Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023/24.

Izvještaj za laboratorijsku vježbu 6

Prekidi i tajmeri

Ime i prezime: **Kanita Kadušić**
Broj index-a: **19327**

Sarajevo, april 2024.

Sadržaj

1 Pseudokod

1.1 Zadatak 1

```
function click()
    if valueMili(debounce) < 200
        return
    else
        reset(debounce)

    onClickCounter ← (onClickCounter + 1) mod 16
    value(onClickLeds) ← value(onClickCounter)

function update()
    counter ← (counter + 1) mod 16
    value(leds) ← value(counter)

start(debounce)
rise(button, click)
attach(ticker, update, T)
```

1.2 Zadatak 2a

```
counter, released ← 0, true

while true
    value(signal) ← not value(signal)
    pause(T)

    clicked ← value(button)
    if not clicked
        released ← true

    if clicked and released
        counter ← (counter + 1) mod 100
        released ← false

    value(first) ← counter ÷ 10
    value(second) ← counter mod 10
```

1.3 Zadatak 2b

```
function click()
    counter ← (counter + 1) mod 100

    value(first) ← counter ÷ 10
    value(second) ← counter mod 10

function update()
    value(signal) ← not value(signal)

rise(button, click)
attach(ticker, update, T)
```

1.4 Zadatak 3

```
function reset()
    counter ← 0
    value(leds) ← 0

function encoder(pin)
    currentA ← value(A)
    currentB ← value(B)

    if currentA ≠ previousA or currentB ≠ previousB
        if pin = A and currentA ≠ previousA
            if currentA = currentB
                counter ← (counter − 1) mod (256 · 4)
            else
                counter ← (counter + 1) mod (256 · 4)
        else if pin = B and currentB ≠ previousB
            if currentA = currentB
                counter ← (counter + 1) mod (256 · 4)
            else
                counter ← (counter − 1) mod (256 · 4)

        previousA, previousB ← currentA, currentB

    value(leds) ← counter ÷ 4

riseOrFall(A, encoder)
riseOrFall(B, encoder)
rise(R, reset)
```

2 Analiza programskog rješenja

2.1 Zadatak 1

[**1**-**14**] uvoz odgovarajućih biblioteka, te deklaracija i inicijalizacija potrebnih varijabli

[**16**-**22**] funkcija čiji se poziv vrši pritiskom na taster, a koja inkrementira brojač čiju vrijednost prikazuje putem LED dioda, pritom vodeći računa o *debouncing*-u

[**24**-**27**] funkcija čiji se poziv vrši periodično, a koja inkrementira brojač čiju vrijednost prikazuje putem LED dioda

[**31**-**32**] gašenje svih LED dioda

[**34**] pokretanje tajmera za *debouncing*

[**35**] slanje pokazivača na funkciju *click*, kao parametra funkcije *rise*, koja se poziva nad objektom *button*

[**36**] slanje pokazivača na funkciju *update* i vremenskog trajanja perioda, kao parametre funkcije *attach*, koja se poziva nad objektom *tick*

2.2 Zadatak 2a

[**1**-**11**] uvoz odgovarajućih biblioteka, te deklaracija i inicijalizacija potrebnih varijabli

[**20**] naizmjenično postavljanje vrijednosti signala na 0 i 1

[**21**] pauza u izvršavanju programa

[**23**-**29**] inkrementiranje brojača na uzlaznu ivicu dovedenog signala

[**31**] postavljanje vrijednosti LED dioda koje odgovaraju cifri desetica

[**32**] postavljanje vrijednosti LED dioda koje odgovaraju cifri jedinica

2.3 Zadatak 2b

[**1**-**14**] uvoz odgovarajućih biblioteka, te deklaracija i inicijalizacija potrebnih varijabli

[**16**-**21**] funkcija čiji se poziv vrši na uzlaznu ivicu dovedenog signala, a koja inkrementira brojač čiju vrijednost prikazuje putem LED dioda

[**23**-**25**] funkcija čiji se poziv vrši periodično, a koja naizmjenično postavlja vrijednost signala na 0 i 1

[**30**] slanje pokazivača na funkciju *click*, kao parametra funkcije *rise*, koja se poziva nad objektom *button*

[**31**] slanje pokazivača na funkciju *update* i vremenskog trajanja perioda, kao parametre funkcije *attach*, koja se poziva nad objektom *tick*

2.4 Zadatak 3

[**1**-**2**] uvoz odgovarajućih biblioteka

[**4**-**15**] klasa koja olakšava rad s proizvoljnim brojem digitalnih izlaza (njen interfejs sadrži metodu za čitanje i pisanje)

[**17**-**26**] deklaracija i inicijalizacija potrebnih varijabli

[**28**-**32**] funkcija čiji se poziv vrši pritiskom na enkoder, a koja postavlja vrijednost brojača na 0, te nju prikazuje putem LED dioda

[**34**-**55**] funkcija koja realizira rad s enkoderom, odnosno inkrementira brojač pri rotaciji u smjeru kazaljke na satu, u suprotnom dekrementira

[**57**-**59**] slanje odgovarajućih funkcija i željenih ivica na koje će se desiti poziv istih funkcija, kao parametara funkcije *irq*, koja se poziva nad odgovarajućim objektima

3 Korišteni hardverski resursi

3.1 LPC1114ETF

	Komponenta	*Opis*	*Količina*
1	LPC1114FN28	mikrokontroler	1
2	LED dioda	digitalni izlaz	8
3	taster	digitalni ulaz	1
4	generator signala	ulazna komponenta	1
5	osciloskop	izlazna komponenta	1
6	sonda	uvezivanje sistema	2
7	konektor	uvezivanje sistema	
8	USB A kabal	napajanje i komunikacija	1

3.2 picoETF

	Komponenta	*Opis*	*Količina*
1	RP2040	mikrokontroler	1
2	LED dioda	digitalni izlaz	8
3	rotacijski enkoder	ulazna komponenta	1
4	konektor	uvezivanje sistema	
5	USB A – USB Micro	napajanje i komunikacija	1

4 Zaključak

Zadaci u okviru Laboratorijske vježbe 6 su bili zanimljivi, a hardverska realizacija tokom laboratorijske vježbe je protekla uredno i bez problema.

U okviru Zadatka 1 je bilo potrebno analizirati ponašanje sistema, realiziranog bez korištenja sistema prekida, prilikom korištenja tastera na sistemu, i to s manjim i većim trajanjem pauze u izvršavanju programa. Nedostatak ovog rješenja je što se, s porastom trajanja pauze, onemogućava pravovremena reakcija na pritisak tastera. Odnosno, ne može se obezbijediti proizvoljna frekvencija automatskog paljenja i gašenja dioda zajedno s trenutačnim odgovorom na pritisak tastera.

Zadatak 2a je zahtijevao da se testira ispravnost aplikacije dovođenjem impulsa sa generatora signala, uz postepeno povećavanje frekvencije četvrtki. Naravno, očekivano ponašanje sistema je da, povećavanjem frekvencije, dolazi do bržeg inkrementiranja brojača, te odgovarajućeg prikaza na LED diodama. Međutim, bez korištenja sistema prekida dolazi do određenih neželjenih efekata. Naime, s obzirom da brojač impulsa nije realiziran kao primarna operacija u programu, pri frekvenciji od oko $450Hz$, brojač, umjesto da nastavi ubrzavati, počinje da usporava. S druge strane, primarna operacija programa, odnosno realizacija generatora četvrtki, teče uredno.

U Zadatku 2b je bilo potrebno realizirati funkcionalnosti iz Zadatka 2a, ali korištenjem sistema prekida, te provjeriti koja je maksimalna frekvencija pri kojoj se aplikacija korektno izvršava. U ovom slučaju se očekivano ponašanje nastavilo i pri frekvencijama većim od $450Hz$. Štaviše, sve do oko $1000Hz$ se moglo uočiti regularno izmjenjivanje stanja LED dioda. Međutim, pri nešto višim frekvencijama, ljudsko oko više nije bilo sposobno uočiti izmjene stanja dioda, pa je izgledalo kao da su sve LED diode konstantno uključene.

Za kraj, nova znanja stečena u okviru Laboratorijske vježbe 6 podrazumijevaju rad s prekidima i tajmerima, te s novim hardverskim komponentama, konkretno, generatorom signala i rotacijskim enkoderom.

5 Prilog

5.1 Zadatak 1: Izvorni kôd

```
01: #include "mbed.h"
02: #include "lpc1114etf.h"
03:
04: Timer debounce;
05: Ticker tick;
06:
07: BusOut leds(LED3, LED2, LED1, LED0);
08: BusOut onClickLeds(LED7, LED6, LED5, LED4);
09:
10: DigitalOut enable(LED_ACT);
11: InterruptIn button(Taster_1);
12:
13: const float T = 0.2;
14: int counter = 0, onClickCounter = 0;
15:
16: void click() {
17:     if (debounce.read_ms() < 200) return;
18:     else debounce.reset();
19:
20:     onClickCounter = (onClickCounter + 1) % 16;
21:     onClickLeds.write(onClickCounter);
22: }
23:
24: void update() {
25:     counter = (counter + 1) % 16;
26:     leds.write(counter);
27: }
28:
29: int main() {
30:     enable.write(0);
31:     leds.write(0);
32:     onClickLeds.write(0);
33:
34:     debounce.start();
35:     button.rise(&click);
36:     tick.attach(&update, T);
37:
38:     while (true) wait_us(1e3);
39: }
```

5.2 Zadatak 2a: Izvorni kôd

```
01: #include "mbed.h"
02: #include "lpc1114etf.h"
03:
04: BusOut first(LED3, LED2, LED1, LED0);
05: BusOut second(LED7, LED6, LED5, LED4);
06:
07: DigitalOut enable(LED_ACT);
08: DigitalOut signal(dp18);
09: DigitalIn button(dp9);
10:
11: const float T = 2e-3;
12:
13: int main() {
14:     enable.write(0);
15:
16:     int counter = 0;
17:     bool released = true;
18:
19:     for (;;) {
20:         signal.write(!signal.read());
21:         wait_us(T / 2 * 1e6);
22:
23:         bool clicked = button.read();
24:         if (!clicked) released = true;
25:
26:         if (clicked && released) {
27:             counter = (counter + 1) % 100;
28:             released = false;
29:         }
30:
31:         first.write(counter / 10);
32:         second.write(counter % 10);
33:     }
34: }
```

5.3 Zadatak 2b: Izvorni kôd

```
01: #include "mbed.h"
02: #include "lpc1114etf.h"
03:
04: Ticker tick;
05:
06: BusOut first(LED3, LED2, LED1, LED0);
07: BusOut second(LED7, LED6, LED5, LED4);
08:
09: DigitalOut enable(LED_ACT);
10: DigitalOut signal(dp18);
11: InterruptIn button(dp9);
12:
13: const float T = 2e-3;
14: int counter = 0;
15:
16: void click() {
17:     counter = (counter + 1) % 100;
18:
19:     first.write(counter / 10);
20:     second.write(counter % 10);
21: }
22:
23: void update() {
24:     signal.write(!signal.read());
25: }
26:
27: int main() {
28:     enable.write(0);
29:
30:     button.rise(&click);
31:     tick.attach(&update, T / 2);
32:
33:     while (true) wait_us(1e6);
34: }
```

5.4 Zadatak 3: Izvorni kôd

```
01: from machine import Pin
02: from time import sleep
03:
04: class BusOut:
05:   def __init__(self, *pins):
06:     self.pins = [ Pin(pin, Pin.OUT) for pin in pins ]
07:     self.value = 0
08:
09:   def write(self, value):
10:     self.value = value
11:     for i, pin in enumerate(self.pins):
12:       pin.value((value >> i) & 1)
13:
14:   def read(self):
15:     return self.value
16:
17: leds = BusOut(11, 10, 9, 8, 7, 6, 5, 4)
18:
19: A = Pin(28, Pin.IN)
20: B = Pin(27, Pin.IN)
21: R = Pin(26, Pin.IN)
22:
23: previousA = A.value()
24: previousB = B.value()
25:
26: counter = 0
27:
28: def reset(pin):
29:     global counter, leds
30:
31:     counter = 0
32:     leds.write(0)
33:
```

```
34: def encoder(pin):
35:     global previousA, previousB, counter, leds
36:
37:     currentA = A.value()
38:     currentB = B.value()
39:
40:     if currentA != previousA or currentB != previousB:
41:         if pin == A and currentA != previousA:
42:             if currentA == currentB:
43:                 counter = (counter - 1) % (256 * 4)
44:             else:
45:                 counter = (counter + 1) % (256 * 4)
46:         elif pin == B and currentB != previousB:
47:             if currentA == currentB:
48:                 counter = (counter + 1) % (256 * 4)
49:             else:
50:                 counter = (counter - 1) % (256 * 4)
51:
52:         previousA = currentA
53:         previousB = currentB
54:
55:     leds.write(counter // 4)
56:
57: A.irq(handler=encoder, trigger=Pin.IRQ_RISING | Pin.IRQ_FALLING)
58: B.irq(handler=encoder, trigger=Pin.IRQ_RISING | Pin.IRQ_FALLING)
59: R.irq(handler=reset, trigger=Pin.IRQ_RISING)
60:
61: while True:
62:     sleep(0.5)
```