Systemy Komputerowe w Sterowaniu i Pomiarach

Laboratorium 2

Aleksander Kruk
Michał Sobiech

31 października 2023

Zadanie 1 Uruchomienie OpenWRT na RPI 4B

Użyte komendy

`...`	Uruchomienie Raspberrypi4 i podłączenie minicoma jak na pierwszym laboratorium
`wget https://downloads.openwrt.org/releases/21.02.1/targets/bcm27xx/bcm2711/ openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz`	Pobranie systemu z openwrt
`gzip -d openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz`	Rozpakowanie systemu
`losetup -P -f openwrt-21.02.1-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz`	Rozstawienie pętli urządzeń blokowych
`losetup -a`	Sprawdzenie, że przydzielono nam loop0
`dd if=/dev/loop0p2 of=/dev/mmcblk0p2 bs=4096`	Kopiowanie partycji drugiej obrazu OpenWRT na kartę SD
`mkdir /mnt/boot /mnt/owrt`	
`mount /dev/loop0p1 /mnt/owrt`	Zamontowanie pierwszych partycji obrazu i karty SD
`mount /dev/mmcblk0p1 /mnt/boot`	
`cp /mnt/owrt/cmdline.txt /mnt/boot/user/`	Kopiowanie zawartości świeżo zamontowanej partycji
`cp /mnt/owrt/kernel8.img /mnt/boot/user/`	
`cp /mnt/owrt/bcm2711-rpi-4-b.dtb /mnt/boot/user/`	
`resize2fs /dev/mmcblk0p2`	Poszerzenie rozmiaru systemu plików
`poweroff + przytrzymanie przycisku`	System pomyślnie się uruchomił

Konfiguracja sieci

`vi /etc/config/network`	Modyfikacja ustawień sieciowych zgodnie z treścią skryptu
`/etc/init.d/network reload`	Wczytanie nowej konfiguracji

Uzupełnienie pakietów

```
opkg update
opkg install python3
opkg install i2c-tools
opkg install gpiod-tools
opkg install spi-tools
opkg install python3-pip
opkg install python3-gpiod
opkg install python3-smbus
pip install gpio4
```

Zadanie 2 Podłączenie podstawowych akcesoriów i ich obsługa przez sysfs oraz za pomocą Pythona

GPIO - wyjście dla LED

```
import gpio4
import time

led_gpio_no = 27
iteration_count = 10
interval_time = 1

led_pin = gpio4.SysfsGPIO(led_gpio_no)
led_pin.export = True
led_pin.direction = 'out'
led_pin.value = 0

for i in range(iteration_count):
    led_pin.value = 1
    time.sleep(interval_time)
    led_pin.value = 0
    time.sleep(interval_time)

led_pin.export = False
```

GPIO - wyjście dla LED z płynną zmianą jasności

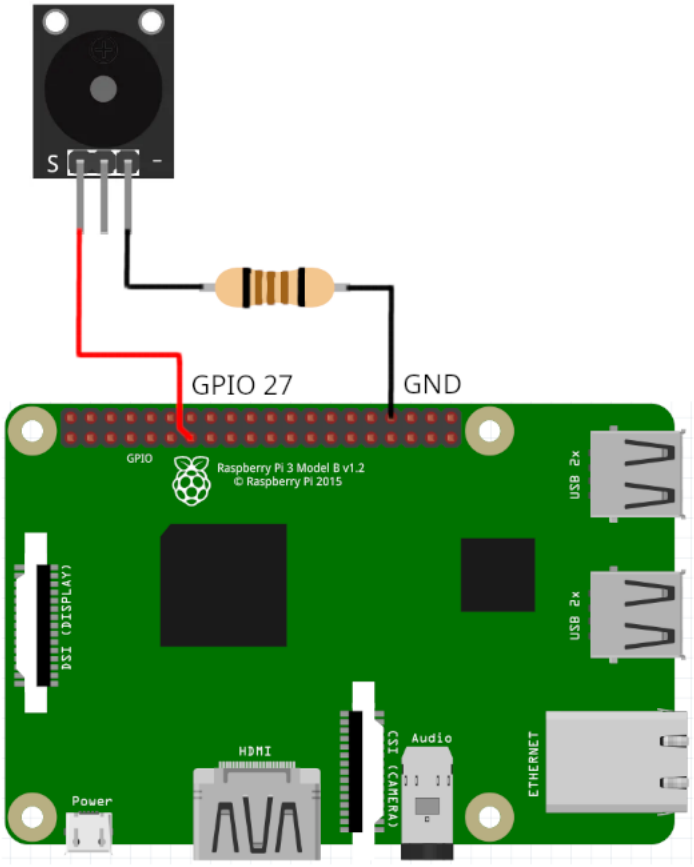
```
import gpio4
import math
import time


def init_pin(gpio_no):
    pin = gpio4.SysfsGPIO(gpio_no)
    pin.export = True
    pin.direction = 'out'
    pin.value = 0
    return pin

def generate_pwm_signal(duty_cycle, period, pin):
    on_time = duty_cycle * period
    off_time = period - on_time
    pin.value = 1
    time.sleep(on_time)
    pin.value = 0
    time.sleep(off_time)

def generate_sin_signal(frequency, time, pin):
    ticks_per_s = 100
    tick_length = 1/ticks_per_s
    for tick_no in range(int(time * ticks_per_s)):
        intensity = (math.sin(frequency * tick_no * tick_length) + 1) / 2
        generate_pwm_signal(intensity, tick_length, pin)

if __name__ == '__main__':
    gpio_no = 27
    frequency = 3
    time_of_working = 10
    pin = init_pin(gpio_no)
    generate_sin_signal(frequency, time_of_working, pin)
    pin.export = False
```

GPIO - wejście

```
import gpio4
import time

def init_output_pin(pin_no):
    pin = gpio4.SysfsGPIO(pin_no)
    pin.export = True
    pin.direction = 'out'
    pin.value = 0
    return pin

def init_input_pin(pin_no):
    pin = gpio4.SysfsGPIO(pin_no)
    pin.export = True
    pin.direction = 'in'
    return pin

if __name__ == '__main__':
    wait_time = 0.5
    output_pin_number = 27
    input_pin_number = 18
    output_pin = init_output_pin(output_pin_number)
    input_pin = init_input_pin(input_pin_number)
    while True:
        if input_pin.value == 0:
            output_pin.value ^= 1
            time.sleep(wait_time)
    else:
        output_pin.export = False
        input_pin.export = False
```

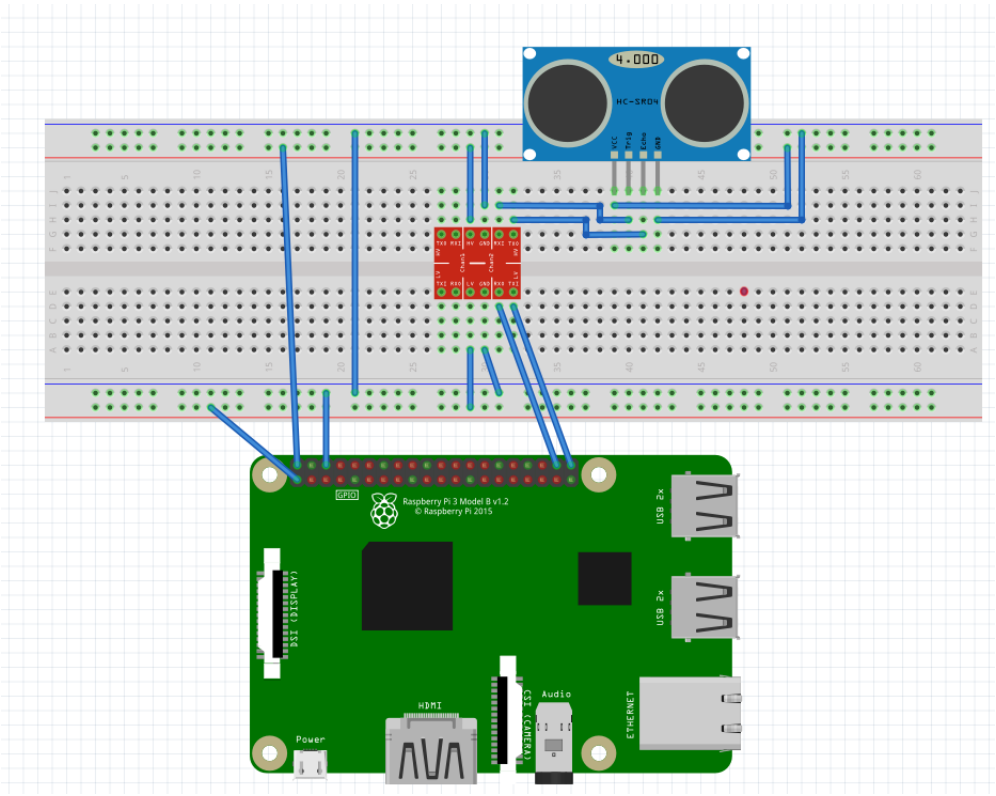
GPIO - wyjście PWM, buzzer pasywny

Niestety nie udało nam się wykonać zdjęcia schematu do tego zadania w trakcie laboratorium, dlatego narysowaliśmy go w gimpie.



```
import gpio4
import time
def init_output_pin(gpio_no):
    pin = gpio4.SysfsGPIO(gpio_no)
    pin.export = True
    pin.direction = 'out'
    return pin

def generate_note_pwm(duty_cycle, note_time, frequency, pin):
    period = 1/frequency
    start_time = time.time()
    while True:
        time_elapsed = time.time() - start_time
        if time_elapsed + period > note_time:
            return
        generate_pwm_signal(duty_cycle, period, pin)

def generate_pwm_signal(duty_cycle, period, pin):
    on_time = duty_cycle * period
    off_time = period - on_time
    pin.value = 1
    time.sleep(on_time)
    pin.value = 0
    time.sleep(off_time)

if __name__ == '__main__':
    key_frequencies = [
        65.41,
        73.42,
        82.41,
        87.31,
        98.00,
        110.00,
        123.47,
        130.81,
        146.83,
        164.81,
        174.61,
        196.00,
        220.00,
        246.94
    ]
    key_frequencies_next_octave = [4 * f for f in key_frequencies]
    output_pin_no = 27
    key_time = 0.25
    duty_cycle = 0.5
    output_pin = init_output_pin(output_pin_no)

    for f in key_frequencies + key_frequencies_next_octave:
        generate_note_pwm(duty_cycle, key_time, f, output_pin)
```

GPIO - ultradźwiękowy czujnik odległości




```python
import gpio4
import time

def init_input_pin(gpio_no):
    pin = gpio4.SysfsGPIO(gpio_no)
    pin.export = True
    pin.direction = 'in'
    return pin

def init_output_pin(gpio_no):
    pin = gpio4.SysfsGPIO(gpio_no)
    pin.export = True
    pin.direction = 'out'
    pin.value = 0
    return pin

def measure_distance(trig_pin, echo_pin, sound_speed):
    trig_pin.value = 1
    start_time = time.time()
    time.sleep(0.001)
    trig_pin.value = 0

    end_time = None
    while True:
        if echo_pin.value == 1:
            end_time = time.time()
            break

    time_elapsed = end_time - start_time

    distance = time_elapsed * sound_speed / 2
    return distance

if __name__ == '__main__':
    speed_of_sound = 343
    echo_gpio_np = 27
    trig_gpio_no = 26

    trig_pin = init_output_pin(trig_gpio_no)
    echo_pin = init_input_pin(echo_gpio_np)

    while True:
        print(measure_distance(trig_pin, echo_pin, speed_of_sound))
```