]
Saving to: 'kernel8.img'

kernel8.img          100%[=====] 45.03M  11.2MB/s   in 4.0s

1970-01-01 01:24:07 (11.2 MB/s) - 'kernel8.img' saved [47213056/47213056]
```

```
bcm2711-rpi-4-b.dtb 100%[=====] 48.58K  --.-KB/s   in 0.004s

1970-01-01 01:25:31 (13.4 MB/s) - 'bcm2711-rpi-4-b.dtb' saved [49749/49749]

# ls
bcm2711-rpi-4-b.dtb  cmdline.txt  kernel8.img
```

Po reboocie RPi system włączył się poprawnie:

```
Welcome to Buildroot
buildroot login: █
```

Kopię pliku .config zapisaliśmy i umieściliśmy w repozytorium.

Zbudowanie obrazu Linuxa dla RPi z systemem plików na trwałym nośniku

Zaczęliśmy od usunięcia poprzedniego obrazu poleceniem *make linux-dirclean*.

W menuconfig zmieniliśmy ustawienia dla initramfs oraz włączyliśmy wsparcie trwałego systemu plików ext2.

Niestety, kompilacja za pomocą *make clean all* się nie powiodła ze względu na zbyt mały rozmiar systemu plików. Zwiększyliśmy go zatem do 120 MiB w menuconfig.

```
Allocating group tables: done
Writing inode tables: done
Copying files into the device: __populate_fs: Could not allocate block in ext2 f
ilesystem while writing file "ocfs2.ko"
mkfs.ext2: Could not allocate block in ext2 filesystem while populating file sys
tem
*** Maybe you need to increase the filesystem size (BR2_TARGET_ROOTFS_EXT2_SIZE)
fs/ext2/ext2.mk:46: recipe for target '/home/user/buildroot-2021.08/output/image
s/rootfs.ext2' failed
make[1]: *** [/home/user/buildroot-2021.08/output/images/rootfs.ext2] Error 1
Makefile:84: recipe for target '_all' failed
make: *** [_all] Error 2
user@lab-18:~/buildroot-2021.08$ █
```

Skopiowaliśmy pliki tak samo jak wcześniej, pamiętając o dodatkowym rootfs.ext2

Tym razem obraz ma mniejszy rozmiar, gdyż nie zawiera również informacji o systemie plików.

```
user@lab-18: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
1970-01-01 00:03:19 (1.91 MB/s) - 'cmdline.txt' saved [65/65]  
  
# ls  
bcm2711-rpi-4-b.dtb  cmdline.txt          kernel8.img  
# pwd  
/mnt/user  
# cd  
# pwd  
/root  
# wget http://10.42.0.1:8000/output/images/rootfs.ext2  
--1970-01-01 00:05:32-- http://10.42.0.1:8000/output/images/rootfs.ext2  
Connecting to 10.42.0.1:8000... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 125829120 (120M) [application/octet-stream]  
Saving to: 'rootfs.ext2'  
  
rootfs.ext2      100%[=====] 120.00M  11.2MB/s   in 11s  
1970-01-01 00:05:42 (11.2 MB/s) - 'rootfs.ext2' saved [125829120/125829120]  
  
# dd if=rootfs.ext2 of=/dev/mmcblk0p2 bs=4096  
30720+0 records in  
30720+0 records out  
#
```

System prawidłowo uruchomił się, a w celu sprawdzenia działania utworzyliśmy plik testowy. Następnie po reboocie możemy zauważyć, że w przeciwieństwie do systemu z ramdyskiem startowym, system plików jest trwały i zmiany nie są tracone przy reboocie.

```
Welcome to Buildroot  
buildroot login: root  
# ls  
# pwd  
/root  
# touch test.txt  
# ls  
test.txt  
#
```

```
udhcpd: lease of 10.42.0.188 obtained, lease time 3600  
deleting routers  
adding dns 10.42.0.1  
OK  
  
Welcome to Buildroot  
buildroot login: root  
# ls  
test.txt  
#
```