

# Raport z laboratorium 2 - 20.03.2024

---

Sebastian Abramowski, 325142

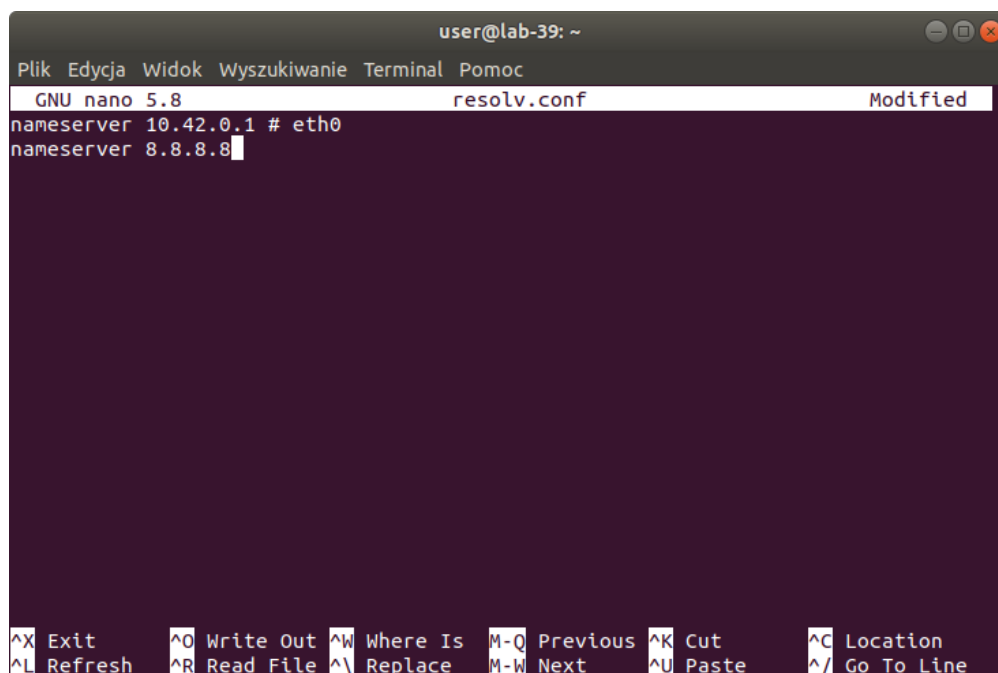
Bogumił Stoma, 325233

---

Plik wygenerowany automatycznie na podstawie pliku raport.md

## Instalacja OpenWRT

Na początku dodaliśmy serwer DNSa na wszelki wypadek do pliku `/etc/resolv.conf` na systemie ratunkowym



```
user@lab-39: ~
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Pomoc
GNU nano 5.8 resolv.conf Modified
nameserver 10.42.0.1 # eth0
nameserver 8.8.8.8
^X Exit      ^O Write Out ^W Where Is  M-Q Previous ^K Cut       ^C Location
^L Refresh   ^R Read File ^\ Replace   M-W Next     ^U Paste     ^_ Go To Line
```

Pobraliśmy obraz systemu przez wget, rozpakowaliśmy go, ustawiliśmy obraz systemu jako urządzenie "loop" i sprawdziliśmy jego nazwę

```
user@lab-39: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
dev  
etc  
init  
lib  
lib64  
linuxrc  
media  
mnt  
openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz  
opt  
proc  
root  
run  
sbin  
sys  
tmp  
usr  
var  
# gzip -d openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz  
# losetup -P -f openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img  
[ 734.321171] loop0: p1 p2  
# losetup -a  
/dev/loop0: 0 openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img  
#
```

Następnie zamontowaliśmy partycję pierwszą obrazu OpenWRT oraz pierwszą partycję karty SD w odpowiednie miejsca, następnie przetrzuciliśmy wymagane pliki do zbudowania pełnego obrazu systemu na karte SD i powiększyliśmy system plików

```
user@lab-39: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
usr  
var  
# gzip -d openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz  
# losetup -P -f openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img  
[ 734.321171] loop0: p1 p2  
# losetup -a  
/dev/loop0: 0 openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img  
# dd if=/dev/loop0p2 of=/dev/mmcblk0p2 bs=4096  
26624+0 records in  
26624+0 records out  
# mkdir /mnt/boot /mnt/owrt  
# mount /dev/loop0p1 /mnt/owrt  
# mount /dev/mmcblk0p1 /mnt/boot  
[ 822.412976] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data  
may be corrupt. Please run fsck.  
# cp /mnt/owrt/cmdline.txt /mnt/boot/user/  
# cp /mnt/owrt/kernel8.img /mnt/boot/user/  
# cp /mnt/owrt/bcm2711-rpi-4-b.dtb /mnt/boot/user/  
# resize2fs /dev/mmcblk0p2  
resize2fs 1.46.2 (28-Feb-2021)  
Resizing the filesystem on /dev/mmcblk0p2 to 161792 (4k) blocks.  
The filesystem on /dev/mmcblk0p2 is now 161792 (4k) blocks long.  
#
```

Odpaliliśmy system

[illegible]

Następnie zmieniliśmy konfigurację sieci, aby system nie próbował być routerem

```
user@lab-39: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
root@OpenWrt:/# cat /etc/config/network  
  
config interface 'loopback'  
    option device 'lo'  
    option proto 'static'  
    option ipaddr '127.0.0.1'  
    option netmask '255.0.0.0'  
  
config globals 'globals'  
    option ula_prefix 'fdbe:f6b7:4667::/48'  
  
#config device  
#    option name 'br-lan'  
#    option type 'bridge'  
#    list ports 'eth0'  
  
config interface 'lan'  
    option device 'eth0'  
    option proto 'dhcp'  
#    option ipaddr '192.168.1.1'  
#    option netmask '255.255.255.0'  
#    option ip6assign '60'  
  
root@OpenWrt:/#
```

Potem, zrestartowaliśmy ustawienia sieci poleceniem

```
/etc/init.d/network reload
```

Sprawdziliśmy czy jest dostęp do naszego systemu przez HTTP

The screenshot shows the OpenWrt web interface. At the top, there's a navigation bar with tabs for 'Status', 'System', 'Network', and 'Logout'. Below this, a yellow warning box states 'No password set!'. The main content area is divided into three sections: 'Status', 'Memory', and 'Network'. The 'Status' section shows system information like hostname, model, architecture, firmware version, kernel version, local time, uptime, and load average. The 'Memory' section shows memory usage with progress bars for total available, used, buffered, and cached memory. The 'Network' section shows the network configuration, including the protocol (DHCP client) and the address (10.42.0.188/24).

System	
Hostname	OpenWrt
Model	Raspberry Pi 4 Model B Rev 1.4
Architecture	ARMv8 Processor rev 3
Firmware Version	OpenWrt 21.02.1 r16325-88151b6303 / LuCI openwrt-21.02 branch git-21.295.67054-13df80d
Kernel Version	5.4.154
Local Time	2024-03-20 11:56:07
Uptime	0h 3m 42s
Load Average	0.00, 0.00, 0.00

Memory	
Total Available	3.62 GiB / 3.69 GiB (98%)
Used	42.71 MiB / 3.69 GiB (1%)
Buffered	864.00 KiB / 3.69 GiB (0%)
Cached	10.36 MiB / 3.69 GiB (0%)

Network	
Protocol: DHCP client	IPv4 Upstream
Address: 10.42.0.188/24	

Następnie pobraliśmy potrzebne pakiety i zaczęliśmy robić zadania

## Zad. 1 (plik gpio\_led\_1.py)

```
import gpio4
from time import sleep

gpio27 = gpio4.SysfsGPIO(27)
gpio27.export = True
gpio27.direction = 'out'

for i in range(10):
    gpio27.value = 1
    sleep(0.5)
    gpio27.value = 0
    sleep(0.5)

gpio27.export = False
```

## Zad. 2 (plik gpio\_led\_2.py) - przebudowaliśmy kod z pracy domowej

```
from gpio4 import SysfsGPIO
from time import sleep

duration = 10
epsilon = 1e-6

def calc_periods(frequency, duty_cycle):
```

```

period = 1 / frequency
high_signal_period = period * duty_cycle
low_signal_period = period - high_signal_period
return high_signal_period, low_signal_period

def variable_duty_cycle(time, duration, min_duty=0, max_duty=1):
    half_duration = duration / 2
    if time <= half_duration:
        return min_duty + (max_duty - min_duty) * (time / half_duration)
    else:
        return max_duty - (max_duty - min_duty) * (
            (time - half_duration) / half_duration)

def generate_values_for_pwm(gpio, frequency=100):
    current_t = 0
    while current_t < duration:
        variable_duty = variable_duty_cycle(current_t, duration)
        high_period, low_period = calc_periods(frequency, variable_duty)
        gpio.value = 1
        current_t += high_period
        sleep(high_period)
        gpio.value = 0
        current_t += epsilon
        gpio.value = 0
        current_t += low_period
        sleep(low_period)

gpio = SysfsGPIO(27)
gpio.export = True
gpio.direction = 'out'
generate_values_for_pwm(gpio)
gpio.export = False

```

### Zad. 3 (plik gpio\_in.py)

```

from gpio4 import SysfsGPIO
import time

gpio_led = SysfsGPIO(27)
gpio_led.export = True
gpio_led.direction = 'out'

gpio_button = SysfsGPIO(10)
gpio_button.export = True
gpio_button.direction = 'in'

gpio_led.value = 0

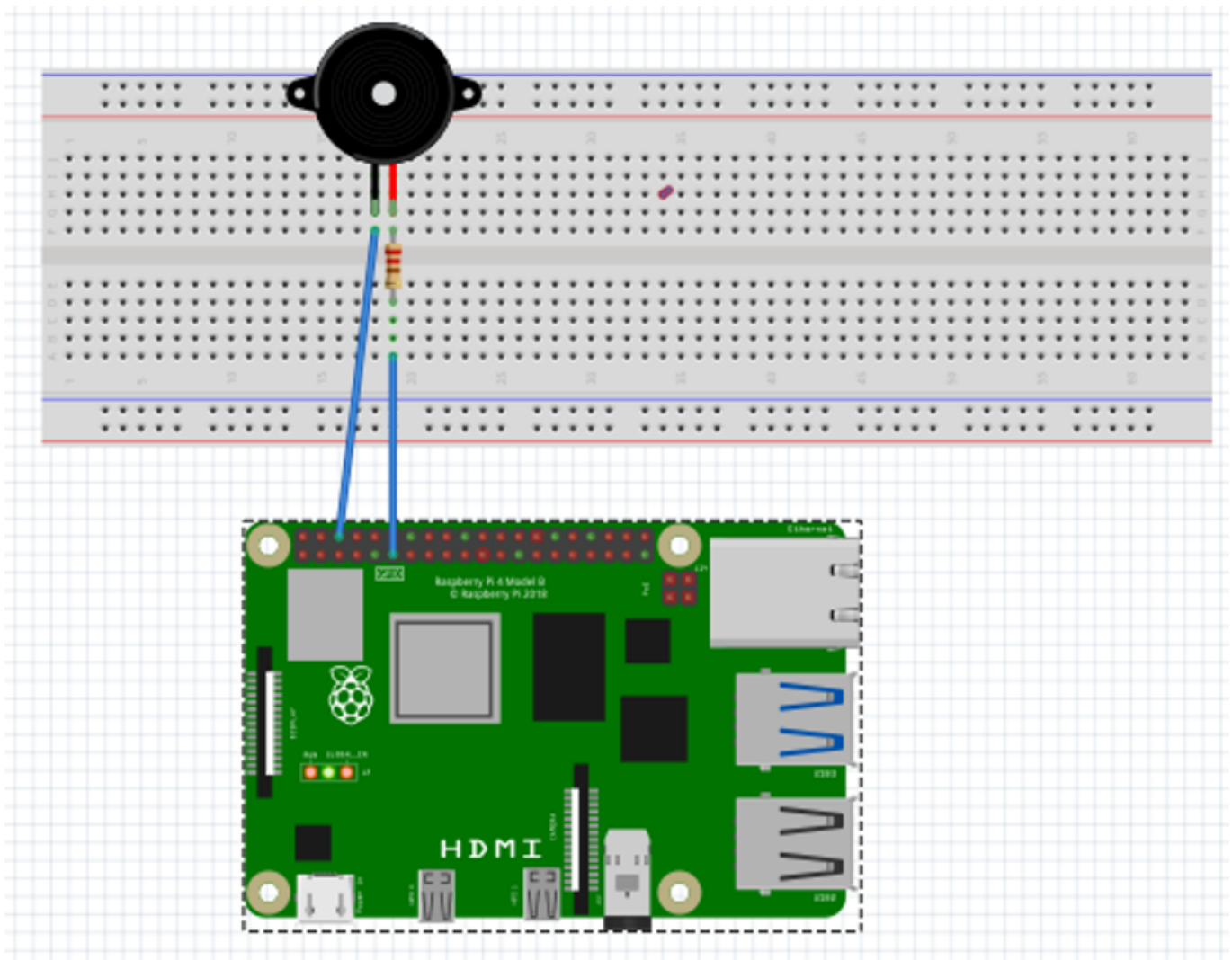
```

```
while True:
    if gpio_button.value == 0:
        gpio_led.value = 1 - gpio_led.value
        time.sleep(0.5)

# gpio_led.export = False
# gpio_button.export = False
```

## Zad. 4 (plik buzzer.py)

Schemat podłączenia



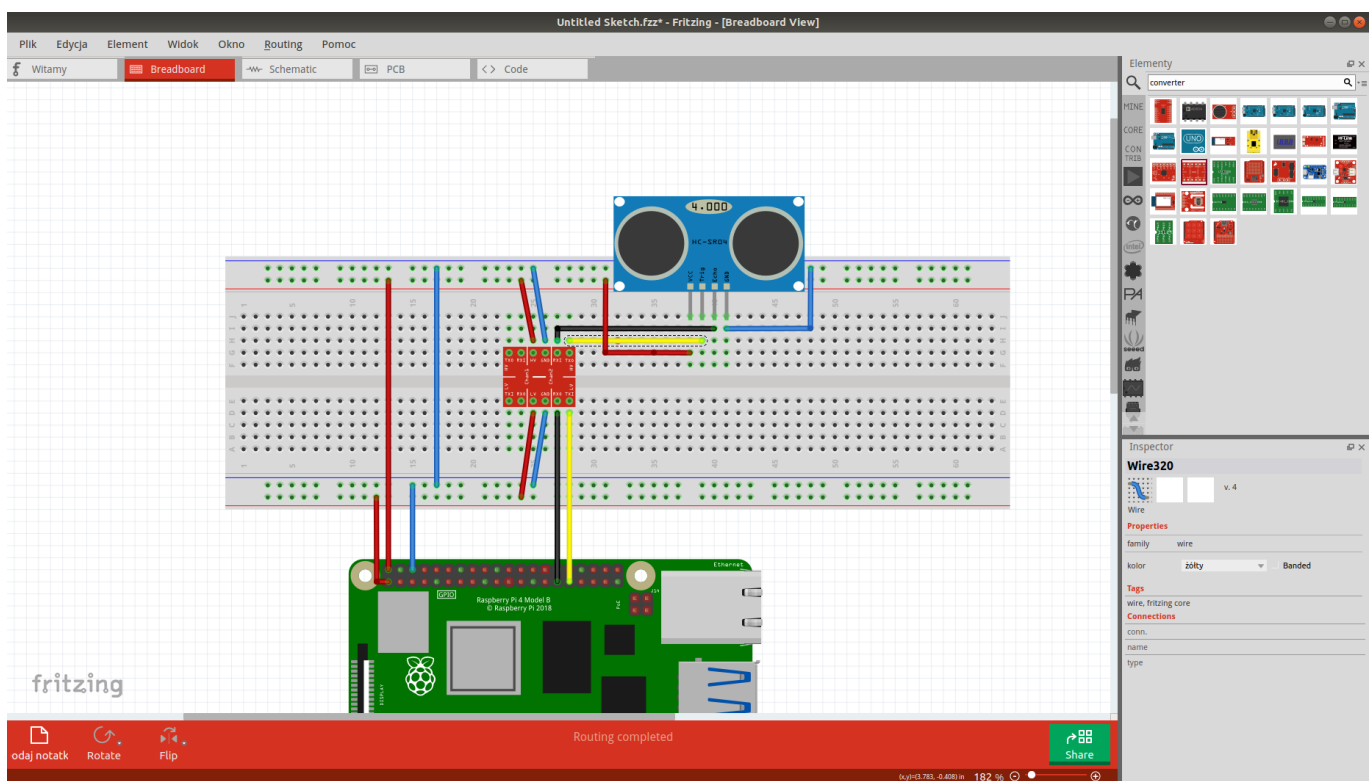
```
import time
import gpio4

buzzer = gpio4.SysfsGPIO(17)
buzzer.export = True
buzzer.direction = 'out'
```

```
c_dur_frequencies = [261.63, 293.66, 329.63, 349.23, 392, 440, 493.88,  
                    523.25, 587.33, 659.25, 698.46, 783.99, 880, 987.77]  
  
duty_cycle = 0.5  
  
for freq in c_dur_frequencies:  
    start = time.time()  
    stop = 0  
    period = 1/freq  
    while (stop - start < 1):  
        buzzer.value = 1  
        time.sleep(duty_cycle * period)  
        buzzer.value = 0  
        time.sleep((1 - duty_cycle) * period)  
        stop = time.time()  
  
buzzer.export = False
```

## Zad. 5 (plik proximity\_sensor.py)

Schemat podłączenia



```
import gpio4  
import time  
  
SPEED_OF_SOUND = 340
```

```
trigger = gpio4.SysfsGPIO(5)
trigger.export = True
trigger.direction = 'out'

echo = gpio4.SysfsGPIO(6)
echo.export = True
echo.direction = 'in'

while True:
    trigger.value = 1
    time.sleep(0.01)
    trigger.value = 0

    while echo.value == 0:
        continue

    time_start = time.time()

    while echo.value == 1:
        continue

    elapsed_time_in_sec = time.time() - time_start

    distance_in_meters = elapsed_time_in_sec * SPEED_OF_SOUND / 2
    distance_in_cm = distance_in_meters * 100

    print(f'{distance_in_cm:.2f}')
    time.sleep(0.5)

# trigger.export = False
# echo.export = False
```