Univerzitet u Sarajevu
Elektrotehnički fakultet
**Ugradbeni sistemi 2023 / 24**

# Izvještaj za laboratorijsku vježbu br. 4

Analogni ulazi i displeji

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

**2. april 2024. godine**

# Sadržaj

# 1 Pseudokod

Zbog relativno kompleksne implementacije programskih rješenja, pseudokod je dat striktno kao intuitivni uvid u to kako kod zapravo funkcionira. Dovoljno je razumjeti način na koji funkcionira programsko rješenje, a detaljne implementacije su date u prilogu.

## 1.1 Zadatak 1

```
adc potMeter
while(true)
      t = podesi_vrijeme(potMeter)
      trci_desno(t)
      upali_sve_diode()
      sleep(t)
      gasi_lijevo(t)
```

## 1.2 Zadatak 2

```
while(true)
      ocitaj_senzor()
      temperatura = izracunaj_temperaturu()
      napon = izracunaj_napon()
      prikazi_na_displeju(temperatura, napon, vrijeme)
      crtaj_tacku_na_grafu(temperatura)
      sleep(1s)
      vrijeme += 1
```

## 1.3 Zadatak 3

```
Bus leds
analogniUlaz potMeter

while(true)
      leds.upaliOdgovarajuce(potmeter)
      sleep(0.01)
```

## 1.4 Zadatak za dodatne bodove 1

```
ucitaj_displej()
modPregleda = otpor
while(true)
      if(taster1 ukljucen) modPregleda.toggle()
      if(modPregleda == otpor)
        prikazi_otpor()
      else prikazi_napon()
      sleep(0.01)
```

# 2 Analiza programskog rješenja

## 2.1 Zadatak 1

Prvi zadatak zahtijeva implementaciju „trćećeg svjetla“ kojem se vrijeme za koje su diode upaljene, odnosno brzina „trčanja“ ubrzava i usporava korištenjem potencoimetra kao analognog ulaza. Program je potrebno bilo implementirati na razvojnom sistemu picoETF, pa je tako i kodiran u MicroPythonu.

Potenciometar je povezan na pin 28, a inicijaliziran je kao objekat klase ADC. Metoda read_u16() klase ADC vraća vrijednost od 0 – 65535, pa se prema tome izračunava vrijeme za koje su diode upaljene, odnosno ugašene. Podešenje potenciometra (0 – 100kΩ) odgovara vremenskom periodu od 0.1s to 2s, pa je tako period i računat.

Trćeće svjetlo je implementirano na sličan način kao u laboratorijskoj vježbi 2, što je već poznata implementacija u čije detalje se ovdje neće ulaziti. Bitno je naglasiti da je računanje vremena odrađeno prije svakog update-a dioda, čime se postiže da se momentalno čita vrijednost napona s analognog ulaza, te se brzina „trčanja“ dioda na taj način dinamički prilagođava.

Sam update dioda se radi pomoću funkcije update_leds() koja prima trenutnu vrijednost brojaca i objekat leds (niz Pin objekata koji odgovaraju pinovima na koje su povezane diode), te podešava redom vrijednosti dioda u odnosu na binarnu vrijednost brojača, poredanu u listu uz pomoć funkcije decimal_to_binary_8bits().

## 2.2 Zadatak 2

U zadatku 2 bilo je potrebno programirati razvojni sistem picoETF tako da na TFT ILI9341 displeju prikazuje temperaturu očitanu sa LM35 temperaturnog senzora, koji je također povezan na uređaj. Za implementaciju programskog koda korištene su brojne dodatne biblioteke koje su potrebne kako za upravljanje displejem, tako i za lakšu manipulaciju i ispisivanje na istom. Detaljan pregled ovih biblioteka i paketa je dostupan u implementaciji rješenja u prilogu.

Konkretno, display je inicijaliziran na sličan način kao u primjeru korištenja datom u prilogu vježbe. Za ispisivanje temperature i napona, koji su računati sa analognog ulaza, kao i vremena koje se update-uje svake sekunde, korištena je funkcije display.print().

Za crtanje linija grafa, kao i „tačaka“ koje predstavljaju krivulju promjene temperature po jedinici vremena, korištena je biblioteka gfx, koja je adaptirana specifično za ovaj display. Link na biblioteku dat je u prilogu zajedno sa izvornim kodom. Konkretno, linije su crtane koristeći metodu gfx.line(), a graf promjene temperature je iscrtavan kao niz tačaka koje predstavljaju trenutno očitanje temperature sa senzora. Ovo je implementirano metodom gfx.fill_circle(), koja se poziva u svakoj iteraciji glavne while True: petlje, odnosno svake sekunde.

## 2.3 Zadatak 3

Ovaj zadatak tražio je implementaciju VU metra korištenjem 8 LED dioda integrisanih u LPC1114ETF razvojni sistem. Programsko rješenje je implementirano na „školski“ način. Naime, definirani su macro-i koji predstavljaju u hex zapisu broj koji je potrebno proslijediti BusOut objektu (LED diodama). Ovaj objekat kontroliše diode na način da podešava „nivo“, odnosno koliko dioda treba biti upaljeno, onda kada mu se odgovarajući macro proslijedi.

Potenciometar je spojen na pin dp9 kao analogni ulaz. Sa ovog ulaza se očitava float vrijednost u opsegu [0.0 – 1.0]. Ovaj opseg podijeljen je na ukupno 9 podioka, koji odgovaraju stanjima od 0 upaljenih dioda do svih 8. Dakle, ukoliko je očitana vrijednost sa analognog ulaza unutar nekog od datih opsega, odgovarajući broj dioda će biti uključen.

Suštinski, u while(true) petlji se kontinualno očitava vrijednost sa analognog ulaza i na osnovu nje se podešavaju diode kroz nekoliko uslova. Sleep vrijeme između iteracija petlje je postavljeno na veoma malo, da bi se očitanje potenciometra i update dioda mogli raditi bez osjetne odgode.

## 2.4 Zadatak 4 – dodatni bodovi

Konačno, u četvrtom zadatku je potrebno bilo spojiti Nokia display na LPC1114ETF razvojni sistem, kao i jedan potenciometar totalne otpornosti 100 kΩ kao analogni ulaz. Potrebno je bilo na displeju prikazivati trenutno očitanu vrijednost otpora u Ohmima, dok se klikom na taster 1 taj prikaz mijenja, tako da se prikazuje očitani napon.

Realizirano programsko rješenje zasniva se na dvije osnovne funkcije: prikaziNapon() i prikaziOtpor(), kao i pomoćnoj funkciji floatToString(). Funkcija floatToString() koristeći sintaksu programskog jezika C dinamički alocira null – terminirani string u koji upisuje float vrijednost poslanu kao parametar, te pointer na isti string vraća kao rezultat.

Funkcija prikaziOtpor() jednostavno trenutnu vrijednost otpornosti izračunava sa analognog ulaza, te je zapisuje u varijablu tipa float. Ova varijabla se zatim uz pomoć funkcije sprintf() konvertuje u null – terminirani string koji se može poslati metodi printString() za ispis na displeju, klase N5110. Bitna napomena je to da je u konkretnoj implementaciji korišten format za ispis cijelog broja, s obzirom na to da kada se pokuša iskoristiti formater za floating-point broj, ova funkcija ne radi ispravno. Problem je vjerovatno bug ili u mbed.h ili u samoj N5110.h biblioteci, mada ovo nije detaljno ispitano niti potvrđeno. Međutim, budući da se prikazuje vrijednost u opsegu od 0 – 100000Ω, pretjerana decimalna preciznost nije neophodna, stoga je ovaj prikaz zadovoljavajući.

Kod funkcije prikaziNapon() implementacija je nešto drugačija. Ovaj put se koristi pomoćna funkcija floatToString(), čije je djelovanje prethodno opisano. Naime, s obzirom na to da je očitana vrijednost napona između 0.0 – 3.3V, prikaz te vrijednosti kao cjelobrojne (funkcijom sprintf()) nije ni približno zadovoljavajući. Iz tog razloga je i korištena pomoćna funkcija, te je očitanje ispisano na displej sa preciznošću od 6 decimalnih mjesta, ponovno koristeći funkciju printString(). Samo refreshanje display-a se radi pozivom metode clear() nad istim, koja „čisti“ kompletan displej i sve piksele podešava na 0. Ova metoda se poziva prije svakog novog ispisa na ekran, sa dovoljno malim sleep vremenom između poziva za ispis, tako da se refreshanje ne može primijetiti golim okom.

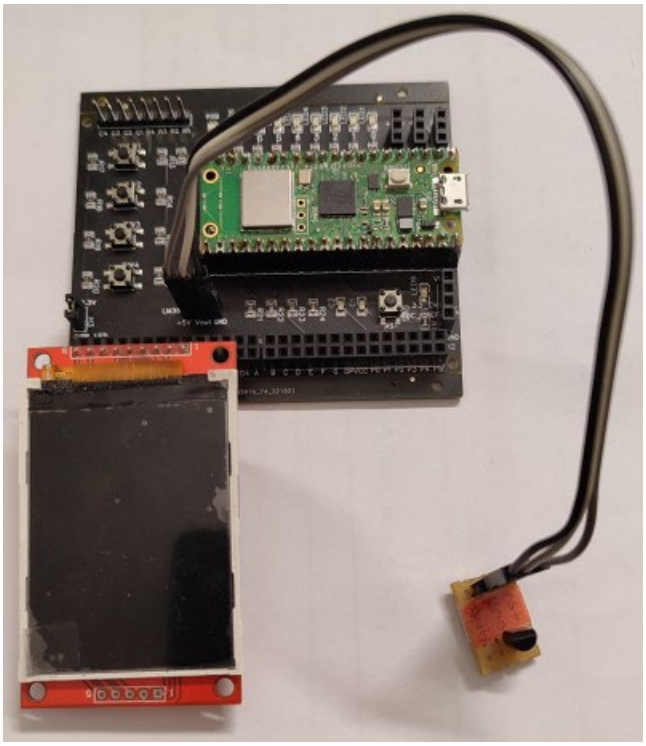
Konačno, u main() funkciji je displej inicijaliziran metodom init(), te se varijabla viewMode postavlja inicijalno na vrijednost false, što odgovara prikazu otpora na displeju. Pritiskom na taster 1, ova vrijednost se toggle-a, za čime se u narednim linijama koda vrši provjera stanja varijable viewMode i prema istom se prikazuje odgovarajuća vrijednost na displeju.

# 3 Korišteni hardverski resursi

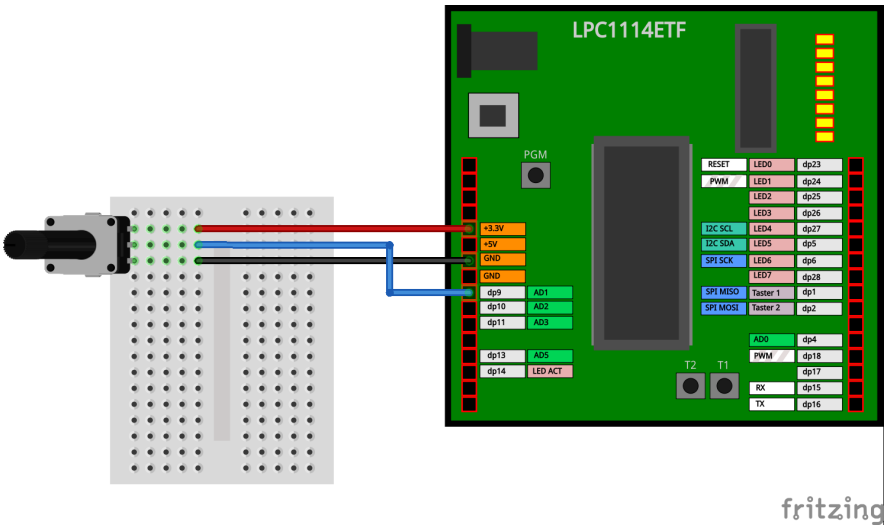
U sklopu ove laboratorijske vježbe korišteni su razvojni sistemi picoETF i LPC1114ETF. Ostala oprema koja je korištena je kako slijedi:

- Nokia N5110 display
- ILI9314 TFT LCD color display
- 1x fizički taster (integrisan na sistemu LPC1114ETF)
- Rotacijski potenciometar (totalne otpornosti 100kΩ)
- LM35 analogni temperaturni senzor

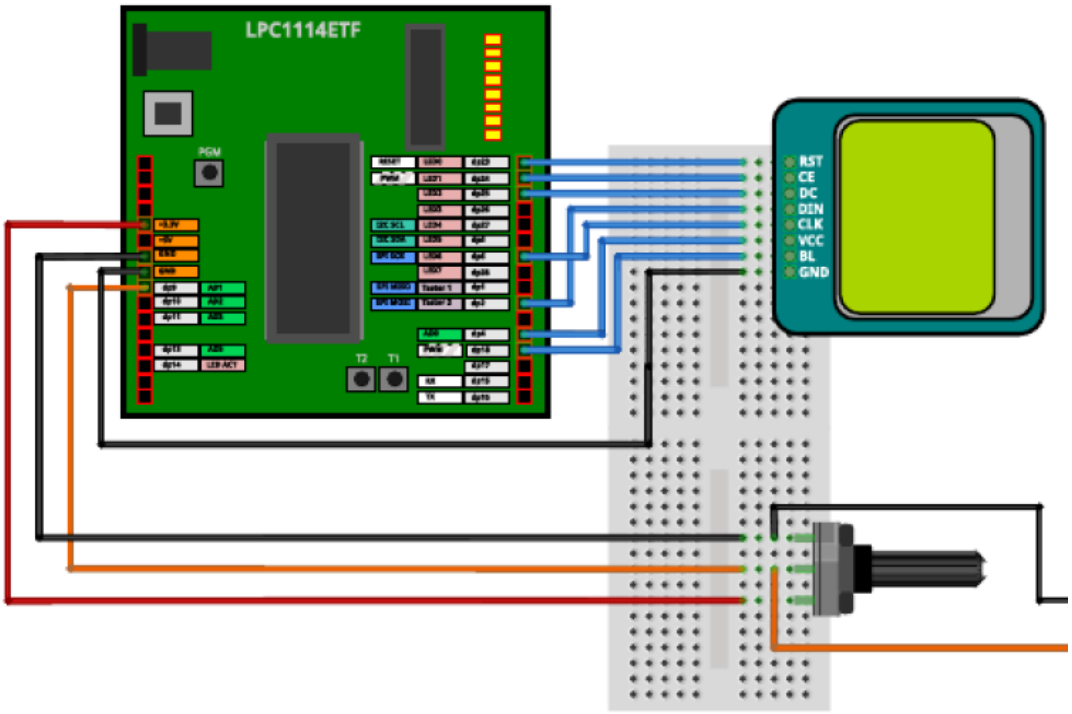
| ULAZI | IZLAZI |
|---|---|
| Taster1 (digitalni, LPC1114ETF) | Nokia display (digitalni output pinovi) |
| LM35 (analogni, picoETF) | ILI9314 TFT display (digitalni output pinovi) |
| Potenciometar (analogni, oba sistema) | |



Slika 1 - Povezivanje TFT displeja i LM35 senzora na sistem picoETF (zadatak 2)




Slika 2 – Povezivanje potenciometra na LPC1114ETF (zadatak 3)




Slika 3 – Šema povezivanja displeja i potenciometra sa sistemom LPC1114ETF (zadatak 4)

# 4 Zaključak

U okviru ove laboratorijske vježbe upoznali smo se sa konceptom rada analognih ulaza na različitim razvojnim sistemima. Sa principom rada korištenih razvojnih sistema picoETF i LPC1114ETF smo se na prethodnim vježbama dovoljno upoznali, te stoga nije bilo poteškoća prilikom korištenja istih, što je i očekivano. Korištene su i različite elektroničke komponente čija je obrada izlazne vrijednosti dosta varirala sa implementacijom. Međutim, uz solidnu podršku dokumentacije (kako o samim komponentama, tako i programske), implementacija programskih rješenja nije predstavljala problem ni u jednom zadatku.

Konkretno govoreći o zadatku br. 3, bilo je adekvatnijih načina za kodiranje stanja i podešavanja niza LED dioda. Cjelobrojni opsezi koji bi se prosljeđivali BusOut objektu mogli su se ekvivalentirati sa float očitanjem sa AnalogIn objekta (potenciometra), te na taj način izračunavati i slati samom BusOut-u, ali to rješenje je numerički zahtjevnije. Može se diskutovati o performansama implementiranog rješenja i razlike sa navedenom modifikacijom. U svakom slučaju, implementirano rješenje je potpuno validno i funkcionalno.

Jedina prepreka prilikom implementacije predstavljala je u 4. zadatku pogrešno funkcionisanje funkcije sprintf(). Kao što je u segmentu 2.4 opisano, ova funkcija ne radi korektno za floating – point parametre, što je posljedica bug-a u nekoj od korištenih third – party biblioteka. Konkretan bug nije utvrđen, međutim, utvrđivanje istog svakako izlazi van opsega vježbe. Implementirana funkcija sa dinamičkom alokacijom niza char-ova se ispostavila kao kvalitetno rješenje ovog problema, te isto funkcionira bez problema.

Cilj vježbe je bio upoznavanje sa analognim ulazima i obradom istih u različitim embedded programskim jezicima, što je uspješno postignuto.

# 5 Prilog

## 5.1 Zadatak 1/Izvorni kod

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

leds = [Pin(i, Pin.OUT) for i in range(4, 12)]
adc = ADC(Pin(28))

def decimal_to_binary_8bits(n):
    # Pretvori decimalni oblik u binarni i skloni '0b' prefiks
    binary_str = bin(int(n))[2:]

    # Dodaji nule na početak dok veličina ne bude 8
    while len(binary_str) < 8:
        binary_str = '0' + binary_str

    # Pretvori binarni string u listu int-ova
    binary_list = [int(bit) for bit in binary_str]

    return binary_list

def update_leds(num, leds):
    num_binList = decimal_to_binary_8bits(num)
    for i in range(0, 8):
        leds[i].value(num_binList[i])

def set_t(adc_val):
    return 0.1 + 1.9*adc_val/65535

brojac = 0
t = 0.1
adc_val = adc.read_u16()

while True:
    print(adc.read_u16())

    t = set_t(adc.read_u16())

    brojac += 1
    update_leds(brojac, leds)
    time.sleep(t)

    for i in range(0, 8):
        brojac *= 2
        update_leds(brojac, leds)
        t = set_t(adc.read_u16())
        time.sleep(t)

    brojac = 255
    update_leds(brojac, leds)
```

```
    t = set_t(adc.read_u16())
    time.sleep(t)

    pomocna = 256
    for i in range(0, 8):
        brojac -= pomocna/2
        update_leds(brojac, leds)
        t = set_t(adc.read_u16())
        time.sleep(t)
        pomocna /= 2

    brojac = 0
```

## 5.2 Zadatak 2/Izvorni kod

Spomenuta dokumentacija za biblioteku gfx.py može se naći na linku:
https://github.com/adafruit/Adafruit-GFX-Library

Sama biblioteka je adaptirana u microPython, ali je u potpunosti bazirana na datom library-u.

```
from ili934xnew import ILI9341, color565
from machine import Pin, SPI
from micropython import const
import glcdfont
import tt14
import tt24
import tt32
import time
import gfx

from machine import ADC
adc = ADC(Pin(26))

# Dimenzije displeja
SCR_WIDTH = const(320)
SCR_HEIGHT = const(240)
SCR_ROT = const(2)
CENTER_Y = int(SCR_WIDTH/2)
CENTER_X = int(SCR_HEIGHT/2)

# Podešenja SPI komunikacije sa displejem
TFT_CLK_PIN = const(18)
TFT_MOSI_PIN = const(19)
TFT_MISO_PIN = const(16)
TFT_CS_PIN = const(17)
TFT_RST_PIN = const(20)
TFT_DC_PIN = const(15)

# Fontovi na raspolaganju
fonts = [glcdfont, tt14, tt24, tt32]
text = 'RPi Pico/ILI9341'
print(text)

print("Fontovi:")
```

```
for f in fonts:
    print(f.__name__)

spi = SPI(0,
          baudrate=62500000,
          miso=Pin(TFT_MISO_PIN),
          mosi=Pin(TFT_MOSI_PIN),
          sck=Pin(TFT_CLK_PIN))
print(spi)

display = ILI9341(spi,
                  cs=Pin(TFT_CS_PIN),
                  dc=Pin(TFT_DC_PIN),
                  rst=Pin(TFT_RST_PIN),
                  w=SCR_WIDTH,
                  h=SCR_HEIGHT,
                  r=SCR_ROT)

# Brisanje displeja i odabir pozicije (0,0)
display.erase()
display.set_pos(0,0)

# Ispis teksta različitim fontovima, počevši od odabrane pozicije
for ff in fonts:
    display.set_font(ff)
    display.print(text)

# Ispis teksta u drugoj boji
display.set_font(tt14)
display.set_color(color565(150, 200, 0), color565(0, 0, 0))
time.sleep(1)

#Brisanje displeja
display.erase()

# Dodatna funkcija za crtanje kružnice ispisom pojedinačnih piksela
display.set_font(tt14)
display.erase()

# Različita orijentacija teksta na displeju
display.set_pos(10,100)
display.rotation=0
display.erase()
display.set_pos(10,100)
display.rotation=1
display.init()
display.erase()
display.set_pos(10,100)
display.rotation=2
display.init()
display.erase()
display.set_pos(10,100)
```

```
display.rotation=3
display.init()
display.erase()

display.set_pos(10,100)
display.rotation=4
display.init()
display.erase()
display.set_pos(10,100)
display.rotation=5
display.init()
display.erase()
display.set_pos(10,100)
display.rotation=6
display.init()
display.erase()
display.set_pos(10,100)
display.rotation=7
display.init()
display.erase()

def fast_hline(x, y, width, color):
    display.fill_rectangle(x, y, width, 1, color)

def fast_vline(x, y, height, color):
    display.fill_rectangle(x, y, 1, height, color)

def line(self, x0, y0, x1, y1, *args, **kwargs):
# Funkcija za crtanje linije.
# Crta liniju koja počinje na (x0, y0) i završava na (x1, y1)
    steep = abs(y1 - y0) > abs(x1 - x0)
    if steep:
        x0, y0 = y0, x0
        x1, y1 = y1, x1
    if x0 > x1:
        x0, x1 = x1, x0
        y0, y1 = y1, y0
    dx = x1 - x0
    dy = abs(y1 - y0)
    err = dx // 2
    ystep = 0
    if y0 < y1:
        ystep = 1
    else:
        ystep = -1
    while x0 <= x1:
        if steep:
            self._pixel(y0, x0, *args, **kwargs)
        else:
            self._pixel(x0, y0, *args, **kwargs)
        err -= dy
        if err < 0:
```

```
            y0 += ystep
            err += dx
        x0 += 1

gfx = gfx.GFX(240, 320, display.pixel, hline=fast_hline, vline=fast_vline)

vrijeme = 5
t=0
while True :
    var = adc.read_u16()
    tmp = var/260
    volt = var/26
    display.set_pos(180,10)
    display.rotation=1
    display.init()
    display.print('Temp: '+ str(tmp) + " C")
    display.print('Napon: ' + str(volt) + 'mV')
    display.print('Vrijeme: ' + str(t) + 's')

    gfx.line(20, 20, 20, 220, color565(255, 255, 255))
    gfx.line(20, 220, 260, 220, color565(255, 255, 255))
    display.set_pos(14, 220)
    display.print("20")
    display.set_pos(14, 160)
    display.print("25")
    display.set_pos(14, 100)
    display.print("30")
    display.set_pos(14, 0)
    display.print("40")
    gfx.fill_circle(20+vrijeme,220-int((tmp-22)*18),4, color565(150, 200, 0) )
    vrijeme = vrijeme + 5
    if(20 + vrijeme) > 260:
        vrijeme = 5
        display.erase()
    t+=1
```

## 5.3 Zadatak 3/Izvorni kod

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

#define nula   0xff
#define jedan  0x7f
#define dva    0x3f
#define tri    0x1f
#define cetiri 0x0f
#define pet    0x07
#define sest   0x03
#define sedam  0x01
#define osam   0x00

BusOut leds(LED0, LED1, LED2, LED3, LED4, LED5, LED6, LED7);
```

```
AnalogIn potMeter (dp9);
DigitalOut e(LED_ACT);

int main() {
    e = 0;
    while(true) {

        if(potMeter < 1./9) leds = nula;
        else if(potMeter >=1./9 && potMeter < 2*1./9) leds = jedan;
        else if(potMeter >=2*1./9 && potMeter < 3*1./9) leds = dva;
        else if(potMeter >=3*1./9 && potMeter < 4*1./9) leds = tri;
        else if(potMeter >=4*1./9 && potMeter < 5*1./9) leds = cetiri;
        else if(potMeter >=5*1./9 && potMeter < 6*1./9) leds = pet;
        else if(potMeter >=6*1./9 && potMeter < 7*1./9) leds = sest;
        else if(potMeter >=7*1./9 && potMeter < 8*1./9) leds = sedam;
        else if(potMeter > 8*1./9) leds = osam;

        wait_us(10);
    }

}
```

## 5.4 Zadatak 4 – dodatni bodovi/Izvorni kod

```
#include "mbed.h"
#include "N5110.h"
#include "lpc1114etf.h"
#include <stdio.h>

AnalogIn potMeter(dp9);
DigitalIn taster(Taster_1);
N5110 lcd(dp4,dp24,dp23,dp25,dp2,dp6,dp18);

char* floatToString(float num) {
    //Alokacija memorije za string
    char* str = (char*)malloc(20 * sizeof(char));

    //Tretman negativnih brojeva
    if (num < 0) {
        *str++ = '-';
        num = -num;
    }

    //Cjelobrojni dio
    int intPart = (int)num;
    int index = 0;
    do {
        str[index++] = intPart % 10 + '0';
        intPart /= 10;
    } while (intPart != 0);

    //Obrtanje cjelobrojnog dijela zbog upisa
```

```
    for (int i = 0; i < index / 2; i++) {
        char temp = str[i];
        str[i] = str[index - i - 1];
        str[index - i - 1] = temp;
    }

    //Decimalna tacka
    str[index++] = '.';

    //Necijeli dio
    float fracPart = num - (int)num;
    for (int i = 0; i < 6; i++) { //6 decimalnih mjesta, modifyable
        fracPart *= 10;
        int digit = (int)fracPart;
        str[index++] = digit + '0';
        fracPart -= digit;
    }

    //Null-terminacija stringa
    str[index] = '\0';

    return str;
}

void prikaziOtpor() {
    char otporString[10];
    float otpor = 100000 * potMeter.read();
    sprintf(otporString, "%d", (int)otpor);
    //Refresh displeja
    lcd.clear();
    lcd.printString("Otpor (Ohm): ", 0, 0);
    lcd.printString(otporString, 0, 2);
}

void prikaziNapon() {
    float refNapon = 3.3;
    float napon = refNapon * potMeter.read();
    char* naponString;
    naponString = floatToString(napon);
    //Refresh displeja
    lcd.clear();
    lcd.printString("Napon (V): ", 0, 0);
    lcd.printString(naponString, 0, 2);
    free(naponString);
}

int main() {
    //false za prikaz otpora, true za prikaz napona
    lcd.init();
    bool viewMode = false;

    while (true) {
```

```
        if(taster == 1) viewMode = !viewMode;

        if(viewMode == false) prikaziOtpor();
        else prikaziNapon();
        wait_us(100);
    }
}
```