Raport z laboratorium 2 - 20.03.2024

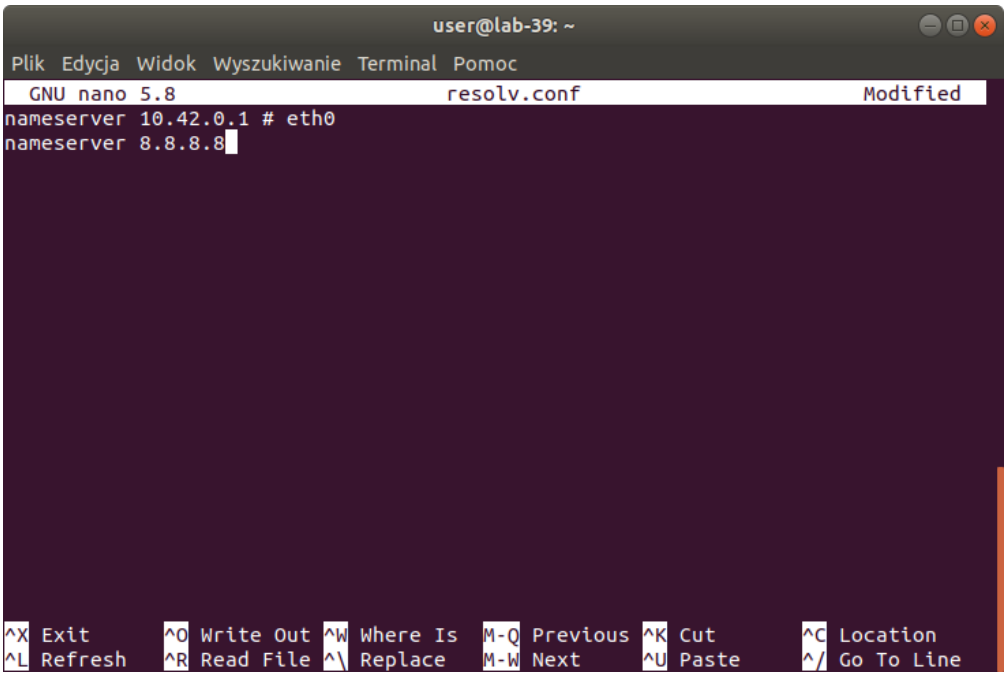
Sebastian Abramowski, 325142

Bogumił Stoma, 325233

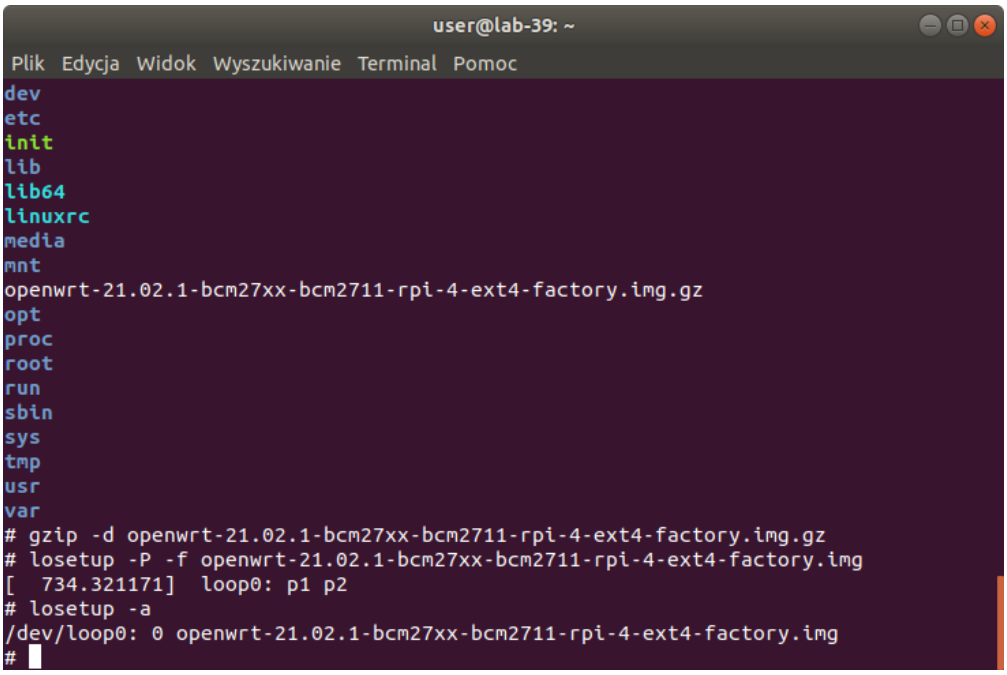
Raport wygenerowany automatycznie na podstawie pliku raport.md

Instalacja OpenWRT

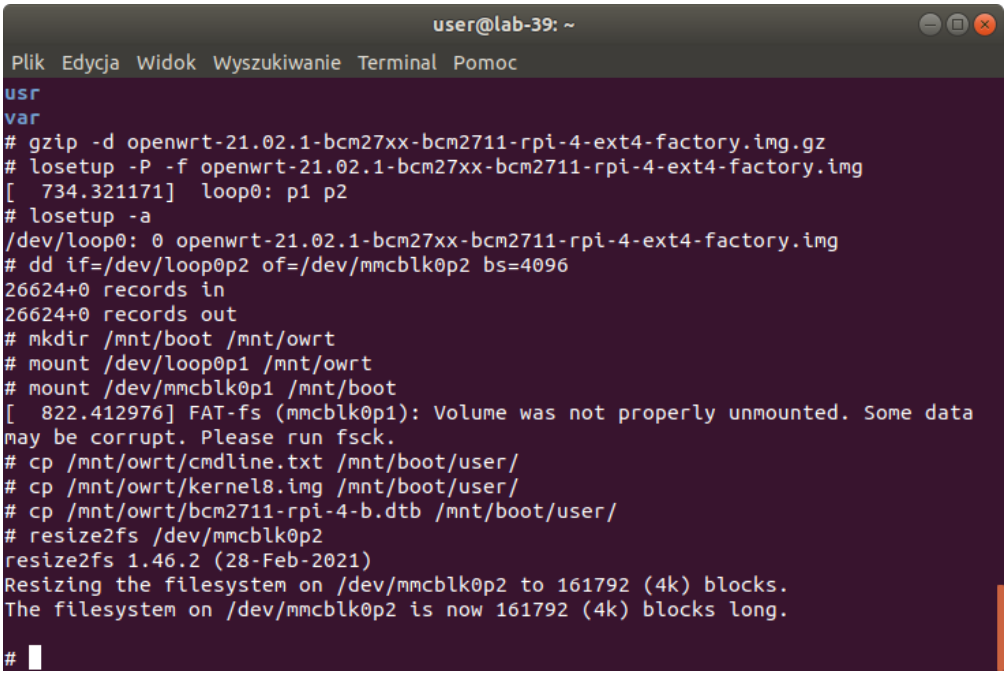
Na początku dodaliśmy serwer DNSa na wszelki wypadek do pliku /etc/resolv.conf na systemie ratunkowym



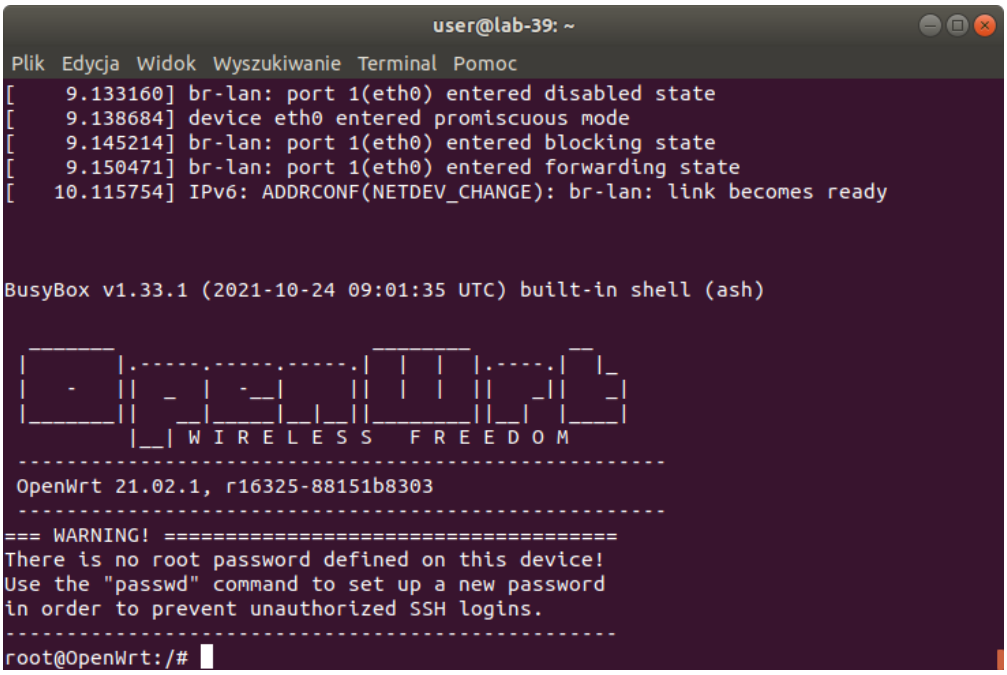
Pobraliśmy obraz systemu przez wget, rozpakowaliśmy go, ustawiliśmy obraz systemu jako urządzenie "loop" i sprawdziliśmy jego nazwę



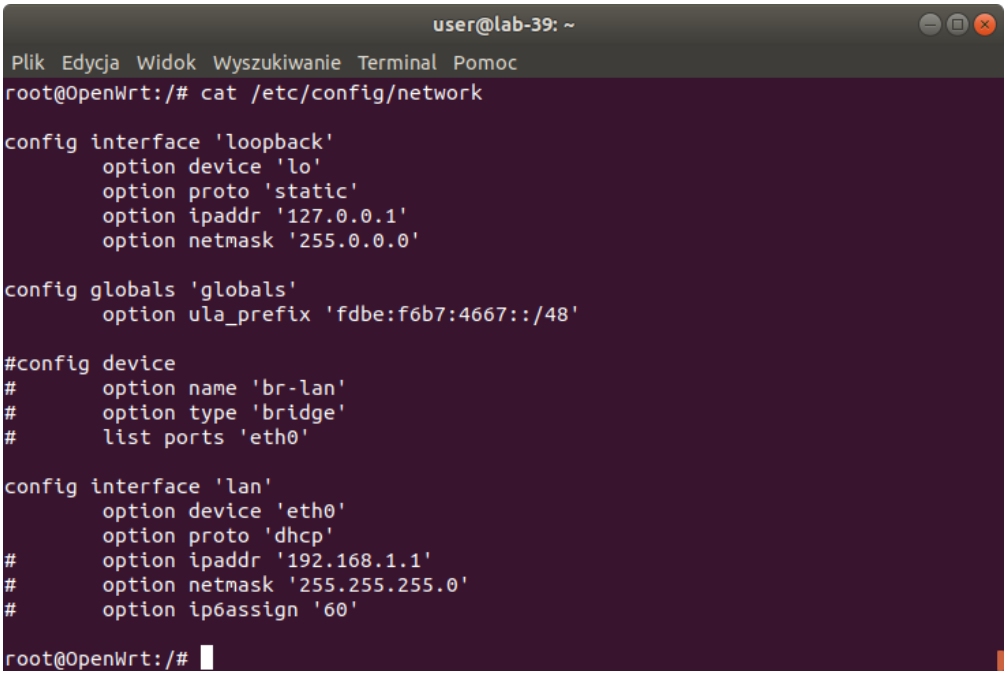
Następnie zamontowaliśmy partycję pierwszą obrazu OpenWRT oraz pierwszą partycję karty SD w odpowiednie miejsca, następnie przerzuciliśmy wymagane pliki do zbudowania pełnego obrazu systemu na karte SD i powiększyliśmy system plików



Odpaliliśmy system



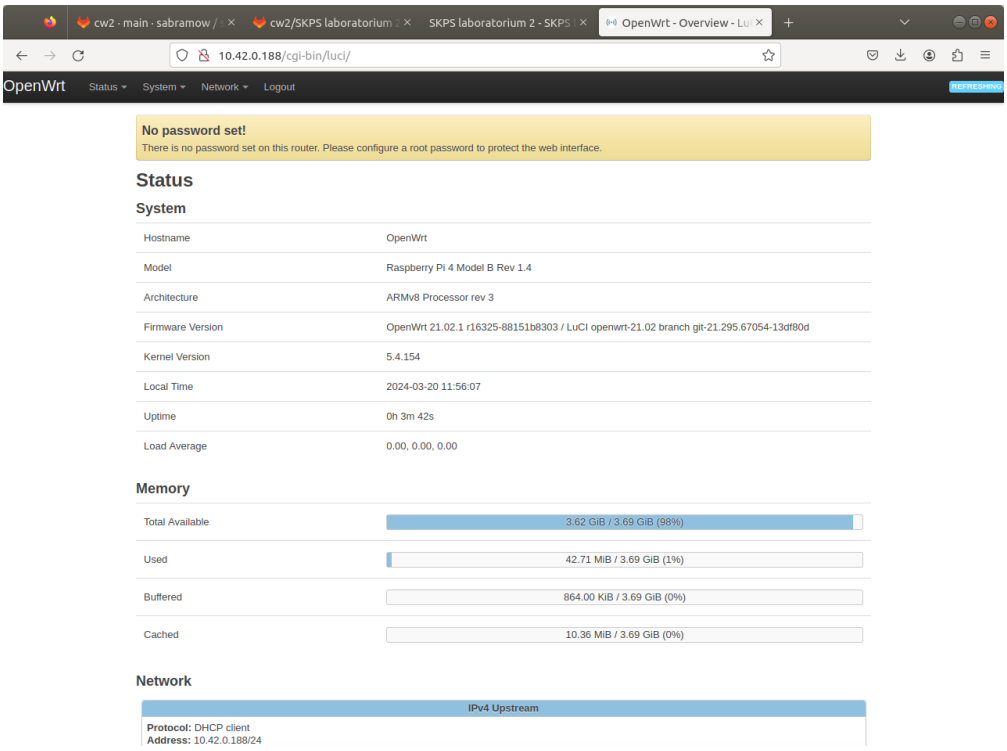
Następnie zmieniliśmy konfigurację sieci, aby system nie próbował być routerem



Potem, zrestartowalismy ustawienia sieci poleceniem

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

Sprawdziliśmy czy jest dostęp do naszego systemu przez HTTP



Następnie zainstowaliśmy 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
```

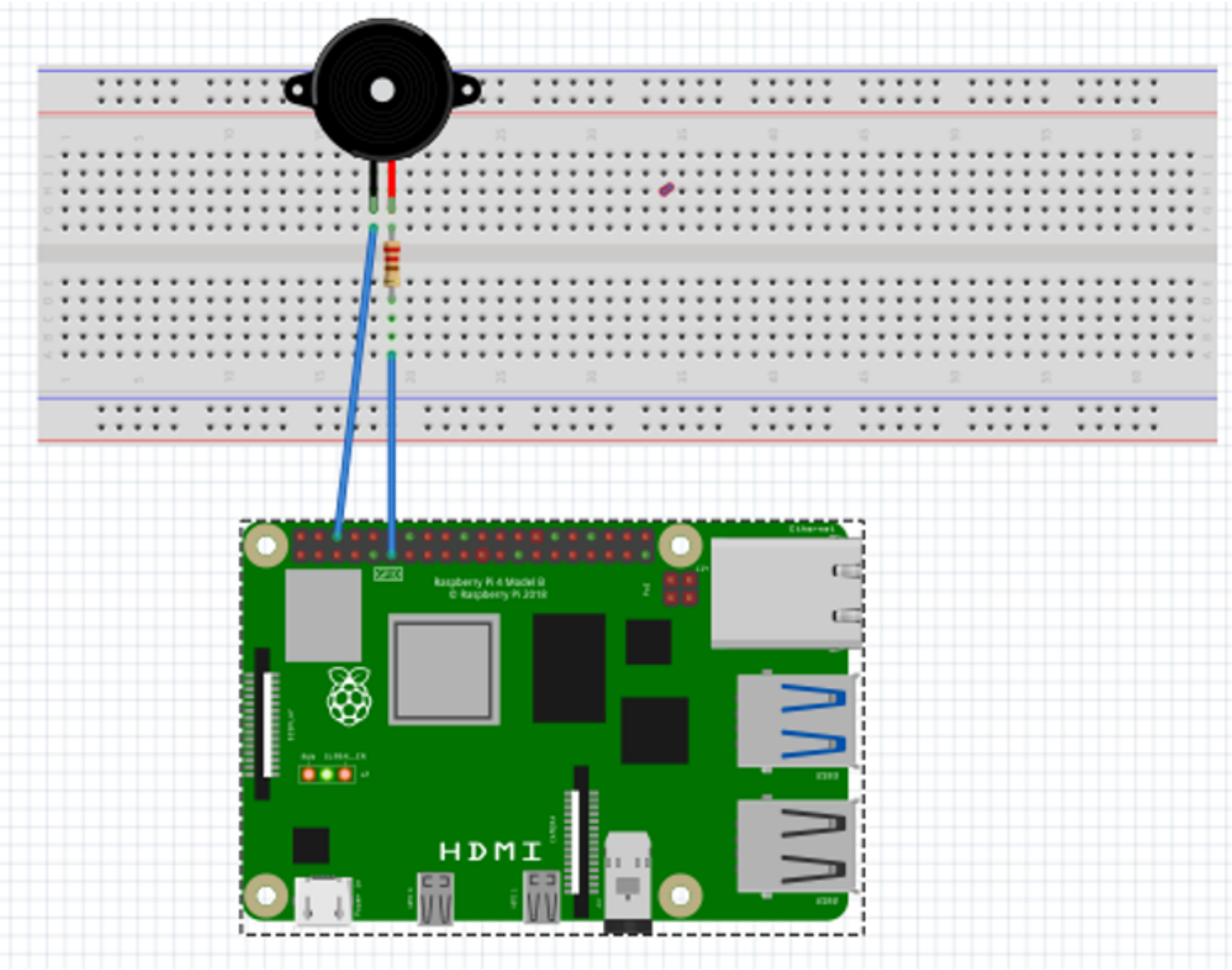
Zad. 3 (plik gpio_in.py)

```
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 podłączenie

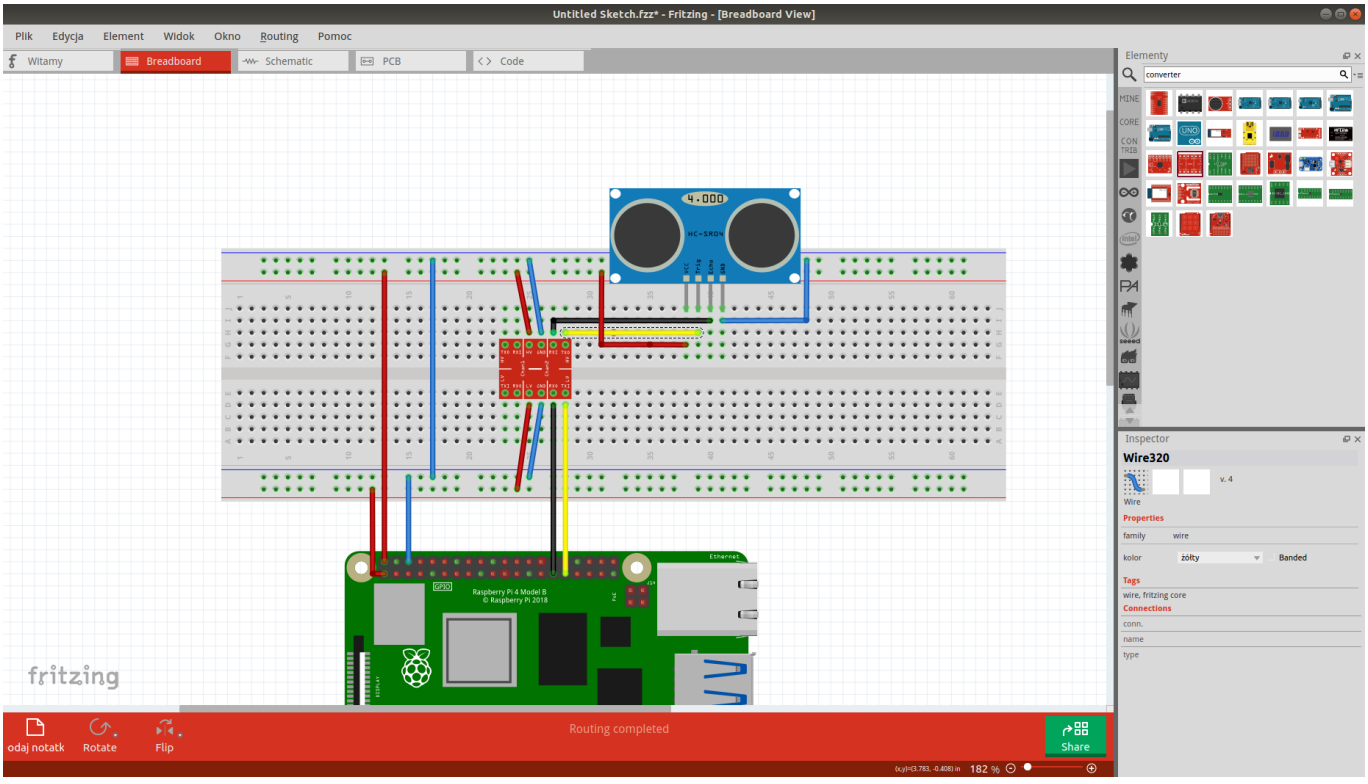



```
import time
import gpio4

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 podłączenie




```
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
```