Laboratorul 3: Timere, Pulse Width Modulation (PWM)

Capitole utile din Datasheet ATmega324

- 1. Pin Configurations
  - secțiunea 1.1 - pag. 2
- 16. 16-bit Timer/Counter1 and Timer/Counter3 with PWM
  - *secțiunile 16.1-16.3* - pag. 111
  - *secțiunea 16.5* - pag. 117
  - *secțiunea 16.7* - pag. 120
  - secțiunea 16.8 - pag. 122
  - secțiunile 16.9.3-16.9.5 - pag. 124
  - secțiunea 16.11 - pag. 132
- 15. 8-bit Timer/Counter0 with PWM
  - secțiunea 15.9 - pag. 104
- 17. 8-bit Timer/Counter2 with PWM and asynchronous operation
  - secțiunea 17.11 - pag. 156
- Note:
  - secțiunile cu *italic* au fost studiate și în laboratorul anterior.
  - secțiunile pentru Timer/Counter0 și Timer/Counter2 conțin doar detaliile registrelor; secțiunile de funcționare sunt omise fiind foarte asemănătoare cu Timer/Counter1

Capitolele sunt din Datasheet ATmega324, document pe care îl aveți și pe desktop-ul calculatorului din laborator. Nu garantăm aceeași ordine a capitolelor în cazul utilizării altui document!

PWM (**P**ulse **W**idth **M**odulation) este o tehnică folosită pentru a varia în mod controlat tensiunea dată unui dispozitiv electronic. Această metodă schimbă foarte rapid tensiunea oferită dispozitivului respectiv din ON în OFF și invers (treceri rapide din HIGH în LOW, de exemplu 5V - 0V). Raportul dintre perioada de timp corespunzătoare valorii ON și perioada totală dintr-un ciclu ON-OFF se numește factor de umplere (*duty cycle*) și reprezintă, în medie, tensiunea pe care o va primi dispozitivul electronic. Astfel, se pot controla circuite analogice din domeniul digital. Practic, asta înseamnă că un LED acționat astfel se va putea aprinde / stinge gradual, iar în cazul unui motor acesta se va învârti mai repede sau mai încet.

În cazul unui motor, căruia i se aplică un semnal PWM cu factor de umplere de 0%, viteza de rotație a acestuia va fi egală cu 0 rpm. Un factor de umplere de 100% va duce la o turație maximă a acestuia.

1. Principiul de funcționare

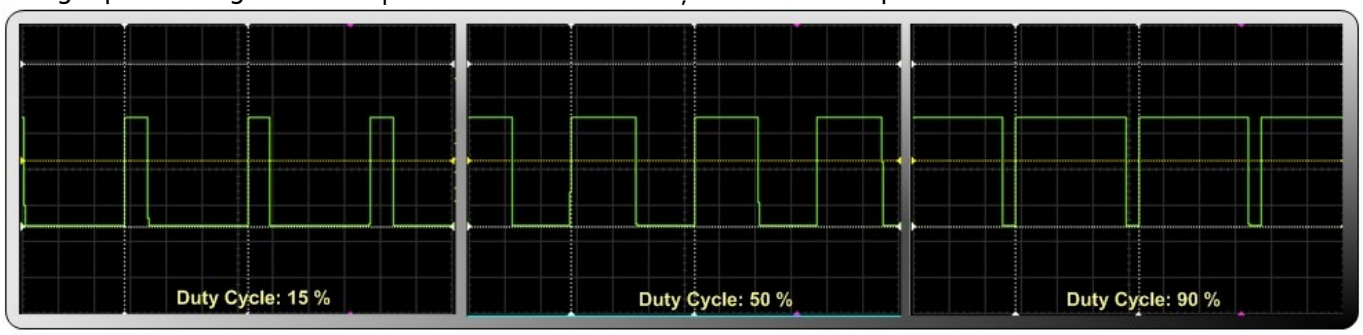
Factorul de umplere se exprimă în procente și reprezintă cât la sută din perioada unui semnal acesta va fi pe nivelul ON. În Figura 1 se pot observa semnale PWM cu factori de umplere diferiți. Astfel, se

poate deduce foarte ușor formula pentru a obține valoarea factorului de umplere (D):

```
D = t_on / (t_on + t_off) * 100 = pulse_width / period * 100
```

Astfel, tensiunea medie care ajunge la dispozitiv este dată de relația: D * Vcc.

<imgcaption image8 center |Semnal PWM cu diferiți factori de umplere >



</imgcaption>

Modularea folosește variația factorului de umplere a unei forme de undă dreptunghiulară pentru a genera la ieșire o tensiune analogică. Considerând o formă de undă dreptunghiulară *f(t)* cu o valoare minimă *ymin* și o valoare maximă *ymax* și factorul de umplere *D* (ca în figura <imgref image8>) valoarea medie a formei de undă e dată de relația:

cum *f(t)* este o formă de undă dreptunghiulară valoarea sa maximă se atinge pentru 0<t<D*T.

Multe circuite digitale pot genera semnale PWM. Majoritatea microcontroller-elor oferă această facilitate, pe care o implementează folosind un numărător care este incrementat periodic (conectat direct sau indirect la o unitate de ceas) și care este resetat la sfârșitul fiecărei perioade a PWM-ului. Când valoarea numărătorului este mai mare decât valoarea de referință, ieșirea PWM (output-ul) trece din starea HIGH în starea LOW (sau invers).

1.1. PWM folosind Atmega324

În cadrul laboratorului trecut am văzut că microcontroller-ul Atmega324 are 3 timere: **Timer0** pe 8 biți, **Timer1** pe 16 biți și **Timer2** pe 8 biți.

Fiecare dintre aceste timere putea fi configurat în diferite moduri. În cazul timer-ului 1 configurarea se realiza prin intermediul registrelor **TCCR1A** și **TCCR1B** cu ajutorul biților `WGM13 - WGM10`. Printre aceste moduri se numărau:

- **Normal mode**
- **CTC mode** (Clear Timer on Compare match) cu TOP la **OCRnA**
- **CTC mode** cu TOP la **ICRn**

Atmega324 dispune de 6 canale de PWM distribuite astfel:

- **timer0 - OCR0A, OCR0B** respectiv **timer2 - OCR2A, OCR2B** - în total 4 canale pe 8 biți
- **timer1 - OCR1A, OCR1B** în total 2 canale pe 16 biți

Controller-ul dispune de 6 canale de PWM în cazul în care acestea sunt configurate în moduri ce au ca TOP valorile 0xFF respectiv 0xFFFF. Alternativ, putem configura fiecare timer (atât cele de 8 biți cât și cele de 16 biți) în moduri ce au ca TOP o valoare specificată într-unul din registrele OCRnA sau OCRnB. Un timer configurat astfel are doar un canal ce genereaza output așa cum ne-am aștepta. Și canalul corespunzător registrului ce conține valoare TOP va genera semnal, însa nu în modul dorit.

Conform tabelului **16-5.** din datasheet, Timer-ul 1 mai poate fi configurat să funcționeze și în modul PWM. Din punct de vedere al microcontroller-ului Atmega324 există 3 tipuri de PWM:

- **Fast PWM**
- **Phase Correct PWM**
- **Phase and Frequency Correct PWM**

1.1.1. Fast PWM

Numărarea se face doar pe frontul crescător al semnalului de ceas. În modul Fast PWM, modificarea factorului de umplere se realizează instant, în schimb semnalul nu este centrat (este defazat, practic apare un “glitch” la modificarea semnificativă a factorului de umplere). Se utilizează pentru majoritatea aplicațiilor, mai puțin cele în care este nevoie de un control precis (de exemplu motoare BLDC, audio). Există mai multe moduri de Fast PWM oferite de către microcontroller. De exemplu, pentru Timer-ul 1 avem:

- Fast PWM pe 8 biți, cu valoarea de TOP = 0x00FF.
- Fast PWM pe 9 biți, cu valoarea de TOP = 0x01FF.
- Fast PWM pe 10 biți, cu valoarea de TOP = 0x03FF.
- Fast PWM cu valoarea de TOP în **ICR**
- Fast PWM cu valoarea de TOP în **OCRnA**

<imgcaption image11 center |Fast PWM Timing></imgcaption>

Pentru variantele de Fast PWM oferite de către celelalte timere (0 și 2), studiați datasheet-ul.

1.1.2. Phase Correct PWM

Numărarea se face pe ambele fronturi ale ceasului. Pentru aceleași configurații ca Fast PWM, Phase Correct PWM este de două ori mai lent (dar are o rezoluție mai bună). În comparație cu modul Fast PWM, semnalul este centrat (modificare simetrică a factorului de umplere), efectul fiind practic vizibil la modificarea factorului de umplere.

Există și pentru acest mod mai multe variante de configurare, asemănătoare celor de la Fast PWM.

<imgcaption image12 center |Phase Correct PWM Timing></imgcaption>

Comparație între Fast PWM și Phase Correct PWM:

<imgcaption image9 center |Fast PWM (stanga) vs. Phase Correct PWM (dreapta)></imgcaption>

1.1.3. Phase and Frequency Correct PWM

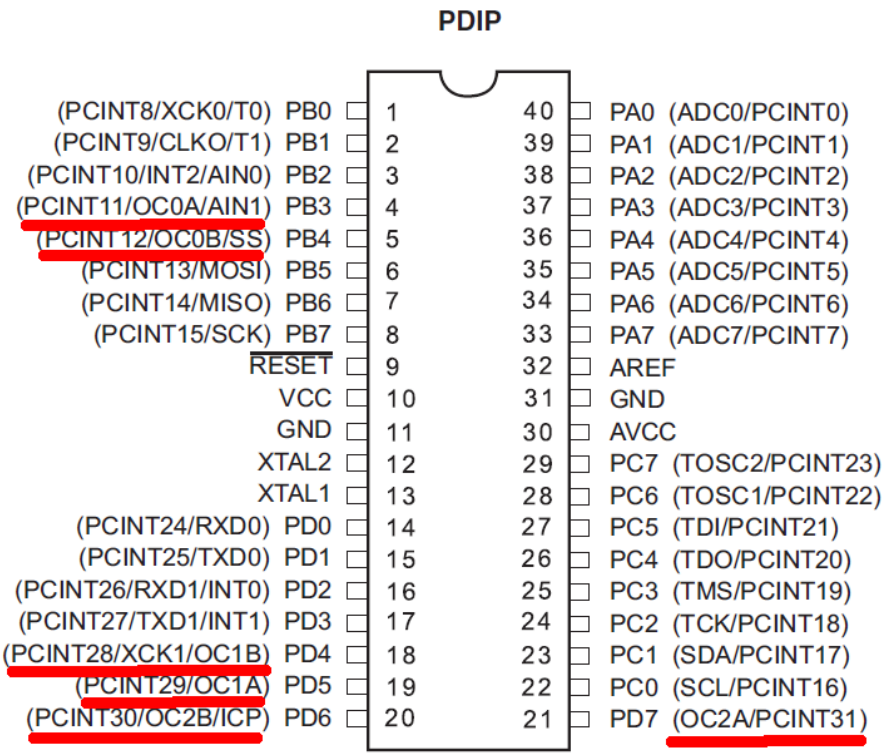
Principala diferență între acest mod și cel de Phase Correct este momentul în care este actualizat registrul OCRnx. Practic, spre deosebire de modul Phase Correct PWM se păstrează durata semnalului ON pe toată perioada *T_on + T_off* și rezultă un semnal "curat", fără distorsiuni (de exemplu frecvențe nedorite care pot avea ca efect pierderi de energie în motoare DC).

<imgcaption image12 center |Phase And Frequency Correct PWM Timing></imgcaption>

2. Lucrul cu PWM-ul

Lucrul cu PWM-ul presupune inițializarea unui timer și configurarea output-ului pe pini. Fiecare timer are doi pini pe care genera ca output un astfel de semnal (cele două canale): Timer0 are `OC0A` și `0C0B`, Timer1 are `OC1A` și `OC1B` etc.

<imgcaption image123 center |ATmega324 PWM Output Pins>



</imgcaption>

De exemplu, presupunem că avem Timer1 configurat pe modul de funcționare Fast PWM (modul de funcționare nu trebuie neapărat să conțină cuvântul 'PWM' ca să poată fi folosit pentru generarea unui semnal). Fast PWM este caracterizat de o frecvență fixă și un prag stabilit de programator, ce poate fi modificat în timpul execuției.

Vom configura tipul de output cu biții numiți `COM..` din registrele `TCCR..` ale timer-ului corespunzător (în exemplul nostru timer-ul 1). Descoperim în capitolul 16 (*16-bit Timer/Counter1 and Timer/Counter3 with PWM*) - secțiunea *Register Description - TCCR1A*, *TCCR1B* și *TCCR1C*

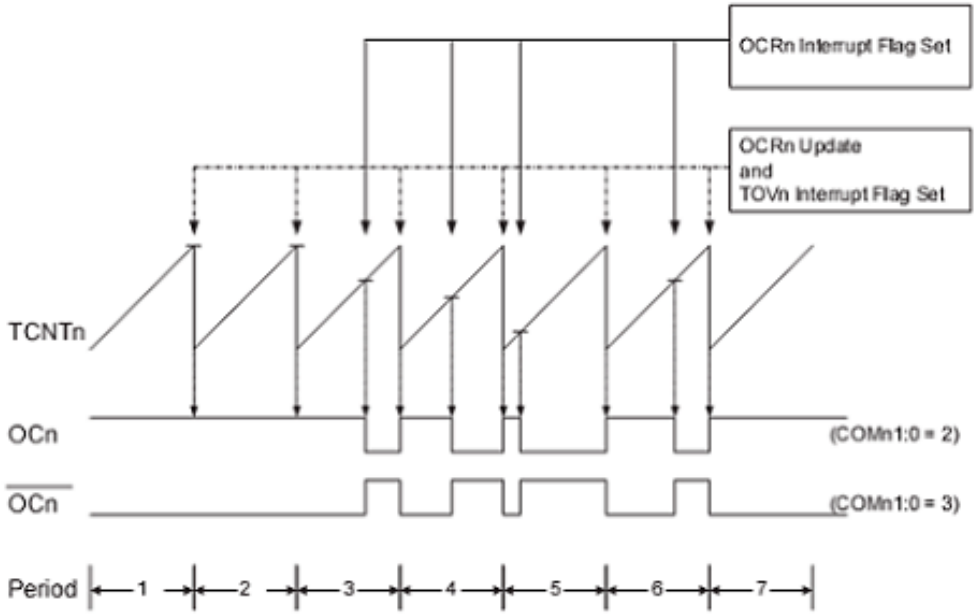
Vedem că:

- Există un tabel pentru biții `COM..` special pentru modul Fast PWM

Atenție! Există câte un tabel de configurare a biților `COM` pentru fiecare mod de funcționare a PWM-ului!

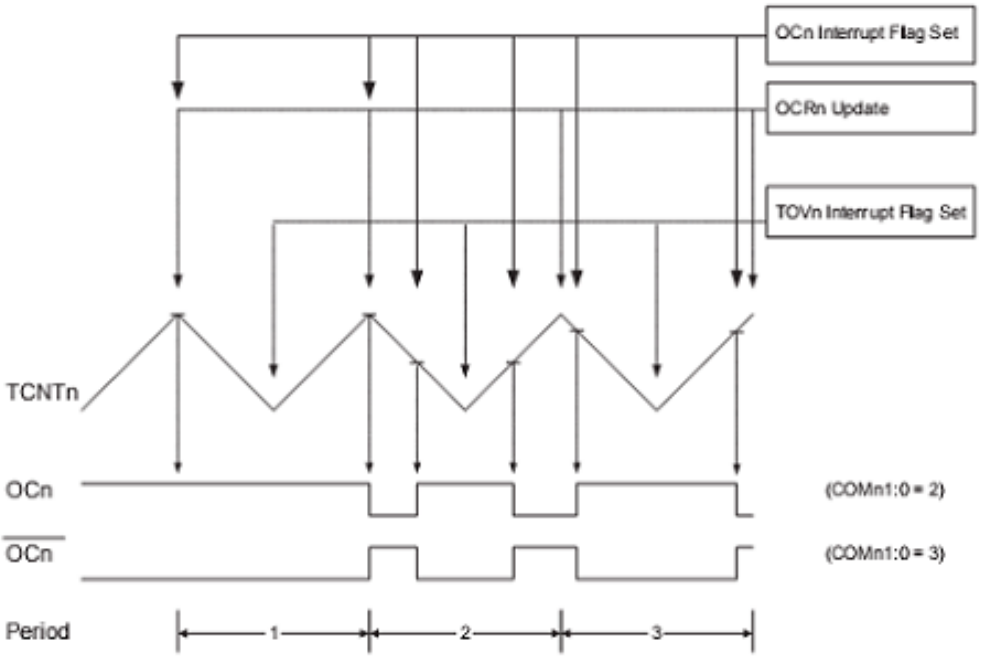
- Modul `1 0` pentru biții `COM1A1 COM1A0` va lăsa semnalul de pe pinul `OC1A` pe 1 în timpul numărătorii (până la atingerea pragului) și va pune semnalul pe 0 de la atingerea pragului până la capătul unui ciclu (numărare completă până la TOP). Modificările făcute asupra semnalului în funcție de prag și counter în acest mod se pot observa în desenul din Figura 4.
- Pentru a obține un factor de umplere x%, aici vom seta `OCR1A = x * TOP / 100` (pentru Fast PWM, TOP poate fi 0xFF, 0x1FF sau 0x3FF, în funcție de configurația aleasă)

<imgcaption image11 center |Fast PWM Timing>



</imgcaption>

<imgcaption image12 center |Phase Correct PWM Timing>



</imgcaption>

Exemplu de inițializare a Timer1 in modul Fast PWM 8-bit non-inverting, cu Prescaler la valoarea 1024 (va genera un semnal asemănător celui din Figura 4) :

```
 //PD5 output - OC1A este PD5
 DDRD |= (1 << PD5);
 //Conform tabelului 16-5 din datasheet, pentru modul Fast PWM 8-bit, biții
WGMn0 si WGMn2 au valoarea 1
 TCCR1A |= (1 << WGM10);
 //TCCR1A conține doar biții WGM10 si WGM11, WGM12 și WGM13 se găsesc in
TCCR1B
 TCCR1B |= (1 << WGM12);
 //Conform tabelului 16-3 din datasheet, pentru modul non-inverting COM1A1 =
1 și COM1A0 = 0
 TCCR1A |= (1 << COM1A1);
 //Conform tabelului 16-6 din datasheet, pentru Prescaler de 1024 scriem 1
pe CS12 si CS10
 TCCR1B |= (1 << CS12) | (1 << CS10);
 //Pragul la care se schimbă semnalul (Fig. 4); fill-rate 0.5: jumătate din
timp 1, jumătate din timp 0
 //Deoarece în acest mod TOP este 0xFF, OCR1A va fi 50 * 255 / 100
 OCR1A = 255 / 2;
```

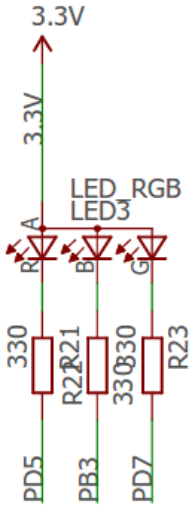
3. Exerciții

Obiectivul exercițiilor este să controlăm culoarea unui LED RGB cu ajutorul PWM. Un LED RGB este compus din 3 diode care emit culori diferite: una roșie, una verde și una culoare albastră. Cu acest LED se poate obține orice culoare printr-o combinație de intensități pe fiecare diodă în parte. Cele trei

LED-uri sunt conectate la:

- `OC1A` este asociat timer-ului 1 (pinul PD5). Acest timer controlează LED-ul `roșu`.
- `OC2A` este asociat timer-ului 2 (pinul PD7). Acest timer controlează LED-ul `verde`.
- `OC0A` este asociat timer-ului 0 (pinul PB3). Acest timer controlează LED-ul `albastru`.

LED-urile sunt conectate in modul “active-low” (LED-ul este aprins atunci cand pinul aferent este LOW, si stins atunci cand pinul este HIGH).



<imgcaption ledrgb center|> </imgcaption>

1. **(1p)** Descărcați arhiva cu scheletul de cod și rulați exemplul. Ce observați că se întamplă pe plăcuță?
   - Ce reprezintă valorile afișate pe LCD?
   - Ce puteți spune despre comportamentul LED-ului?
   - Modificați funcția de inițializare a timer-ului 1 astfel încât sa funcționeze în modul Fast PWM cu `TOP` la `ICR1`.
   - Calculați valoarea lui `ICR1` pentru ca frecvența la care funcționează timer-ul sa fie **exact 1Hz**.
   - Aveți grijă ca duty cycle-ul pentru `canalul A` să rămână 25%, iar pentru `canalul B` să rămână 50%.
2. **(1p)** Plecând de la implementarea exemplului, modificați codul astfel încât sa obțineți același comportament folosind semnal PWM în locul întreruperilor.
   - Valorile biților `COM` trebuie setate conform tabelului 16-3 din datasheet (Capitolul 16.11.1). **Hint**: Deoarece avem LED-uri intr-o configuratie active-low, trebuie ales modul 'inverting'.
   - LED-ul este conectat pe canalul A al timer-ului.
3. **(2p)** Dorim să vizualizăm efectele prescaler-ului asupra frecvenței cu care clipește LED-ul. Pentru aceasta, vom folosi butonul legat la PB2 pentru a cicla prin valorile posibile ale acestuia.
   - Implementați rutina de tratare a întreruperii generate de butonul PB2.
   - Pentru a seta prescaler-ul folosim biții CS - tabelul 16-6 din datasheet (Capitolul 16.11.2)
   - Valoarea inițială a prescaler-ului trebuie să fie 1.
   - Trebuie să implementați și debounce pentru buton. Recomandăm folosirea unui delay de minim **50ms**.
   - Observați ce se întâmplă cu frecvența afișată pe LCD. Care frecvență corespunde cărui prescaler?
4. **(1p)** Acum că putem controla prescaler-ul, vrem să putem controla și factorul de umplere. Vom folosi butonul legat la PD6 pentru a controla valoarea factorului de umplere.
   - Adăugați logica de modificare a factorului de umplere la rutina de tratare a întreruperii implementată anterior.

- Pentru afișare optimă pe LCD, incrementați în multipli de 1/16 sau 1/8.
- Atenție la debounce!
- Ce puteți spune despre LED atunci când variați factorul de umplere?

5. Pentru a completa LED-ul RGB, avem nevoie și de celelalte două LED-uri, conectate pe output-urile timer-elor 0 și 2 (OC0A și respectiv OC2A).
   - **(2p)** Implementați inițializarea Timer-elor 0 și 2 pentru a funcționa în modul următor:
     - Mode of Operation: Fast PWM
     - Prescaler: 1
     - Compare Output Mode: Inverting (set atunci când ajunge la prag, clear la overflow)
     - pragul inițial 0xC0
   - **(1p)** Implementați funcția `update_color` pentru a da o nouă culoare LED-ului RGB. Această funcție este apelată periodic de bucla principală.
     - Funcția `update_color` va trebui să funcționeze după următorul automat de stări:
       - Starea 1 (cea inițială): `roșu` și `verde` au duty cycle de 100%, `albastru` de 0%. În această stare, `roșu` scade și `albastru` crește. Se trece în starea 2 când `roșu` are duty cycle 0%, `albastru` are duty cycle 100%.
       - Starea 2: În această stare `roșu` crește și `verde` scade. Se trece în starea 3 când `roșu` are duty cycle 100% și `verde` are duty cycle 0%.
       - Starea 3: În această stare `verde` crește și `albastru` scade. Se trece în starea 1 când `verde` are duty cycle 100% și `albastru` are duty cycle 0%.
6. **(2p)** Speaker-ul de pe placă este conectat pe pinul PD4, așa cum am văzut în primul laborator. PD4 este și ieșirea B a timer-ului 1, deci am putea folosi un factor de umplere de 50% pentru a produce sunet pe speaker. Scopul acestui exercițiu este să redați melodia pusă în scheletul de laborator.
   - `surprise_notes` sunt diferitele note folosite, cu valorile în Hertzi
   - `durations` este un vector cu duratele asociate fiecărei note din `surprise_notes`
   - Modificați Timer-ul 1 să funcționeze în modul CTC cu TOP la OCR1A și setați prescaler-ul la valoarea 1024.
   - Implementați funcția `update_song` astfel încât la intervalele date de `durations` schimbați frecvența Timer-ului 1 (care e determinată de OCR1A) pentru a reda frecvențele din `surprise_notes`; funcția este apelată în bucla principală la fiecare 25ms.
   - Resetați TCNT la fiecare update al TOP-ului. Altfel, în unele situații, va trebui să așteptați un overflow pentru ca TCNT să se reseteze.
7. **(Bonus 1p)** Implementarea melodiei de la exercițiul anterior a stricat funcționarea LED-ului conectat la pin-ul PD5. Faceți modificările necesare astfel încât LED-ul și melodia să funcționeze în același timp.

4. Resurse

- Schelet laborator
- PDF laborator
- Datasheet ATmega324
- Responsabili: Cristi Trancă

5. Linkuri utile

- [1] Tutorial video PWM
- [2] Lucrul cu PWM pentru microcontroller-ele Atmel AVR
- [3] Newbie's Guide to AVR PWM
- [4] Diferenta dintre Phase Correct si Fast PWM