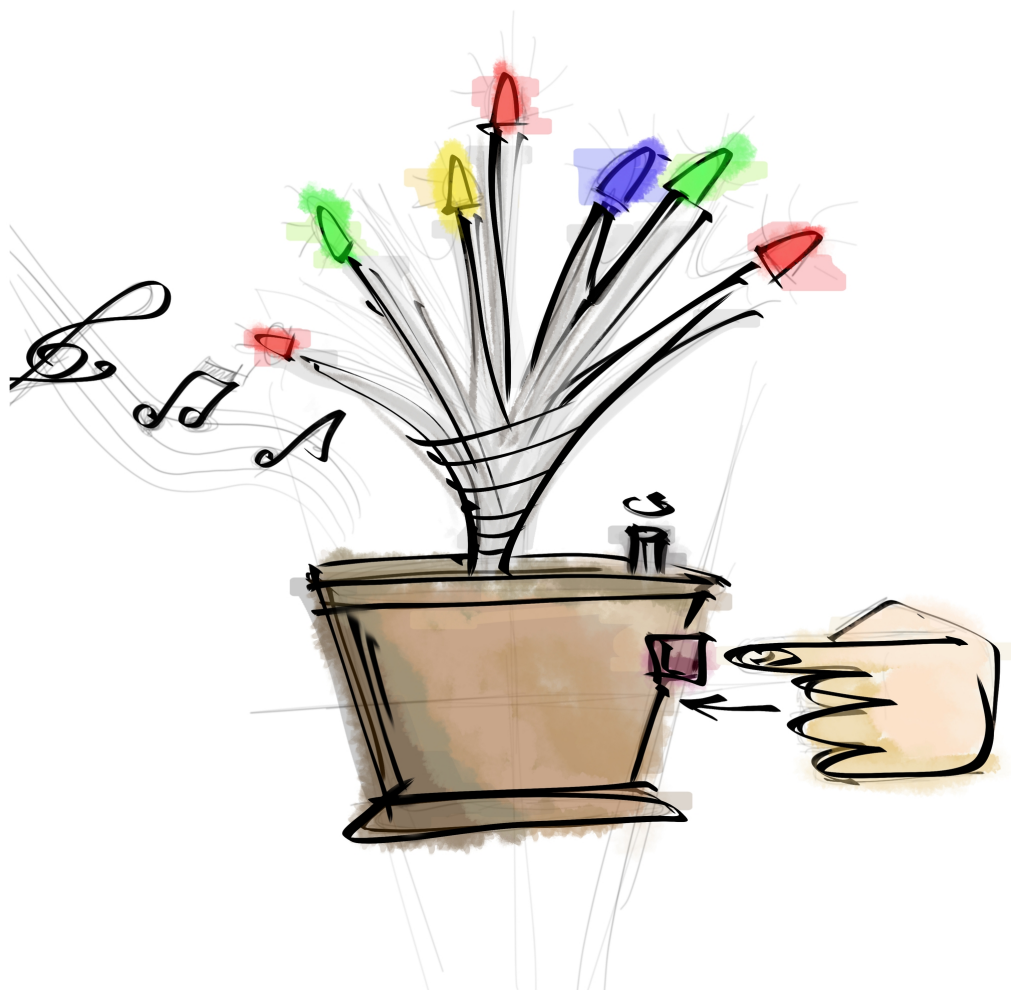


# Copăcelul Muzical

Mureșan Bianca-Maria

Ianuarie 2019



# Descriere

Proiectul constă în construirea unui copac electronic care iese dintr-un ghiveci cu 20 de ramuri, iar la sfârșitul 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 potențiometrul pentru reglarea volumului.

## Date de intrare și de ieșire

Date de intrare:

- starea butonului (Low/High).

Date de ieșire:

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

# Proiectarea fizică

Obiecte componente:

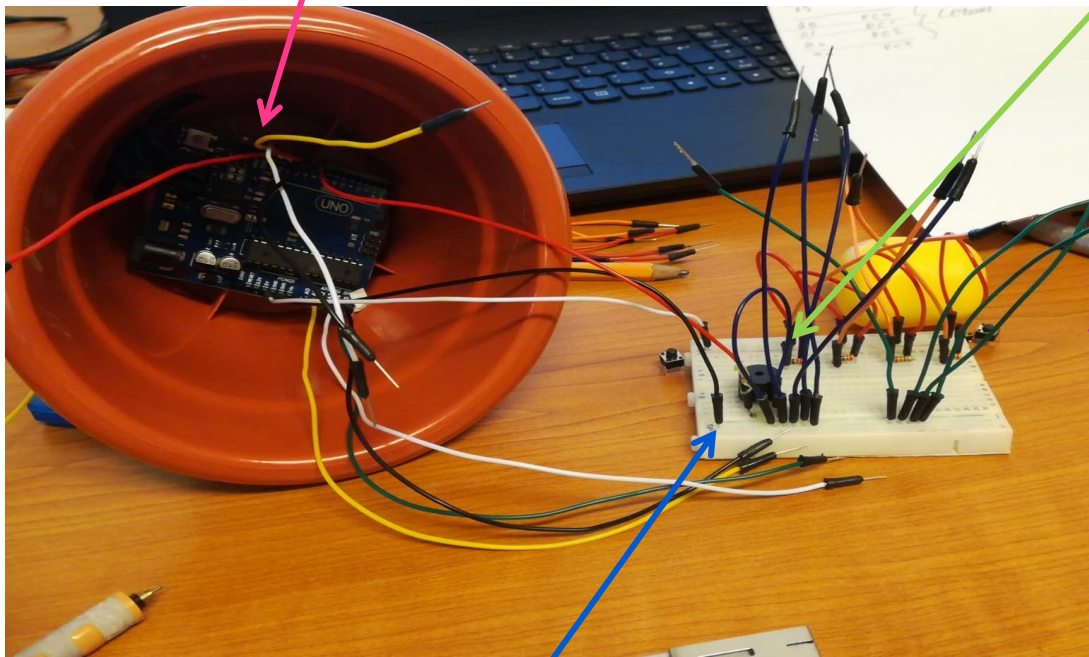
- o placă de Arduino (ATmega328P);
- un buzzer și un potențiometrul;
- un buton;
- 5 LED-uri a câte 4 culori (roșu, galben, verde, albastru);
- un breadboard, rezistențe și fire.

## Proiectarea ghiveciului

Pentru a proiecta arborele conform descrierii, pe fundul ghiveciului vom lipi placa de Arduino și vom găuri ghiveciul pe lateral pentru a-l putea alimenta și 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 potențiometrul.

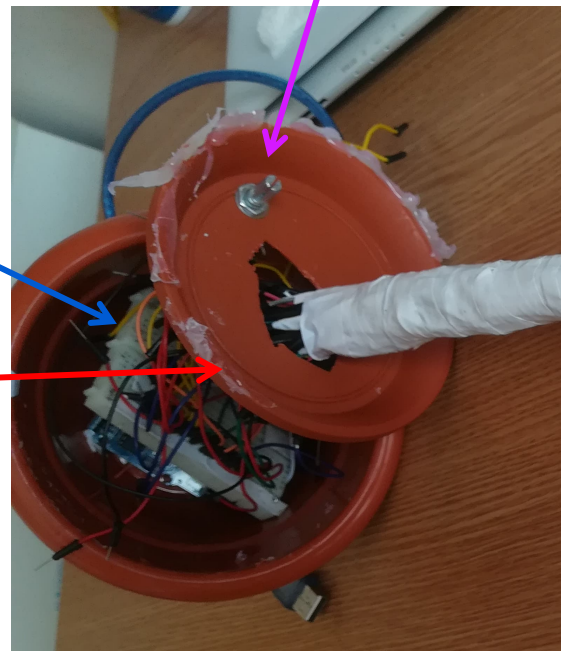
Placa de Arduino

Buzzer



Potentiometru

Breadboard



Capac



Buton

Alimentare

# Proiectarea buzzer-ului

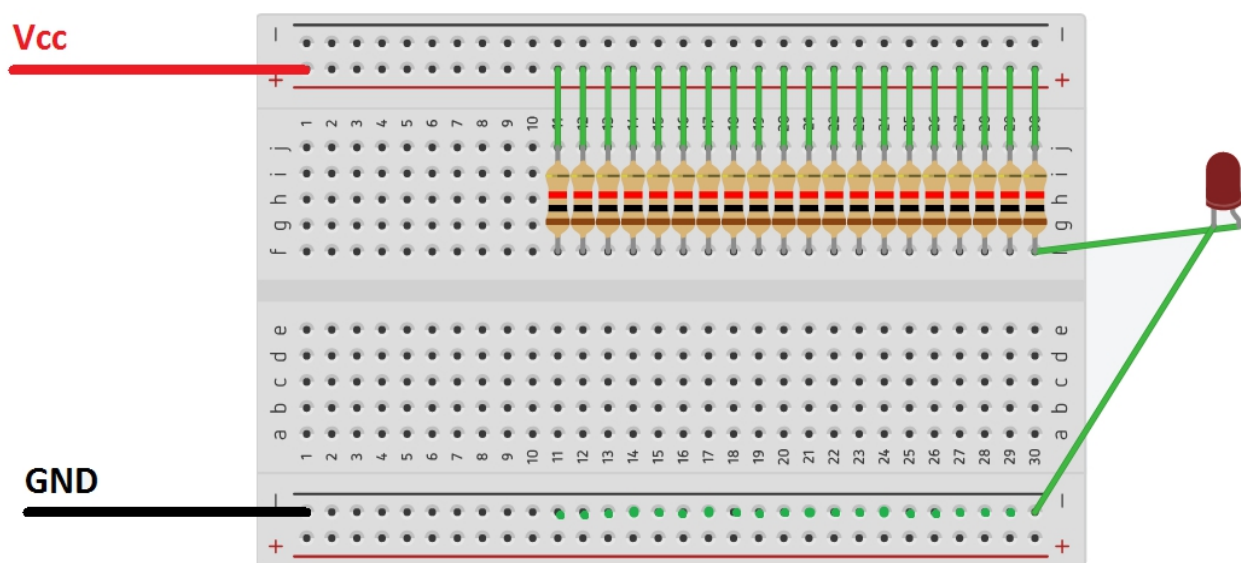
Implementarea melodiilor se va realiza cu ajutorul unui buzzer pentru redarea notelor și a unui potențiometrul pentru reglarea volumului. Buzzer-ul va fi amplasat pe breadboard, iar potențiometrul pe capacul ghiveciului. Un pin de la buzzer va fi conectat la GND, iar celălalt la potențiometrul. Ceilalți doi pini ai potențiometrului vor fi conectați 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 din acești pini vor fi conectați 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

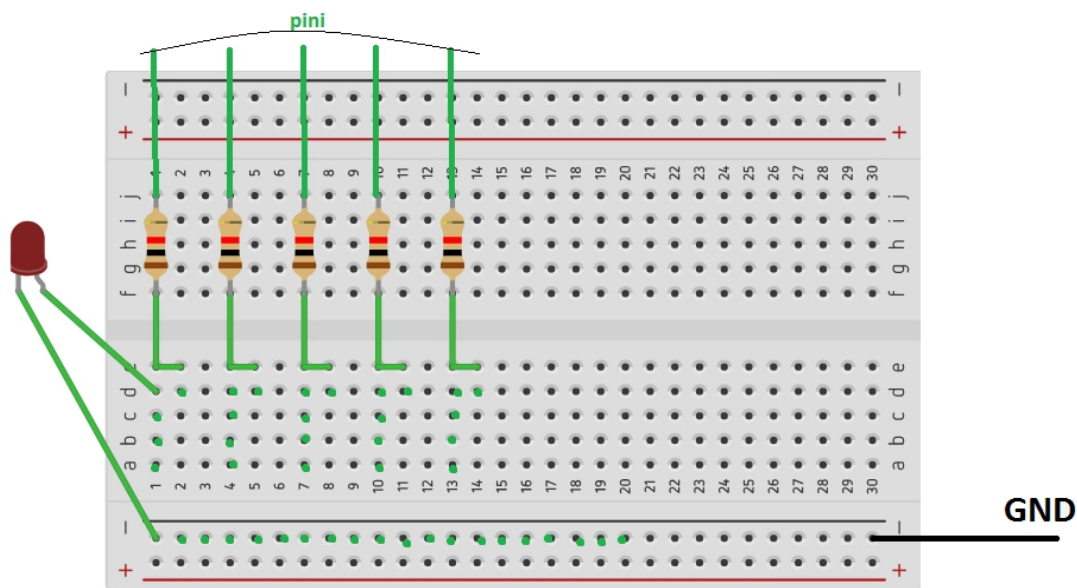
## Varianta 1: Legarea în paralel la Vcc



Presupunând că de fiecare rezistență este legat un LED, precum la ultima rezistență din poza de mai sus. Tensiunea pe ansamblul rezistență-LED înseriate va fi egală cu  $V_{cc}$  (5V), tensiunile pe LED-uri variind în funcție de rezistențe.

Această variantă va ține 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 ieși semnalele de comandă pentru un ansamblu de 5 rezistențe înseriate cu 5 LED-uri precum în imaginea de mai sus, fiecare ansamblu din cele 4 reprezentând o culoare(roșu, galben, verde sau albastru). Tensiunea pe ansamblul rezistență-LED înseriate va fi egală cu tensiunea de pe pin-ul de comandă când e ON (5V), tensiunile LED-urilor variind în funcție de rezistențe.

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

tență-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 și o rezistență, rămânând un LED pe dinafară, legat în serie cu o rezistență, având astfel un curent maxim pe fiecare ansamblu. Excepție fac LED-urile albastre care, fiind cele mai consumatoare, vor fi legate în serie cu o rezistență, fiecare cu pinul ei de comandă. Folosind legea lui Ohm și știind de câți volți avem nevoie pentru a aprinde fiecare LED în funcție de culoarea sa, putem afla valorile rezistențelor noastre.

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

Această variantă rămâne și varianta finală.



# Alegerea pinilor

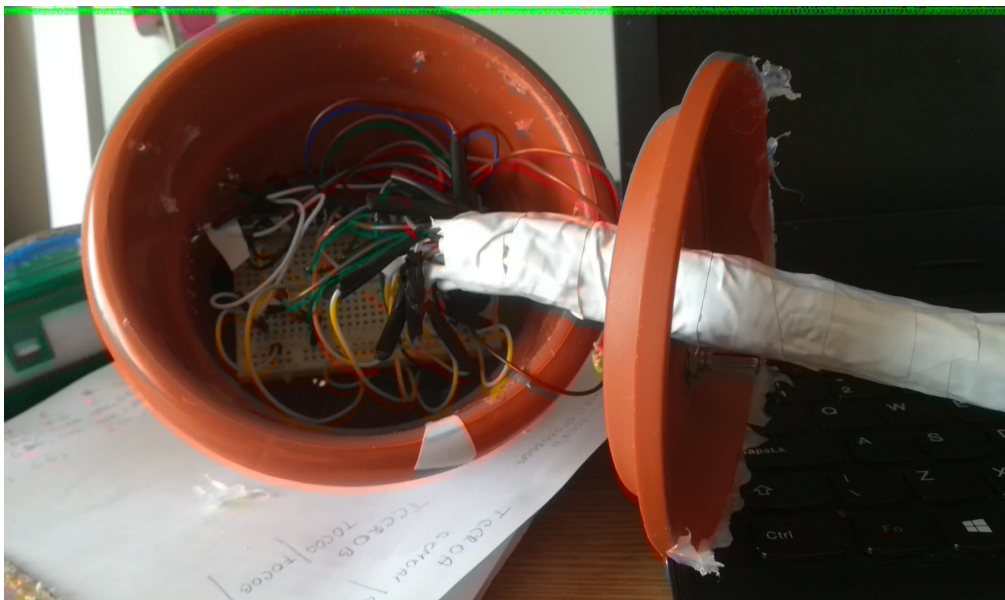
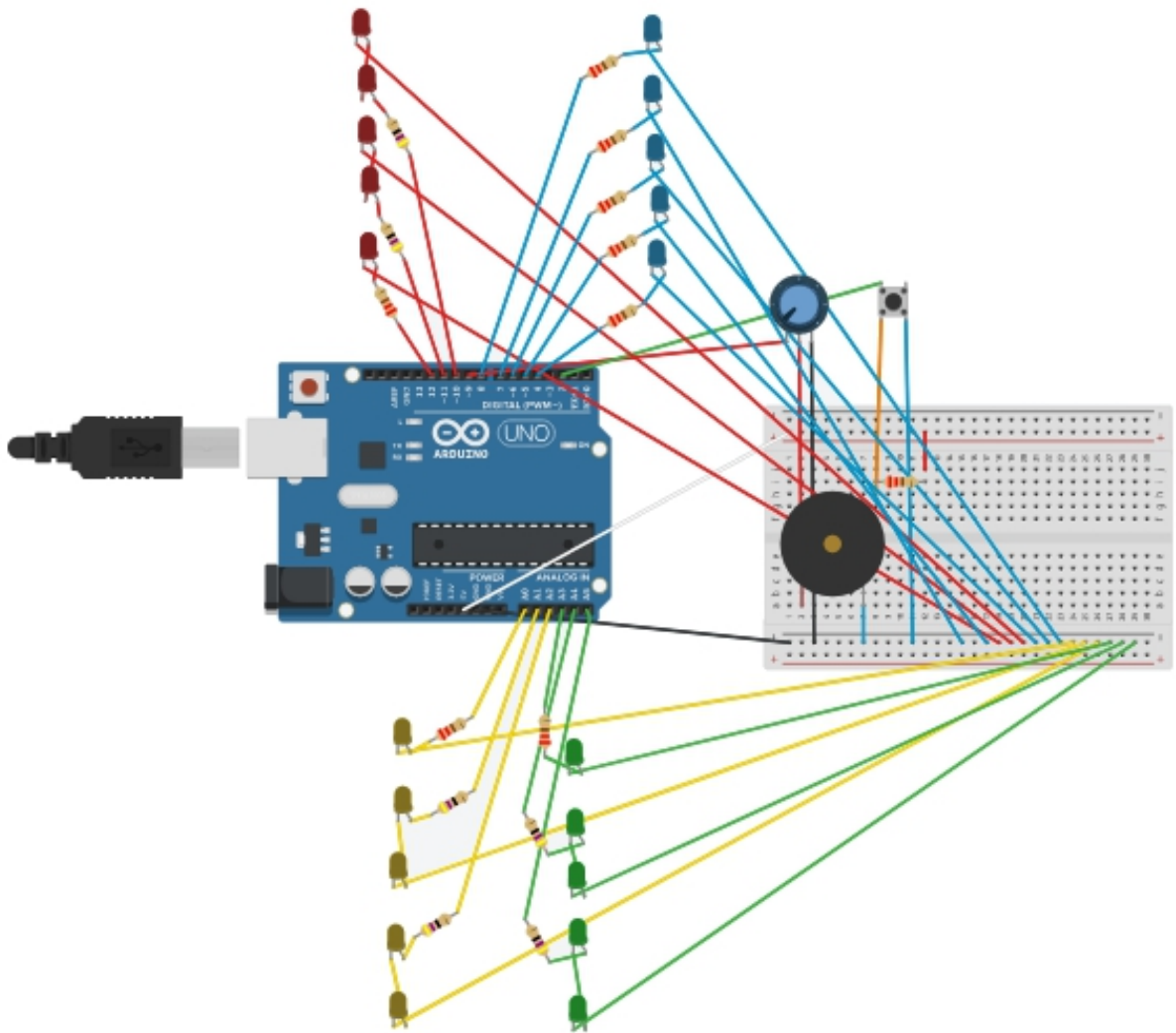
Arduino function		Arduino function
reset	(PCINT14/RESET) PC6 1	PC5 (ADC5/SCL/PCINT13) analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0 2	PC4 (ADC4/SDA/PCINT12) analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1 3	PC3 (ADC3/PCINT11) analog input 3
digital pin 2	(PCINT18/INT0) PD2 4	PC2 (ADC2/PCINT10) analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3 5	PC1 (ADC1/PCINT9) analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4 6	PC0 (ADC0/PCINT8) analog input 0
VCC	VCC 7	GND GND
GND	GND 8	AREF analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6 9	AVCC VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7 10	PB5 (SCK/PCINT5) digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5 11	PB4 (MISO/PCINT4) digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6 12	PB3 (MOSI/OC2A/PCINT3) digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7 13	PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0 14	PB1 (OC1A/PCINT1) digital pin 9 (PWM)

Digital Pins 11, 12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17, 18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Pinii colorați în albastru, roșu, galben și verde reprezintă pinii aleși pentru controlul LED-urilor, iar cei cu mov reprezintă restul pinilor aleși, astfel:

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

# Schema fizică



# Codul

## Inițializare Timer0:

```
void timer0_init() {  
    SREG = 1<<7; // Activează întreruperile globale  
  
    TCCR0A = 0b00000010; //Port normal de operare  
                        //OC0A și OC0B deconectați  
    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 ieșire  
}  
                        //pentru OCR0A
```

## Inițializare 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 ieșire pentru
                          //semnalul PWM al buzzerului

    TCCR1A = 0b01000000; // timerului1: CTC cu OC1A,
    TCCR1B = 0b00001010; //prescaler de 8, pinul OC1A
    TCCR1C = 0b00000000; //setat ca ieșire pentru
                          //semnalul PWM
}
```

## Inițializare buton:

```
void button_int(){
    DDRD &= ~ 0b000000100; //PD2 setat ca intrare
}
```

## Inițializare LED-uri

```
void LEDs_init(){
    DDRC |= 0b000000111; //ieșire galben
    DDRC |= 0b00111000; //ieșire verde
    DDRB |= 0b00111101; //ieșire roșu
    DDRD |= 0b11100000; //ieșire 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 conțin notele în ordine și cei ce conțin timpul necesar fiecărei note în funcție de tempoul melodiei;
4. Milisecundele(long int ms = 0;), contorul pentru vectorul de melodii(int i =0;) și variabila TOP folosită la frecvență(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
    oprește și 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 OCR0 este valoarea scrisă în acest registru.

$$\begin{aligned} TOP &= 1000000 / \text{frecventa} \\ OCR1A &= (TOP + 1) / 2 \end{aligned}$$

Unde frecvența înseamnă frecvența pe care dorim să o implementăm.