

LABORATOR 10 – PWM

Până acum toate aplicațiile scrise la laborator sau la curs au avut de îndeplinit o singură sarcină. Am calculat funcții logice în laboratoarele 3 și 4, am afișat caracterul preluat de la tastatură în laboratorul 7, am controlat interfonul în laboratorul 8 sau am afișat timpul în laboratorul 9. În realitate o aplicație micro nu are de îndeplinit o singură funcție, ci mai multe. Un exemplu de funcție secundară executată de foarte multe aplicații micro este afișarea timpului. Cum în multe aplicații este necesară măsurarea timpului (cuptorul cu microunde încălzește mâncarea timpul programat) este foarte simplu să se afișeze timpul în standby.

O altă caracteristică a aplicațiilor scrise până acum constă în timpul scurt de execuție. Timpul maxim de execuție a buclei principale este probabil de câteva microsecunde pentru aplicațiile funcții logice și tastatură și ajunge la aproape o milisecundă la ceas. Ceasul necesită 1 ms nu pentru ca am face multe calcule ci pentru că avem de afișat 16 caractere.

După cum s-a menționat anterior, majoritatea aplicațiilor reale au de îndeplinit mai multe funcții iar unele dintre aceste funcții necesită timp de execuție care pot ajunge de ordinul secundelor. În acest laborator vom scrie o aplicație care are de îndeplinit trei funcții dintre care una necesită un timp mare de execuție.

Scopul lucrării

Se va scrie o aplicație care are de îndeplinit două sarcini:

1. Să controleze prin intermediul tastelor ,C' și ,D' luminozitatea unui LED,
2. Să determine dacă numărul c preluat de la tastatură aparține unei triplete pitagoreice. Detalii suplimentare la pasul 3. Sarcina 3 este opțională în acest laborator, dar devine obligatorie în laboratorul următor. Această sarcină poate necesita un timp de execuție de ordinul secundelor.

După implementare, afișajul LCD va arăta astfel:

C	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1
L										0	1	2	3	4	5	6
1	9	9	9					P	Y	T	H					
2	L	=	7	5	%											

figura 1

Desfășurarea lucrării

Pasul 1: Crearea proiectului

Se va crea un proiect nou cu numele **pwm** care va conține aceleași surse ca proiectul kbd (secțiunea obligatorie) din laboratorul 7. Pentru crearea acestui proiect procedați după cum urmează:

- a) **Se va crea (obligatoriu cu NEXT)** proiectul cu numele **pwm**. La crearea proiectului **obligatoriu bifați** căsuța **Create initial file**.
- b) **Închideți** AVR Studio.
- c) **Copiați IOfn.c și defs.h** din folderul **kbd** în folderul proiectului **pwm**.
- d) În folderul **pwm ștergeți fișierul pwm.c** (care este gol).

- e) **Copiați** **pwm.c** din platformă în folderul proiectului **pwm**.
- f) **Deschideți** (obligatoriu cu **NEXT**) proiectul **pwm**.
- g) Faceți click dreapta pe **Source Files** și din meniul contextual ce va apare selectați **Add existing Source File(s): Adăugați** la proiect fișierul **IOfn.c** din folderul **pwm**.
- h) Faceți click dreapta pe **Header Files** și din meniul contextual ce va apare selectați **Add existing Header File(s): Adăugați** la proiect fișierul **defs.h** din folderul **pwm**.

În **pwm.c** a fost modificată secțiunea de afișare a caracterului preluat de la tastatură astfel încât să se afișeze întotdeauna caracterul tastat pe prima poziție a primei linii (1,1). Pentru aceasta au fost adăugați liniile roșii din secțiunea de cod următoare:

```
...
if(kbhit){
    gotoLC(1,1);
    kbhit=0;
    putcharLCD (kbcode);
}
...
```

Verificați că tipul de oscilator este setat conform indicațiilor din laboratorul 9, pasul 3. Dacă nu este, modificați-l.

Deoarece fiecare sarcină are un loc propriu unde afișează, în acest program nu se va folosi `clrLCD()`. Poziționarea se va face cu `gotoLC(...)` și afișarea noilor valori se va face prin suprascriere.

Pasul 2: LED cu luminozitate variabilă

Controlul luminozității prin PWM este prezentat în prelegerea 5, **capitolul 3.5 „Modul fast PWM – fast Pulse Width Modulation”**. În implementarea din acest laborator diferența față de curs constă numai în frecvența ceasului procesor: în curs f_{CLK_CPU} este 14,4MHz iar la laborator este 8MHz.

Procedura de setarea timerului 0 este foarte asemănătoare cu setarea timerului 2 din laboratorul 9 – ceas. Registrele Timerului 0 sunt descrise în documentația ATmega16A începând de la pagina 82. Pentru a determina valorile biților din registrul TCCR0 procedați după cum urmează:

- a) Determinați valorile biților WGM00 și WGM01 pentru a programa timerul 0 în mod fast PWM.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 14-2. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

- b) Determinați p astfel încât T_{cycle} să fie **cât mai aproape de 10 ms**. Determinați valorile biților CS02, CS01 și CS00 pentru p -ul calculat.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 14-6. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$\text{clk}_{\text{IO}}/(\text{No prescaling})$
0	1	0	$\text{clk}_{\text{IO}}/8$ (From prescaler)
0	1	1	$\text{clk}_{\text{IO}}/64$ (From prescaler)
1	0	0	$\text{clk}_{\text{IO}}/256$ (From prescaler)
1	0	1	$\text{clk}_{\text{IO}}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

- c) Determinați valorile biților COM01 și COM00 care controlează comportarea pinului OC0 conform tabelului următor.

Table 14-4. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

- a) Programați TCCR0 cu valorile determinate la pașii a-c. Documentați setarea lui TCCR0 ca în programul de la pagina 22 din prelegerea 5. Setarea lui TCCR0 se face **după** `sysinit()` și înainte de `while(1)`. (**NU** în `sysinit()`!)
- b) Modificați cu `clrbits` sau `setbits` DDRB pentru ca pinul OC0 să fie ieșire. Nu modificați biții din DDRB programați anterior. Modificarea lui DDRB se face după `sysinit()` și înainte de `while(1)`. (**NU** în `sysinit()`!)
- d) **Conectați un LED pe pinul OC0 ca în figura 10 din prelegerea 5.**

Luminozitatea LED-ului se va modifica prin intermediul tastelor ,C'(rește) și ,D'(escrește).

Luminozitatea se modifică în trepte de 5% din luminozitatea maximă. Apăsarea lui ,C' mărește luminozitatea cu 5% iar apăsarea lui ,D' scade luminozitatea cu 5%. Creșterea și descreșterea sunt saturate, adică luminozitatea nu poate crește mai mult de 100% și nu poate scădea sub 0% indiferent de numărul de apăsări ale tastelor ,C' și ,D'.

Registrul care controlează luminozitatea are 8 biți și astfel valorile luminozității sunt în intervalul 0 -255. Luminozitatea care se introduce de la tastatură este în plaja 0-100. Din acest motiv este nevoie să calculăm valoarea ce se va scrie în registru în funcție de valoarea L setată de la tastatură: pentru L=100 în registru trebuie scris 255 iar pentru L=0 trebuie scris 0. Valoarea care se scrie în registru se calculează în funcție de L cu regula de trei simplă.

Structura codului pentru variația luminozității este comentată în `pwm.c`. Decomentați și completați.

Afișați valoarea luminozității ca în figura 1. Noua valoare se scrie peste vechea valoare. Valoarea inițială a luminozității este 50%.

Testați dacă se afișează timpul, tasta apăsată și dacă se poate modifica luminozitatea LED-ului!

Dacă funcționează, chemați profesorul pentru verificarea funcționării. Veți prezenta și calculele pentru p -ul și T_{cycle} -ulul timerului 0.

Dacă ați ajuns aici aveți nota 5.

Pasul 3: c pitagoreic. Opțional în acest laborator, obligatoriu în laboratorul următor.

Pe lângă controlul luminozității se cere să determine dacă numărul c preluat de la tastatură aparține unei triplete pitagoreice. O tripletă pitagoreică este format din trei numere naturale nenule a , b și c , cu proprietatea că $a^2 + b^2 = c^2$. De exemplu, $a=3$, $b=4$, $c=5$ formează o tripletă pitagoreică. Numărul $c=999$ aparține unei triplete pitagoreice? Numere pitagoreice cu $c < 2100$ se pot descărca de la <http://www.tsm-resources.com/alists/trip.html>

Numărul c se va reprezenta pe **exact** 3 cifre zecimale, adică va lua valori între 1 și 999. Numerele mai mici ca 100 se vor introduce cu zerouri în față. De exemplu, pentru $c=5$ vom introduce 005. Acest mod stângaci de operare va simplifica scrierea programului.

În continuare este sugerată o variantă de implementare. Dacă nu vă place această variantă, puteți scrie propria variantă. Totul este să nu aibă o complexitate mai mare și, cel mai important, să funcționeze.

Pentru început vom elimina secțiunea care afișează tasta apăsată:

```
—— if(kbhit){  
—— gotoLCD(1,1);  
—— kbhit=0;  
—— putcharLCD(kbeode);  
—— }
```

Implementarea se face cu o SFSM, ca în laboratorul 8. SFSM are 3 stări:

- În starea C1 se va șterge prima linie a LCD-ului prin scrierea unui string format din 16 caractere spațiu (blanc), cifra citită de la tastatură se va memora în stringul `buf_pit`, se va poziționa cursorul pe linia 1, coloana1 și se va afișa cifra. Poziționarea este necesară deoarece între două preluări de cifră poate să intervină o modificare a intensității LED-ului. Afișarea ceasului va poziționa cursorul pe linia 2. Dacă nu repositionăm cursorul, cifra preluată va fi afișată pe linia 2.
- În starea C2 cifra citită de la tastatură se va memora în stringul `buf_pit`, se va poziționa cursorul pe linia 1, coloana2 și se va afișa cifra.
- În starea C3 cifra citită de la tastatură se va memora în stringul `buf_pit`, se va poziționa cursorul pe linia 1, coloana3 și se va afișa cifra. După ce s-a preluat a treia cifră se va calcula c . Deși calculul este foarte simplu, dacă nu știți să-l faceți, folosiți `atoi` din `stdlib`.

Afișați mesajul „Busy...” și apoi verificați prin forță brută dacă există doi întregi a și b astfel încât $a^2 + b^2 = c^2$. Generați pe a și b cu două `for`-uri. Dacă se îndeplinește condiția $a^2 + b^2 = c^2$ setați o variabilă spion.

În funcție de valoarea spionului afișați din linia 1, coloana 8 fie „PYTH ” dacă c este pitagoreic, fie „nonNPYTH ” dacă nu este. Ca optimizare, puteți ieși din cele două `for`-uri cu `goto`.

Structura codului pentru c pitagoreic este comentată în `pwm.c`. De comentați și completați.
Testați pentru $c=005$ și pentru $c=999$. Ce se întâmplă pe durata calculelor cu controlul luminozității?

Dacă nu se fac calcule pentru a determina dacă c este pitagoreic sau nu, controlul luminozității trebuie să funcționeze ca la pasul 2 (pentru nota 5).

Dacă se fac calcule pentru a determina dacă c este pitagoreic sau nu, **ce se întâmplă cu controlul luminozității?**

**Dacă c pitagoreic funcționează și aveți un răspuns la întrebarea de mai sus, chemați profesorul.
Veți obține între 0 și 5 puncte în funcție de cât ați implementat din c pitagoreic și de calitatea implementării.**