LABORATOR 10 – PWM

Până acum toate aplicaţiile scrise la laborator sau la curs au avut de îndeplinit o singură sarcină. Am calculat funcţii logice în laboratoarele 3 şi 4, am afişat caracterul preluat de la tastatură în laboratorul 7, am controlat interfonul în laboratorul 8 sau am afişat timpul în laboratorul 9. În realitate o aplicaţie micro nu are de îndeplinit o singură funcţie, ci mai multe. Un exemplu de funcție secundară executată de foarte multe aplicaţii micro este afişarea timpului. Cum în multe aplicaţii este necesară măsurarea timpului (cuptorul cu microunde încălzește mâncarea timpul programat) este foarte simplu să se afişeze 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 aplicaţiile funcții logice şi tastatură şi ajunge la aproape o milisecundă la ceas. Ceasul necesită 1 ms nu pentru ca am face multe calcule ci pentru că avem de afişat 8 caractere.

După cum s-a menționat anterior, majoritatea aplicaţiilor reale au de îndeplinit mai mute funcţii iar unele dintre aceste funcţii necesită timpi de execuție care pot ajunge de ordinul secundelor. În acest laborator vom scrie o aplicaţie care are de îndeplinit trei funcţii dintre care una necesită un timp mare de execuţie.

Scopul lucrării

Se va scrie o aplicaţie care are de îndeplinit trei sarcini:

1. Să afişeze timpul.
2. Să controleze prin intermediul tastelor ‚C' şi ‚D' luminozitatea unui LED,
3. Să determine dacă numărul *c* preluat de la tastatură aparţine unei triplete pitagoreic. Detalii suplimentare la pasul 3. Sarcina 3 este opţională în acest laborator, dar deveni obligatorie în laboratorul următor. Această sarcină poate necesita un timp de execuție de ordinul secundelor.

După implementare, afişajul LCD va arăta astfel:

C / L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	9	9	9										P	Y	T	H
2	1	0	:	1	5	:	0	7			L	=	7	5	%	

figura 1

Desfăşurarea lucrării

Pasul 1: Integrarea ceasului cu tastaura

Se va crea un proiect nou cu numele **pwm** care va conţine aceleaşi surse ca proiectul kbd (secţiunea obligatorie) din laboratorul 7. Pentru crearea acestui proiect procedaţi după cum urmează:

a) **Se va crea** un proiectul cu numele **pwm.** La crearea proiectului **NU** bifaţi căsuța **Create initial file** deoarece sursele deja există.

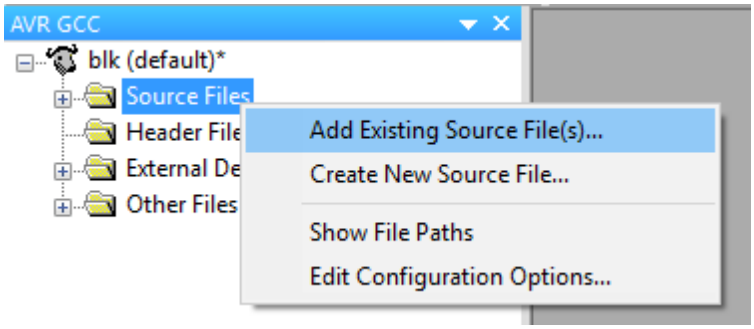
b) **Închideţi** AVR Studio.

c) **Copiaţi IOfn.c** din folderul KBD în folderul proiectului **pwm**.

d) **Copiaţi** **pwm.c** din platforma de laborator în folderul proiectului **pwm**. pwm.c este o copie aproape identica a lui kbd.c

e) **Deschideţi** proiectul pwm.

f) În fereastra **AVR GCC** intrarea **Source Files** este goală. Faceţi click dreapta pe **Source Files** şi din meniul contextual ce va apare selectaţi **Add existing Source File(s):**



Din folderul pwm **Adăugați** la proiect fişierele **pwm.c** şi **IOfn.c**.

n pwm.c a fost modificată secţiunea de afişare a caracterului preluat de la tastatură astfel încât să se afişeze întotdeauna caracterul tastat pe prima poziție a primei linii (1,1). Pentru aceasta au fost adăugaţi liniile roşii din secțiunea de cod următoare:

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

În continuare se va adăuga în **pwm.c** funcţionalitatea de ceas din proiectul **ceas** din laboratorul 9:

a) Verificați că tipul de oscilator este setat conform indicaţiilor din laboratorul 9, pasul 3. Dacă nu este, modificaţi-l.

b) Adăugaţi **după** sysint() şi **înainte** de while() setarea lui TCCR2 şi a lui OCR2din laboratorul 9 - Ceas. (**NU** în sysinit!)

c) În `while`(1), după afişarea tastei, adăugaţi codul pentru calcularea, actualizarea şi afişarea timpului.

d) Deoarece fiecare sarcină are un loc propriu unde afișează, în acest program nu se va folosi `clrLCD`(). Poziţionarea se va face cu `gotoLC`(…) şi afişarea noilor valori se va face prin suprascriere. Înainte de afişarea timpului poziţionaţi cursorul pe linia 2, coloana 1.

Testaţi dacă se afişează timpul şi tasta apăsată! Dacă da, chemaţi profesorul.

Pasul 2: LED cu luminozitate variabilă

Controlul luminozităţii prin PWM este prezentat în prelegerea 5, capitolul **3.5 „Modul fast PWM – fast Pulse Width Modulation”**. În implementarea din acest laborator diferenţa faţă de curs constă numai în frecvenţa 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 documentaţia ATMega16A începând de la pagina 82. Pentru a determina valorile biților din registrul TCCR0 procedaţi după cum urmează:

a) Determinaţi valorile biţilor WGM00 şi 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. Determinaţi valorile biţilor CS02, CS01 şi 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	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/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) Determinaţi valorile biţilor COM01 şi 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)

e) Programaţi TCCR0 cu valorile determinate la paşii a-c. Documentaţi setarea lui TCCR0 ca în programul de la pagina 22 din prelegerea 5. Setarea lui TCCR0 se face după sysint() şi înainte de while(1). (**NU** în sysinit!)

f) Modificaţi cu clrbit sau setbit DDRB pentru ca pinul OC0 să fie ieşire. Nu modificaţi biţii din DDRB programaţi anterior. Modificarea lui DDTB se face după sysint() şi înainte de while(1). (**NU** în sysinit!)

d) Conectaţi un LED pe pinul OC0 ca în figura 10 din prelegerea 5.

Luminozitatea LED-ului se va modifica prin intermediul tastelor ‚**C’**(reşte) şi ‚**D**’(escreşte).

Luminozitatea se modifică în trepte de 5% din luminozitatea maximă. Apăsarea lui ‚C' măreşte luminozitatea cu 5% iar apăsarea lui ‚D' scade luminozitatea cu 5%. Creşterea şi descreşterea sunt saturate, adică luminozitatea nu poate creşte mai mult de 100% şi nu poate scădea sub 0% indiferent de numărul de apăsări ale tastelor ‚C' şi ‚D'.

Registrul care controlează luminozitatea are 8 biţi şi astfel valorile luminozităţii 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 variaţia luminozităţii este comentată în pwm.c. Decomentaţi şi completați.

Afişaţi valoarea luminozităţii ca în figura 1. Noua valoare se scrie peste vechea valoare. Valoarea iniţială a luminozităţii este 50%.

Testaţi dacă se afişează timpul, tasta apăsată şi dacă se poate modifica luminozitatea LED-ului!

Dacă funcţionează, chemaţi profesorul pentru verificarea funcţionării. Veţi prezenta şi calculele pentru *p*–ul şi Tcycle-ului timerului 0.

Dacă aţi ajuns aici aveţi nota 5.

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

Pe lângă ceas şi controlul luminozităţii se cere să determine dacă numărul *c* preluat de la tastatură aparţine 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 aparţine 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 şi 999. Numerele mai mici ca 100 se vor introduce cu zerouri în faţă. 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, puteţi scrie propria variantă. Totul este să nu aibă o complexitate mai mare şi, cel mai important, să funcționeze.

Pentru început vom elimina secțiunea care afișează tasta apăsată:

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

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 şi se va afișa cifra. Poziționarea este necesară deoarece între două preluări de cifră poate să intervină o afișarea a ceasului. Afișarea ceasului va poziționa cursorul pe linia 2. Dacă nu repoziţionăm 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 poziţiona cursorul pe linia 1, coloan2 şi se va afişa cifra.
- În starea C3 cifra citită de la tastatură se va memora în stringul buf_pit, se va poziţiona cursorul pe linia 1, coloan3 şi se va afişa cifra. După ce s-a preluat a treia cifră se va calcula *c*. Deşi calculul este foarte simplu, dacă nu știți să-l faceţi, folosiți `atoi` din `stdlib`.

  Apoi afişaţi mesajul „*Busy…*” şi verificaţi prin forţă brută dacă există doi întregi *a* şi *b* astfel încât $a^2 + b^2 = c^2$. Generaţi pe *a* şi *b* cu două `for`-uri. Dacă se îndeplinește condiția $a^2 + b^2 = c^2$ setaţi o variabilă spion.

  În final, ştergeţi mesajul „*Busy…*” şi în funcţie de valoarea spionului afişaţi fie „*PYTH*” dacă *c* este este pitagoreic, fie „*NPYTH*” dacă nu este. Ca optimizare, după îndeplinirea condiției puteţi ieși din cele două *for*-uri cu goto.

  Structura codului pentru c pitagoreic este comentată în pwm.c. Decomentaţi şi completați.

Testaţi pentru c=005 şi pentru c=999. Ce se întâmplă pe durata calculelor cu ceasul şi cu controlul luminozităţii?

Nu uitaţi, ceasul şi controlul luminozităţii trebuie să funcționeze ca la pasul 2 (pentru nota 5)

Dacă *c* pitagoreic funcţionează şi aveţi un răspuns la întrebarea de mai sus, chemaţi profesorul.

Veţi obține între 0 şi 5 puncte in funcţie de cât aţi implementat din c pitagoreic şi de calitatea implementării.