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: **Ismar Muslić**
Broj index-a: **19304**

09. april 2024.

Sadržaj

1 Pseudokod

Zbog genealne kompleksnosti implementacije svih zadataka, pseudokod je dat kao intuitivna ideja kako pristupiti rješavanju problema, dok su konkretne implementacije svih zadataka date u prilogu ovog izvještaja.

1.1 Zadatak 1

```
PWM led
led.set_period(time)
while(true)
      led.set_duty(potentiometer.read())
      sleep(0.1)
```

1.2 Zadatak 2

```
PWM LEDs
AnalogIn photoResistor

LEDs.set_frequency()
LEDs.set_duty()

delta = 65535/8

lum = scale_voltage(photoResistor.read())

if (lum < 0)
  ugasi_diode()

else if (lum >= 0 and lum < delta)
  update_leds(0)

else if (lum >= delta and lum < 2*delta)
  update_leds(1)

else if (lum >= 2*delta and lum < 3*delta)
  update_leds(2)

else if (lum >= 3*delta and lum < 4*delta)
  update_leds(3)

else if (lum >= 4*delta and lum < 5*delta)
  update_leds(4)

else if (lum >= 5*delta and lum < 6*delta)
  update_leds(5)

else if (lum >= 6*delta and lum <= 7*delta)
  update_leds(6)

else if (lum >= 7*delta and lum < 8*delta)
  update_leds(7)

else
  update_leds(8)
  sleep(0.1)
```

1.3 Zadatak 3

```
AnalogOut signal

while(true)
      generate_signal()
```

2 Analiza programskog rješenja

2.1 Zadatak 1

U prvom zadatku bilo je potrebno na razvojni sistem LPC1114ETF spojiti potenciometar na pin za analogni ulaz (dp24), te sa istim kontrolisati intenzitet osvjetljenja LED diode koja je na sistemu ugrađena. Ovo se postiglo na način da je ulaz sa potenciometra deklarisan kao objekat klase AnalogIn, te se s njega očitavala vrijednost pomoću metode read(). S obzirom na to da se duty cycle PWM objekta može podešavati u istom 16 – bitnom opsegu pomoću metode write(), dovoljno je bilo samo učitavati kontinualno vrijednost očitanu sa AnalogIn ulaza kao duty cycle PWM izlaza. Time je postignuto da se intenzitet svjetla LED diode mijenja u odnosu na otklon potenciometra.

2.2 Zadatak 2

Zadatak je zahtijevao implementaciju VU – metra na LED diodama LED0 – LED7 razvojnog sistema picoETF, s tim da isti pokazuje intenzitet osvijetljenosti fotootpornika koji je na sistem povezan kao analogni ulaz. Također je bilo potrebno podesiti diode tako da svijetle od LED0 do LED7 postepeno rastućim intenzitetom.

Implementirano programsko rješenje uključuje deklarisanje ADC ulaza za fotootpornik, kao i niza od 8 elemenata koji predstavljaju PWM objekte, odnosno LED diode. Varijabla 'delta' služi kao pomoćna varijabla u velikom dijelu programa, s obzirom na to da je potrebno 16-bitne opsege kako duty cycle-a, tako i ulaznog napona sa fotootpornika podijeliti na 8 dijelova. Dijeljenje na tačno 8 dijelova se vrši iz razloga što imamo tačno 8 LED dioda, koje ne samo da moraju da svijetle postepeno jačim intenzitetom, nego i da se određen broj njih pali ili gasi u odnosu na očitanu vrijednost napona sa ADC ulaza, pa nam stoga poznavanje vrijednosti 65535/8 olakšava implementaciju.

Tako je na osnovu ove varijable inicijaliziran niz duty[] koji predstavlja vrijednosti duty cycle-a koje svaka od LED dioda treba da ima da bi se postigao efekt gradijenta s intenzitetom njihovog osvjetljenja. Ova vrijednost se na PWM objekte i postavlja procedurom update_leds(n_leds). Ova funkcija prima int parametar koji predstavlja broj dioda koje je potrebno upaliti (određuje se na osnovu intenziteta svjetlosti na fotootpotniku), te uz pomoć jedne for-petlje podešava odgovarajući duty cycle za onoliko dioda koliko je proslijeđeno kao parametar. Ukoliko je petlja izvan tog opsega, ta dioda ne treba ni biti upaljena, pa se stoga njen duty cycle postavlja na 0.

Funkcija scaleVoltage služi, kako joj ime i nalaže, za skaliranje napona. Naime, s obzirom na to da otpor fotootpornika ima eksponencijalnu karakteristiku, te da pad napona na njemu nikad ne može biti 0V, ne možemo ni pretpostaviti da će se vrijednost koju vraća metoda read_u16() kretati tačno između 0 i 65535. Stoga je potrebno zapamtiti vrijednosti očitanja sa fotootpornika pri stanju kada je u potpunosti osvijetljen i u stanju kada je u potpunosti zatamnjen, te na osnovu tih vrijednosti izvršiti linearizaciju opsega očitanja da bi se isto moglo adekvatno koristiti.

Posljednji dio koda predstavlja samo očitavanje napona sa fotootpornika, te paljenje onoliko LED dioda koliki intenzitet osvjetljenja samo očitanje predstavlja. Ovdje je prikazani intenzitet na LED diodama proporcionalan sa očitanjem sa fotootpornika, u linearnom odnosu.

2.3 Zadatak 3

U zadatku 3 bilo je potrebno koristeći analogni izlaz na sistemu FRDM-KL25Z kreirati diskretizirane oblike određenih signala sa osnovnom frekvencijom od 500 Hz, te realizacijom jednog perioda sa 50 uzoraka, što znači da je širina u kojoj signal treba da zadržava vrijednost 40μs. Isto se postiglo koristeći AnalogOut objekat kojem se vrijednost podešavala na osnovu toga koju vrijednost signal treba da ima u datom vremenskom trenutku.

Prvobitna ideja rješenja ovog zadatka uključivala je za 5 različitih signala pisanje 5 različitih procedura koje će na izlazu kreirati jedan period za svoj respektivni signal. Međutim, ovo se pokazalo kao neadekvatno rješenje iz razloga što samo pozivanje ovih procedura iz main funkcije zahtijeva dodatno vrijeme koje nije svaki put jednako, te ga je stoga dosta teško uzeti u obzir prilikom iscrtavanja signala. Naravno, iz tog razloga signal nije korektno ni iscrtavan jer je bilo skoro nemoguće pravilno podesiti sleep time.

Prešlo se zatim na podešavanje vrijednosti signala direktno u main funkciji, što je uvelike olakšalo podešavanja sleep vremena, mada je bitno naglasiti da je ponovno i ono bilo drastično manje od zadanih 40μs.

Uspješno su na ovaj način generisani 'Sawtooth' signal sa uzlaznom ivicom, kao i sinusni signal, otklonjen za +0.5 po ordinati.

Kod sinusnog signala je također prva ideja bila pozivanje bibliotečke funkcije sin(), ali ovo rješenje je bilo neadekvatno iz jednakih, pa i jačih razloga kao što je to bilo za pisanje posebnih procedura. Naime, vrijeme koje je potrebno da funkcija sin() izračuna return vrijednost je također previše veliko da bi se signal mogao realizirati potrebnom frekvencijom.

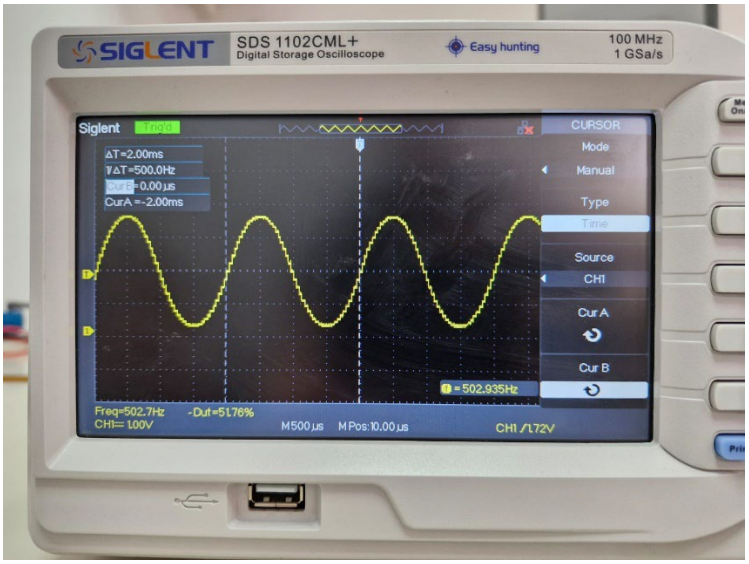
Rješenje koje je implementirano zasniva se na računanju potrebne vrijednosti sinusa za opseg od 0 – π/2, te smještanju istih vrijednosti u niz. Zatim su se te vrijednosti očitavale u adekvatnom poretku da bi se signal pravilno iscrtavao. Ovo rješenje je daleko bolje iz razloga što se računanje samih vrijednosti funkcije sinus vrši samo jednom, i to za ¼ perioda. Čitanje vrijednosti iz niza se izvršava veoma brzo, pa je bilo lakše i podesiti vrijeme čekanja za ovaj signal.

3 Korišteni hardverski resursi

Za potrebe laboratorijske vježbe 5 korišteni su razvojni sistemi LPC1114ETF, picoETF, kao i FRDM-KL25Z. Na istima su korištene i integrisane LED diode, kao i od dodatne opreme potenciometar, fotootpornik, te osciloskop.

ULAZI	IZLAZI
Potenciometar (analogni, DP9 na LPC1114ETF)	LED1 (digitalni PWM, ugrađena na LPC1114ETF)
Fotootpornik (analogni, GP28 na picoETF)	LED0 – LED7 (digitalni PWM, ugrađene na picoETF)

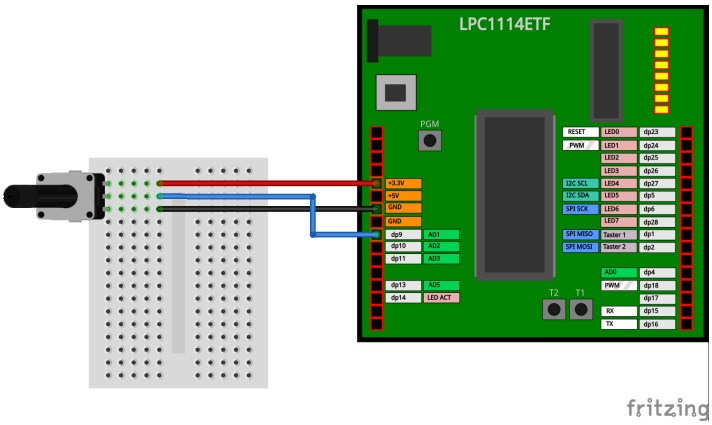
Pomoću osciloskopa su očitavani generisani signali na sistemu FRDM-KL25Z. Izgled očitanih signala je na slikama koje slijede:



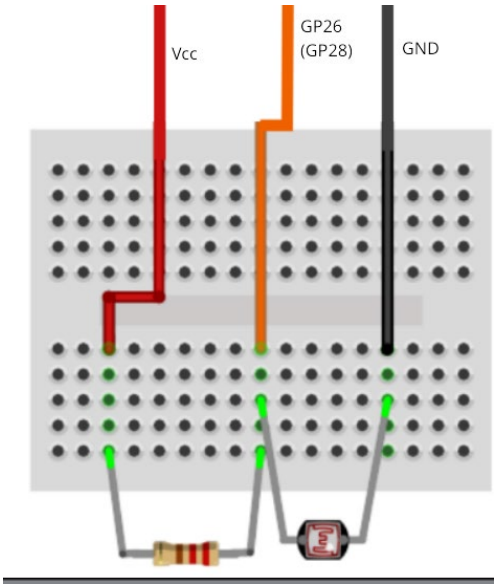
Slika 1 – Sinusni signal

Slika 2 – 'Sawtooth' signal sa uzlaznom ivicom

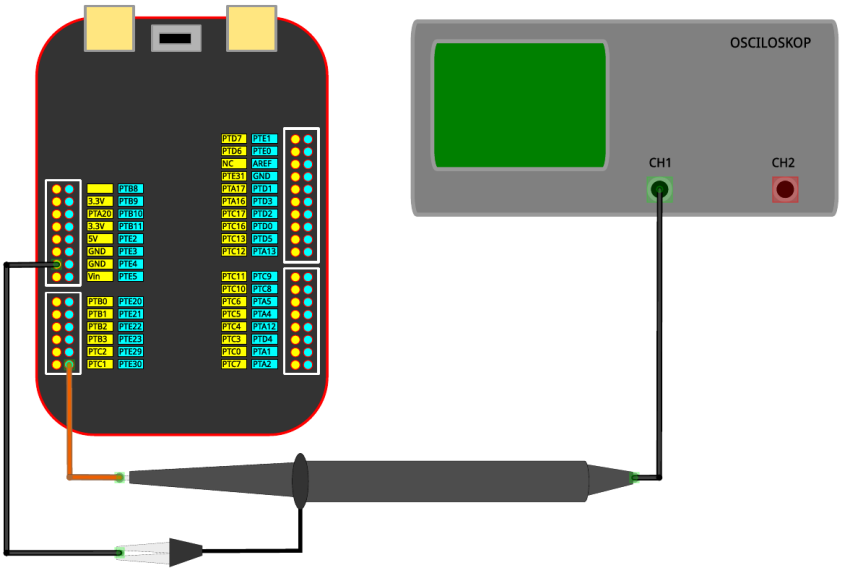
Načini povezivanja dodatnih komponenata su na shemama ispod:



Slika 3 – Povezivanje potenciometra na LPC1114ETF



Slika 4 – Povezivanje fotootpornika na picoETF



Slika 5 – Povezivanje osciloskopa na izlaz sa FRDM-KL25Z

4 Zaključak

Najveći i ujedno jedini izazov vezan za ovu laboratorijsku vježbu bio je podešavanje vremena čekanja kod generisanja diskretnih signala u zadatku 3. Naime, svaki aspekt izvršenja programa na mikrokontroleru zahtijeva određeno vrijeme koje se ne može adekvatno predvidjeti, pa je stoga bilo potrebno modificirati prvobitne ideje o programskom rješenju, koje bi jedino bile funkcionalne u idealnom okruženju.

Osim izloženog, nije bilo daljnih problema niti poteškoća prilikom izvođenja laboratorijske vježbe. Cilj vježbe bio je upoznavanje sa analognim izazima i PWM-om, što je uspješno i postignuto.

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);
    //kod perioda u us se intenzitet svjetla mijenja
    //sa manjim promjenama polozaja potenciometra, te nema flickeringa

    while (1) {
        led.write(pot.read()); //podesen duty
        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 is maybe overkill
for led in leds:
    led.freq(1000)

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

# Initialize 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 by setting duty
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

Napomena: zakomentirani su dijelovi koda koji generišu druge signale, dok je „vidljivi“ kod onaj za generisanje sinusoidnog signala. Svi signali su radi uštede vremena izvršavanja generisani u main() funkciji, pa je toga bilo potrebno komentirati ostale dok se radi na jednom.

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

AnalogOut signal(PTE30);

//uzlazna rampa
void signal2() {
    for(float i = 0.0; i < 1.0; i += 1./25) {
        signal = i;
        wait_ns(20500);
    }
}

//silazna rampa
void signal3() {
    for(float i = 1.0; i>0; i -= 1./25) {
        signal = i;
        wait_us(39);
    }
}

//triangularni
void signal4() {
    for(float i = 1.0; i>0; i -= 1./12) {
        signal = i;
        wait_us(20);
    }
    for(float i = signal; i < 1; i += 1./12) {
        signal = i;
        wait_us(20);
    }
}

//sinusoida
void signal1() {
    for(float i = PI/6; i<=PI/2; i += float((PI/2-PI/6)/12)) {
        signal = sin(i);
        wait_us(20);
    }
    for(float i = PI/2; i<=PI; i += float(PI/50)) {
        signal = sin(i);
        wait_us(20);
    }
    for(float i = PI; i<7*PI/6; i += float(PI/66)) {
        signal = sin(i);
        wait_us(20);
```

```
    }
}

// |sin(x)|
void signal5() {
    for(float i = 0; i <= PI; i += float(PI/50)) {
        signal = sin(i);
        wait_us(20); //podesiti frekvenciju
    }
    /*for(float i = PI/2; i<=PI; i += float(PI/24)) {
        signal = sin(i);
        wait_us(20);
    }*/
}

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) {
        //signal 1
        //signal1();
        /*signal=sin(fi)/2+0.5;
        fi+=inc_sin;
        if(fi>=2*PI) fi=0;
        wait_ns(2); *///treba pogledati sta je sa frekv


        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;*/

        //SIGNAL 3
        //inicijalizirati i na 1 prije petlje
        /*signal = i;
        wait_ns(15550);
        i-=incr;
        if(i<0) i=1;*/

        //SIGNAL 4
        //signal4();

        //SIGNAL 5
        //signal5();

    }
}
```