Copăcelul Muzical

Mureşan Bianca-Maria

Ianuarie 2019

Descriere

Proiectul constă în construirea unui copac electronic care iese dintr-un ghiveci cu 20 de ramuri, iar la sfârşitul fiecăreia se va afla un LED colorat (în total vor fi patru culori). Pe ghiveci se va afla un buton care la fiecare apăsare va schimba melodia curentă (în total vor fi 3 melodii). De asemenea, tot pe ghiveci se va afla un potenţiometru pentru reglarea volumului.

Date de intrare şi de ieşire

Date de intrare:

- starea butonului (Low/High).

Date de ieşire:

- cele 3 melodii (frecvenţe pe buzzer);
- luminile de pe ramuri(tensiunea de pe LED-uri).

Proiectarea fizică

Obiecte componente:

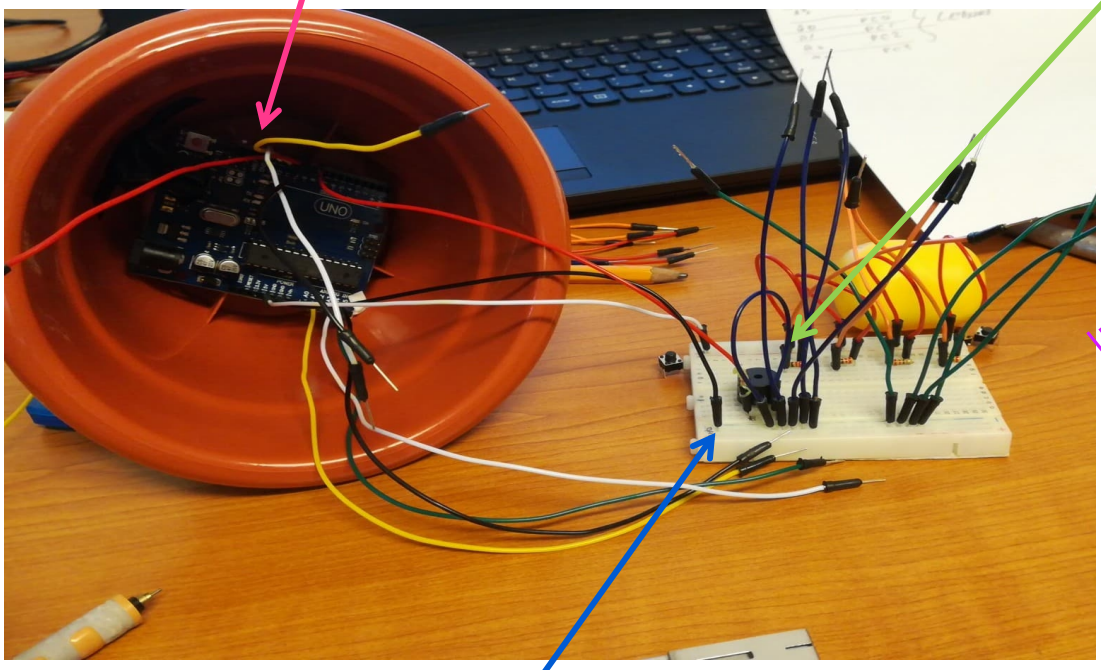
- o placă de Arduino (ATmega328P);
- un buzzer şi un potenţiometru;
- un buton;
- 5 LED-uri a cate 4 culori (roşu, galben, verde, albastru);
- un breadboard, rezistențe și fire.

Proiectarea ghiveciului

Pentru a proiecta arborele conform descrierii, pe fundul ghiveciului vom lipi placa de Arduino și vom găurii ghiveciul pe lateral pentru a-l putea alimenta şi pentru a putea să conectăm butonul. Peste placa de Arduino va fi plasat breadboard-ul de care este conectat buzzer-ul, iar peste el un capac cu rol de protejare a circuitelor interioare. De asemenea, pe capac va fi amplasat un potenţiometru.

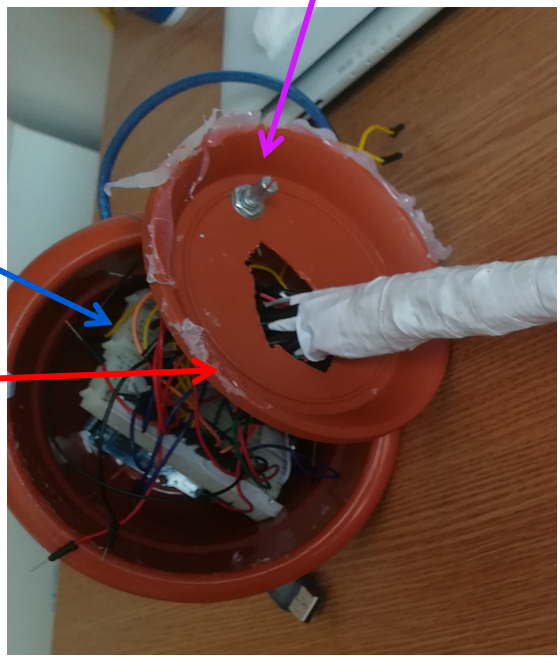
Placa de Arduino

Buzzer



Potentiometru

Breadboard



Capac

Buton

Alimentare

Proiectarea buzzer-ului

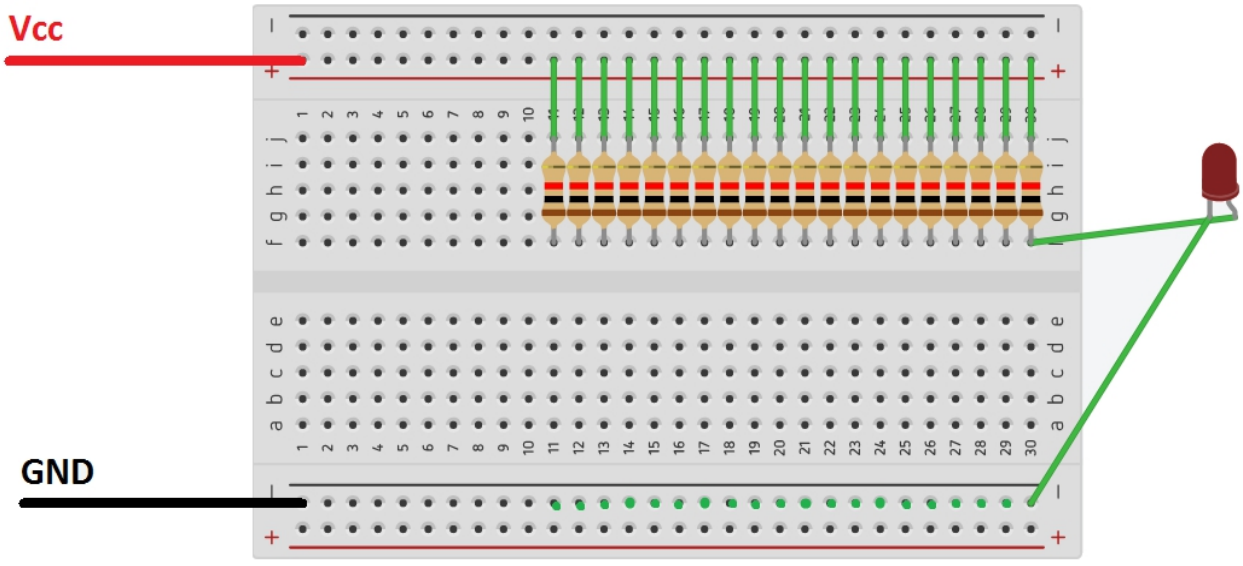
Implementarea melodiilor se va realiza cu ajutorul unui buzzer pentru redarea notelor şi a unui potenţiometru pentru reglarea volumului. Buzzer-ul va fi amplasat pe breadboard, iar potenţiometrul pe capacul ghiveciului. Un pin de la buzzer va fi conectat la GND, iar celălalt la potenţiometru. Ceilalţi doi pini ai potenţiometrului vor fi conecţati la GND și respectiv la pinul PB1 de pe Arduino.

Proiectarea butonului

Butonul va fi amplasat pe exteriorul ghiveciului, deasupra alimentării. Ghiveciul va fi găurit în 4 puncte în care vor intra pinii butonului. Doi dintre aceşti pini vor fi conectaţi la GND, respectiv la Vcc pe breadboard, iar al treilea la placa de Arduino pe pinul PD2 pentru a-i putea verifica starea (Low/High).

Proiectarea LED-urilor

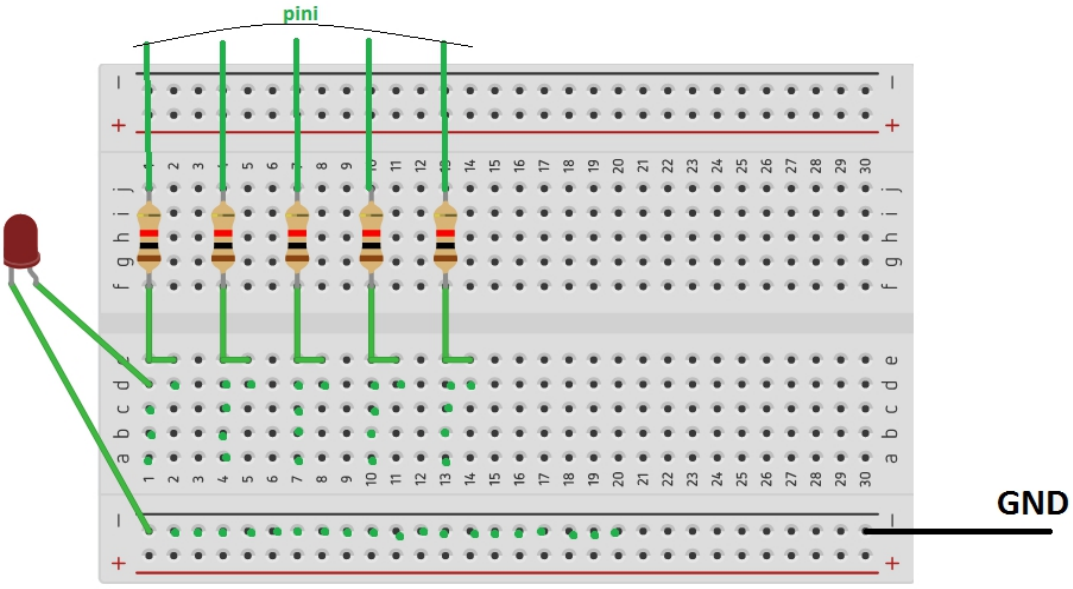
Varianta 1: Legarea în paralel la Vcc



Presupunem că de fiecare rezistenţă este legat un LED, precum la ultima rezistenţă din poza de mai sus. Tensiunea pe ansamblul rezistenţă-LED înseriate va fi egală cu Vcc (5V), tensiunile pe LED-uri variind în funcţie de rezistenţe.

Această variantă va ţine LED-urile mereu pe ON cât timp circuitul este alimentat, neputând să le controlăm după bunul plac.

Varianta 2: Legarea în paralel a câte 5 LED-uri pe un pin



Presupunem că avem 4 pini pe Arduino de unde vor ieşi semnalele de comandă pentru un ansamblu de 5 rezistenţe înseriate cu 5 LED-uri precum în imaginea de mai sus, fiecare ansamblu din cele 4 reprezentând o culoare(roşu, galben, verde sau albastru). Tensiunea pe ansamblul rezistenţă-LED înseriate va fi egală cu tensiunea de pe pin-ul de comandă când e ON (5V), tensiunile LED-urilor variind în funcţie de rezistenţe.

Această variantă ne va lăsa să controlăm fiecare set de culoare după bunul plac, însă avem prea puţin curent pe un pin de comandă (intensitate de maxim 40 mA) care trebuie împărţit mai apoi la cele 5 ansambluri rezis-

tenţă-LED înseriate, rezultând un curent maxim de 8 mA, iar pentru ca o diodă să se aprindă avem nevoie de minim 10 mA.

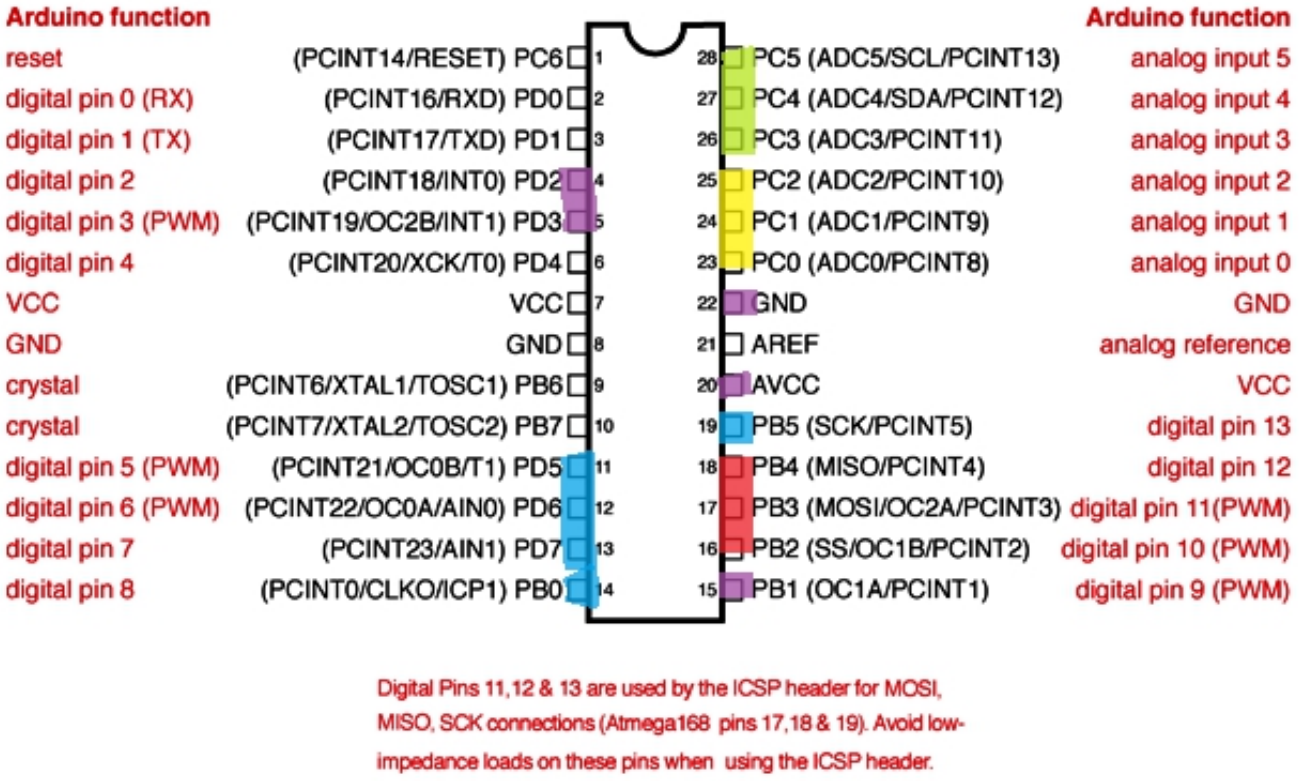
Varianta 3: Legarea în serie

Această variantă presupune legarea în serie de două ori a câte două LED-uri de o culoare şi o rezistenţă, rămânând un LED pe dinafară, legat în serie cu o rezistenţă, având astfel un curent maxim pe fiecare ansamblu. Excepţie fac LED-urile albastre care, fiind cele mai consumatoare, vor fi legate în serie cu o rezistenţă, fiecare cu pinul ei de comandă. Folosind legea lui Ohm şi ştiind de câţi Volţi avem nevoie pentru a aprinde fiecare LED în funcţie de culoarea sa, putem afla valorile rezistenţelor noastre.

Astfel vom avea 3 pini necesari pentru roşu, galben, verde si 5 pini pentru albastru, în total fiind folosiți 14 pini pentru LED-uri.

Acestă variantă rămâne şi varianta finală.

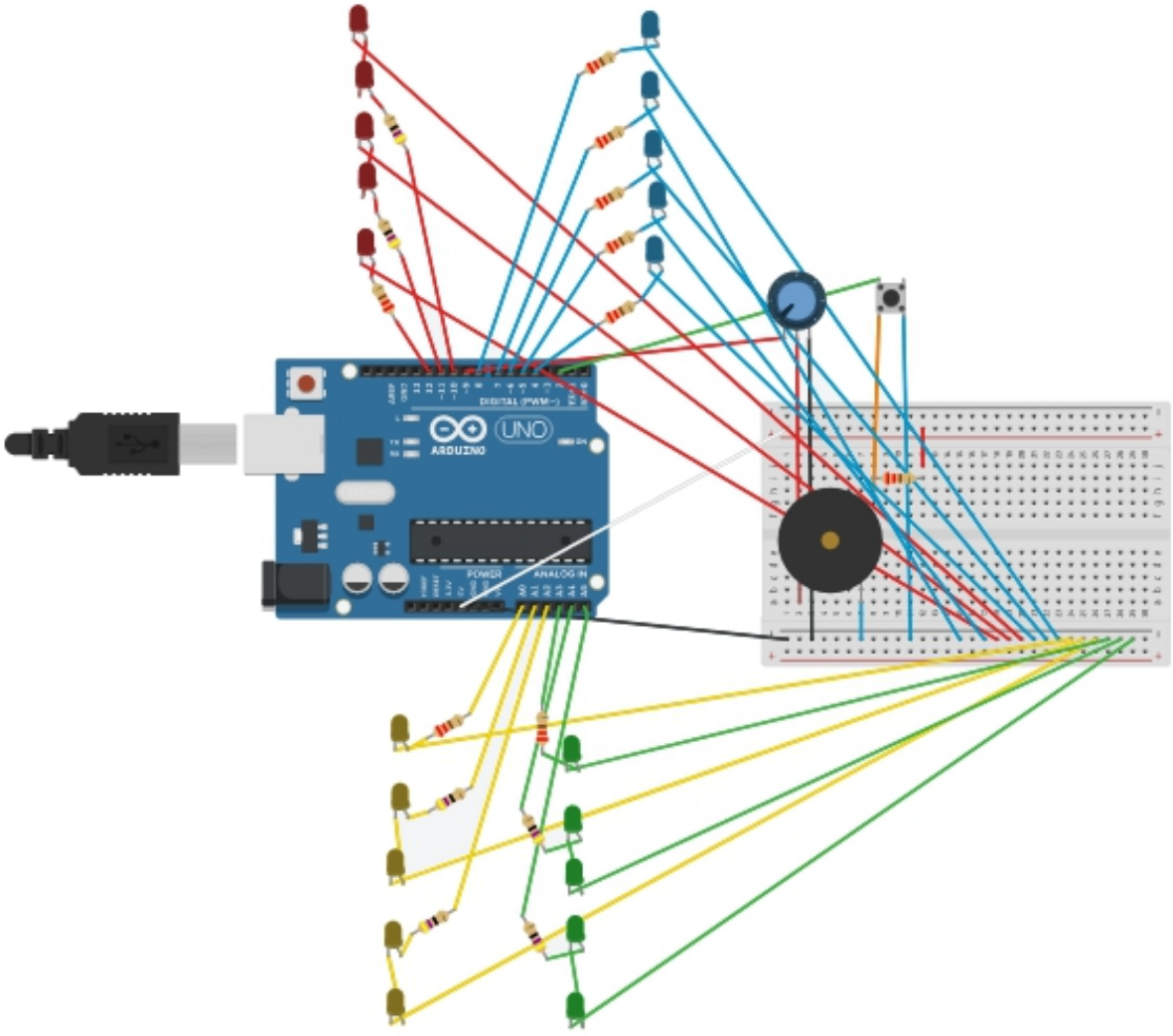
Alegerea pinilor

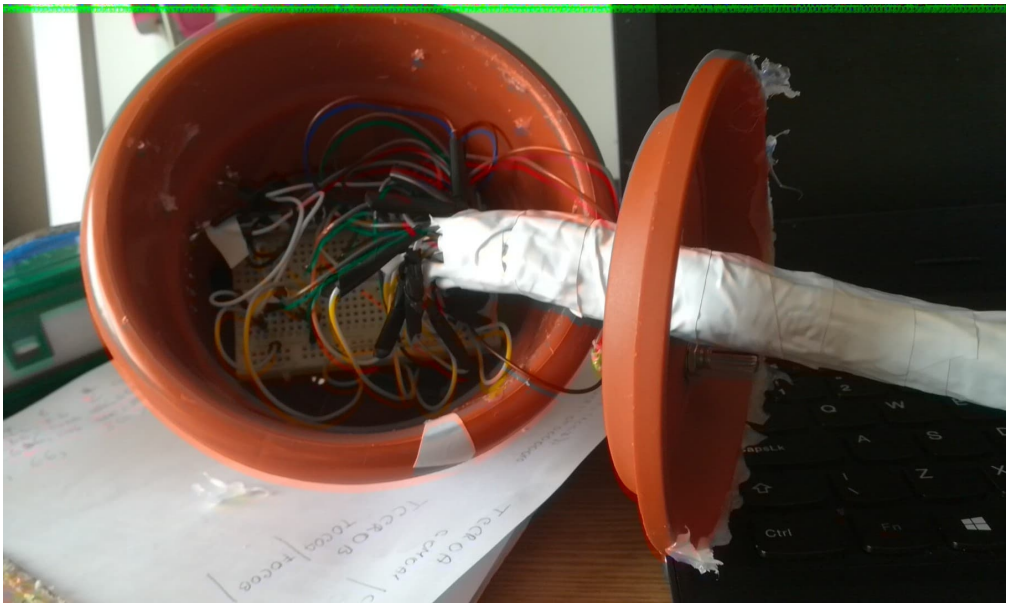


Pinii coloraţi în albastru, roşu, galben şi verde reprezintă pinii aleşi pentru controlul LED-urilor, iar cei cu mov reprezintă restul pinilor, aleşi astfel:

- PB1 - pinul care comandă buzzerul. Acesta are nevoie de un semnal PWM pentru a reda frecvenţa şi durata notelor melodiei. Pentru a ne putea creea PWM-ul după cerinţele problemei, portul trebuie să dispună de OCR1A sau OCR2B de pe pinul PD3.
- PD2 - pinul care citeşte semnalele de intrare de pe buton.

Schema fizică





Codul

Iniţializare Timer0:

```
void timer0_init() {
SREG = 1<<7;  // Activează întreruperile globale

TCCR0A = 0b00000010;  //Port normal de operare
                      //OC0A şi OC0B deconectaţi
TCCR0B = 0b00000011;  //CTC cu OC0RA,
                      //prescaler de 64

TCNT0 = 0;  //registru de numărare timer0
OCR0A = 250;  //16.000.000/64=250.000/250=1.000Hz
              //=>T=1ms
TIMSK0 |= 0b00000010;  //Activează întreruperile
                       //interne de comparare la ieşire
                       //pentru OCR0A
}
```

Iniţializare întreruperi:

```
void interrupt_init(){
EIMSK=0b00000011;  //Activează întreruperile externe

EICRA=0b00001010;  //Frontul crescător pentru INT0
                   //(PD2)
}
```

Inițializare buzzer:

```
void buzzer_init(){
  DDRB |= 0b00000010; //PB1 e setat ca ieşire pentru
                      //semnalul PWM al buzzer-ului

  TCCR1A =  0b01000000; // timer-ul1: CTC cu OC1A,
  TCCR1B =  0b00001010; //prescaler de 8, pinul OC1A
  TCCR1C =  0b00000000; //setat ca ieşire pentru
                        //semnalul PWM
 }
```

Inițializare buton:

```
void button_int(){
    DDRD &=~ 0b00000100;  //PD2 setat ca intrare
}
```

Inițializare LED-uri

```
void  LEDs_init(){
DDRC |= 0b00000111;  //ieşire galben
DDRC |= 0b00111000;  //ieşire verde
DDRB |= 0b00111101;  //ieşire roşu
DDRD |= 0b11100000;  //ieşire albastru
}
```

Variabile globale:

1. Totalitatea notelor pe care le vom folosi (ex: float C4 = 261.63;);
2. Tempoul pentru melodii (int sec;);
3. Vectorii ce conţin notele în ordine şi cei ce conţin timpul necesar fiecărei note în funcţie de tempoul melodiei;
4. Milisecundele(long int ms = 0;), contorul pentru vectorul de melodii(int i =0;) şi variabila TOP folosită la frecvenţă(long int TOP;);
5. Contorul folosit pentru schimbarea melodiilor (int contor= -1;).

Main:

```
int main(){
  timer0_init();
  buzzer_init();
  button_int();
  LEDs_init();
  interrupt_init();

  while(1){
  }
}
```

Întreruperi:

```
ISR(INT0_vect){
  contor++;
  i=0;
  ms=0;
  if(contor==3)
      contor=0;
}
```

//Se activează la apăsarea butonului (PD2) programul se opreşte şi execută întreruperea

Semnalul PWM

Pentru a genera un semnal de ieșire în modul CTC, pinul OC0 trebuie setat ca și pin de ieșire digitală. Modul de ieșire pentru OC0 poate fi setat ca să basculeze de fiecare dată când se găsește egalitate între TCNT0 și OCR0. Frecvența semnalului este definită de ecuația următoare:

$$f_{OC0} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCR0)}$$

unde $f_{clk\_I/O}$ reprezintă frecvența ceasului intern al sistemului, N este valoarea prescaler-ului (1, 8, 64, 256, 1024) iar ORC0 este valoarea scrisă în acest registru.

TOP = 1000000/frecventa
OCR1A = (TOP+1)/2

Unde frecvenţa înseamnă frecvenţa pe care dorim să o implementăm.