Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023/24

Izvještaj za laboratorijsku vježbu br. 5

Analogni izlazi i širinsko-impulsna modulacija (PWM)

Ime i prezime: **Mirza Mahmutović**
Broj index-a: **19320**

12. april 2024. godine

Sadržaj

1 Pseudokod i/ili dijagram toka

1.1 Zadatak 1

```
AnalogIn pot()
PwmOut led()
led.period(T)
while (true) do
      led = pot
      wait()
end_while
```

1.2 Zadatak 2

```
photoRes = ADC(Pin())
leds = [PWM(Pin()),...,PWM(Pin())]
leds.setFrequency(10000)
delta = int(65535/8)
duty = [delta,2*delta,3*delta,4*delta,5*delta,6*delta,7*delta,65535]
update_leds(n_leds)
      for(i = 0; i < 7; i++) do
            if(i < n_leds) then
                  leds[i].duty() = duty[i]
            else
                  leds[i].duty() = +
            end_if
      end_for
scaleVoltage(readVoltage):
      k = 64000/(64000-47900)
      return int(k*readVoltage - 47900*k)

while(true) do
      lum = scaleVoltage(photoRes.read())
      if(lum < 0) then
            update_leds(-1)
            sleep()
      else_if(lum >= 0 and lum < delta) then
```

```
            update_leds(0)
            sleep()
      else_if(lum >= delta and lum < 2*delta) then
            update_leds(1)
            sleep()
      else_if(lum >= 2*delta and lum < 3*delta) then
            update_leds(2)
            sleep()
      else_if(lum >= 3*delta and lum < 4*delta) then
            update_leds(3)
            sleep()
      else_if(lum >= 4*delta and lum < 5*delta) then
            update_leds(4)
            sleep()
      else_if(lum >= 5*delta and lum < 6*delta) then
            update_leds(5)
            sleep()
      else_if(lum >= 6*delta and lum <= 7*delta) then
            update_leds(6)
            sleep()
      else_if(lum >= 7*delta and lum < 8*delta) then
            update_leds(7)
            sleep()
      else
            update_leds(8)
            sleep()
      end_if
      sleep()
end_while
```

1.3 Zadatak 3

```
AnalogOut signal()
float i=0
const float incr=1./50
double fi=0
const float inc_sin=2*PI/50
float signals[13]
```

```
int count = 0
for(float i = PI/6; i<=PI/2; i += float((PI/2-PI/6)/12)) do
      signals[count++] = sin(i);
end_for
while(true) do
      //SIGNAL 1
      for(int i=0; i<13; ++i) do
            signal = signals[i];
            wait_us(21);
      end_for
      for(int i=12; i>0; --i) do
            signal = signals[i];
            wait_us(21);
      end_for

      for(int i=0; i<13; ++i) do
            signal = 1.0-signals[i];
            wait_us(21);
      do
      for(int i=12; i>0; --i) do
            signal = 1.0-signals[i];
            wait_us(21);
      end_for

      //SIGNAL 2
      signal = i
      wait_ns(16000)
      i += incr
      if(i>1) then
            i=0
      end_if
end_while
```

2 Analiza programskog rješenja

2.1 Zadatak 1

Zadatak 1 predstavlja prvi zadatak u radu sa PWM-om. Fokus ovog zadatka je više na razumijevanju perioda i duty cycle-a PWM-a. Potrebno je deklarisati PWM izlaz, te odgovarajući analogni ulaz koji će upravljati duty cycle-om PWM-a. Pored toga potrebno je upaliti enable signal da bi se program ispravno izvršavao na LPC1114ETF sistemu. Prije ulaska u beskonačnu petlju se postavlja period PWM izlaza, a onda se u petlji očitava vrijednost analognog ulaza, i njegova vrijednost postavlja kao vrijednost duty cycle-a. Postoji više metoda za postavljanje perioda PWM signala, dok su u zadatku korištene metode *period_us()* i *period_ms()*. Korištenjem prve metode se period postavljao na 50 mikrosekundi. Ovo omogućava da se intenzitet svjetla mijenja sa manjim promjenama položaja potenciometra u odnosu na period od 500ms koji se postavljao korištenjem druge metode..

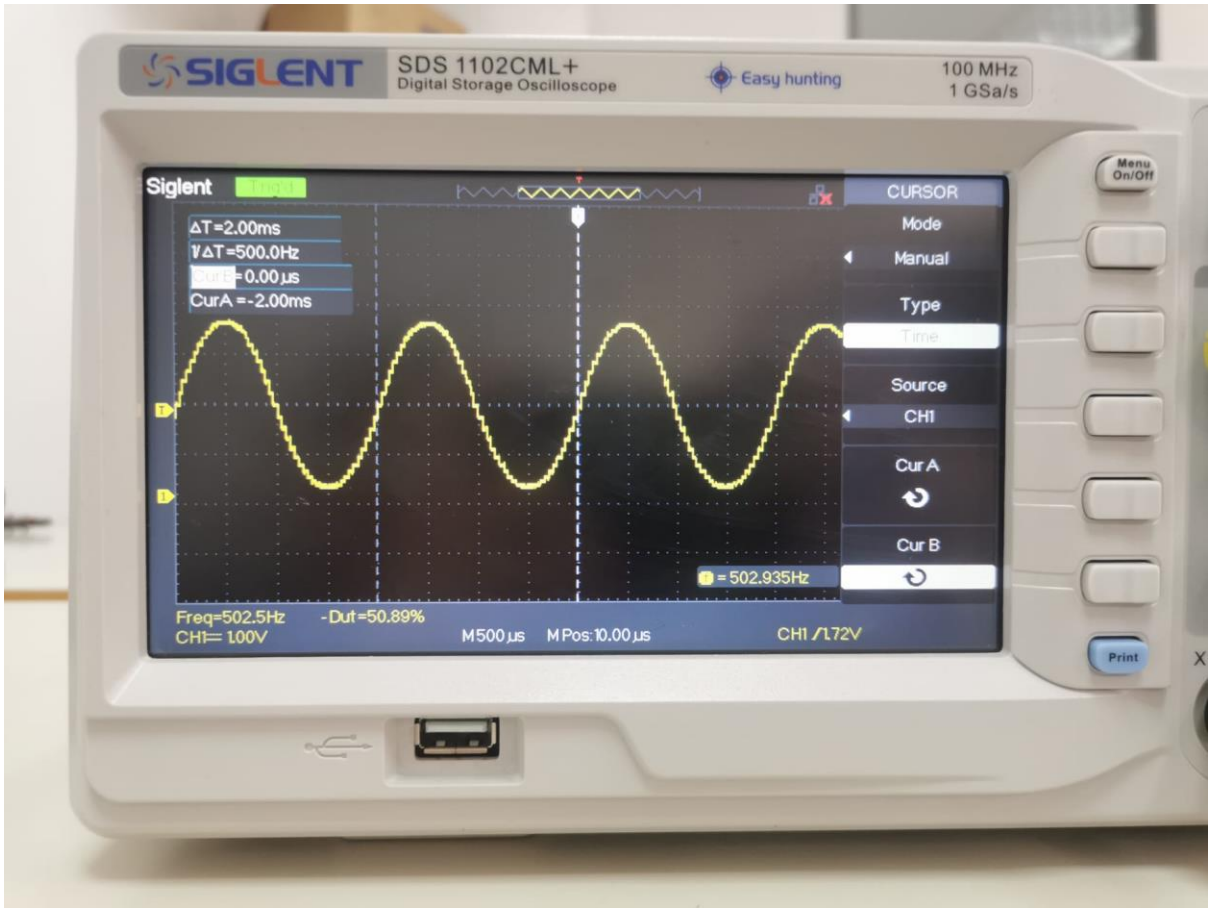
2.2 Zadatak 2

U drugom zadatku je potrebno na sistem povezati fotootpornik preko kojeg će se upravljati sa intenzitetom svjetla svih 8 LED dioda sistema. Diode LED1 do LED6 se postepeno uključuju kako se smanjuje osvjetljenost fotootpornika, dok se za potpunu zatamnjenost fotootpornika pale sve diode. U kodu se prvo deklariše odgovarajući analogni ulaz za fotootpornik, te niz PWM izlaza za 8 LED dioda sistema. Postavlja se frekvencija (period) signala na ovim izlazima, te se računa različit duty cycle za svaku od dioda u nizu. S ovim se postiže razlika u intenzitetu svjetlosti na diodama. U zadatku se koriste i dvije metode *scaleVoltage()* i *update_leds()*. Metoda *scaleVoltage()* se koristi za skaliranje napona na analognom ulazu fotootpornika. Ovo se radi da bi se fotootpornik prilagodio okruženju u kojem se nalazi, jer napon na ulazu fotootpornika nikad nije 0V. Metoda *update_leds()* služi da bi se na osnovu skalirane vrijednosti napona na fotootporniku upalio potrebni broj dioda, postavljanjem njihovih duty cycle-a na odgovarajuće vrijednosti. U beskonačnoj petlji se pomoću metode *scaleVoltage()* računa skalirana vrijednost napona, te na osnovu nje metodi *update_leds()* prosljeđuje broj dioda koje se trebaju upaliti (0 znači da se treba upaliti 1 dioda).

2.3 Zadatak 3

U trećem zadatku je korištenjem analognom izlaza bilo potrebno realizovati jedan od ponuđenih signala. U ovom rješenju su realizovana prva dva signala. Potrebno je prvo deklarisati analogni izlaz, a zatim za realizaciju prvog signala je potrebno izračunati 12 vrijednosti funkcije sin(x) i spremiti ih u niz. U beskonačnoj petlji je dovoljno samo čitati te

vrijednosti korištenjem 4 for petlje jer je sin(x) periodična funkcija. U prvoj petlji se vrijednosti čitaju u uzlaznoj putanji, zatim u drugoj u silaznoj putanji. U trećoj petlji se vrijednosti čitaju u uzlaznoj putanji, međutim uzimaju se njihove komplementarne vrijednosti (1.0 – sin(x)), dok se u četvrtoj petlji vrijednosti ponovo čitaju u silaznoj putanji uzimanjem njihovih komplementarnih vrijednosti. Što se tiče drugog signala, potrebno je prije beskonačne petlje postaviti početnu vrijednost signala (0.0), te korak inkrementacije (1.5). Zatim se u beskonačnoj petlji vrijednost signala povećava dok ne dostigne vrijednost 1, kada se resetuje nazad na početnu vrijednost.



Slika 1: Generisani 1. signal

Slika 2: Generisani 2. signal

3 Korišteni hardverski resursi

Na ovoj vježbi su korišteni sljedeći razvojni sistemi:

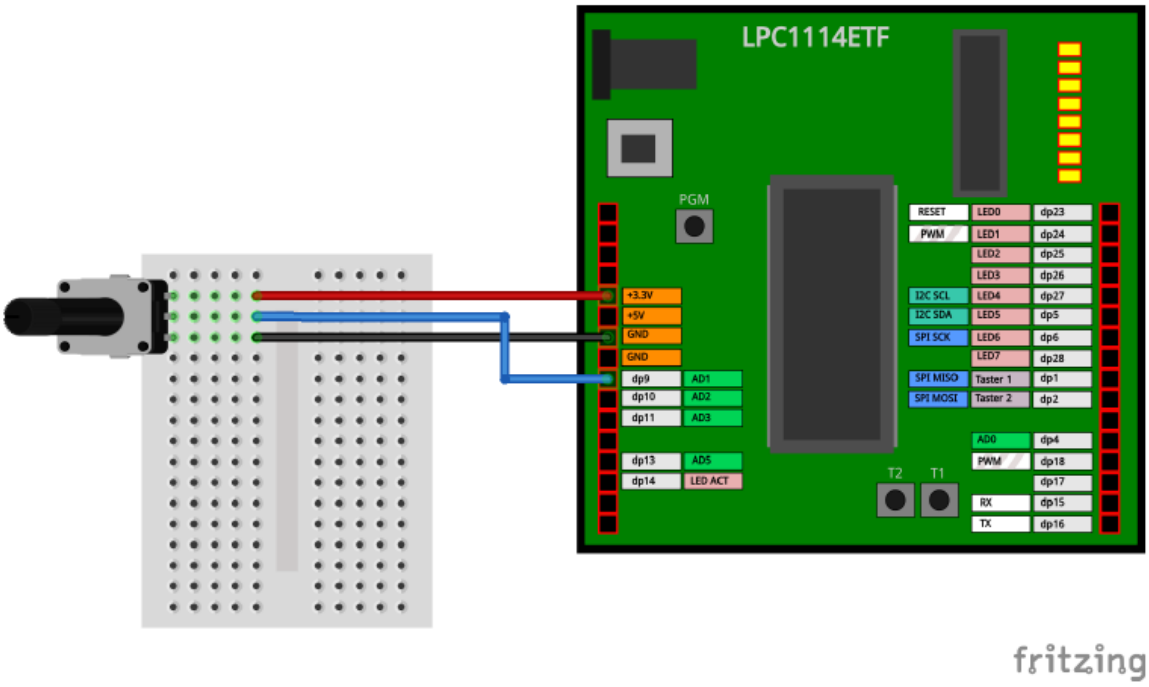
- *LPC1114ETF,* baziran na mikrokontroleru *NXP LPC1114FN28*
- *picoETF*, baziran na mikrokontroleru *RP2040*
- *FRDM-KL25Z*
- potenciometar
- fotootpornik
- osciloskop

Na sistemu *LPC1114ETF* su za ovu vježbu korišteni sljedeći elementi:

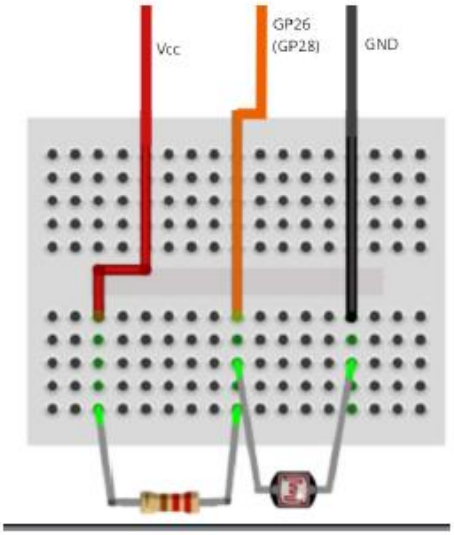
- 1x LED dioda

Na sistemu *picoETF* su za ovu vježbu korišteni sljedeći elementi:

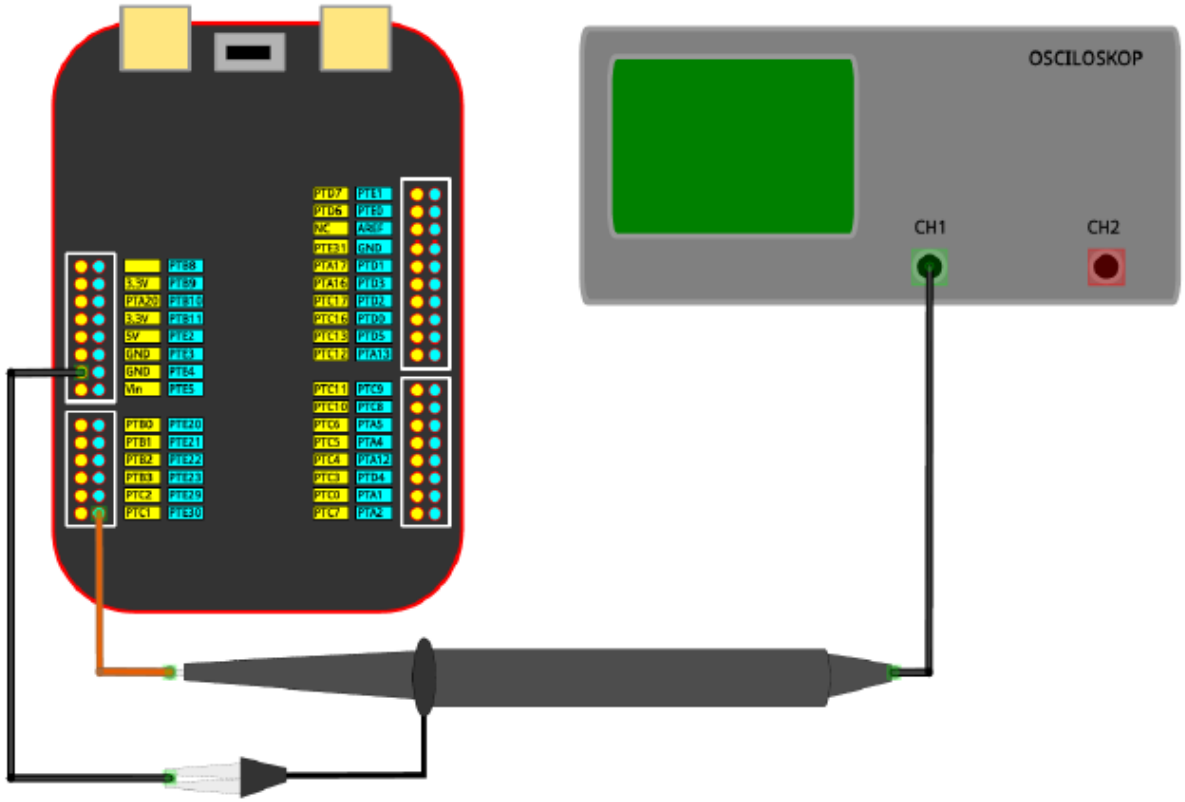
- 8x LED dioda



Slika 3: Povezivanje potenciometra na LPC1114ETF sistem



Slika 4: Povezivanje fotootpornika na picoETF sistem



Slika 5: Povezivanje osciloskopa na FRDM-KL25Z sistem

4 Zaključak

Ova vježba je za cilj imala da se studenti upoznaju sa načinom korištenja analognih izlaza i PWM izlaza. Kroz ove zadatke upoznali smo se sa pojmom „duty cycle“, te kako se pomoću njega može upravljati izlaznim naponom. Upoznali smo se i sa periodom (frekvencijom) PWM signala. U ovim konkretnim zadacima smo ova dva pojma koristili za upravljanje intenzitetom osvjetljenja LED dioda. Pored toga, u 3. zadatku smo se upoznali i sa principima rada sa analognim izlazima i konstrukcijom signala uz pomoć istih. Nakon uspješno odrađenih postavljenih zadataka, cilj vježbe je postignut.

5 Prilog

5.1 Zadatak 1/izvorni kod

```
#include "mbed.h"
#include "lpc1114etf.h"


AnalogIn pot(AD1);
```

```
PwmOut led(LED1);
BusOut leds(LED0, LED2, LED3, LED4, LED5, LED6, LED7);
DigitalOut E(LED_ACT);

int main() {
    E=0;
    leds = 0;
    //led.period_ms(500);
    led.period_us(50);

    while (1) {
        led.write(pot.read());
        printf("%f \n",pot.read());

        wait_us(10000);
    }
}
```

5.2 Zadatak 2/izvorni kod

```
from machine import Pin, PWM, ADC
from time import sleep
sleep(0.1) # Wait for USB

# Initialize the photoresistor
photoRes = ADC(Pin(28))

# Initialize the LEDs as PWM outputs
leds = [PWM(Pin(i)) for i in range(4, 12)]

# Set the PWM frequency, 1kHz should do
for led in leds:
    led.freq(10000)

# 1/8 of the max duty cycle
delta = int(65535/8)

# Set the duty cycle values for LEDs (gradient)
```

```
duty = [delta, 2*delta, 3*delta, 4*delta, 5*delta, 6*delta, 7*delta, 65535]
# For good measure
print(duty)

# Turn on as many LEDs as necessary
def update_leds(n_leds):
  for i in range(0, 8):
    if i <= n_leds:
      leds[i].duty_u16(duty[i])
    else:
      leds[i].duty_u16(0)

# Scale the input voltage.
#The voltage drop across the photoresistor is never 0V
# so it needs to be scaled accordingly. Performs a linear transformation.
def scaleVoltage(readVoltage):
  k = 64000/(64000-47900)
  return int(k*readVoltage - 47900*k)



while True:
  # Read off the voltage from the photoresistor,
  # scale it and print it for reference
  lum = scaleVoltage(photoRes.read_u16())
  print(lum)

  # Depending on the voltage, turn on as many LEDs as necessary:
  if lum < 0:
    update_leds(-1)
    sleep(0.1)
  elif lum >= 0 and lum < delta:
    update_leds(0)
    sleep(0.1)
  elif lum >= delta and lum < 2*delta:
    update_leds(1)
    sleep(0.1)
```

```
  elif lum >= 2*delta and lum < 3*delta:
    update_leds(2)
    sleep(0.1)
  elif lum >= 3*delta and lum < 4*delta:
    update_leds(3)
    sleep(0.1)
  elif lum >= 4*delta and lum < 5*delta:
    update_leds(4)
    sleep(0.1)
  elif lum >= 5*delta and lum < 6*delta:
    update_leds(5)
    sleep(0.1)
  elif lum >= 6*delta and lum <= 7*delta:
    update_leds(6)
    sleep(0.1)
  elif lum >= 7*delta and lum < 8*delta:
    update_leds(7)
    sleep(0.1)
  else :
    update_leds(8)
    sleep(0.1)

  sleep(0.01)
```

5.3 Zadatak 3/izvorni kod

```
#include "mbed.h"
#define PI 4*atan(1)

AnalogOut signal(PTE30);

int main(){
    float i=0;
    const float incr=1./25;
    double fi=0;
    const float inc_sin=2*PI/50;

    float signals[13];
```

```
int count = 0;
for(float i = PI/6; i<=PI/2; i += float((PI/2-PI/6)/12))
    signals[count++] = sin(i);


while(true) {

    for(int i=0; i<13; ++i) {
        signal = signals[i];
        wait_us(21);
    }

    for(int i=12; i>0; --i){
        signal = signals[i];
        wait_us(21);
    }

    for(int i=0; i<13; ++i) {
        signal = 1.0-signals[i];
        wait_us(21);
    }
    for(int i=12; i>0; --i) {

        signal = 1.0-signals[i];
        wait_us(21);
    }



    //SIGNAL 2
    //inicijalizirati i na 0 prije petlje
    //increment na 1./50
    /*signal = i;
    wait_ns(16000);
    i+=incr;
```

```
            if(i>1)  i=0;*/


        }
    }
```