Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023 / 24

Izvještaj za laboratorijsku vježbu br. 3

Višebitni digitalni ulazi i izlazi

Ime i prezime: **Ismar Muslić**
Broj index-a: **19304**

26. mart 2024. godine

Sadržaj

1 Pseudokod

Napomenuo bih da je u pseudokodu 3. zadatka na jednostavan način predstavljeno podešavanje vrijednosti perioda signala. Ovo je učinjeno da bi se izbjegla nepotrebna „ifologija“ koja je u kodu morala biti upotrijebljena. Smatram da za nju u pseudokodu nije bilo potrebe, s obzirom na to da se jednostavno u ovisnosti od unosa podešava vrijednost varijable, te da pseudokod tu funkcionalnost vjerodostojno predstavlja.

1.1 Zadatak 1

```
diode[]
while(true)
      unos = ocitajUnos()
      if(unos == broj)
        dioda[broj].ukljuci()
      else if(unos == C)
        ugasiSveDiode()
```

1.2 Zadatak 2

```
while(true)
      if(taster1 pritisnut)
        brojac++
      else if(taster2 pritisnut)
        brojac--
      else if(taster3 pritisnut)
        brojac = 0
      else if(taster4 pritisnut)
        toggleAutomatskoBrojanje()

      if(brojac izvan opsega)
        vratiBrojacUOpseg()

      prikaziBrojacNaDisplayu()
```

1.3 Zadatak 3

```
while(true)
      unos = ocitajUnos()
      if(unos == A)
        toggleSignal()
      else if(unos == broj 0-8)
        podesiAdekvatanPeriod(broj)
      else if(unos == C)
        period++
        if(period izvan opsega)
          podesiPeriodUnutarOpsega()
      else if(unos == D)
        period--
        if(period izvan opsega)
          podesiPeriodUnutarOpsega()
      prikaziPeriodNaDisplayu()
```

2 Analiza programskog rješenja

2.1 Zadatak 1

U zadatku 1 potrebno je bilo programirati mikrokontroler LPC1114ETF da radi sa 4x4 matričnom tastaturom, te da na osnovu pritisnutih tipki na tastaturi pali LED sa rednim brojem jednakim pritisnutoj tipki, odnosno da postavi sve diode na 0 pritiskom na tipku „C“. Rješenje zadatka implementirano je u programskom jeziku mbed (C++), koristeći 4x4 matricu elemenata tipa `char` koji predstavljaju tipke na tastaturi. Unos sa tastature se očitava na način da se uzastopno očitavaju redovi tastature, te se na istima traži da li je pritisnuta određena kolona. Na taj način se saznaje koja tipka na tastaturi je pritisnuta, te se odgovarajuća `char` vrijednost očitava u programu iz prethodno spomenute matrice. Ukoliko je očitana cifra od 1 do 8, pali se odgovarajuća LED, dok ukoliko je očitano slovo C, sve diode se gase. Inicijalno gašenje dioda se vrši pomoću `BusOut` objekta, dok su sve ostale manipulacije sa istima odrađene pomoću niza `DigitalOut` objekata. Ovakvi nizovi su također korišteni za očitavanje ulaza sa matrične tastature.

2.2 Zadatak 2

U drugom zadatku tražena je implementacija decimalnog brojača na četverocifrenom 7 – segment display-u, a koji se kontroliše uz pomoć četiri tastera na razvojnom sistemu picoETF. Kontrola 7 – segment display-a je urađena uz pomoć lookup tabele u kojoj su smještene vrijednosti segmenata za svaku cifru od 0 do 9, u binarnoj reprezentaciji. Pojedinačne pozicije od 0 do 4, kao i segmenti cifara i tasteri su svrstani u odgovarajuće nizove `Pin` objekata uz pomoć kojih se sa istim ulazima i izlazima vrše sve softverske manipulacije.

Prikazivanje cifara na display-u je postignuto koristeći nekolicinu pomoćnih funkcija. Funkcija `prikazi_cifru(cifra)` adekvatno uključuje, odnosno isključuje segmente A – G tako da bi cifra proslijeđena kao parametar bila korektno prikazana. Ova funkcija uzima binarnu reprezentaciju njoj proslijeđenog parametra iz lookup tabele, te zatim elemente niza `segmenti[]` u `for` petlji postavlja na odgovarajuće vrijednosti.

Zatim, funkcija `prikazi_poziciju(pozicija)` bira odgovarajuću od 4 cifre na display-u, odnosno, onu koja je proslijeđena kao parametar. Najprije se isključuju sve četiri cifre, a zatim se samo tražena uključuje.

Funkcija `daj_cifru(broj, n)` je jednostavna funkcija koja vraća vrijednost n-te cifre proslijeđenog broja.

Konačno, funkcija `prikazi_broj(broj)` koristi prethodne tri funkcije kao pomoćne da bi efektivno prikazala kompletan broj na 7 – segmentnom display-u. S obzirom na to da, osim što je neefikasno, također je i hardverski nemoguće kontrolisati svaku cifru posebno, utoliko je potrebno osvježavati prikaz na display-u dovoljno brzo da bi ljudskom oku te promjene bile neprimjetne. Tako ova funkcija izdvaja iz trenutne vrijednosti brojača jednu po jednu cifru, te sa sitnim međutrajanjem čekanja osvježava cifre na display-u na način da prikaže svaku od njih u jednoj iteraciji `while()` petlje.

Konačno, u ovisnosti o pritisnutom tasteru onako kako je zadatkom zahtijevano, varijabla `broj` se modificira na odgovarajući način, počev od inicijalizacije nulom. Automatsko

inkrementiranje brojača se vršilo uz pomoć dodatne varijable `count`. Naime, ova varijabla služi da bi se izbrojalo onoliko iteracija koliko je dovoljno za vrijeme od 1 s, s obzirom na vrijeme čekanja glavne `while(1)` petlje. Konkretna implementacija ovog zadatka, kao i svih ostalih je data u prilogu izvještaja.

2.3 Zadatak 3.1

Treći zadatak daje izbor jedne od dvije implementacije, od kojih je u ovom slučaju odabrana prva. Konkretno, potrebno je napisati program koji kreira „četvrtke“, odnosno square – wave signal sa parametrima koji se unose sa matrične tastature. Naime, unosom sa tastature se pokreće ili zaustavlja generacija signala, te se zadaje period istog. Dodatno, na 7 – segmentnom display-u se prikazuje vrijednost trenutnog perioda u milisekundama.

Programsko rješenje implementirano je korištenjem ponovno matrice elemenata tipa `char` koji predstavljaju simbole odnosno vrijednosti koje je moguće unijeti sa tastature, dok su pojedinačni pinovi koji kontolišu kako tastaturu, tako i display, ponovno uvršteni kao elementi odgovarajućih nizova. Na isti način kao i u prethodnim zadacima su implementirane funkcionalnosti kako za display, tako i za matričnu tastaturu, stoga se u te detalje ovdje neće ulaziti.

Glavni program se zasniva na `while(1)` petlji u kojoj se izvršava osnovni funkcionalni kod. Square – wave signal se generiše koristeći PWM izlaz na pinu 19. Inicijalna frekvencija je postavljena na 1kHz. Ukoliko je unos sa tastature slovo „A“, generisanje se uključuje, odnosno isključuje tako da se pri svakom unosu ove vrijednosti mijenja logički toggle koji kontroliše to stanje, te se tako postiže efekat uzastopnog uključivanja i isključivanja pritiskom na tipku. Nadalje, čuvana varijabla je trenutna vrijednost perioda, `T_current`, koja se postavlja na zadatkom zadane vrijednosti, odnosno inkrementira i dekrementira, ovisno o unosu sa tastature. Zatim se na kraju `while(true)` petlje postavlja nova frekvencija PWM izlaza u odnosu na zadani period T, te se aktuelna vrijednost istog prikazuje na display-u. Korištenjem osciloskopa je utvrđeno da je generisani signal ispravan, te da se podešavanje istog pomoću tastature vrši tačno.

3 Korišteni hardverski resursi

U okviru ove vježbe korišteni su razvojni sistemi LPC1114ETF, te picoETF. Dodatno, od opreme je korišteno:

- 8x LED (integrisano na sistemu LPC1114ETF)
- 4x fizički taster (integrisano na sistemu picoETF)
- 4 – cifreni 7 – segmentni display sa zajedničkom anodom
- 4x4 matrična tastatura
- Osciloskop

4 Zaključak

U okviru ove laboratorijske vježbe studenti su se upoznali sa višebitnim digitalnim izlazima u vidu 7 – segmentnog display-a, te višebitnim digitalnim ulazima u vidu matrične tastature. Postojale su sitne poteškoće prilikom testiranja rješenja na hardware – u. Razlog istima je činjenica da su neka od rješenja na raspoloživim simulatorima radila besprijekorno iako su postojali određeni bug-ovi u kodu koji su primijećeni tek prilikom pokretanja programa na mikrokontrolerima. Međutim, nije bilo problema prilikom otklanjanja istih, te je cilj vježbe uspješno postignut.

5 Prilog

5.1 Zadatak 1/Izvorni kod

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

DigitalOut leds[]={LED0, LED1, LED2, LED3, LED4, LED5, LED6, LED7};
BusOut leds_off(LED0, LED1, LED2, LED3, LED4, LED5, LED6, LED7);
char keypad[4][4]={
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}};

DigitalOut rows[4]={dp16, dp15, dp17, dp18};
DigitalIn cols[4]={dp9, dp10, dp11, dp13};

DigitalOut E(LED_ACT);

char readKeypad(){
    for(int i=0; i<4; i++){
        rows[i].write(1);

        for(int j=0; j<4; j++){
            if(cols[j].read()){
                rows[i].write(0);
                return keypad[i][j];
            }
        }

        rows[i].write(0);
    }
    return ' ';
}

int main(){
    E=0;
    //turnLedsOff();
    leds_off=0;
```

```
    char button=' ';
    bool released=true;

    while (true){
        button=readKeypad();
        if(button==' ') released=true;

        if(button!=' ' && released){
            if(button>='1' && button<='8') leds[int(button)-'0'].write(1);
            else if (button=='C') leds_off=0;//turnLedsOff();

            released=false;
        }
    }
}
```

5.2 Zadatak 2/Izvorni kod

```
from machine import Pin
import time
time.sleep(0.1) # Wait for USB to become ready

cifre = {
    0: 0b1000000,
    1: 0b1111001,
    2: 0b0100100,
    3: 0b0110000,
    4: 0b0011001,
    5: 0b0010010,
    6: 0b0000010,
    7: 0b1111000,
    8: 0b0000000,
    9: 0b0010000,
    10: 0b1111111
}

mjesto = [Pin(m, Pin.OUT) for m in range(4, 8)]
segmenti = [Pin(s, Pin.OUT) for s in range(8, 15)]
tasteri = [Pin(i, Pin.IN) for i in range(4)]

# prikazi cifru na jednom mjestu
def prikazi_cifru(cifra):
    binarno = cifre[cifra]
    for i in range(7):
        segmenti[i].value(binarno & 1)
        binarno = binarno >> 1

# biranje pozicije ručno
def prikazi_poziciju(pozicija):
    for i in range(4):
        mjesto[i].value(1)
    mjesto[pozicija].value(0)
```

```
# prikazivanje kompletnog broja
def prikazi_broj(broj):
    n = 3
    cifra = daj_cifru(broj, n)
    while (n>=0):
        prikazi_cifru(10)
        prikazi_poziciju(3-n)
        prikazi_cifru(cifra)
        n -= 1
        cifra = daj_cifru(broj, n)
        time.sleep(0.0001)

# izvuci n-tu cifru broja
def daj_cifru(broj, n):
    return broj // 10 ** n % 10

# inicijalizacija
auto = False
broj = 0

# main loop
while True:
    # tasteri
    if tasteri[0].value():
        broj += 1
        while (tasteri[0].value()):
          continue

    elif tasteri[1].value():
        broj -= 1
        while (tasteri[1].value()):
            continue

    elif tasteri[2].value():
        broj = 0

    elif tasteri[3].value():
        auto = not auto
        count = 0
        while(tasteri[3].value()):
            continue

    # ukoliko je pritisnut taster za automatsko inkrementiranje
    if auto:
        if count == 720:
            broj += 1
            count = 0
        count += 1

    # broj mora ostati u rasponu 0 - 9999
    if broj > 9999:
```

```
        broj = 0

    if broj < 0:
        broj = 9999

    # prikaži na display-u
    prikazi_broj(broj)
    time.sleep(0.0001)
```

5.3 Zadatak 3/Izvorni kod

Napomena: za PWM output korišten je u programu Pin 28 mikrokontrolera RP2040, radi lakšeg povezivanja osciloskopa.

```
from machine import Pin, PWM
import time
time.sleep(0.1) # Wait for USB to become ready

# matrix keyboard
tipke = [['1', '2', '3', 'A'],
         ['4', '5', '6', 'B'],
         ['7', '8', '9', 'C'],
         ['*', '0', '#', 'D']]

# pinovi na koje je povezana
keypad_rows = [21, 22, 26, 27]
keypad_cols = [0, 1, 2, 3]

# prazni nizovi u koje će se dodijeliti Pin(i, Pin.IN/OUT)
col_pins = []
row_pins = []

# dodjela fizickih pinova
for x in range(0,4):
  row_pins.append(Pin(keypad_rows[x], Pin.OUT))
  row_pins[x].value(1)
  col_pins.append(Pin(keypad_cols[x], Pin.IN, Pin.PULL_DOWN))
  col_pins[x].value(0)

# scankeys() - vraca char koji je procitan s tastature
def scankeys():
  for row in range(4):
    for col in range(4):
      row_pins[row].high()
      key = None

      if col_pins[col].value() == 1:
        print("Tipka: ", tipke[row][col])
        key_press = tipke[row][col]
        return key_press
        time.sleep(0.3)
```

```
    row_pins[row].low()
  return None

# 7 - segment display
cifre = {
    0: 0b1000000,
    1: 0b1111001,
    2: 0b0100100,
    3: 0b0110000,
    4: 0b0011001,
    5: 0b0010010,
    6: 0b0000010,
    7: 0b1111000,
    8: 0b0000000,
    9: 0b0010000,
    10: 0b1111111
}

mjesto = [Pin(m, Pin.OUT) for m in range(4, 8)]
segmenti = [Pin(s, Pin.OUT) for s in range(8, 15)]

# prikazi cifru na jednom mjestu
def prikazi_cifru(cifra):
    binarno = cifre[cifra]
    for i in range(7):
        segmenti[i].value(binarno & 1)
        binarno = binarno >> 1

# biranje pozicije ručno
def prikazi_poziciju(pozicija):
    for i in range(4):
        mjesto[i].value(1)
    mjesto[pozicija].value(0)

# prikazivanje kompletnog broja
def prikazi_broj(broj):
    n = 3
    cifra = daj_cifru(broj, n)
    while (n>=0):
        prikazi_cifru(10)
        prikazi_poziciju(3-n)
        prikazi_cifru(cifra)
        n -= 1
        cifra = daj_cifru(broj, n)
        time.sleep(0.00001)

# izvuci n-tu cifru broja
def daj_cifru(broj, n):
    return broj // 10 ** n % 10

T_current = 0.001
pwm_on = False
```

```
while True:

  unos = scankeys()

  prikazi_broj(int(T_current * 1000))

  if unos == 'A' :
    if pwm_on:
      pwm.deinit()
      pwm_on = False
    else:
      pwm = PWM(Pin(28))
      pwm.freq(int(1.0 / T_current))
      pwm.duty_u16(32767) # 50% duty cycle za simetrican talas
      pwm_on = True
    time.sleep(1)

  elif unos == '0':
    T_current = 0.001
    time.sleep(1)

  elif unos == '1':
    T_current = 0.002
    time.sleep(1)

  elif unos == '2':
    T_current = 0.003
    time.sleep(1)

  elif unos == '3':
    T_current = 0.004
    time.sleep(1)

  elif unos == '4':
    T_current = 0.005
    time.sleep(1)

  elif unos == '5':
    T_current = 0.006
    time.sleep(1)

  elif unos == '6':
    T_current = 0.007
    time.sleep(1)

  elif unos == '7':
    T_current = 0.008
    time.sleep(1)

  elif unos == '8':
    T_current = 0.009
```

```
    time.sleep(1)

  elif unos == '9':
    T_current = 0.01
    time.sleep(1)

  elif unos == 'C':
    T_current += 0.001
    if T_current > 0.01:
      T_current = 0.001
    time.sleep(1)

  elif unos == 'D':
    T_current -= 0.001
    if T_current < 0.001:
      T_current = 0.01
    time.sleep(1)

  if pwm_on:
    pwm.freq(int(1.0/T_current))

  prikazi_broj(int(T_current * 1000))

  time.sleep(0.0001)
```