4

PWM dimer LED

U ovoj vežbi je potrebno realizovati dimer za LED koristeći softversku i hardversku realizaciju impulsno-širinske modulacije (eng. *pulse width modulation* – PWM) upotrebom *wiringPi* i BCM2835 biblioteka.

Impulsno-širinska modulacija je veoma rasprostranjena modulacija u oblasti elektrotehnike generalno, zbog svoje osobine da srednja vrednost signala zavisi od odnosa impuls-pauza, tako da se upravljanjem tim odnosom, vrši upravljanje srednjom vrednosti. Detaljnu analizu različitih vrsta i matematičkih modela impulsno-širinske modulacije moguće je pogledati u poglavlju [15]. U smislu potreba ove laboratorijske vežbe PWM se koristi da bi se srednja vrednost napona na LED menjala i time menjao intenzitet osvetljaja čime se realizuje LED dimer.

Raspberry Pi računar ima na 40 pinskom GPIO priključku dostupna 2 pina na kojima se može dobiti PWM izlaz. Reč je o PWM kanalu 0 i on se može koristiti na GPIO12 (pin 32, A+/B+/Pi2 i CM modeli) i GPIO18 (pin12, svi modeli). Drugi kanal postoji, ali je deljen sa audio izlazom pa nije pogodan za korišćenje.

4.1. Primeri za vežbu

U ovom poglavlju će biti dato nekoliko primera za vežbu kao i nekoliko zadataka sa ciljem da se isprobaju i nauče načini realizacije PWM na RPi.

Primere u kojima se koristi *wiringPi* biblioteka neophodno je kompajlirati sa dodatnim prekidačima za uključenje ***pthread*** i ***wiringPi*** biblioteka.

Npr.: `gcc -o primer1 primer1.c -lwiringPi -lpthread`

Primere u kojima se koristi BCM2835 biblioteka neophodno je kompajlirati sa dodatnim prekidačem za uključenje **BCM 2835** bibliteke.

Npr.: `gcc -o primer3 primer3.c -lbcm2835`

4.1.1. SoftPWM upotrebom *wiringPi* biblioteke

U ovom poglavlju će biti dat primer koji koristi softversku realizaciju PWM upotrebom *wiringPi* biblioteke koji se može realizovati na bilo kom izlaznom pinu.

```
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
char i;

int main() {

     wiringPiSetup();

     if (softPwmCreate(28, 0, 100)!=0)
          printf("Doslo je do greske.");

     while (1) {
          for (i=0; i<101; i++) {
               softPwmWrite(28, i);
               delay(10);
          }
     }
}
```

U prethodnom primeru su korišćene 3 funkcije za rad sa *wiringPi* bibliotekom.

Funkcija `wiringPiSetup` služi da se omogući inicijalizacija GPIO koristeći *wiringPi* brojeve pinova. Postoje i druge opcije označavanja (BCM, fizički brojevi pinova), a za detalje pogledati dokumentaciju *wiringPi* biblioteke.

Funkcija `softPwmCreate` omogućava kreiranje softverske realizacije PWM na odabranom pinu i ima format:

```
int softPwmCreate (int pin, int initV, int pwmOpseg)
```

gde su:
`pin` – pin na GPIO
`initV` – početna vrednost
`pwmOpseg` – opseg PWM od 0 do 100

Funkcija `softPwmWrite` omogućava ispis PWM vrednosti na pin i ima format:

```
int softPwmWrite (int pin, int vrednost)
```

gde su:

`pin` – pin na GPIO
`vrednost` – PWM vrednost koja će se poslati na pin.

4.1.2. HardPWM upotrebom *wiringPi* biblioteke

Naredni primer realizuje pojačanje i smanjenje osvetljaja LED upotrebom hardverske realizacije PWM i *wiringPi* biblioteke. Ovaj primer nije moguće probati sa DVK512 dodatnom pločicom!

```
#include <wiringPi.h>
#include <stdio.h>

int main(void) {

     int pin, i;

     if (wiringPiSetup() == -1)
          exit(1);

     pinMode(1, PWM_OUTPUT);

     for(;;) {
          for (i=0; i<1024; ++i) {
               pwmWrite(1, i);
               delay(10);
          }

          for (i=1024; i>=0; --i) {
               pwmWrite(1, i);
               delay(10);
          }
     }

     return 0;
}
```

Funkcija `pinMode` omogućava podešavanje moda odabranog pina i ima format:
`int pinMode (int pin, int pinmod)`

gde su:
`pin` – pin na GPIO
`pinmod` – može imati jednu od 4 vrednosti: `INPUT`, `OUTPUT`, `PWM_OUTPUT` ili `GPIO_CLOCK`. Paziti da samo *wiringPi* pin 1 (BCM_GPIO 18, fizički pin 12 na GPIO priključku) podržava PWM izlaz, a jedino *wiringPi* pin 7 (BCM_GPIO 4, fizički pin 7 na GPIO priključku) podržava izlazni CLOCK mod.

Funkcija `pwmWrite` se koristi kada želimo da koristimo hardversku realizaciju PWM na pinu i ima format:

```
int pwmWrite (int pin, int vrednost)
```

gde su:

`pin` – pin na GPIO

`vrednost` – PWM vrednost koja će se poslati na pin u opsegu 0-1024.

4.1.3. Kontrola dimera tasterom

U narednom primeru je realizovan PWM LED dimer koji menja intenzitet osvetljaja u 5 nivoa na pritisak tastera. Pored toga, u ovom primeru ćemo se podsetiti i softverskog diferenciranja (primena radi jednostruke detekcije promene stanja tastera) što je rađeno u ranijim predmetima.

```
// NA DVK512 ploči se koristi KEY0 i LED3,
// a LED0 svetli dok taster nije pritisnut

#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>

int main() {

     int i=0, taster_pritisnut=0;
     int r_value;

     if (wiringPiSetup() == -1)
          exit(1);

     pinMode(21, INPUT);
     pinMode(25, OUTPUT);

     digitalWrite(25, HIGH);

     if (softPwmCreate(28, 0, 100) != 0)
          exit (2);

     i = 0;
     taster_pritisnut = 0;

     while (1) {
          r_value = digitalRead(21); // procitaj KEY0
          digitalWrite(25, r_value);
```

```
          //cekamo na pritisak tastera
          if (!r_value && !taster_pritisnut) {
               taster_pritisnut = 1;

               i += 20;
               if (i > 100) i = 0;

               printf("i = %d\n", i);
               fflush(stdout);

               softPwmWrite(28, i);
          }

          // cekamo da se taster otpusti
          if (r_value) taster_pritisnut = 0;
     }
     return 0;
}
```

Funkcije `digitalRead` i `digitalWrite` služe za čitanje i ispis na navedenom pinu.

4.1.4. Realizacija dimera upotrebom BCM2835 biblioteke

U narednom primeru će biti realizovan dimer upotrebom hardverskog PWM i BCM2835 biblioteke.

```
// PWM kontrola GPIO pinova upotrebom BCM2835 biblioteke
// i hardverskog PWM-a za dimovani blink LED
// gcc -o pwm4 pwm4.c -l bcm2835
// sudo ./pwm4
#include <bcm2835.h>
#include <stdio.h>

// PWM izlaz je na pin 12 na RPi2 GPIO prikljucku
// sto je BCM pin 18 u ALT fun broj 5
// ovo je jedini PWM pin slobodan na RPi GPIO
#define PIN 18

// koristi PWM kanal 0
#define PWM_CHANNEL 0

// max range PWM signala
#define RANGE 1024
```

```
int main(int argc, char **argv)
{
    if (!bcm2835_init())
        return 1;

    // Setuj izlazni pin u Alt Fun 5,
    // da bi PWM kanal 0 bio izlaz

    bcm2835_gpio_fsel(PIN, BCM2835_GPIO_FSEL_ALT5);

    // Clock divider = 2048
    // Sa divider = 2048 i RANGE = 1024,
    // u MARKSPACE modu,
    // frekvencija ponavljanja impulsa
    // ce biti 19.2MHz/2048/1024= oko 18Hz

bcm2835_pwm_set_clock(BCM2835_PWM_CLOCK_DIVIDER_2048);
    bcm2835_pwm_set_mode(PWM_CHANNEL, 1, 1);
    bcm2835_pwm_set_range(PWM_CHANNEL, RANGE);

    // realizacija dimera promenom PWM odnosa u
    // opsegu 1/RANGE i (RANGE-1)/RANGE

    int direction = 1;
    int data = 1;
    while (1)
      {
        if (data == 1)
          direction = 1;
        else if (data == RANGE-1)
          direction = -1;
        data += direction;
          bcm2835_pwm_set_data(PWM_CHANNEL, data);
        bcm2835_delay(10);
      }
    bcm2835_close();
    return 0;
}
```

Detalje oko podešavanja PWM koristeći BCM2835 biblioteku moguće je pogledati u PWM delu opšte dokumentacije ove biblioteke na [9]. Rezultat rada programa je moguće vizuelno ispratiti povezivanjem LED ili pregledati dobijen PWM signal povezivanjem sa osciloskopom.

4.2. Zadaci

4.2.1. Zadatak 1

Koristeći osciloskop pogledati dobijene PWM signale u prethodnim primerima.

4.2.2. Zadatak 2

Koristeći tastere KEY0 i KEY1 na DVK512 ekspanzionoj ploči napisati program u C-u koji radi dimovanje LED3 tako što se jedan taster koristi za povećanje intenziteta, a drugi za smanjenje i to za 20% (dakle imamo 5 nivoa osvetljaja). Program testirati povezivanjem sa RPi koji ima na sebi DVK512.