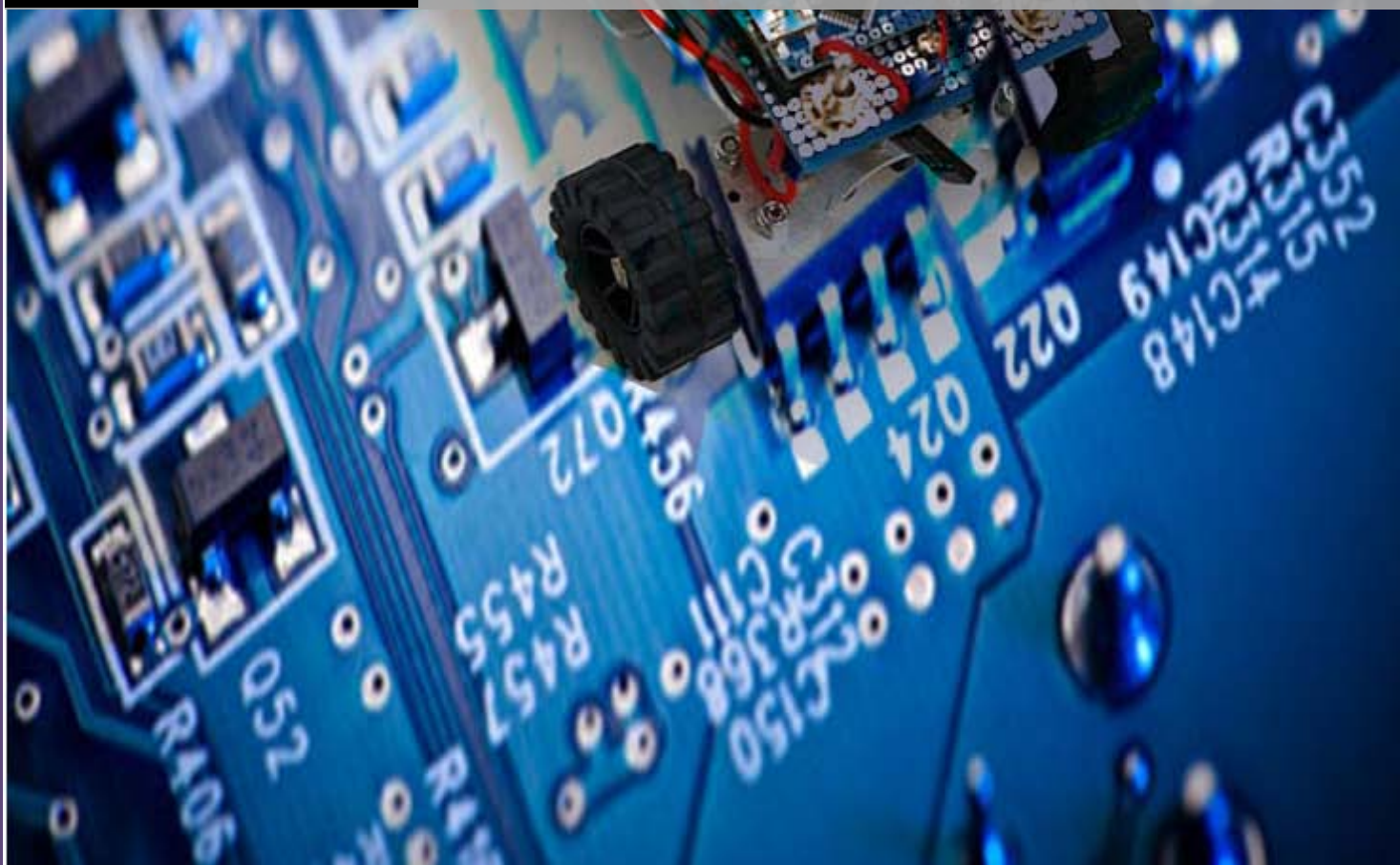


**mastINVENT**

GHIDUL ELECTRONISTULUI INCEPATOR



## Cuprins

<b>REZISTORUL</b>	<b>7</b>
<b>CODUL CULORILOR PENTRU REZISTOARE</b>	<b>9</b>
<b>IDENTIFICAREA VALORII REZISTOARELOR INSTALATE PE PCB</b>	<b>11</b>
<b>VALORI STANDARDIZATE</b>	<b>12</b>
<b>REZISTOARE SMD</b>	<b>12</b>
<b>MARCAREA REZISTOARELOR SMD</b>	<b>13</b>
MARCARE CU TREI SIMBOLURI ALFA-NUMERICE	13
MARCARE CU PATRU SIMBOLURI ALFA-NUMERICE	13
MARCARE CONFORM STANDARDULUI EIA 96	14
<b>POTENTIOMETRE. REZISTOARE SEMIREGLABILE.</b>	<b>15</b>
<b>REZISTOARE SEMIREGLABILE</b>	<b>15</b>
<b>MARCAREA POTENTIOMETRELOR SI A REZISTOARELOR SEMIREGLABILE</b>	<b>17</b>
<b>DIODA SEMICONDUCTOARE</b>	<b>19</b>
<b>TIPURI DE DIODE</b>	<b>22</b>
DIODE REDRESOARE	22
DIODE DE COMUTATIE	22
DIODE SCHOTTKY	23
DIODA VARICAP	23
DIODA ZENER	23
<b>L.E.D. – DIODA ELECTROLUMINISCENTA</b>	<b>24</b>
<b>TESTAREA UNUI REZISTOR</b>	<b>26</b>
<b>TESTARE REZISTORULUI UTILIZAND OHMMETRUL</b>	<b>26</b>
<b>TESTAREA UNEI DIODE</b>	<b>27</b>
<b>TESTARE CU AJUTORUL OHMMETRULUI</b>	<b>27</b>
TESTAREA REZISTENTEI ANOD-CATOD A DIODEI	27
TESTAREA REZISTENTEI CATOD-ANOD A DIODEI	28
DIODA DESCHISA	29
DIODA IN SCURT-CIRCUIT	29
<b>TESTAREA DIODEI CU AJUTORUL VOLTMETRULUI</b>	<b>29</b>
DIODA DESCHISA	30
DIODA IN SCURT-CIRCUIT	30
<b>TESTAREA DIODEI UTILIZAND CIRCUITUL SPECIALIZAT DE TESTARE AL MULTIMETRULUI</b>	<b>30</b>

<b>PRIMA UTILIZARE A DISPOZITIVULUI ARDUINO</b>	<b>33</b>
INTRODUCERE	33
CABLUL USB	34
SISTEMUL SOFTWARE ARDUINO IDE	34
PROCEDURA DE INSTALARE A DRIVERELOR PENTRU ARDUINO PENTRU WINDOWS XP SI 7	35
<b>PRIMA APLICATIE CU ARDUINO – APRINDEREA INTERMITENTA A UNUI LED</b>	<b>39</b>
CIRCUITUL ASOCIAT PRIMULUI EXPERIMENT	46
SCHEMA ELECTRICA	47
CODUL SURSA	47
<b>EXPERIMENTUL 2. CONTROLUL INTENSITATII LUMINOASE A UNUI LED.</b>	<b>50</b>
CIRCUITUL ASOCIAT	50
SCHEMA ELECTRICA	51
CODUL SURSA	52
<b>EXPERIMENTUL 3. UTILIZAREA LED-ULUI RGB</b>	<b>53</b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	53
CULORI	55
CODUL SURSA	56
UTILIZAREA CODURILOR DE CULOARE	58
PWM – MODULATIA IN FRECVENTA A IMPULSURILOR	59
<b>EXPERIMENTUL 4. UTILIZAREA INTRARILOR ANALOGICE. CONECTAREA UNUI POTENTIOMETRU LA DISPOZITIVUL ARDUINO.</b>	<b>60</b>
CIRCUITUL ASOCIAT	60
SCHEMA ELECTRICA	62
CODUL SURSA	62
<b>EXPERIMENTUL 5. UTILIZAREA UNUI POTENTIOMETRU ROTATIV LINIAR PENTRU A CONTROLA INTENSITATEA LUMINOASA.</b>	<b>64</b>
CIRCUITUL ASOCIAT	64
SCHEMA ELECTRICA	65
CODUL SURSA	65

<b><u>EXPERIMENTUL 6. UTILIZAREA COMUTATOARELOR CU REVENIRE.</u></b>	<b>68</b>
CIRCUITUL ASOCIAT	68
SCHEMA ELECTRICA	69
CODUL SURSA	70
<b><u>EXPERIMENTUL 7. DETECTAREA SCHIMBARII STarii COMUTATORULUI CU REVENIRE.</u></b>	<b>72</b>
CIRCUITUL ASOCIAT	72
CODUL SURSA	73
<b><u>EXPERIMENTUL 8. UTILIZAREA A DOUA COMUTATOARE CU REVENIRE PENTRU A CONTROLA ILUMINAREA UNUI LED.</u></b>	<b>75</b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	76
CODUL SURSA	77
<b><u>EXPERIMENTUL 9. CALIBRAREA. UTILIZAREA UNEI FOTOCELULE PENTRU A CONTROLA INTENSITATEA LUMINII EMISE DE UN LED.</u></b>	<b>79</b>
CIRCUIT ASOCIAT	79
SCHEMA ELECTRICA	80
CODUL SURSA	81
<b><u>EXPERIMENTUL 10. UTILIZAREA UNUI POTENTIOMETRU ROTATIV LINIAR PENTRU A CONTROLA O GRUPARE DE LED-URI.</u></b>	<b>84</b>
REZISTOARE VARIABLE. POTENTIOMETRE.	86
SCHEMA DE MONTARE PE PLACA BREADBOARD	87
CODUL SURSA	88
<b><u>EXPERIMENTUL 11. SESIZAREA INTESITATII LUMINII. UTILIZAREA UNEI FOTOCELULE PENTRU A CONTROLA O GRUPARE DE LED-URI.</u></b>	<b>90</b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	90
FOTOCELULA	91
CODUL SURSA	92
<b><u>FIRE DE LEGATURA</u></b>	<b>94</b>

<b>EXPERIMENTUL 12. UTILIZAREA UNUI POTENTIOMETRU DIGITAL.</b>	<b>95</b>
SCHEMA ELECTRICA	96
CODUL SURSA	96
<b>EXPERIMENTUL 13. UTILIZAREA UNUI POTENTIOMETRU DIGITAL PENTRU A VARIA INTENSITATEA LUMINOASA A UNUI LED.</b>	<b>99</b>
SCHEMA ELECTRICA	99
CODUL SURSA	100
<b>EXPERIMENTUL 14. EMITEREA SUNETELOR UTILIZAND ARDUINO SI UN DIFUZOR. CONSTRUCTIA UNUI PSEUDO-THEREMIN.</b>	<b>103</b>
GENERAREA UNEI SECVENTE DE NOTE MUZICALE	103
PSEUDO-THEREMIN	105
CODUL SURSA	106
MODIFICARI POSIBILE	107
<b>EXPERIMENTUL 15. UTILIZAREA UNUI BARGRAF CU LED-URI.</b>	<b>108</b>
CIRCUITUL ASOCIAT	108
SCHEMA ELECTRICA	109
CODUL SURSA	110
<b>EXPERIMENTUL 16. BARGRAF CONTROLAT PRIN INTERMEDIUL DISPOZITIVULUI ARDUINO SI A UNUI CONTOR CD4017</b>	<b>112</b>
CIRCUITUL INTEGRAT CD4017	112
SCHEMA DE MONTARE PE PLACA BREADBOARD	113
CODUL SURSA	114
<b>EXPERIMENTUL 17. BARGRAF CONTROLAT PRIN INTERMEDIUL DISPOZITIVULUI ARDUINO, CONTORULUI CD4017 SI A UNUI POTENTIOMETRU LINIAR</b>	<b>115</b>
CODUL SURSA	116

<b><u>EXPERIMENTUL 18. MULTIPLEXAREA A OPT LED-URI UTILIZAND REGISTRUL DE DEPLASARE 74HC595</u></b>	<b><u>118</u></b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	118
REGISTRUL DE DEPLASARE 74HC595	119
CODUL SURSA	120
CONTROLUL INTENSITATII LUMINOASE	123
<b><u>AFISAJE CU 7 SEGMENTE</u></b>	<b><u>125</u></b>
<b><u>EXPERIMENT 19. RULAREA CIFRELOR DE LA 0 LA 9 PE UN AFISAJ CU 7 SEGMENTE UTILIZAND DISPOZITIVUL ARDUINO</u></b>	<b><u>127</u></b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	127
CODUL SURSA	128
<b><u>EXPERIMENT 20. CONSTRUIREA UNUI ZAR ELECTRONIC UTILIZAND AFISAJ COMPUS DIN DOI DIGITI CU 7 SEGMENTE</u></b>	<b><u>130</u></b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	130
CODUL SURSA	132
<b><u>EXPERIMENT 21. AFISAREA UNUI NUMAR PE UN AFISAJ CU 4 DIGITI FORMATI DIN 7 SEGMENTE</u></b>	<b><u>135</u></b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	136
CODUL SURSA	137
<b><u>MATRICE CU LED-URI</u></b>	<b><u>140</u></b>
STRUCTURA UNEI MATRICE CU LED-URI	140
GENERAREA UNOR SIMBOLURI CU AJUTORUL ARDUINO SI A MATRICEI CU LED-URI	141
<b><u>EXPERIMENT 22. REPREZENTAREA UNOR SIMBOLURI PE MATRICEA CU LED-URI</u></b>	<b><u>144</u></b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	144
CODUL SURSA	145
<b><u>EXPERIMENTUL 23. REPREZENTAREA LITERELOR DIN ALFABET PE MATRICEA CU LED-URI</u></b>	<b><u>151</u></b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	151
CODUL SURSA	151

<b><u>EXPERIMENTUL 24. RULAREA UNUI TEXT PE MATRICEA CU LED-URI</u></b>	<b>157</b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	157
CODUL SURSA	157
<b><u>EXPERIMENTUL 25. PROIECTAREA UNUI DISPOZITIVUL DE INREGISTRARE A TEMPERATURII, UTILIZAND UN TERMISTOR SI DISPOZITIVUL ARDUINO.</u></b>	<b>162</b>
SCHEMA ELECTRICA	164
CODUL SURSA	165
<b><u>EXPERIMENTUL 26. PROIECTAREA UNUI TEMPORIZATOR ELECTRONIC UTILIZAND UN AFISAJ CU PATRU DIGITI SI DISPOZITIVUL ARDUINO</u></b>	<b>172</b>
SCHEMA ELECTRICA	172
CODUL SURSA	173
<b><u>EXPERIMENTUL 27. TEMPORIZATOR ELECTRONIC CU POTENTIOMETRU DIGITAL SI BUZZER</u></b>	<b>179</b>
SCHEMA ELECTRICA	179
CODUL SURSA	181
<b><u>EXPERIMENTUL 28. MOTORUL IN CURENT CONTINUU</u></b>	<b>187</b>
SCHEMA DE MONTARE PE PLACA BREADBOARD	187
CODUL SURSA	188
TRANZISTOARE	189
SCHEMA ELECTRICA	191



## Rezistorul

**Rezistorul electric** este un element de circuit pasiv care se opune trecerii curentului electric atunci cand la bornele sale se aplica o tensiune electrica. El absoarbe pe la borne putere electrica activa pe care o transforma in caldura prin efectul electrocaloric.

Caracterizarea globala a unui rezistor se face cu ajutorul parametrului fizic numit **rezistenta electrica**, notat cu  $R$ .

Unitatea de masura, in sistem international, a rezistentei electrice se numeste *Ohm* si se noteaza cu simbolul  $\Omega$ .

**Rezistorul liniar** are rezistenta electrica independenta de valoarea intensitatii curentului electric ce il strabate.

Ecuatia tensiune-curent are, in cazul rezistorului liniar, expresia:

$$u = R * i$$

Caracteristica tensiune-curent este *liniara* si *bilaterală*; liniara, in sensul ca aceasta caracteristica este o linie dreapta ce trece prin origine. O consecinta importanta a liniaritatii este faptul ca tensiunea este intotdeauna proportionala cu curentul si viceversa. Bilateral inseamna ca aceasta caracteristica are simetrie para fata de origine:  $u(-i) = -u(i)$ . Datorita proprietatii de bilateralitate, schimbarea polaritatii tensiunii aplicate schimba sensul curentului, dar nu sivaloarea sa.

Rezultatul este ca **orice rezistor se poate conecta in circuit fara a tine seama de semnificatia in notarea bornelor sale**.

**Rezistorul neliniar** are rezistenta electrica dependenta de curentul care il parcurge.

Un alt criteriu de clasificare imparte rezistoarele in:

- rezistoare variabile;
- rezistoare invariabile.

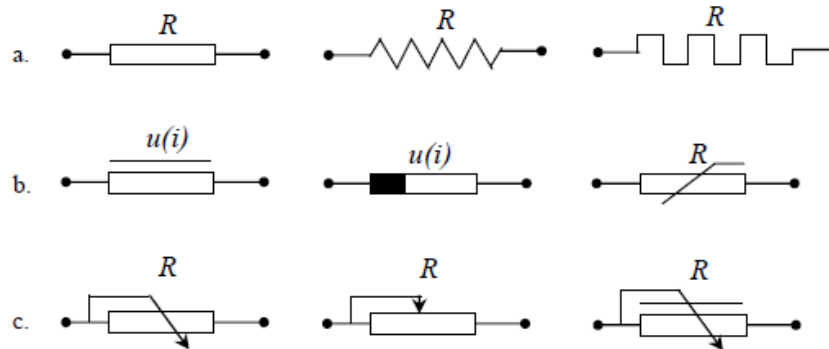
Spre deosebire de rezistoarele **invariabile**, a caror valoare a rezistentei nu poate fi modificata de catre utilizator, rezistoarele **variabile** permit utilizatorului sa modifice valoarea rezistentei lor, prin deplasarea unui cursor (rezistoarele numindu-se in acest caz *potentiometre* si *reostate* – in functie de modalitatea de conectare) sau prin realizarea unei comutatii, ce realizeaza schimbarea conexiunilor unor grupuri de rezistoare (la rezistoarele cu variatie *in decade*).

**Puterea electrica** asociata unui rezistor poate fi dedusa din formula generala a puterii electrice:

$$p = u * i$$



Simbolurile grafice uzuale ale rezistoarelor sunt prezentate in figura de mai jos: pentru rezistoarele liniare sunt date in figura a, rezistoarele neliniare au simbolurile din figura b., iar simbolurile pentru rezistoarele variabile sunt date in figura c.



Din punct de vedere constructiv se pot identifica trei tipuri principale de rezistoare: **rezistorul bobinat** - consta dintr-un fir conductor lung infasurat in jurul unui cilindru izolator; **rezistorul tip pelicula superficiala**, denumite *film resistor*, - fabricat prin depunerea, in mediu vidat, a unui strat subtire de metal pe un substrat izolator; **rezistorul din amestec de carbon** - alcătuit dintr-un element rezistiv cilindric in care este inclus un fir ale carui capete sunt accesibile sau care are borne terminale metalice de care sunt atasate firele de legatura.

O alta categorie de rezistoare sunt rezistoarele variabile sau *potentiometrele*. Acestea sunt dispozitive cu trei borne de acces; doua borne sunt conectate la capetele unui element rezistiv, iar a treia borna este un contact mobil cu elementul rezistiv intr-un punct intermediar. Cele mai utilizate tipuri de potentiometre sunt prezentate in figura 2 (a. - potentiometru rotativ, b.- potentiometru liniar).

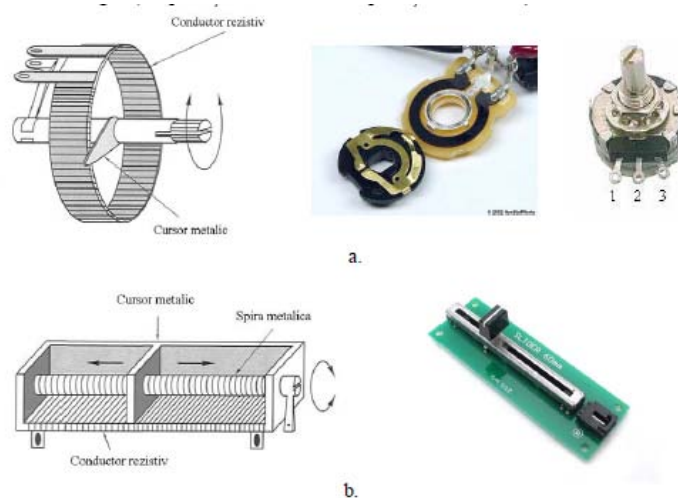


Figura 2. Tipuri de potentiometre

### Codul culorilor pentru rezistoare

Datorita automatizarii industriei si a usurintei de citire a valorii rezistorului, majoritatea rezistoarelor sunt marcate prin intermediul unui cod al culorilor. Astfel nu se mai intampla ca marcajul in clar al unui rezistor lipit pe cablaj sa ajunga catre placa si sa nu mai poata fi citit.

Majoritatea rezistoarelor sunt marcate prin intermediul a patru benzi colorate; exista insa si rezistoare marcate cu cinci sau sase benzi colorate. Benzile colorate trebuie citite dinspre capat spre centru, prima banda este foarte aproape de margine.

Banda 1: Prima cifra din valoarea rezistentei.

Banda 2: A doua cifra din valoarea rezistentei.



Banda 3: Multiplicator, putere a lui 10

Banda 4: Existenta sau chiar absenta benzii 4 marcheaza toleranta rezistorului, adica procentajul de abatere posibila de la valoarea inscrisa prin primele 3 benzi.

In tabel este prezentata corespondenta intre culorile marcate pe rezistor si valorile pe care acestea le reprezinta.

Culoare	banda 1	banda 2	banda 3	banda 4
Negru	0	0	x 1	
Maro	1	1	x 10	
Rosu	2	2	x 100	
Portocaliu	3	3	x 1.000	
Galben	4	4	x 10.000	
Verde	5	5	x 100.000	
Albastru	6	6	x 10 <sup>6</sup>	
Violet	7	7	x 10 <sup>7</sup>	
Gri	8	8	x 10 <sup>8</sup>	
Alb	9	9	x 10 <sup>9</sup>	
Auriu			x 0,1	5%
Argintiu			x 0,01	10%
fara culoare				20%

În figura de mai jos este prezentat un exemplu de calcul al valorii rezistenței:

	banda 1 [ gri ] -> 8 banda 2 [rosu] -> 2 banda 3 [negru] -> * 1 => <b>valoarea</b> 82 * 1 = 82 $\Omega$ banda 4 [auriu] -> <b>toleranta</b> 5%
	banda 1 [portocaliu] -> 3 banda 2 [portocaliu] -> 3 banda 3 [galben] -> x10,000 => <b>valoarea</b> 33x10,000 =330 k $\Omega$ banda 4 [maro] -> <b>toleranta</b> 5%

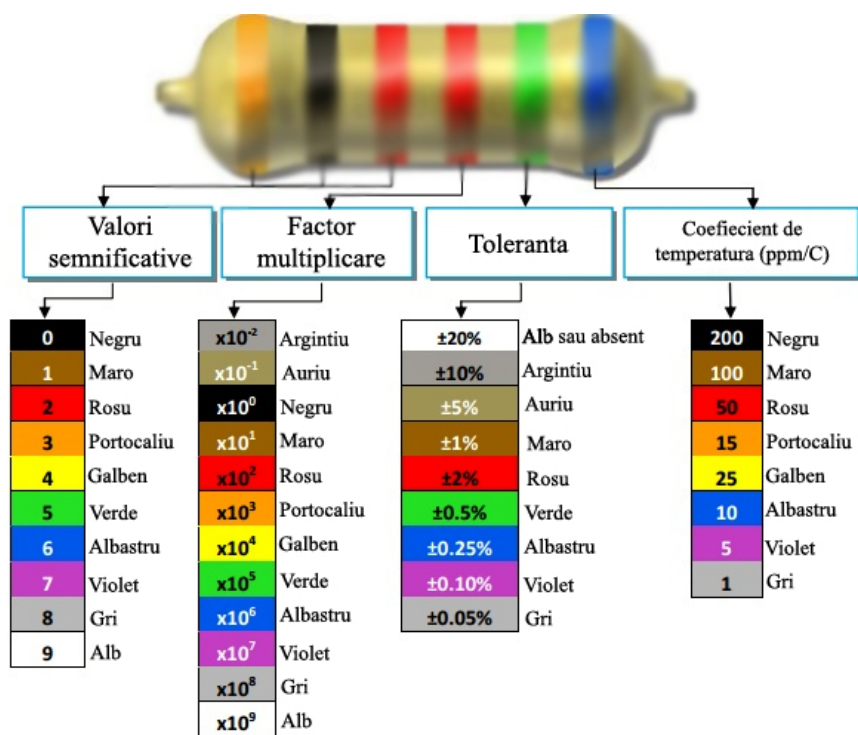


Figura 4. Semnificatia codului culorilor pentru rezistori cu patru, cinci sau sase benzi

### Identificarea valorii rezistoarelor instalate pe PCB

Rezistoarele utilizate pe placile electronice, au, in general, valori ale rezistentei cuprinse intre 1 Ohm si cateva milioane de Ohmi (M $\Omega$ ). De cele mai multe ori, intrucat este mai dificil de utilizat pe un computer, simbolul  $\Omega$  nu este suprimat. Prescurtarea, prin suprimarea simbolului  $\Omega$ , elimina necesitatea punctelor zecimale si riscul de a omite virgula atunci cand sunt copiate documentele. Aceasta functioneaza prin inlocuirea punctului zecimal cu prefixul rezistentei (de exemplu K, in cazul valorilor mai mari de un KiloOhm). Apar astfel urmatoarele notatii prescurtate:

Notatia standard	Notatia prescurtata
100m $\Omega$ = 0,1 $\Omega$	0R1
1 $\Omega$	1R
1,2 $\Omega$	1R2
1 K $\Omega$	1K
1,2 K $\Omega$	1K2
2,5 K $\Omega$	2K5
1,4M $\Omega$	1M4

Tabel 2. Notatie prescurtata a valorii rezistentei

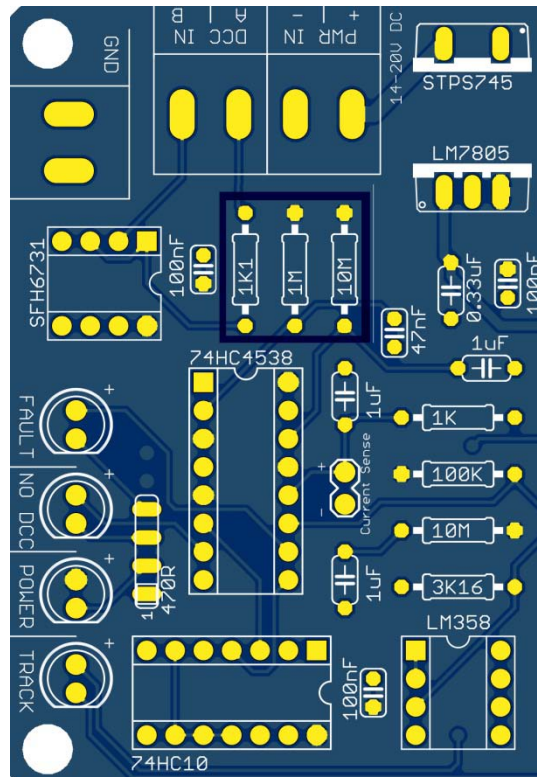


Figura 5. Reprezentarea valorii rezistentei pe PCB.

### Valori standardizate

Pentru a evita anumite situatii, valorile rezistoarelor sunt standardizate si respecta un anumit algoritm. Acestea sunt multipli de zece si fiecare rezistenta standardizata poate fi calculata in functie de cea precedenta conform formulei:

$$R_n = R_{n-1} * \sqrt[12]{10}$$

Astfel, valorile standard sunt: 1,0; 1,2; 1,5; 1,8; 2,2; 2,7; 3,3; 3,9; 4,7; 5,6; 6,8 8,2 si multiplii de zece ai acestora.

Completand lista, valorile standard ale rezistoarelor cuprinse in intervalul  $1\Omega \sim 1k\Omega$  sunt:

1R 1R2 1R5 1R8 2R2 2R7 3R3 3R9 4R7 5R6 6R8 8R2 10R 12R 15R  
18R 22R 27R 33R 39R 47R 56R 68R 82R 100R 120R 150R 180R 220R  
270R 330R 390R 470R 560R 680R 820R

### Rezistoare SMD

Rezistoarele SMD au, in general o sectiune dreptunghiulara si prezinta suprafete metalice la capete. Suprafetele metalice sunt utilizate pentru fixarea, prin cositorire, pe PCB. Rezistorul SMD este fabricat prin depunerea unui strat subtire de oxid metalic pe un substrat ceramic. Grosimea si lungimea substratului de oxid metalic determina valoarea rezistentei electrice. Spre deosebire de rezistoarele obisnuite, cele SMD nu sunt imprimate cu benzi colorate. Pentru a determina valoarea rezistentei, acestea sunt marcate cu un cod numeric sau alfa-numeric.



Figura 6. Rezistor SMD

Rezistoarele SMD sunt disponibile in diverse incapsulari, printre care se numara: 2512, 2010, 1812, 1210, 1206, 0805, 0603, 0402, etc. Acestea reprezinta dimensiunile in zecimi de milimetru, adica 0603 inseamna o dimensiune a rezistorului de 0,6 x 0,3mm.

### **Marcarea rezistoarelor SMD**

Exista trei sisteme, larg raspandite, de marcare a rezistoarelor SMD.



Figura 7. Rezistor SMD, 4.7k $\Omega$

### **Marcare cu trei simboluri alfa-numerice**

În cazul acestui sistem de marcare, prima și a doua valoare reprezintă cifrele semnificative, iar a treia, ordinul de mărime (putere a lui 10). De exemplu, în cazul rezistorului prezentat în figura 6, marcat cu simbolul 472, valoarea este  $4,7k = 47 \times 10^2 \Omega$ . Astfel, rezistorul marcat cu simbolurile 100 nu are rezistența de  $100\Omega$  ci  $10 \times 10^0 = 10 \times 1 = 10\Omega$ . În cazul rezistentelor mai mici de  $10\Omega$ , este utilizată litera R pentru a înlocui punctul zecimal. De exemplu, simbolul 5R6 reprezintă o rezistență de  $5,6\Omega$ .

#### **Exemple de rezistoare marcate cu un cod compus din 3 simboluri:**

$$220 = 22 \times 10^0 (1) = 22\Omega$$

$$471 = 47 \times 10^1 (10) = 470\Omega$$

$$102 = 10 \times 10^2 (100) = 1000\Omega \text{ sau } 1k\Omega$$

$$3R3 = 3,3\Omega$$

### **Marcare cu patru simboluri alfa-numerice**

Acest sistem este utilizat pentru rezistoare cu o precizie mai mare. În cazul acestui sistem de marcare, prima, a doua și a treia valoare reprezintă cifrele semnificative, iar a patra, ordinul de mărime (putere a lui 10). De exemplu, în cazul rezistorului unui rezistor marcat cu simbolul 4702, valoarea este  $47k = 470 \times 10^2 \Omega$ . În cazul rezistentelor mai mici de  $100\Omega$ , este utilizată litera R pentru a înlocui punctul zecimal.

**Exemple de rezistoare marcate cu un cod compus din 4 simboluri:**

$$4700 = 470 \times 10^0 (1) = 470\Omega$$

$$2001 = 200 \times 10^1 (10) = 2000\Omega \text{ or } 2k\Omega$$

$$1002 = 100 \times 10^2 (100) = 10000\Omega \text{ sau } 10K\Omega$$

$$15R0 = 15,0\Omega$$

**Marcare conform standardului EIA 96**

**Standardul EIA 96** este utilizat pentru marcarea rezistoarelor cu o toleranta de 1%. Acesta utilizeaza un cod format din trei caractere. Primul si al doilea digit indicat valoarea rezistorului si al treilea caracter indica factorul de multiplicare.

**Alte simboluri utilizate pentru marcarea rezistoarelor SMD**

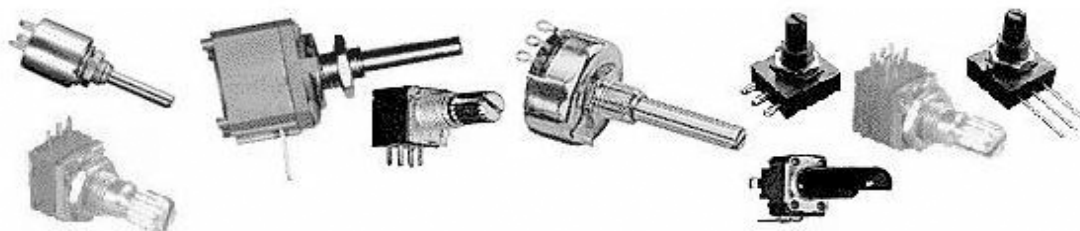
1. Rezistorul SMD marcat cu **0, 00, 000** sau **0000** este denumit jumper sau conexiune cu valoare 0 a rezistentei.
2. Rezistorul SMD marcat cu un cod din trei cifre si o bara scurta sub una dintre acestea prezinta o toleranta de 1%. De exemplu, daca un rezistor este marcat **122** acesta are valoarea 1,2k $\Omega$  si toleranta 1%.
3. Codul marcat pe rezistoarele SMD de ordinul miliohmilor, contine simbolul M sau m, care delimiteaza punctul zecimal. Astfel, valoarea rezistorului marcat **1M50** este 1,50m $\Omega$ .
4. Rezistoarele SMD de ordinul miliohmilor pot fi marcate si printr-o bara care cuprinde intregul cod. Acestea pot fi marcate fie **1m5** = 1.5m $\Omega$ , **R001** = 1m $\Omega$ , sau **101** = 0,101 $\Omega$ , **047** = 0,047 $\Omega$ . Bara este utilizata doar atunci cand litera R a trebuit omisa din cauza lipsei de spatiu pe suprafata rezistorului.



## Potentiometre. Rezistoare semireglabile.

Potentiometrul este cel mai simplu model de traductor electro-mecanic (converteste miscarea rotativa sau liniara intr-un rezistenta variabila). Aceasta variatie a rezistentei electrice poate fi utilizata pentru o multime de aplicatii: de la a controla volumul unui sistem audio pana la a stabili directia unui utilaj industrial.

In figura 8 sunt prezentate o parte din modelele de potentiometre mai raspandite, montabile pe PCB sau pe panou.



*Figura 8. Modele de potentiometre*

Dintre toate aceste modele, probabil cel mai raspandit este cel din centrul imaginii, un potentiometru montabil pe panou, de 25mm, cu diametrul orificiului de montare de 10mm si axul cu o lungime de 6,35mm. Majoritatea potentiometrelor rotative au o rotatie de 270 de grade de la o extremitate la cealalta. Astfel, o rotire completa a unui pontentiometru reprezinta doar trei sferturi din semnificatia matematica a acesteia.

### Rezistoare semireglabile

Acestea sunt utilizate, in general, pentru calibrarea instrumentelor, modificarea fina a tensiunii sau curentului generate de surse in comutatie. In mod normal, acestea pot fi inlocuite de potentiometre inasa, este preferata utilizarea lor atunci cand calibrarea sau controlul trebuie facuta de un specialist si mecanismul nu trebuie sa fie accesibil oricui.



*Figura 9. Modele de rezistoare semireglabile*

Exista putine modele de rezistoare semireglabile disponibile. In cadrul figurii 9, primul si al patrulea model sunt semireglabile multitura, utilizate atunci cand este necesara stabilirea unei rezistente foarte precise. Deoarece acestea sunt ermetice, valoarea rezistentei reglate este oarecum independenta de factorii externi.

Unul dintre parametrii importanti ai potentiometrului este legea de variatie a rezistentei in functie de pozitia cursorului. In functie de acest parametru, pot fi clasificate in: liniare, logaritmice, invers logaritmice, exponentiale si invers exponentiale. Rezistoarele semireglabile sunt, in general, doar liniare. In figura de mai jos, este prezentata variatia procentuala rezistentei raportata la variatia unghiului de rotatie al potentiometrului pentru pricipalele tipuri de potentiometre.

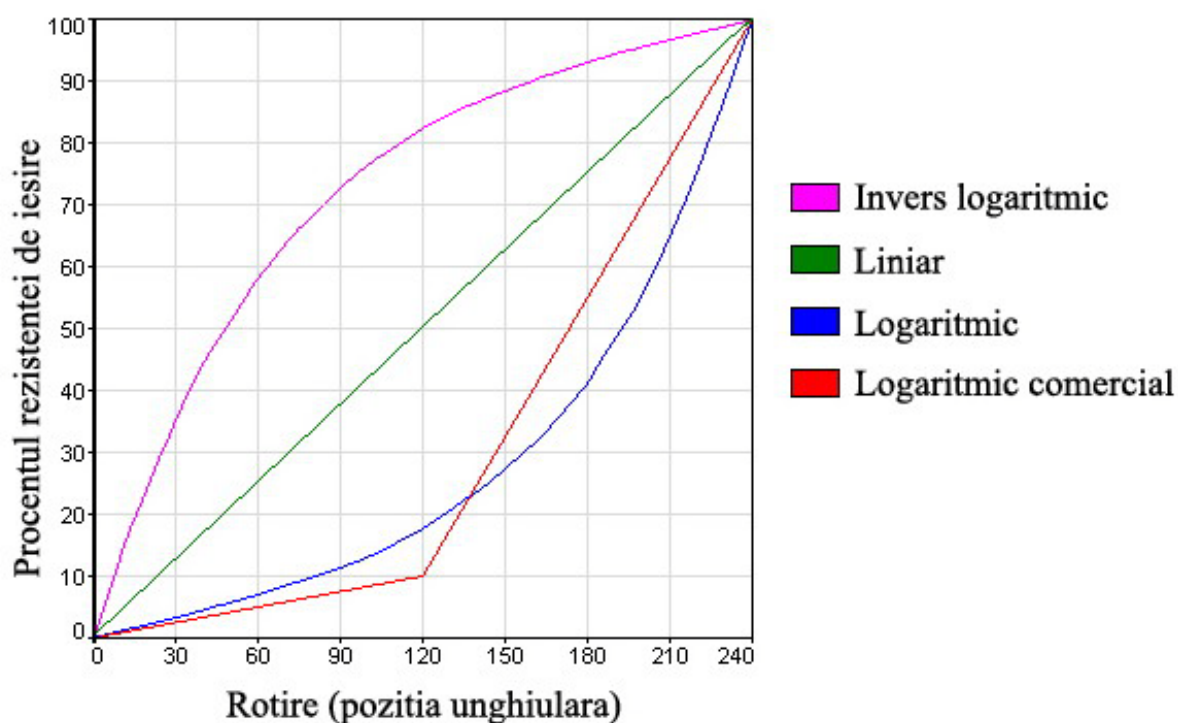


Figura 10. Variatia procentuala rezistentei raportata la variatia unghiului de rotatie

În imaginea de mai jos este prezentat un potentiometru rotativ clasic. Pinul terminal notat cu cifra 2 este cursorul. Într-o aplicație de control al volumului unui sistem audio, pinul 1 este conectat la masă, intrarea este aplicată în pinul 3 și ieșirea este preluată de la pinul 2, reglând semnalul de la minim (conexiune la masă) până la maxim (conexiune directă la intrare).



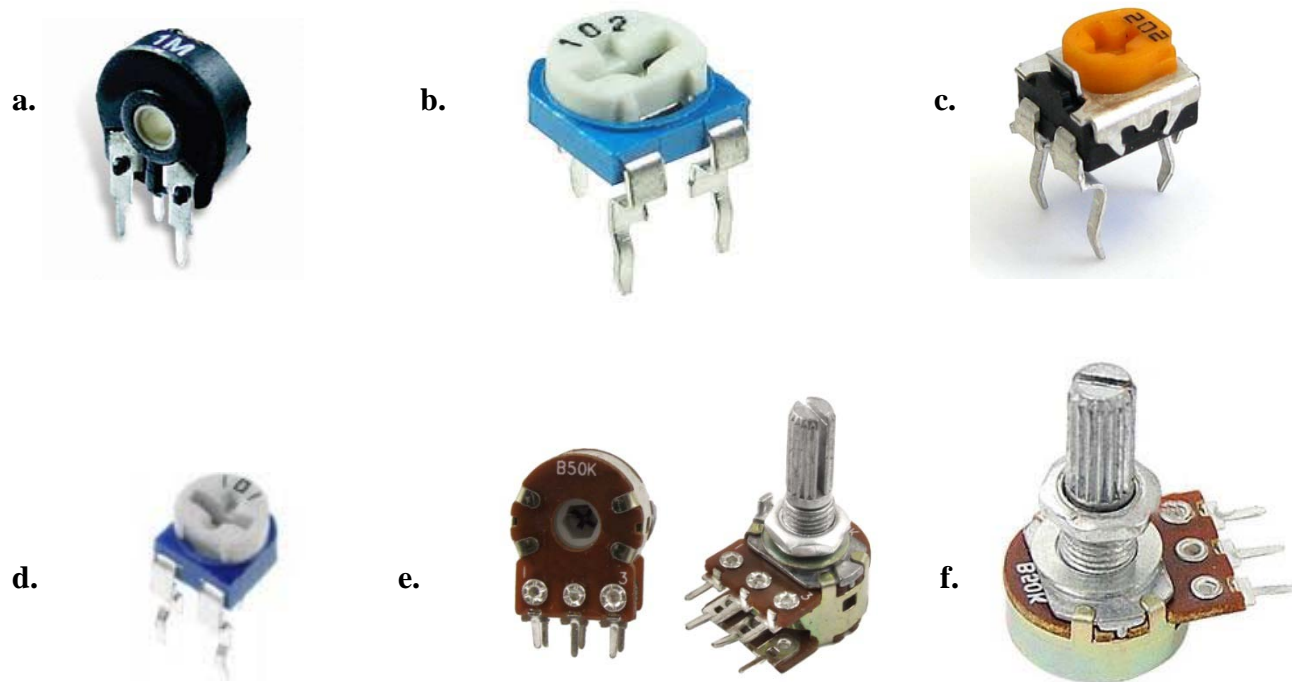
*Figura 11. Elementele constructive ale unui potentiometru*

### **Marcarea potentiometrelor și a rezistoarelor semireglabile**

Așa cum se observă în imaginea de mai jos, potentiometrul este marcat prin intermediul unei succesiuni de caractere alfanumerice. Primul simbol indică legea de variație a rezistenței și poate fi: B pentru o variație liniară sau A pentru variație logaritmică. În condițiile în care nu există un astfel de simbol marcat pe potentiometru sau pe rezistorul semireglabil, variația este liniară. Următoarele simboluri reprezintă valoarea rezistenței maxime reglate. În cazul potentiometrelor, valoarea rezistenței maxime este marcată în clar, cu specificația că pentru multiplu de 1000 este utilizat simbolul K (adică  $1K = 1000\Omega$ ), iar pentru  $10^6$  simbolul M (adică  $1M = 10^6\Omega$ ). Marcajul pentru valoarea rezistenței maxime reglate în cazul rezistoarelor semireglabile este similar cu acela pentru marcată similar cu marcajul cu trei simboluri alfanumerice al rezistoarelor SMD. Astfel, semireglabilul marcat cu simbolul 104 va avea valoarea de  $10 \times 10^4\Omega = 100k\Omega$ .

Ca si in cazul rezistoarelor, valorile potentiometrelor si ale rezistoarelor semireglabile sunt standardizate. In general, acestea respecta secventa 1; 2,5; 5. Astfel, valorile cele mai des intalnite sunt:  $1k\Omega$ ,  $5k\Omega$ ,  $10k\Omega$ ,  $25k\Omega$ ,  $50k\Omega$ ,  $100k\Omega$ ,  $500k\Omega$  si  $1M\Omega$ .

In figura de mai jos sunt prezentate cateva modele de rezistoare semireglabile si potentiometre.



*Figura 12. a) rezistor semireglabil vertical  $1M\Omega$ , b) rezistor semireglabil orizontal  $1k\Omega$ , c) rezistor semireglabil orizontal  $2k\Omega$ , d) rezistor semireglabil orizontal  $100\Omega$ , e) potentiometru liniar  $50k\Omega$ , f) potentiometru liniar  $20k\Omega$*

## Dioda semiconductoare

Dioda este un dispozitiv cu doua terminale, care dispune intr-un sens de rezistenta electrica minima (teoretica egala cu zero) si de o rezistenta apropiata de infinit in celalalt sens. Aceasta poate fi furnizata de catre producatori sub diverse forme, insa, cea mai uzuala este aceea prezentata in Figura 13.

Cele doua terminale ale diodei au denumiri diferite si anume **anod** (polul pozitiv), respectiv **catod** (polul negativ). Pentru a utiliza corect dioda in circuit, catodul este indicat pe capsula diodei prin intermediul unui *inel* desenat, ca in Figura 13. Așa cum am specificat in descrierea sa, proprietatea principala a unei diode este aceea ca permite circulatia curentului intr-un singur sens, fiind un dispozitiv unidirectional.



Figura 13. Dioda semiconductoare.

In circuitele electronice, dioda semiconductoare este simbolizata ca in Fig.14.

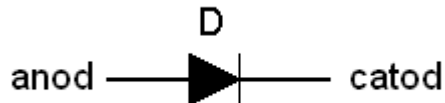
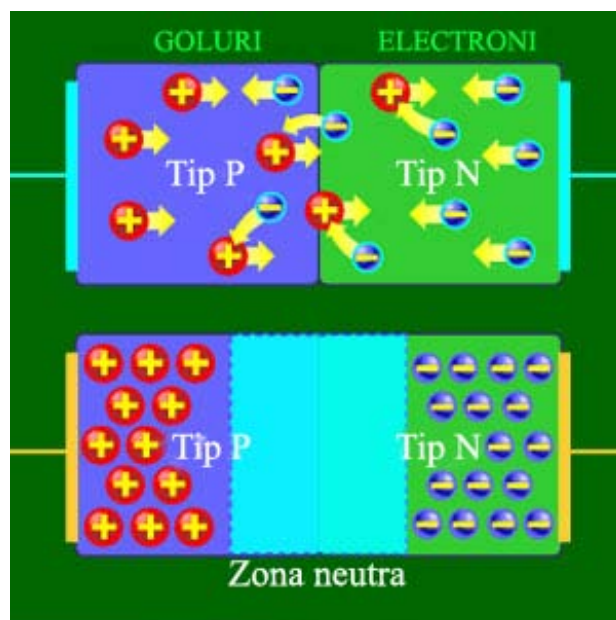


Figura 14. Simbolul electronic al diodei semiconductoare.

In paragrafele care urmeaza vom explica cum functioneaza efectiv dioda. Aceasta este cel mai simplu dispozitiv semiconductor. Semiconductorul este un material a carui proprietate de a conduce curentul electric poate varia in functie de anumite conditii. Majoritatea elementelor semiconductoare sunt construite din materiale slab conductoare carora si le se adauga anumite impuritati. Procedura de adaugare a impuritatilor este denumita dopare.

Un semiconductor cu un numar suplimentar de electroni este denumit material de tip N, intrucat este incarcat cu sarcina negativa. In cadrul elementului de tip N, electronii liberi se deplaseaza din regiunea incarcata negativ spre cea incarcata pozitiv. Pe de alta parte, semiconductorul cu un numar suplimentar de goluri, sarcini pozitive sau regiuni spre care pot migra sarcinile negative, este denumit element de tip P, deoarece este incarcat cu sarcina pozitiva. Electronii se pot deplasa dintr-un spatiu gol intr-altul, dintr-o zona incarcata negativa

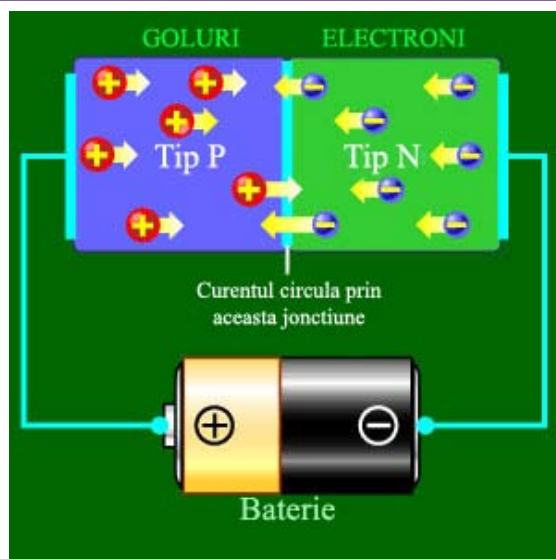
in alta zona, incarcata pozitiv. Drept rezultat, pare ca aceste goluri se deplaseaza dinspre zona incarcata pozitiv catre zona incarcata negativ.



*Figura 15. La jonctiune, electronii liberi din zona de tip N umplu golurile din zona de tip P. Astfel este generat un strat izolator in zona centrala a diodei, denumita zona neutra.*

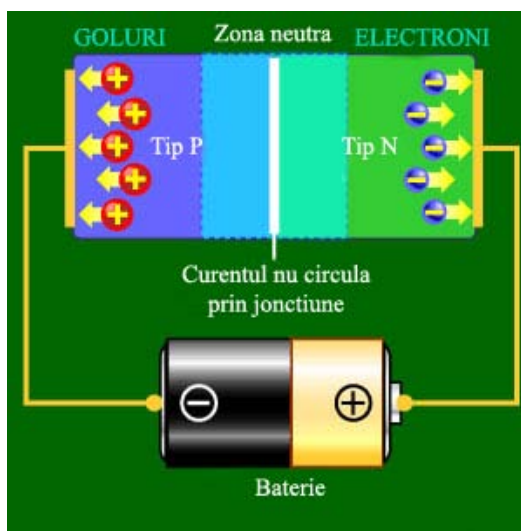
O dioda este compusa dintr-o sectiune de tip N conectata la o sectiune de tip P, cu electrozi la fiecare capat. Acest ansamblu conduce electricitatea intr-un singur sens. Atunci cand nu este aplicata tensiune diodei, electronii din sectiunea de tip N umplu golurile din sectiunea de tip P de-a lungul jonctiunii dintre straturi, formand o zona neutra, fara purtatori liberi. In aceasta zona, materialul semiconductor revina la starea sa normala, in care nu conduce curentul electric.

Pentru a elimina zona neutra, este necesar ca electronii sa se deplaseze din zona de tip N catre zona de tip P si golurile in sens invers. Pentru ca acest lucru sa se realizeze, regiunea de tip N a diodei trebuie conectata la polul negativ al unui circuit si regiunea de tip P la polul pozitiv. Astfel, electronii liberi din zona de tip N sunt respinsi de electrodul negativ si atrasi catre polul pozitiv si golurile se vor deplasa in sensul opus. Atunci cand diferenta de potential, adica tensiunea aplicata la bornele diodei, este suficient de mare electronii din zona libera sunt indepartati din golurile pe care le ocupa si incep sa se deplaseze, din nou, liber. Drept rezultat, curentul electric va trece prin dioda.



*Figura 16. Atunci cand polul negativ al unui circuit este conectat la stratul de tip N al diode si polul pozitiv la stratul de tip P, electronii si golurile incep sa se deplaseze si zona neutra dispare.*

Daca dioda este alimentata in celalalt sens, cu zona de tip P conectata la polul negativ al unui circuit si zona de tip N la polul pozitiv, sarcina nu va mai trece prin dioda. Electronii din zona de tip N sunt atrasi de polul pozitiv, iar golurile din zona de tip P sunt atrase de polul negativ. Curentul nu trece prin dioda deoarece golurile si electronii se deplaseaza in directia opusa sensului in care ar trebui sa circule si zona libera se extinde.



*Figura 17. Atunci cand polul pozitiv al unui circuit este conectat la stratul de tip N al diode si polul negativ la stratul de tip P, electronii si golurile sunt atrasi si zona neutra isi maresc dimensiunea, oprind trecerea curentului.*



În cazul în care tensiunea pe dioda este pozitivă, se spune că aceasta funcționează în *conducție directă*, iar în cazul în care tensiunea pe dioda este negativă, se spune că aceasta funcționează în *conducție inversă*.

- în *conducție directă*: se observă o tensiune de prag (notată  $V_D$ ); dacă valoarea tensiunii pe dioda este sub prag, prin dioda nu trece curent electric; dacă valoarea tensiunii pe dioda atinge pragul, atunci prin dioda trece curent electric. Curentul prin dioda nu trebuie să depășească o anumită valoare maximă, notată în cataloagele de diode  $I_F$ , impusă de puterea maximă pe care o poate disipa dioda fără a se distruge termic.
- în *conducție inversă*: se observă o tensiune specifică denumită tensiune de străpungere, (notată cu  $V_{BR}$ ) a cărei valoare este de ordinul zecilor-sutelor de volți; dacă valoarea în modul a tensiunii pe dioda este mai mică decât valoarea  $V_{BR}$ , curentul electric nu trece prin dioda; dacă valoarea tensiunii pe dioda atinge valoarea  $V_{BR}$ , atunci prin dioda curentul electric crește necontrolat – acest fenomen se numește străpungerea diodei.

### Tipuri de diode

Există variante diverse de diode care se deosebesc prin particularități funcționale și destinație. Principalele categorii sunt prezentate în continuare.

### Diode redresoare

Sunt diode destinate utilizării în circuite redresoare pentru rețeaua de curent alternativ. Parametrii principali sunt curentul maxim,  $I_M$  și tensiunea inversă maximă,  $U_M$ . Plaja de valori ale acestor parametri este:

- amperi-zeci de mii de amperi, pentru  $I_M$ ;
- zeci volți - zeci de mii de volți pentru  $U_M$ .

Diodele de curenți mari sunt construite în așa mod încât să le poată fi atasate radiatoare de răcire. Diodele de curenți mici sunt închise în capsule de plastic sau ceramică și au catodul marcat cu o bandă albă sau neagră.

### Diode de comutație

Sunt diode destinate utilizării în circuite funcționând în comutație sau la frecvențe ridicate. Parametrii principali sunt timpii de comutație.

### Diode Schottky

Sunt diode realizate intr-o tehnologie speciala, de tip metal-semiconductor si au simbolul prezentat in Fig.18. Avantajele acestor diode sunt:

- tensiune mica in conductie, aproximativ 0,3V;
- timpi de comutatie foarte mici.

Dezavantajul principal:

- tensiune inversa maxima mica (zeci de volti).

### Dioda varicap

Denumirea diodei vine de la expresia capacitate variabila. Simbolul este prezentat in Fig.2.16.b. Dioda este utilizata in polarizare inversa si proprietatea principala este ca se comporta in aceasta situatie ca un condensator cu capacitate variabila, dependenta de tensiunea la borne. Domeniul principal de utilizare sunt radiocomunicatiile, mai precis acordul circuitelor oscilante din emitatoare si receptoare.



Figura 18. a) Dioda Schottky. b) Dioda varicap.

### Dioda Zener

Este o dioda construita pentru a fi utilizata in zona de strapungere inversa. Simbolurile utilizate pentru dioda Zener si caracteristica grafica sunt prezentate in Fig. 19. In polarizare directa este similara diodelor redresoare. In polarizare inversa dioda se strapunge la o tensiune numita tensiune Zener,  $U_Z$ , constanta pentru o anumita dioda. Astfel, parametru principal care caracterizeaza dioda Zener este tensiunea de stapungere. Proprietatea de a mentine constanta tensiunea pe o plaja mare de curenti face ca dioda sa fie utilizata indeosebi in circuitele stabilizatoare de tensiune.



Figura 19. Dioda Zener

## L.E.D. – Dioda electroluminiscenta

Interactiunea dintre electroni si goluri din acest ansamblu mai are un efect interesant – poate genera lumina.

Lumina este o forma de energie eliberata de un atom. Aceasta este alcatuita din mici particule care poseda energie dar nu au masa, denumite fotoni. Fotonii sunt eliberati ca rezultat al miscarii electronilor. In interiorul atomului, electronii se deplaseaza pe orbita in jurul nucleului. In functie de distanta fata de nucleu, acestia poseda o anumita energie; cu cat se regasesc la o distanta mai mare de nucleu au o energie mai mare. Pentru ca un electron sa se deplaseze pe un nivel mai indepartat de nucleu, acesta trebuie sa isi mareasca energia. In sens invers, pentru a se deplasa pe un nivel mai apropiat de nucleu, electronul elibereaza energie. Aceasta energie este eliberata sub forma unui foton. Cu cat este eliberata mai multa energie cu atat fotonul eliberat poseda o energie mai mare, care se reflecta intr-o crestere a frecventei luminii emise.

Asa cum am discutat in sectiunea despre functionarea diodei, electronii liberi pot ajunge in goluri din zona de tip P. Aceasta implica deplasarea intr-o regiune mai apropiata de nucleu, astfel ca, electronii vor elibera energie sub forma de fotoni. Fenomenul are loc in orice dioda, dar fotonii pot fi observati doar in conditiile in care dioda este construita dintr-un anumit material. De exemplu, atomii dintr-o dioda de siliciu standard sunt aranjati astfel incat energia emisa de deplasarea electronilor este destul de mica, astfel ca, lumina emisa de acestea se regaseste in spectrul infrarosu si este invizibila pentru ochiul uman.

Diodele care emit lumina in spectrul vizibil sunt construite din materiale caracterizate de o distanta mai mare intre orbitele pe care se deplaseaza electronii. Aceasta distanta determina frecventa fotonului emis, deci culoarea luminii emise.

LED-urile sunt construite pentru a elibera un numar relativ mare de fotoni. In plus, acestea sunt integrate intr-o carcasa care concentreaza lumina intr-o anumita directie. Dupa cum observati in imagine, mare parte a luminii este directionata catre capatul rotund al LED-ului.

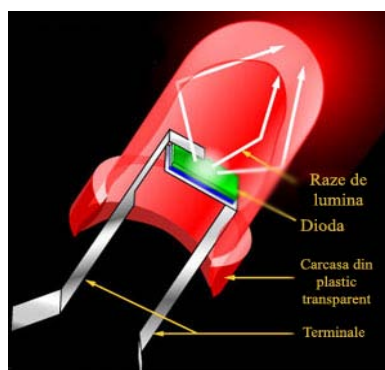


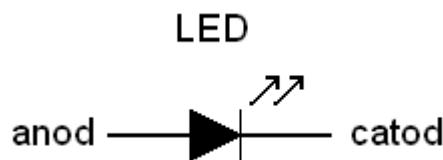
Figura 20. Structura LED

Principalul avantaj, ce recomanda utilizarea LED-urilor, este eficienta. Lampa cu incandescenta genereaza multa caldura in procesul de producere a luminii, avand astfel un randament destul de scazut. Spre deosebire de acestea, LED-urile utilizeaza un procent mult mai mare al puterii electrice pentru a genera lumina. LED-urile ofera o eficienta luminoasa mai mare decat becul incandescent; de exemplu, un LED standard produce in jur de 75 lumeni pentru 1 watt consumat, in comparatie cu un bec ce poate produce 17 lumeni pentru 1 watt consumat. Un alt avantaj este durata de viata, media fiind de aproximativ 50.000 de ore.

La fel ca si dioda, LED-ul permite trecerea curentului doar in directie directa, iar trecerea curentului electric prin LED este semnalizata prin aprinderea acestuia. Spre deosebire de dioda semiconductoare, LED-ul are o tensiune de prag  $V_{LED}$  de aproximativ 1,6V.



a)



b)

Figura 21. a) LED-ul sau dioda electroluminiscenta; b) Simbolul electronic al LED-ului

## Testarea unui rezistor

Pentru a testa daca un rezistor este defect, va trebui sa verificam valoarea rezistentei sale utilizand ohmmetrul unui multimetru.

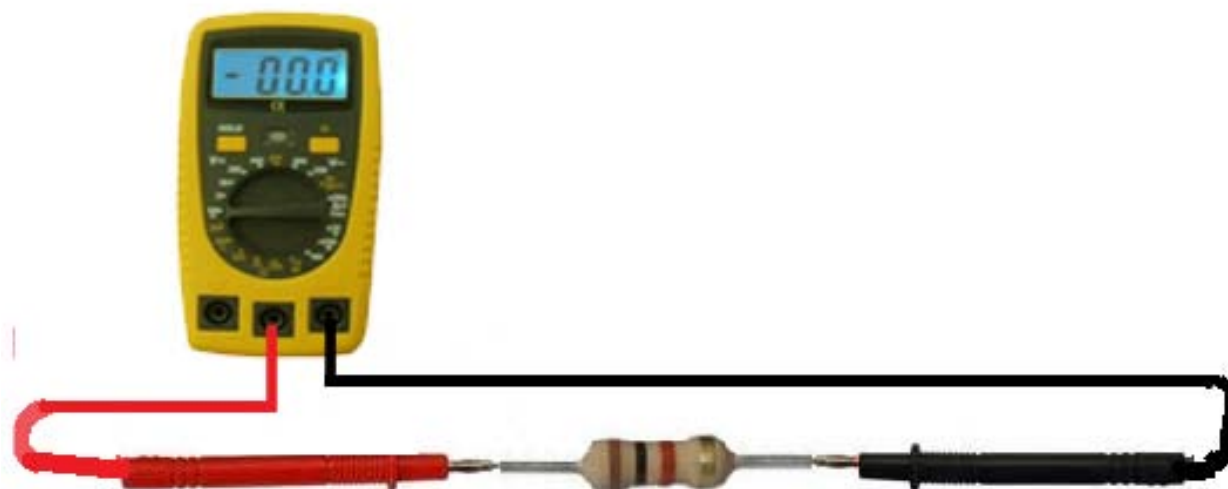
### Testare rezistorului utilizand ohmmetrul

Aceasta este cea mai eficienta modalitate de a testa rezistorul. Pentru a efectua acest test cu o precizie cat mai mare, vom seta multimetrul pe o valoare mai mare dar cat mai apropiata de rezistenta rezistorului pe care urmeaza sa il testam si vom conecta bornele multimetrului la terminalele rezistorului. Sensul in care conectam bornele nu conteaza, intrucat rezistorul este nepolarizat.

Pentru aceasta operatie, sonda de culoare neagra a multimetrului se va conecta la borna *COM* a multimetrului, cea de culoare rosie la borna  $\Omega$ , *V*, *mA* iar cursorul va fi setat pe o valoare cat mai apropiata de rezistenta ce urmeaza a fi evaluata din zona evidentiata in imaginea de mai jos. De exemplu, daca vom masura un rezistor a carui rezistenta este de  $1k\Omega$ , vom seta cursorul multimetrului la  $2k$ . Dacac vom masura un rezistor a carui rezistenta este de  $22k\Omega$  vom seta cursorul multimetrului la  $200k$  (chiar daca  $20k$  este mai apropiata ca valoare, setand pe aceasta valoare vom receptiona o eroare).



*Figura 22. Zonele de interes ale multimetrului atunci cand evaluam valoarea rezistentei unui rezistor*



*Figura 23. Evaluarea rezistentei unui rezistor cu ajutorul multimetrului digital*

Rezistenta pe care ohmmetrul o va afisa trebuie sa fie aproximativ egala cu cea marcata pe rezistor. De exemplu, daca vom masura un rezistor cu valoarea de  $1\Omega$  si toleranta 5%, precum este cel din figura 23, valoarea pe care o vom masura cu ohmmetrul va fi intre  $950\Omega$  si  $1050\Omega$ . Daca valoarea masurata se incadreaza in intervalul de toleranta (in cazul exemplificat de noi,  $950\Omega \sim 1050\Omega$ ) atunci rezistorul este bun. Daca valoarea masurata este cu mult in afara acestui interval, rezistorul este defect.

## Testarea unei diode

Exista mai multe metode de a verifica daca o dioda este functionala. Vom vedea care dintre caracteristicile diode pot fi exploatate pentru a verifica daca aceasta este functionala.

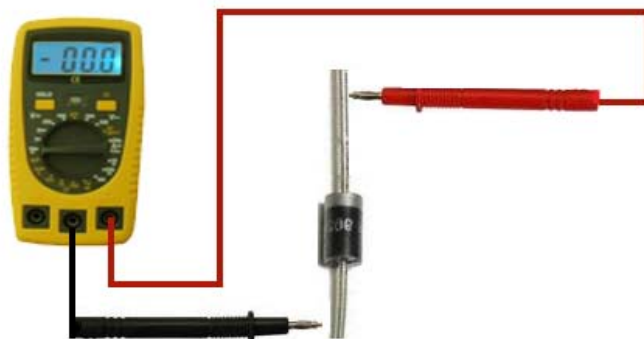
### Testare cu ajutorul ohmmetrului

Prin intermediul acestui test putem afla daca dioda este functionala, deschisa sau scurt-circuitata. De data aceasta este foarte importanta polaritatea.

### Testarea rezistentei anod-catod a diodei

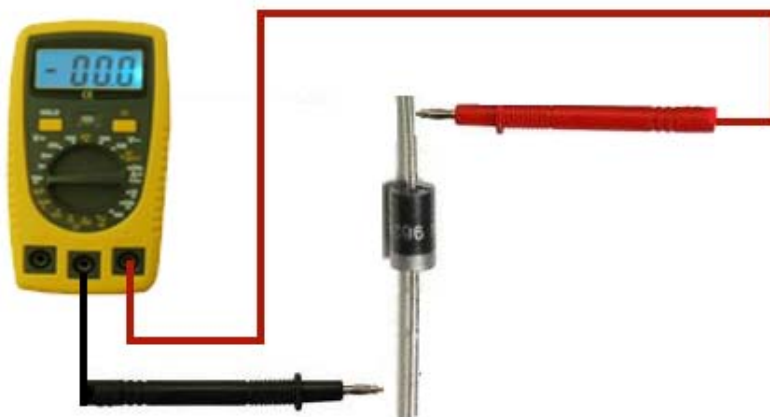
Vom plasa sonda pozitiva a multimetrului la anodul diodei si sonda negativa la catodul diodei (linia argintie marcata pe dioda). Multimetrul trebuie sa citeasca o valoare relativ mica a

rezistentei, adica de ordinul zecilor sau sutelor de Ohmi. De exemplu, aceasta valoare poate fi in jur de 230K $\Omega$ .



### Testarea rezistentei catod-anod a diodei

De data aceasta, vom plasa sonda pozitiva a multimetrului la catodul diodei (linia argintie marcata pe dioda) si sonda negativa la anodul diodei. Multimetrul trebuie sa citeasca o valoare foarte mare a rezistentei, mai mare de 1M $\Omega$  sau chiar OL.





### *Dioda deschisa*

Daca dioda are o rezistenta foarte mare in ambele sensuri, atunci inseamna ca aceasta este defecta si trebuie inlocuita.

### *Dioda in scurt-circuit*

Daca dioda are o rezistenta relativ mica in ambele sensuri, atunci inseamna ca este scurt-circuitata. Drept urmare, este defecta si trebuie inlocuita.

### *Testarea diodei cu ajutorul voltmetrului*

Problema testarii diodei cu ajutorul ohmmetrului este ca reprezinta doar o evaluare calitativa a acesteia, nu si una cantitativa. Intrucat diodele poseda o cadere de tensiune specifica la bornele lor, putem utiliza aceasta proprietate pentru a testa functionalitatea diodei. Pentru a efectua acest test, va trebui sa integram dioda intr-un circuit de curent continuu similar cu acela din figura de jos.



Dioda de siliciu are o cadere de tensiune între terminalele sale de aproximativ  $0,4V \sim 0,7V$ . Deci, măsurând tensiunea de la bornele diodei, în circuitul prezentat în imaginea de mai sus, trebuie să citim o valoare a tensiunii cuprinsă în acest interval,  $0,4V \sim 0,7V$ . Dacă dioda este din germaniu, caderea de tensiune măsurată va avea o valoare aproximativ egală cu  $0,3V$ .

### **Dioda deschisă**

Dacă tensiunea măsurată este foarte mare, atunci dioda este deschisă, deci defectă, și va trebui înlocuită.

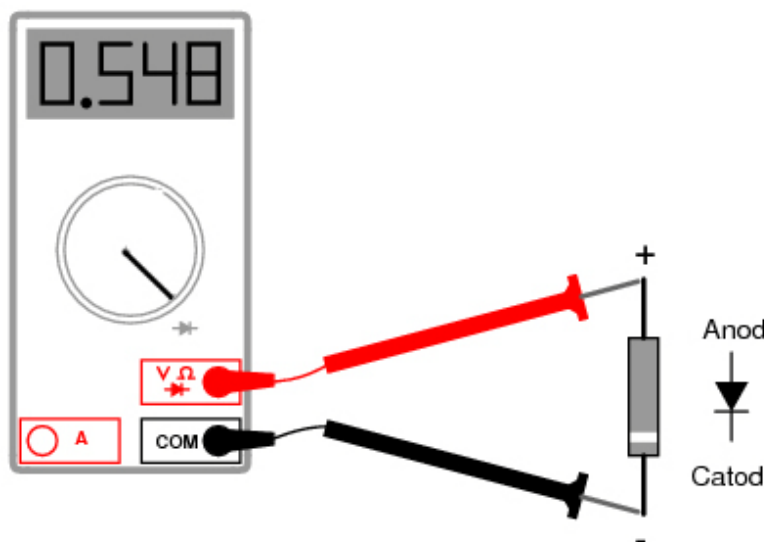
### **Dioda în scurt-circuit**

Dacă tensiunea măsurată este foarte mică, atunci dioda este scurt-circuitată.

### **Testarea diodei utilizând circuitul specializat de testare al multimetrului**

O parte dintre multimetre digitale au integrat un sistem specializat de măsurare a diodelor. Acest sistem de testare evaluează caderea de tensiune în volți, nu rezistența. În condițiile în care multimetrul posedă funcția de testare a diodei, nu mai este necesară construirea circuitului pe care l-am studiat anterior.

Testare tensiunii directe (sonda pozitivă a multimetrului la anodul diodei și sonda negativă la catodul diodei, linia argintie marcată pe dioda) obținută cu un astfel de sistem va fi mai mică decât cea tipică, de  $0,7V$  pentru diodele din siliciu sau  $0,3V$  pentru diodele din germaniu, întrucât curentul furnizat de multimetru este foarte mic, pentru economisirea bateriei. În figura de mai jos este evidențiată această procedură de evaluare a diodei.



1. Utilizand software-ul de pe CD pentru calculul rezistentei rezistorilor in functie de codul culorilor marcat pe acestea, identificati valorile rezistorilor care se regasesc in kit-ul dvs. Dupa identificarea valorilor rezistorilor si sortarea acestora, evaluati rezistenta lor cu ajutorul multimetrului pe care il aveti la dispozitie si notati aceste valori intr-un tabel dupa modelul celui de mai jos.

Nr. crt.	Codul culorilor	Valoarea calculata in functie de codul culorilor ( $\Omega$ )	Toleranta (%)	Valoare masurata ( $\Omega$ )	Abatare fata de valoarea normala ( $\Omega$ )	Abatere procentuala
1.	Maro – Rosu – Rosu - Argintiu	1.200 (1,2k)	10	1180	20	1,6%
2.	Maro – Negru – Maro - Argintiu	100	10	96	4	4%
3.	Rosu – Violet – Galben - Argintiu	270.000 (270k)	10	277.000	7.000	2,6%
4.						
5.						

Utilizand rezultatele obtinute prin efectuarea baza masuratorilor, evaluati starea rezistoarelor din kit-ul dvs.

2. In kit-ul dvs. exista un set de diode redresoare, marcate 1N4007. Identificati-le si precizati denumirea terminalelor.



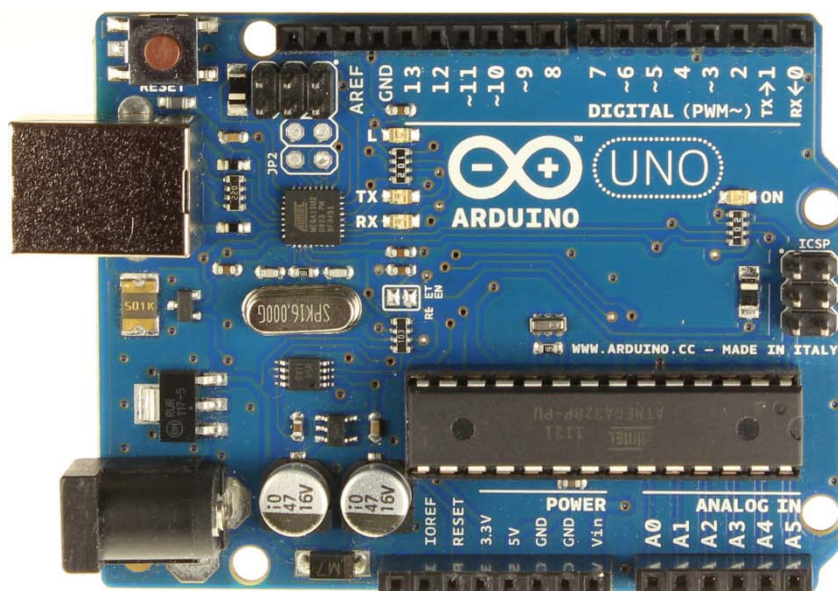
*Diode 1N4007*

3. Folosind metodele care v-au fost descrise mai sus, masurati rezistenta anod-catod, rezistenta catod-anod si caderea de tensiune dintre terminale pentru fiecare dintre diode. Utilizati un tabel similar cu acela de mai jos.

Nr. crt.	Dioda testata	Rezistenta anod-catod( $\Omega$ )	Rezistenta catod-anod( $\Omega$ )	Cadere de tensiune intre terminale (V)	Cadere de tensiune masurata cu testerul DIODA (V)
1.	1N4007	275k	6,5M	0,612	0,635
2.		283k	8,1M	0,628	0,649
3.					
4.					
5.					

Utilizand rezultatele obtinute prin efectuarea baza masuratorilor, evaluati starea diodelor din kit-ul dvs.

## Prima utilizare a dispozitivului Arduino



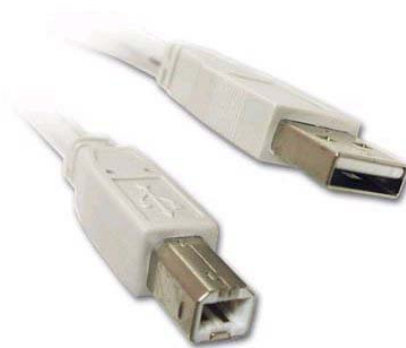
### Introducere

**Arduino Uno** este construit in jurul unui microcontroller ATmega328. Este prevazut cu 14 terminale digitale de intrare/iesire (dintre care 6 pot fi utilizate ca iesiri PWM), 6 intrari analogice, un rezonator ceramic de 16 MHz, o conexiune USB, un conector de alimentare externa si un buton de resetare. Practic, dispozitivul contine toate elementele necesare pentru a programa si utiliza un microcontroller. Spre deosebire de dispozitivele similare precedente, Arduino Uno nu mai utilizeaza un driver suplimentar pentru conversia USB – serial. In locul acestuia, a fost programat un microcontroller Atmega16U2 pentru a asigura comunicatia pe USB.

Reviziile 2 si 3 au adus anumite imbunatatiri dispozitivului: au fost adaugati pinii SCL si SDA, regasiti langa pinul AREF, si pinul IOREF in apropierea celui RESET pentru a permite echipamentelor conectate la Arduino sa se adapteze la tensiunea care le este furnizata de acesta; microcontroller-ul Atmega8U2 a fost inlocuit cu Atmega16U2.

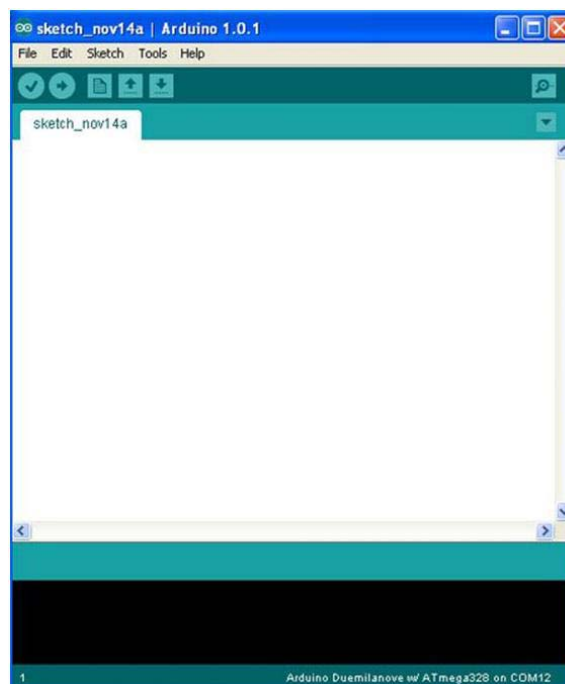
### **Cablul USB**

Cablul de conexiune USB este utilizat pentru a conecta dispozitivul Arduino la computer pentru a incarca programe, alimentare si comunicare seriala. Acest model utilizeaza un cablu USB de tip B, similar cu acela prin intermediul caruia conectati imprimanta la computer.



### **Sistemul software Arduino IDE**

Arduino IDE este sistemul software necesar pentru a scrie cod sursa utilizabil de dispozitivul Arduino si pentru a-l incarca in memoria acestuia. Acest software se regaseste pe CD-ul de instalare sau poate fi descarcat de pe site-ul oficial Arduino, [www.arduino.cc](http://www.arduino.cc), la sectiunea **Download**. Interfata este cea prezentata in figura de mai jos.



Pentru a derula mai usor instalarea, va recomandam sa copiatii fisierele din folderul **Arduino** de pe CD in computerul dvs.

### **Procedura de instalare a driverelor pentru Arduino pentru Windows XP si 7**

Conectati partea USB tip B a cablului la dispozitivul Arduino Uno si cealalta la unul dintre porturile USB ale computerului. Dispozitivul Arduino Uno va fi astfel alimentat din portul USB al computerului. Se vor aprinde continuu LED-urile verde si rosu. La prima utilizare se va instala driver-ul de comunicare prin intermediul port-ului USB. Sistemul de operare va afisa un mesaj prin care anunta utilizatorul ca a fost conectat un nou dispozitiv la computer. Mesajul afisat va fi **Found New Hardware** Arduino Uno.



Sistemul de operare Windows va deschide o fereastră pentru instalarea noului dispozitiv; in cadrul acesteia trebuie sa bifati optiunea **No, not this time** si apoi sa apasati **Next >**. *Daca aceasta fereastră nu a fost afisată din anumite motive sau ati anulat procedura de instalare, accesati Windows Device Manager (Start>Control Panel>Hardware) si indentificati Arduino Uno in lista dispozitivelor conectate la computer. Apasati click dreapta si alegeti **Update driver**.*





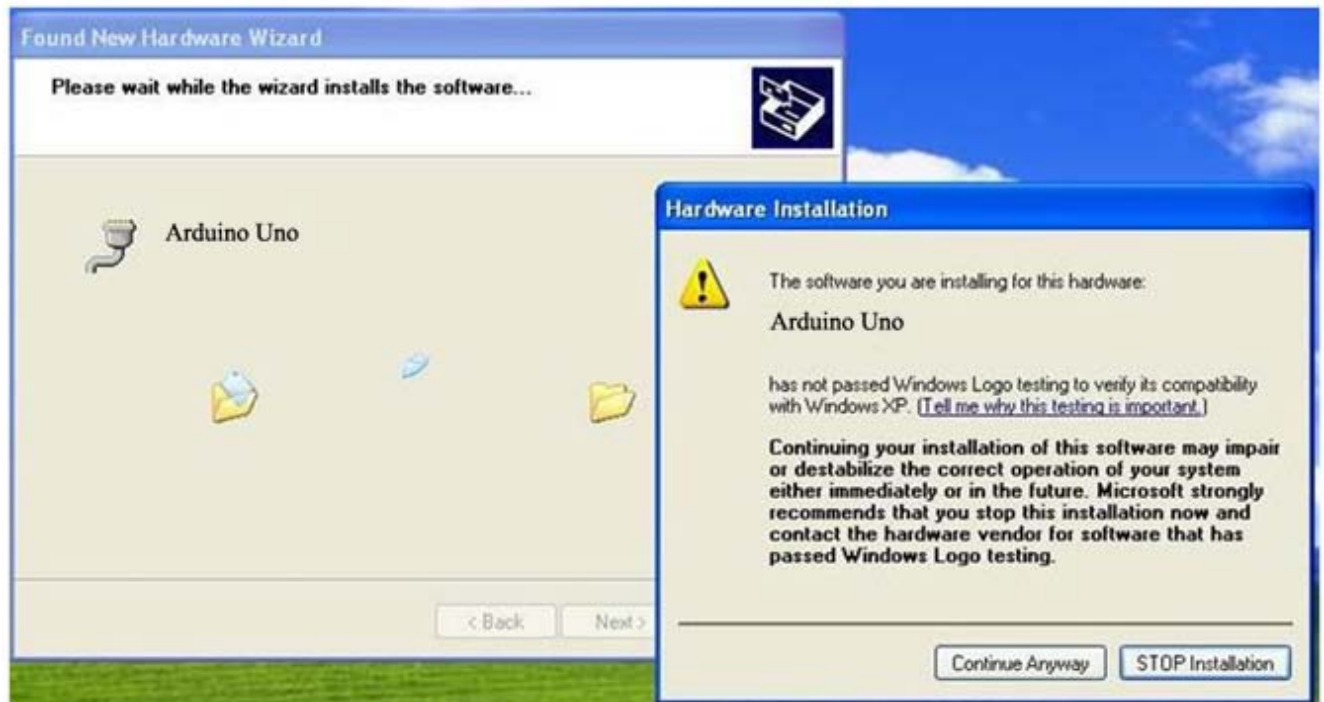
Selectati optiunea **Install from a list of specific location (Advanced)** in fereastra urmatoare apoi apasati **Next >**.



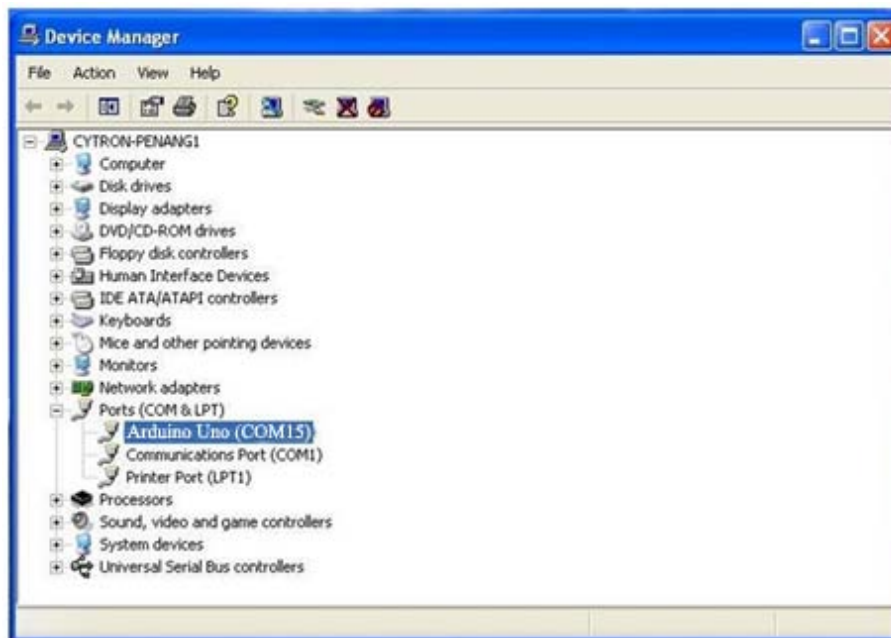
In urmatoarea fereastra asigurati-va ca ati selectat optiunea **Include this location** si cautati folderul in care ati downloadat sau copiat fisierele de instalare ale Arduino. Selectati folderul **drivers** si apasati **OK**.



Sistemul de operare va copia fisierele necesare si va instala driver-ul. In timpul instalarii este posibil sa apara o fereastra de avertizare care va informeaza ca urmeaza sa instalati un dispozitiv care nu a fost testat pentru compatibilitate; apasati **Continue Anyway** pentru a continua instalarea dispozitivului. Dupa finalizarea instalarii driver-ului, apasati **Finish**.



Este necesar sa reporniti computerul pentru instalarea corecta a driver-ului. Dupa repornire, asigurati-va de faptul ca dispozitivul Arduino este conectat la computer prin intermediul cablului USB. LED-ul verde trebuie sa fie ilumineze continuu. In acest moment va trebui sa identificam portul COM asociat dispozitivului Arduino Uno. Deschideti Device Manager (din **Start Menu**, selectati **Settings -> Control Panel**, dublu click pe optiunea **System** si selectati **Hardware**. Apoi apasati pe **Device Manager**).

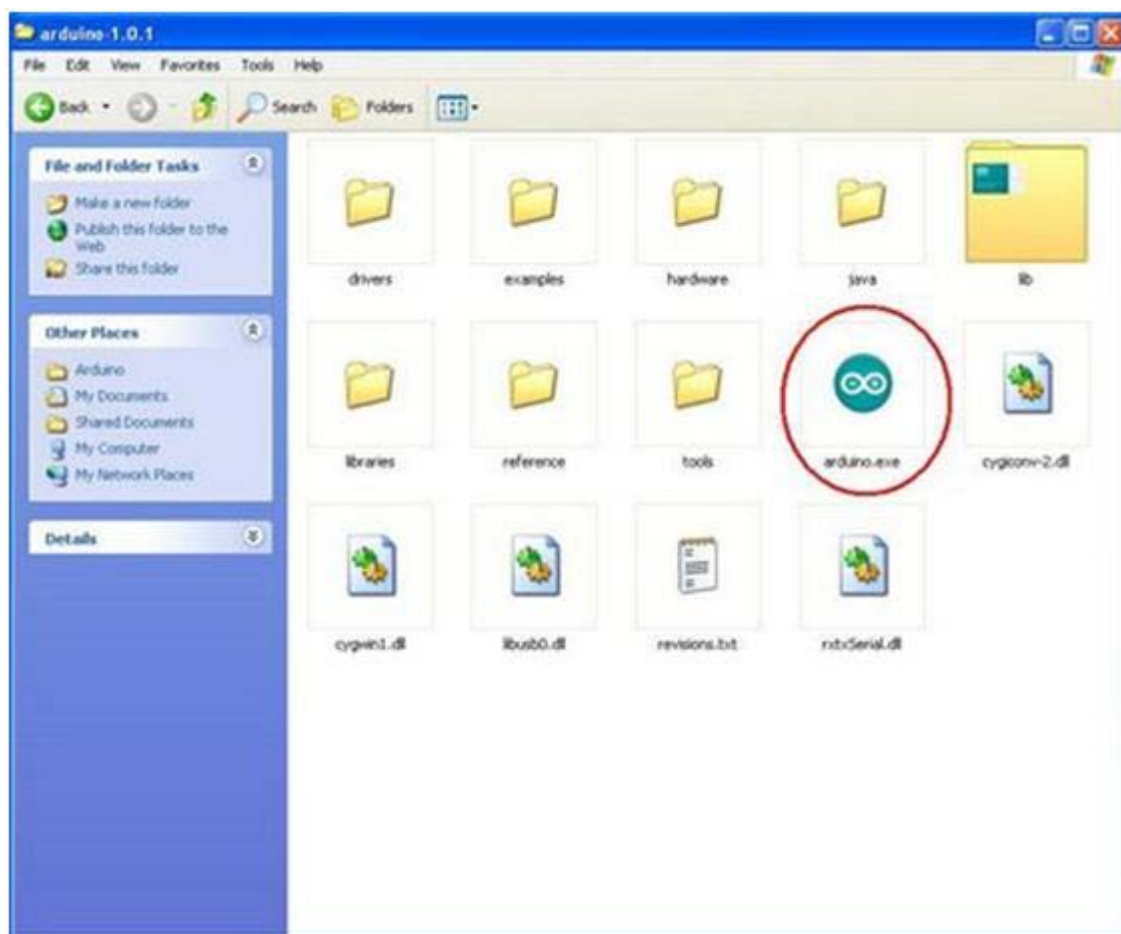


Cautati intrarile asociate port-urilor serial si paralel: **Ports (COM & LPT)** si identificati port-ul denumit **USB Serial Port (COM"X")**. Numarul port-ului asociat dispozitivului Arduino Uno va fi utilizat mai tarziu de catre sistemul software Arduino IDE. In cazul de fata, dupa cum se observa in imagine, port-ul asociat dispozitivului Arduino Uno este COM15. *Daca nu veti identifica in lista de port-uri dispozitivul Arduino, deconectati cablul USB de la computer si conectati-l din nou.*

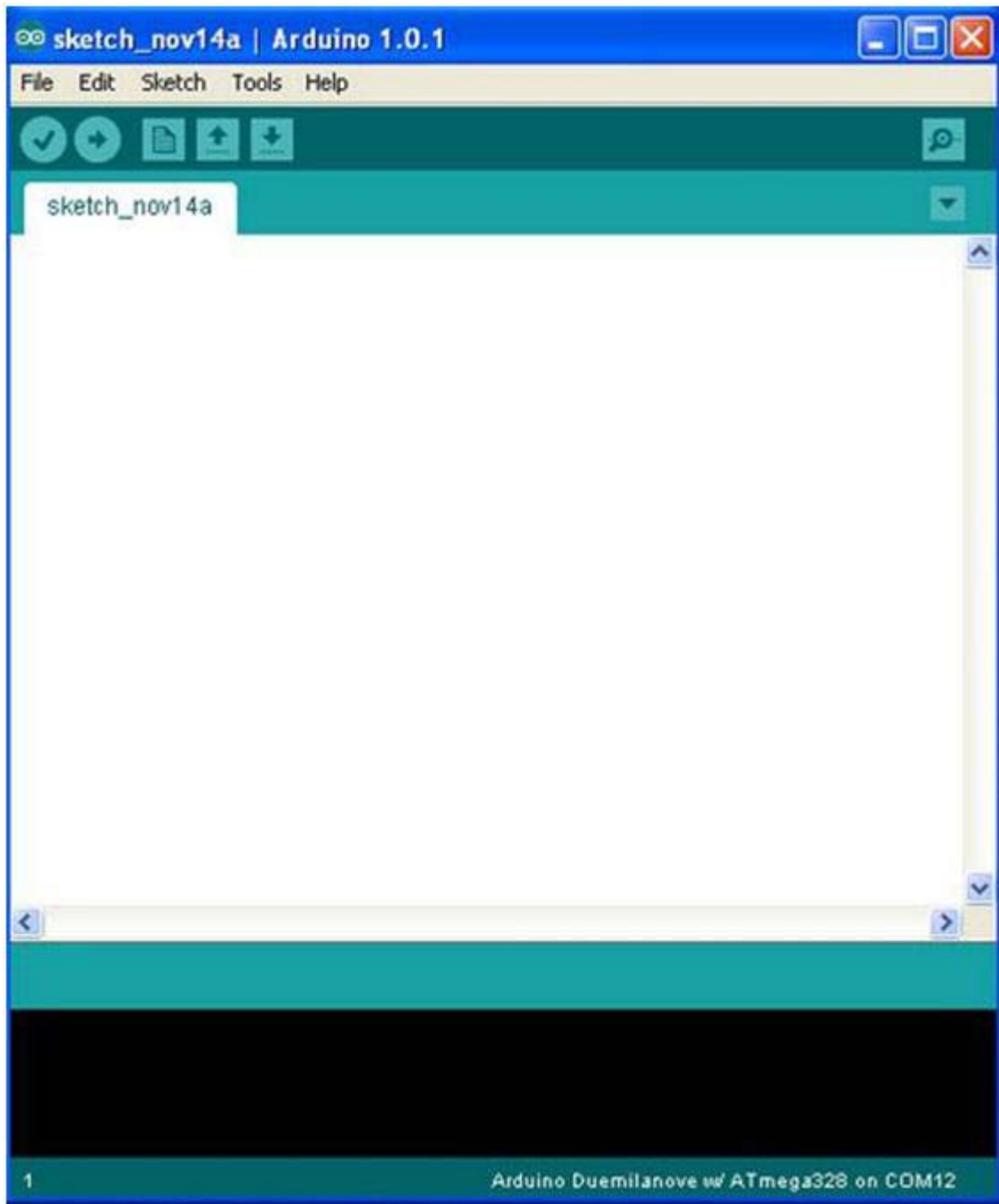
## Prima aplicatie cu Arduino – aprinderea intermitenta a unui LED

In cadrul primelor experimente pe care le vom desfasura, invatam modalitatea de a utiliza pinii digitali pentru a controla diverse dispozitive: LED-uri, difuzoare si cei analogici pentru a prelua informatii de la senzori precum potentiometre sau fotocelule.

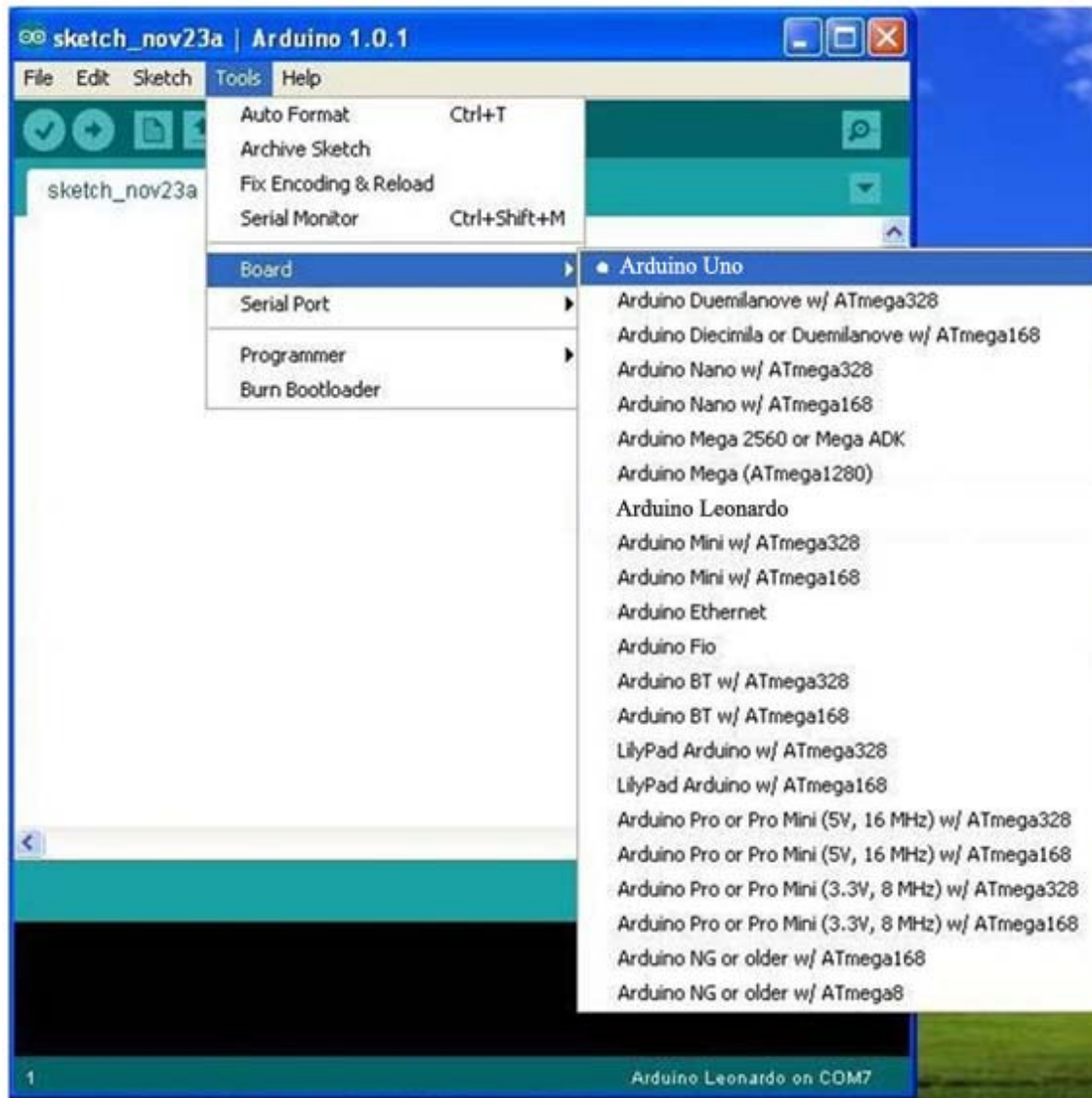
Deschideti folderul in care ati copiat sau downloadat driverele si sistemul software Arduino si identificati fisierul denumit **arduino.exe**; apasati dublu click pe acesta si se va deschide fereastra Arduino IDE.



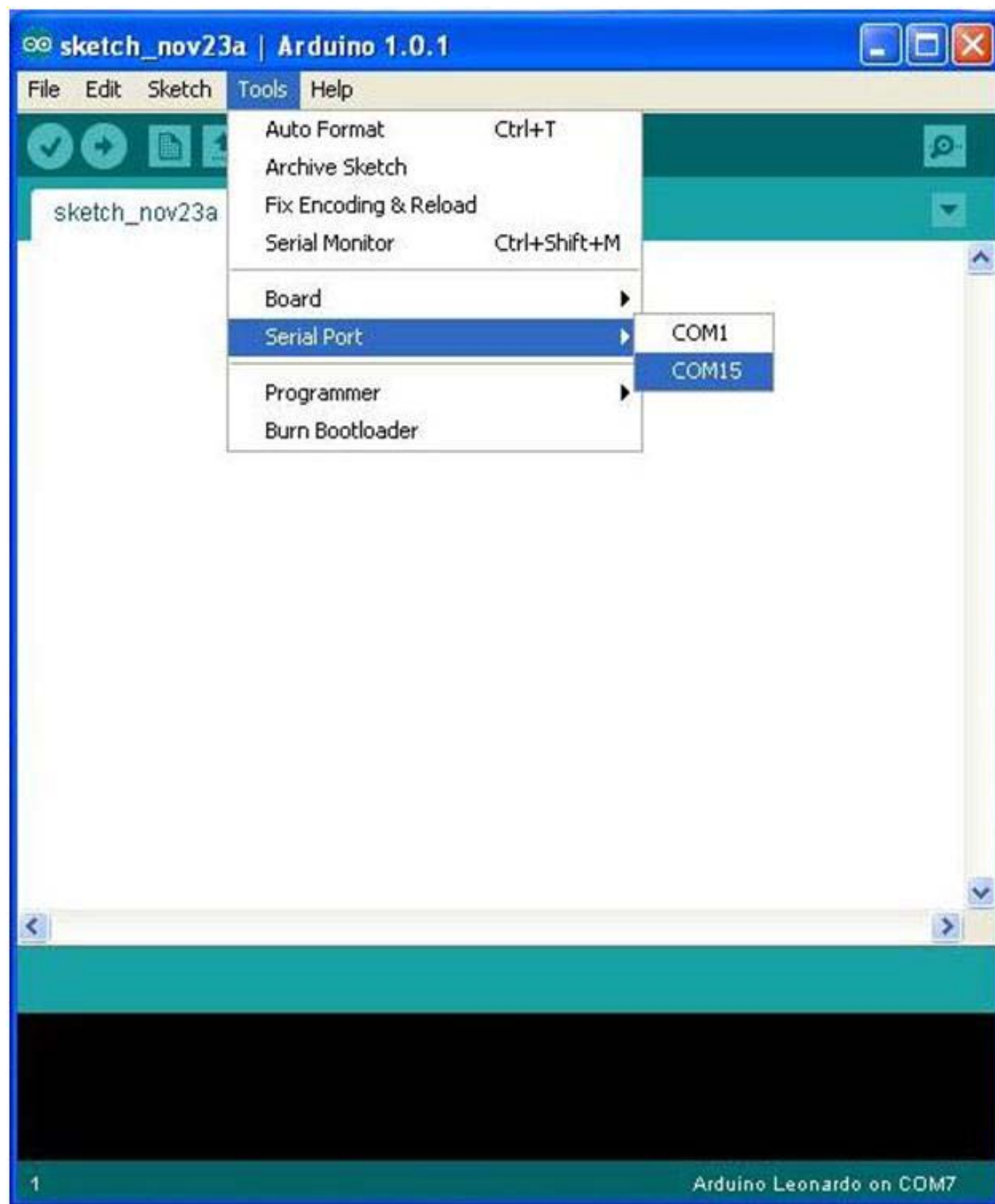
În imaginea de mai jos este prezentată fereastra de pornire a sistemului software Arduino IDE:



Asigurati-va ca dispozitivul Arduino Uno este conectat la computer prin intermediul cablului USB. Sistemul Arduino IDE poate fi utilizat pentru o gama mai larga de dispozitive Arduino, printre care se regaseste si Uno. Pentru ca acesta sa functioneze corect va trebui sa selectati dispozitivul si microcontroller-ul corecte. Selectati din meniul Arduino IDE **Tools -> Board -> Arduino Uno** si apasati o data click dreapta pentru a activa optiunea.

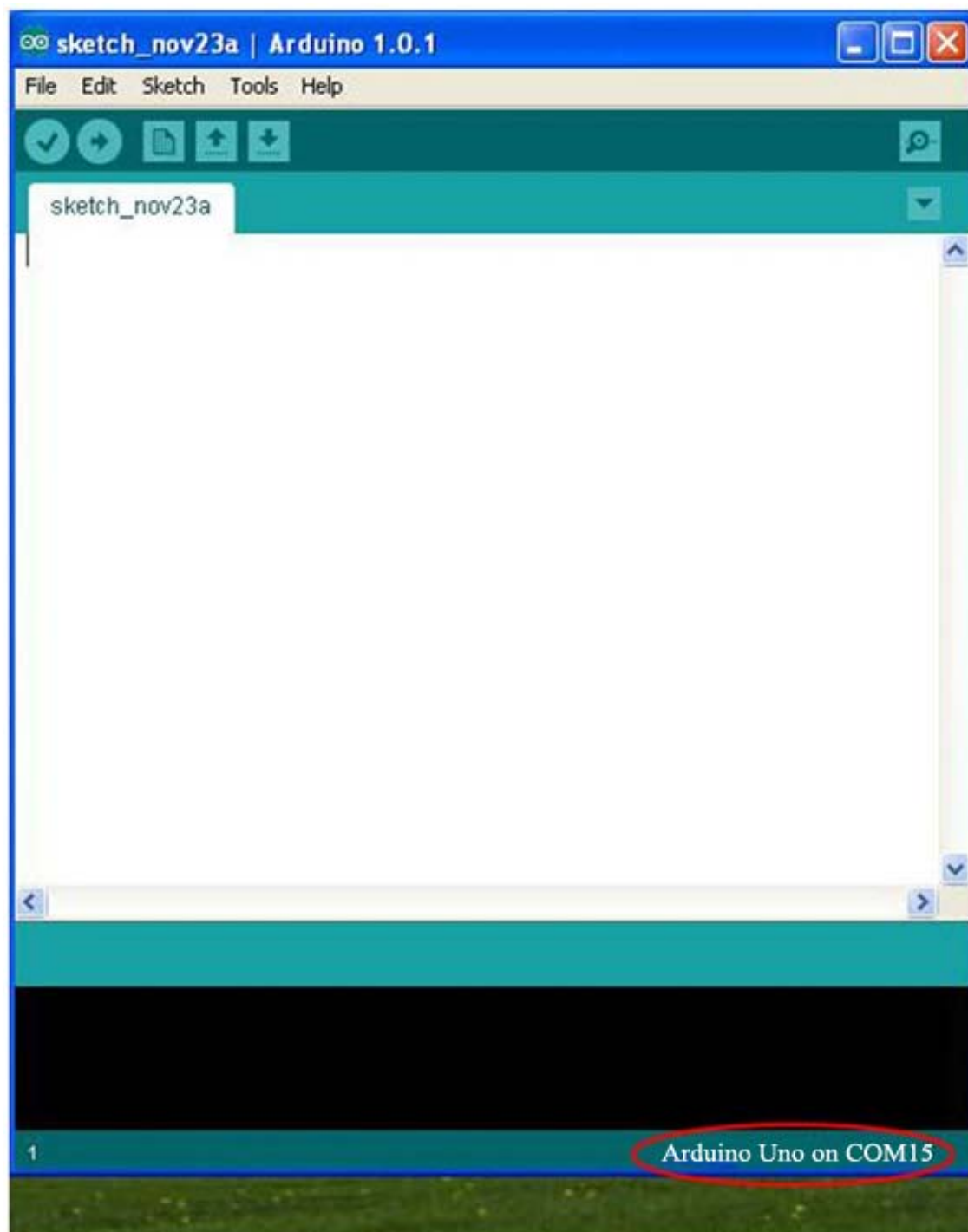


Pentru ca sistemul software sa functioneze corect va trebui sa selectati si port-ul COM asociat dispozitivului conectat la computer. Selectati din meniu **Tools -> Serial Port -> COM?**, in functie de port-ul COM asociat dispozitivului pe care l-ati conectat. Tineti cont ca acest numar poate sa difere in functie de port-ul USB pe care il utilizati.



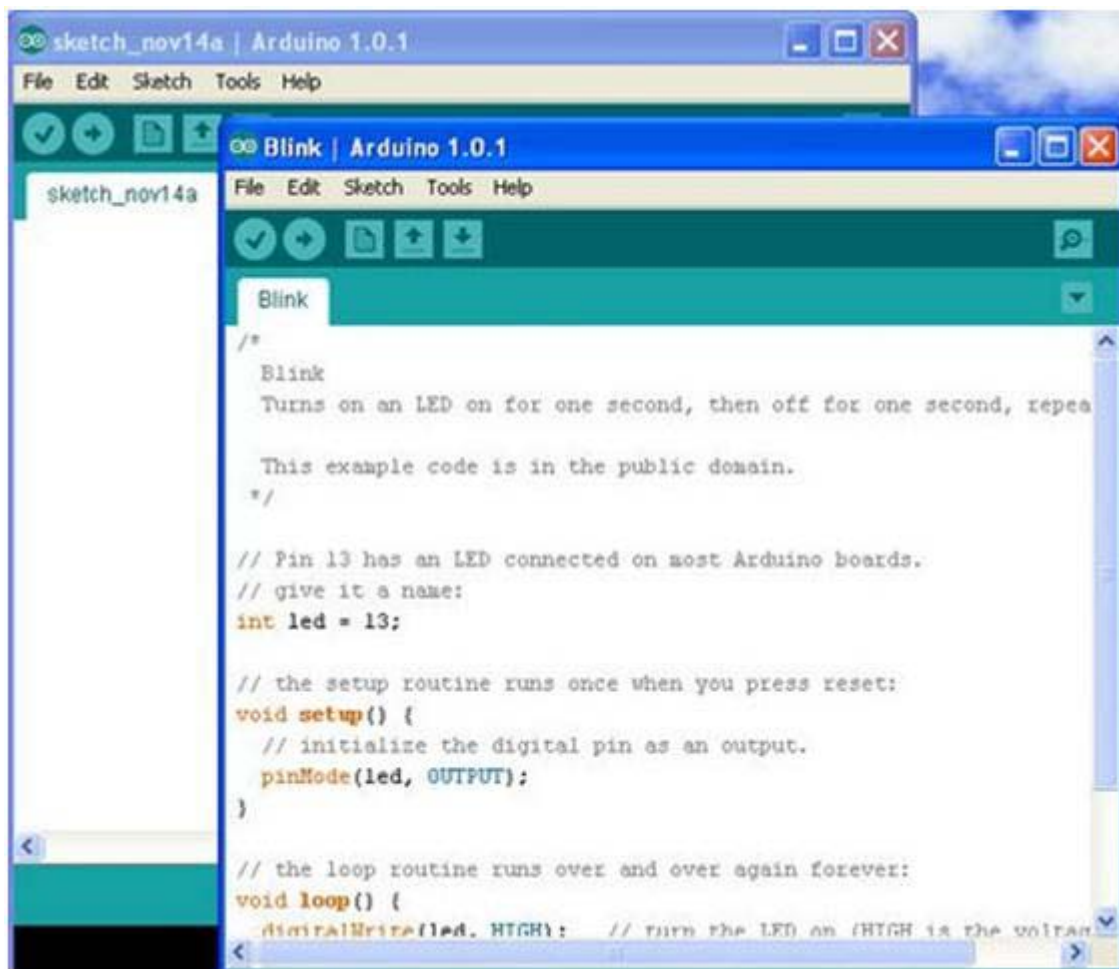


Setarile pe care tocmai le-ati efectuat vor aparea in partea din dreapta jos a sistemului Arduino IDE.





Dupa ce ati efectuat toate aceste setari sunteti pregatiti sa rulati primul experiment pe dispozitivul Arduino. Selectati din meniul Arduino IDE **File -> Examples -> Basic -> Blink**, si efectuati click pe aceasta optiune. Intr-o fereasta noua, va aparea codul sursa al programului selectat.



The screenshot shows the Arduino IDE interface. The main window is titled "Blink | Arduino 1.0.1" and displays the source code for the Blink example. The code is as follows:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repea

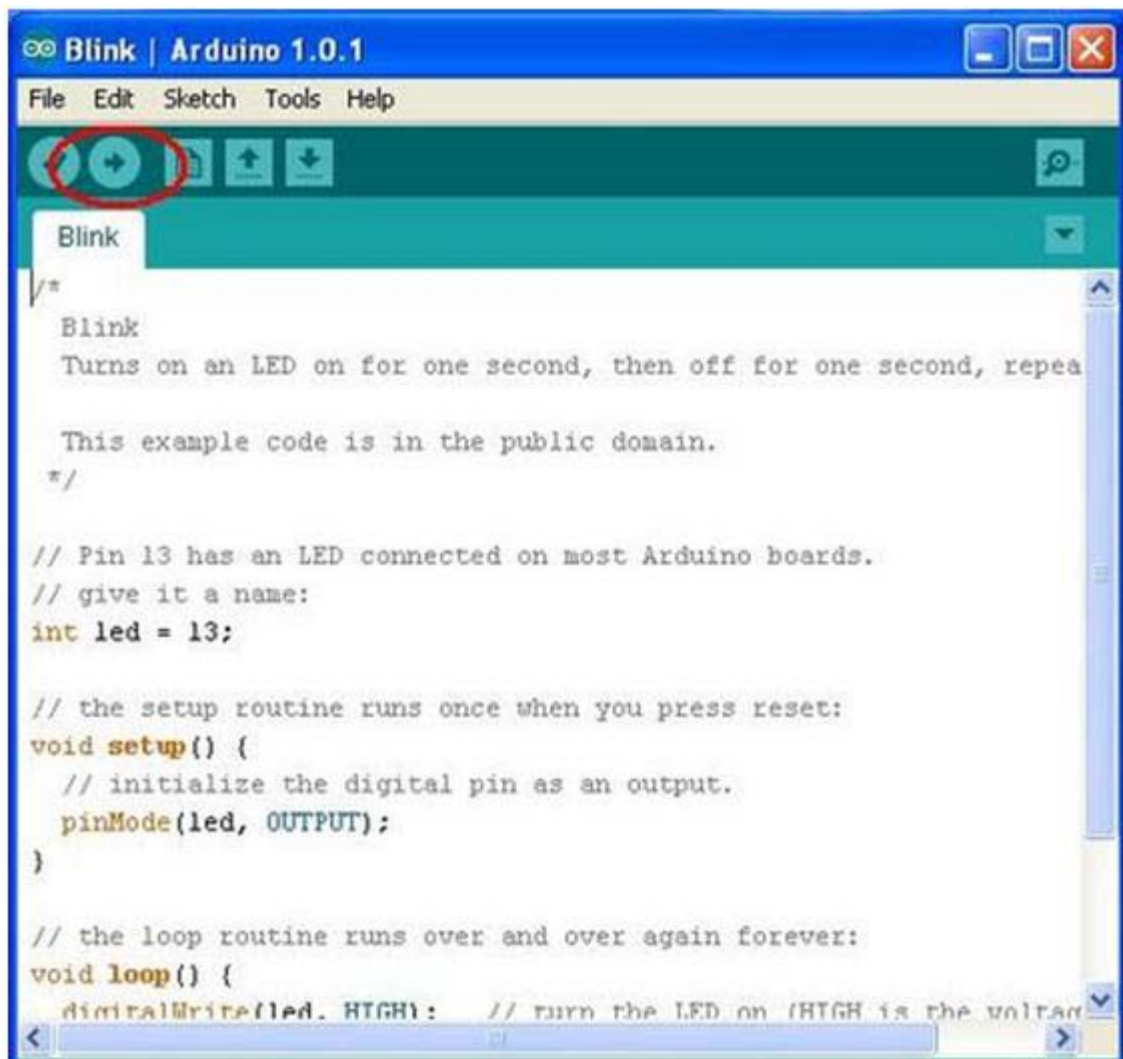
  This example code is in the public domain.
  */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

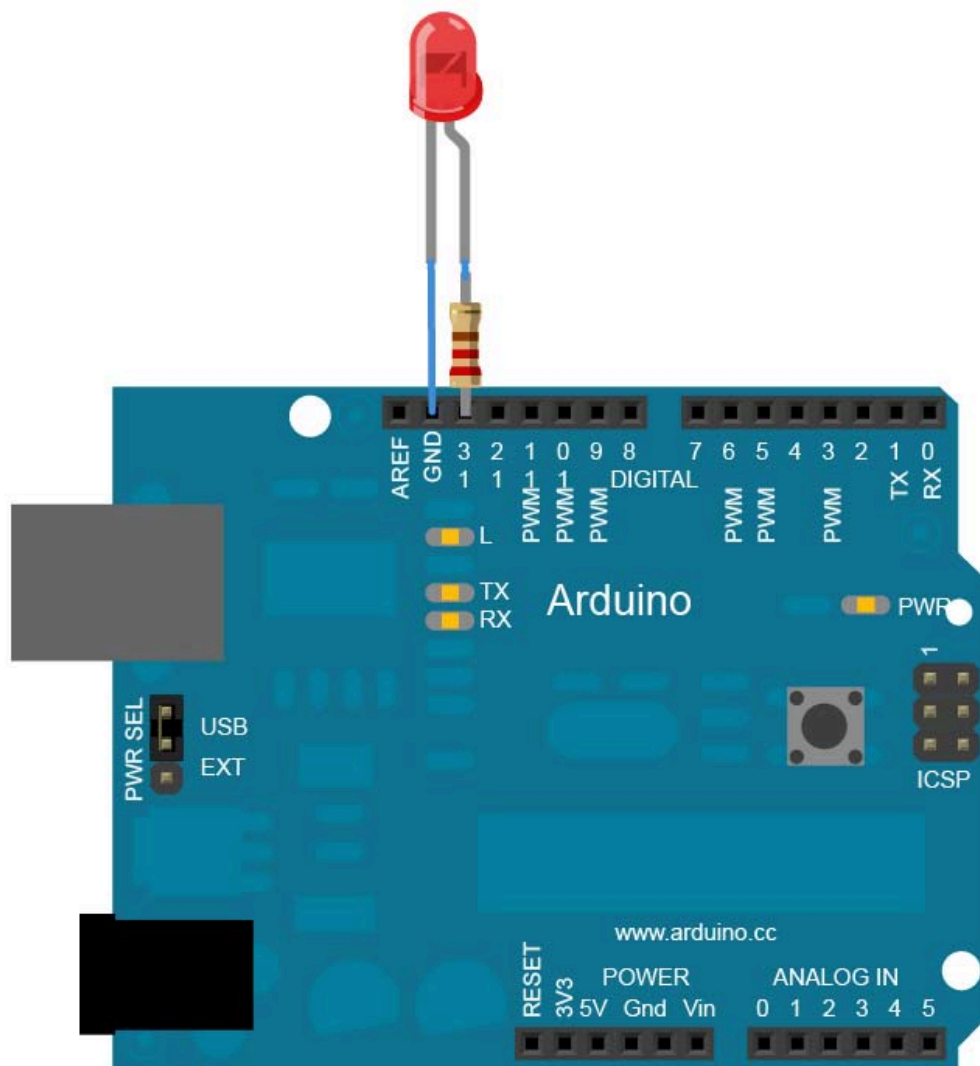
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(led, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Puteti efectua un click pe simbolul de compilare pentru a incarca programul in memoria dispozitivului Arduino conectat la computer. In conditiile in care setarile au fost efectuate corect si nu exista erori veti receptiona mesajul **Upload Done** care va informeaza ca programul a fost incarcat cu succes in memoria dispozitivului Arduino. LED-ul rosu va lumina intermitent la interval de o secunda.

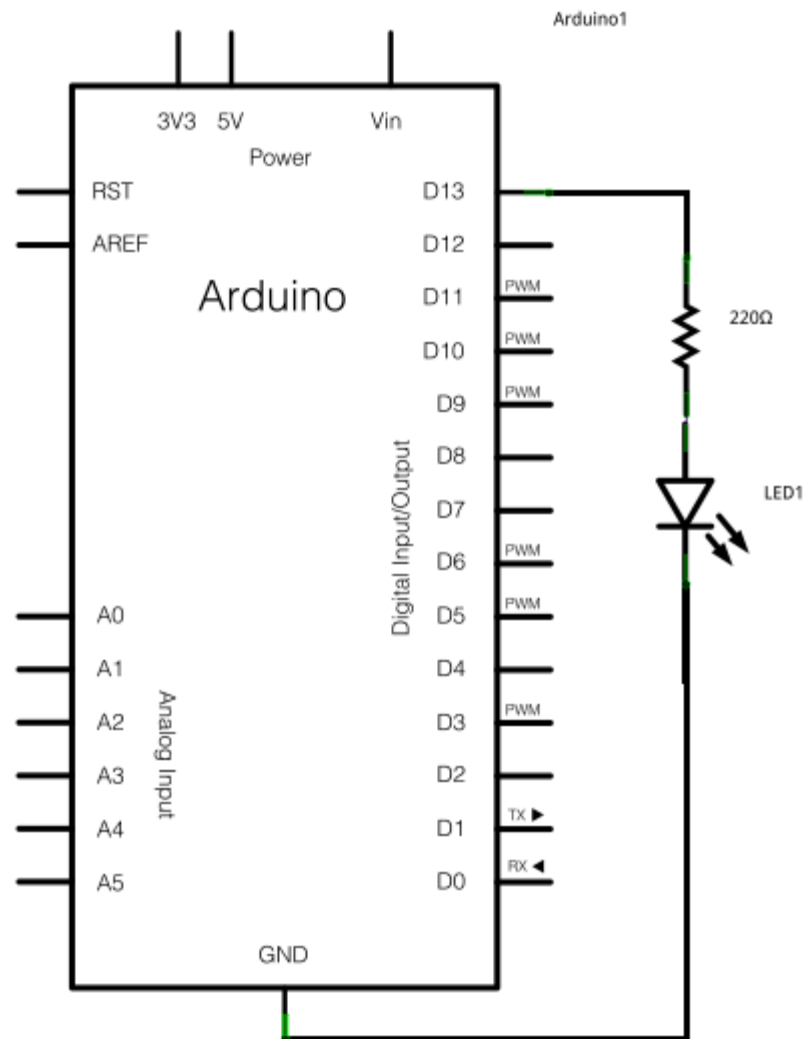


### *Circuitul asociat primului experiment*

Pentru a construi primul nostru circuit, veti atasa un rezistor de  $220\Omega$  (rosu – rosu – maro – ) pinului mai lung (pozitiv, denumit anod) al unui LED. Terminalul opus al rezistorului il veti conecta la pinul 13 al dispozitivului Arduino. Apoi, veti conecta pinul mai scurt (negativ, denumit catod) al LED-ului la cel de masa al dispozitivului Arduino (pinul GND). In conditiile in care programul a fost incarcat pe dispozitiv si acesta este coctat la computer, sau alimentat de la o sursa externa, inclusiv de la o baterie de 9V, LED-ul conectat va lumina intermitent la interval de o secunda.



### Schema electrica



### Codul sursa

În cadrul programului pentru aprinderea intermitentă a unui LED, prima etapă o reprezintă inițializarea pinului 13 ca ieșire prin intermediul codului:

- `pinMode(13, OUTPUT);`

În bucla principală, este aprins LED-ul (parametrul HIGH reprezintă activarea ieșirii):

- `digitalWrite(13, HIGH);`

Astfel este livrata o tensiune de 5V in pinul 13. Aceasta genereaza o diferenta de potential intre pinii LED-ului. Prin intermediul urmatoarei linii de cod, este stins LED-ul (parametrul LOW reprezinta dezactivarea iesirii):

- `digitalWrite(13, LOW);`

Atfel alimentare cu tensiune a pinului 13 este intrerupta si LED-ul se stinge. Intre cele doua stari, de activare si dezactivare a iesirii 13, vom introduce o intarziere prin intermediul functiei `delay()` pentru a observa cum trece dintr-o stare intr-alta LED-ul. Functia `delay(1000)` comanda dispozitivul Arduino sa intarzie urmatoare actiune pentru 1000 de milisecunde, adica o secunda.

*Iata codul sursa, cu explicatii pentru fiecare functie:*

```
/*
  Blink
  Aprinde un LED timp de o secunda, apoi il stinge pentru o secunda, in mod repetat

  */

// Pinul 13 are conectat un LED la majoritatea modelelor de Arduino
// vom denumi variabila:
int led = 13;

// rutina va rula deindata ce va fi apasata tasta reset:
void setup() {

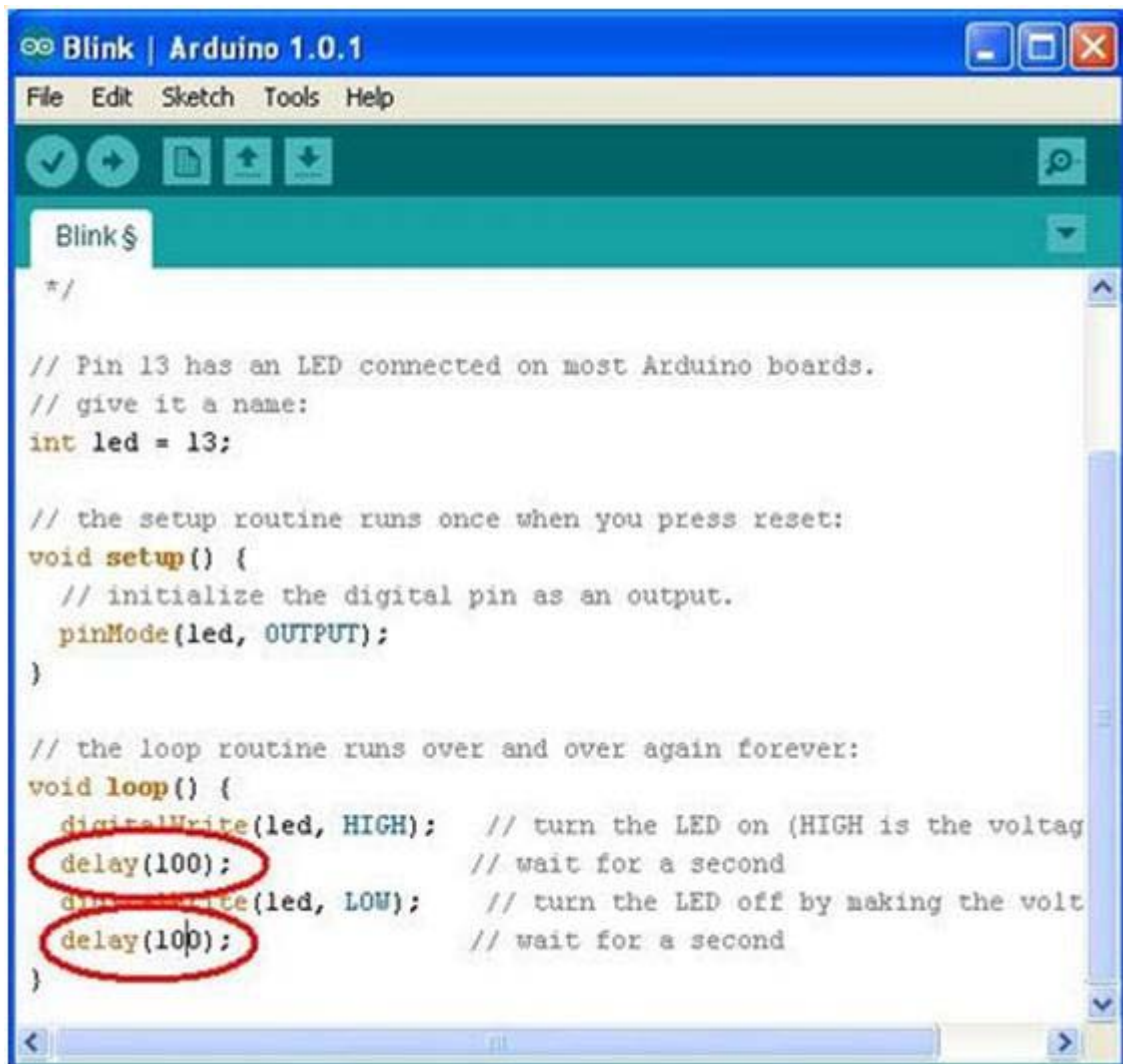
  // initializarea pinului digital ca iesire
  pinMode(led, OUTPUT);

}

// bucla va rula la infinit, pana la oprirea alimentarii:
void loop() {
  digitalWrite(led, HIGH); //aprinde LED-ul (HIGH reprezinta activarea iesirii)
  delay(1000);             // se asteapta o secunda
  digitalWrite(led, LOW);  // stinge LED-ul (LOW reprezinta dezactivarea iesirii)
  delay(1000);             // se asteapta o secunda
}
```

Dupa ce am analizat modalitatea de functionare a programului, putem face cateva modificari. Vom modifica intervalul dintre momentul in care se aprinde si cel in care se stinge LED-ul.

Pentru aceasta, modificam functia **delay(1000)** in **delay(100)**, pentru ambele intervale de intarziere. Apasati simbolul **Upload** pentru a incarca noul program in memoria dispozitivului. Este posibil sa fie necesara deconectarea si reconectarea dispozitivului la computer pentru a face modificari ale programului pe care acesta il va rula.



The screenshot shows the Arduino IDE interface with the 'Blink' sketch open. The code is as follows:

```
/*  
  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage)  
  delay(100);                // wait for a second  
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW  
  delay(100);                // wait for a second  
}
```

In the original image, the `delay(100);` lines in the `loop()` function are circled in red, indicating the modification from `1000` to `100`.

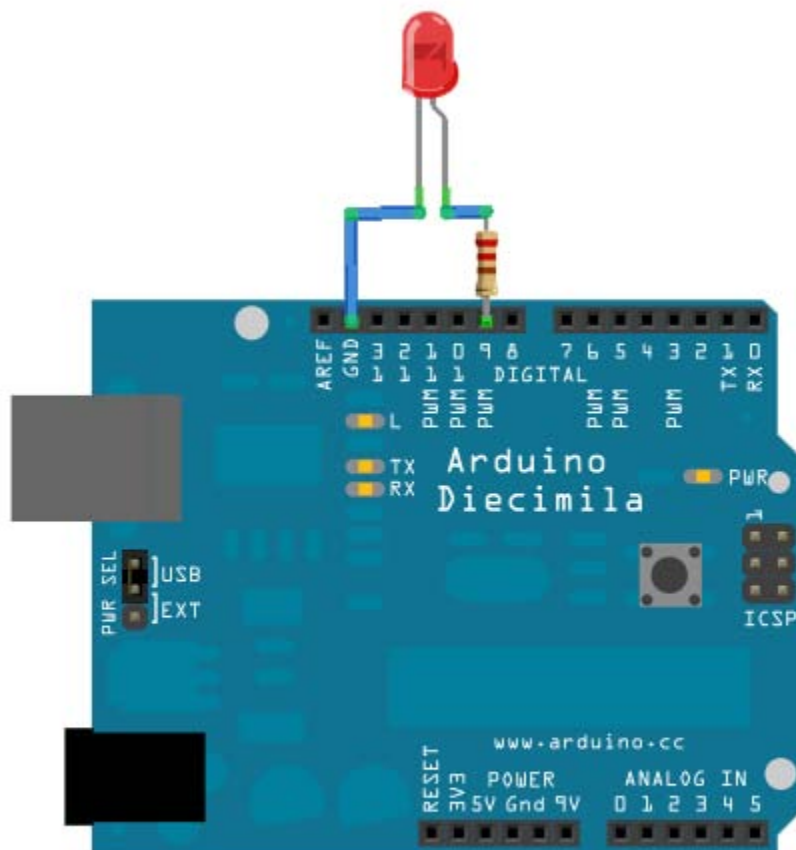
Dupa incarcarea noului program in memoria dispozitivului, veti observa ca s-a modificat rata de clipire a LED-ului.

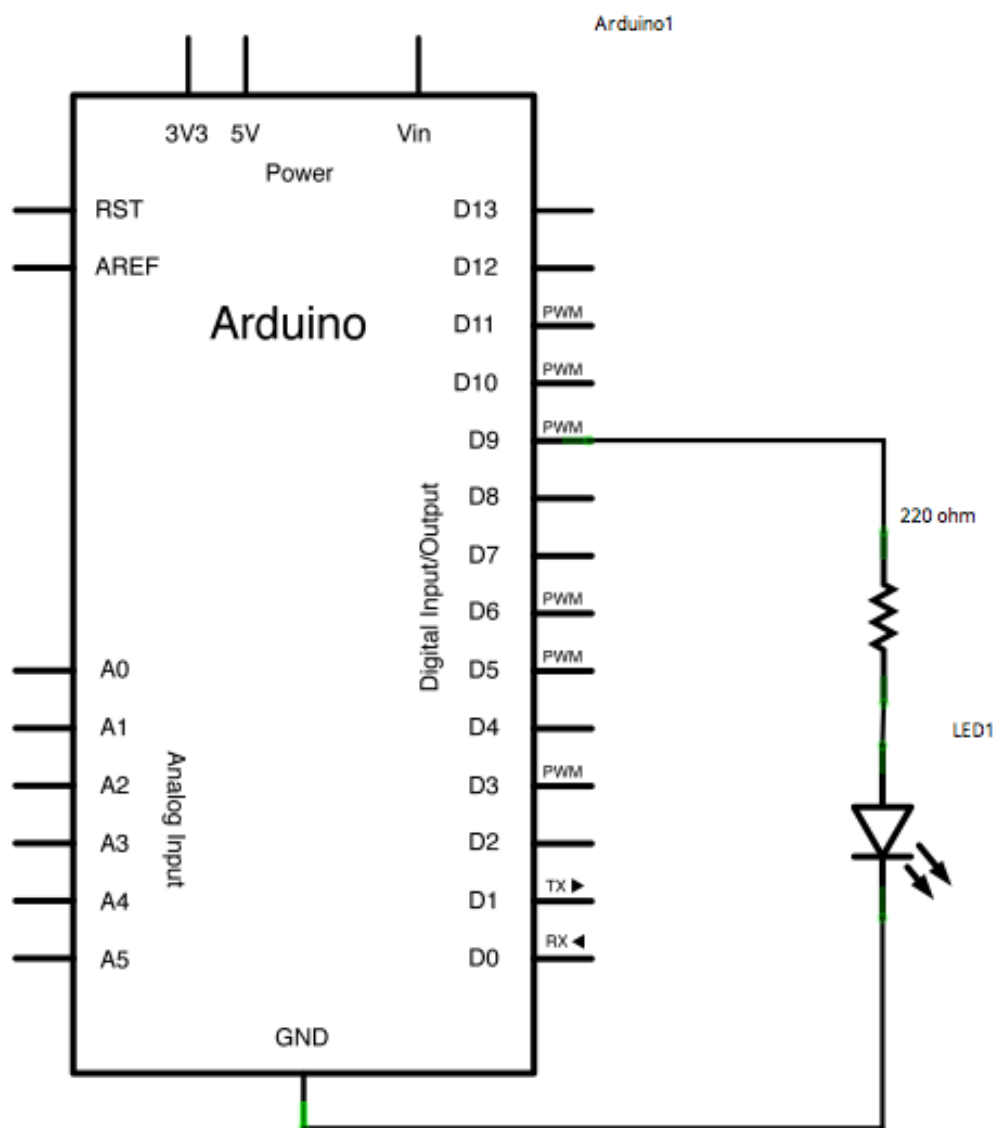
## Experimentul 2. Controlul intensitatii luminoase a unui LED.

Acest experiment demonstreaza modalitatea in care poate fi utilizat un pin digital pentru a controla intensitatea luminoasa a unui LED. In cadrul acestui experiment utilizam tehnica PWM – modulatia in frecventa a impulsurilor, pe care o vom explica ulterior, pentru a regla intensitatea luminoasa a LED-ului. Pe scurt, aceasta tehnica este similara cu actionarea unui intrerupator la intervale foarte scurte de timp, de ordinul miimilor de secunda.

### Circuitul asociat

Similar cu experimentul precedent, pentru a construi acest montaj, veti atasa un rezistor de  $220\Omega$  (rosu – rosu – maro – ) pinului mai lung (pozitiv) al unui LED din kit-ul dvs. Terminalul opus al rezistorului il veti conecta la pinul 9 al dispozitivului Arduino. Apoi, veti conecta pinul mai scurt (negativ) al LED-ului la masa dispozitivului Arduino (pinul GND).



Schema electrica



**Codul sursa**

Pentru a intelege modalitatea in care functioneaza acest program, fiecare linie de cod este insotita de cate un comentariu, delimitat de caracterele //.

/\*

*Utilizarea dispozitivului Arduino pentru a varia intensitatea luminoasa a unui LED*

*Circuitul utilizat:*

*\* LED-ul este conectat, prin intermediul unui rezistor de limitare a curentului, cu valoarea de 220  $\Omega$ , intre pinul digital 9 si masa (GND) a dispozitivului Arduino*  
\*/

```
int ledPin = 9; // LED-ul se conecteaza la pinul 9
```

```
void setup() {
```

```
// functia de stabilirea a parametrilor initiali nu va contine nici o informatie
```

```
}
```

```
void loop() {
```

```
// cresterea intensitatii luminoase, de la zero la maxim, adica 255, cu pasul din cinci in cinci
```

```
for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
```

```
// setarea valorii (in intervalul 0 ~255):
```

```
analogWrite(ledPin, fadeValue);
```

```
// asteptam 30 de milisecunde pentru a observa efectul
```

```
delay(30);
```

```
}
```

```
// scaderea intensitatii luminoase, de la maxim, adica 255, la zero, cu pasul din cinci in cinci
```

```
//observati ca, de data aceasta, daca este sa privim ca pe o progresie aritmetica, ratia este de -5
```

```
for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
```

```
// setarea valorii (in intervalul 0 ~255):
```

```
analogWrite(ledPin, fadeValue);
```

```
// asteptam 30 de milisecunde pentru a observa efectul
```

```
delay(30);
```

```
}
```

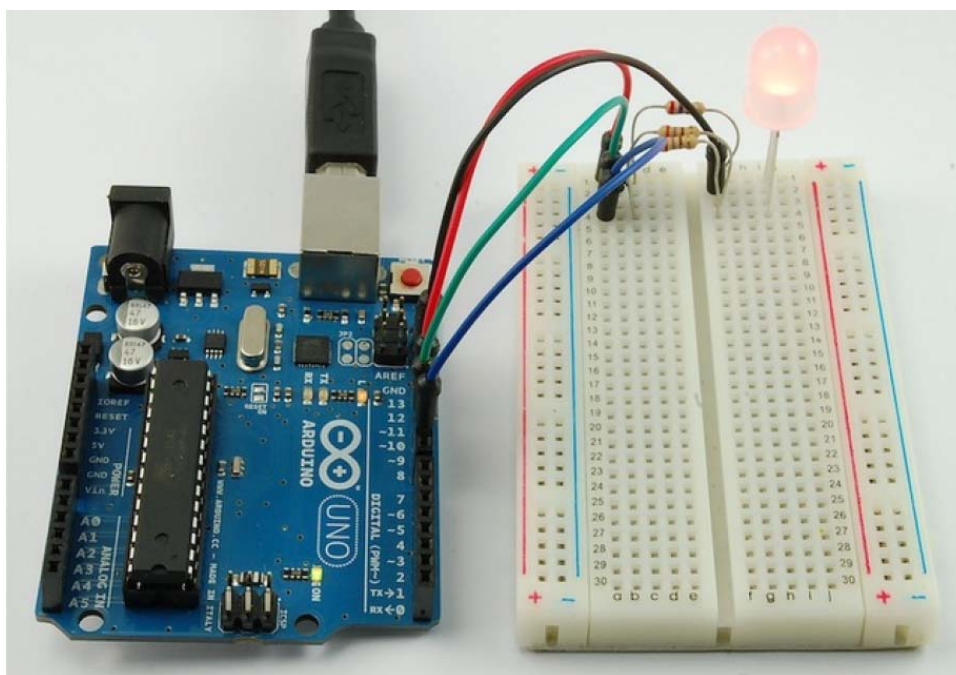
```
}
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File -> Examples -> Basic -> Fade**.

### Experimentul 3. Utilizarea LED-ului RGB

În cadrul acestei lectii, folosind cunostintele dobandite pana acum, vom invata sa utilizam un LED RGB (red – green – blue, adica rosu – verde –albastru) impreuna cu dispozitivul nostru Arduino pentru a obtine anumite culori din spectrul ce poate fi perceput de ochiul uman.

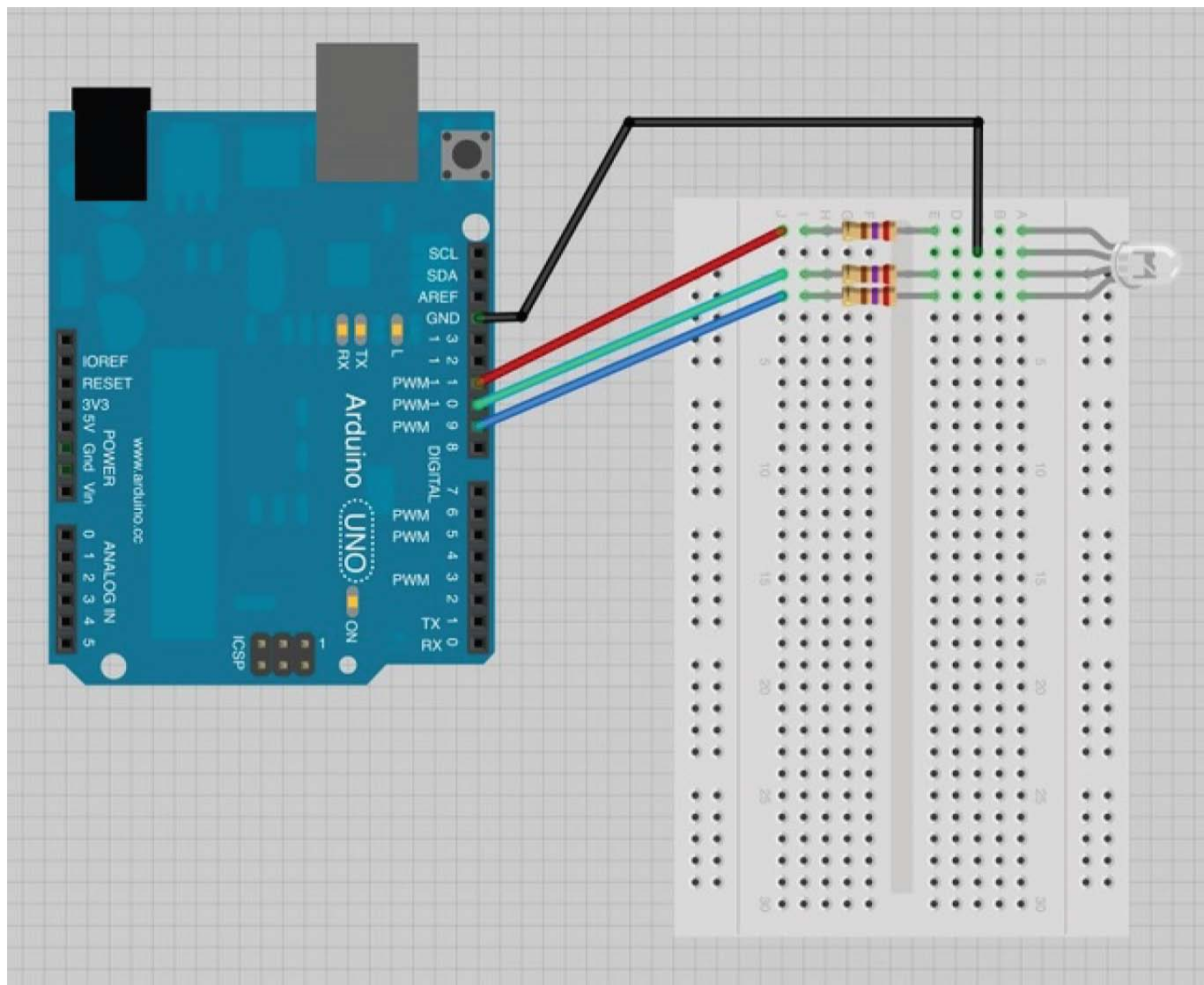
Functia pe care o vom utiliza pentru a controla culoarea luminii emise de LED-ul nostru va fi *analogWrite*. La prima vedere, LED-urile RGB arata precum unele obisnuite, insa, in capsula acestora se regasesc trei LED-uri: unul care emite lumina rosie, unul care emite lumina verde si unul care emite lumina de culoare albastra. Controland intensitatea luminoasa a fiecaruia dintre aceste LED-uri putem obtine aproximativ orice culoare dorim. Varianta dificila de a realiza acest lucru ar fi sa utilizam rezistori cu diverse valori ale rezistentei pentru a controla intensitatea luminoasa a fiecarui LED. Insa, dispozitivul Arduino ne permite, prin intermediul functiei *analogWrite*, sa controlam curentul pe care il livreaza LED-urilor.



#### Schema de montare pe placa Breadboard

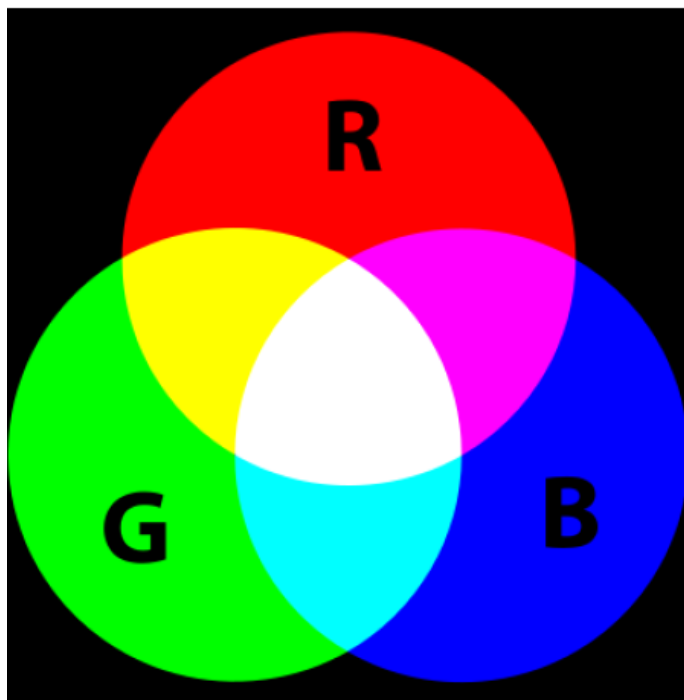
LED-ul RGB are patru terminale. Pentru fiecare dintre LED-urile interne (rosu, verde si albastru) exista cate un terminal care se conecteaza la polul pozitiv al unei surse de alimentare. Al patrulea terminal este comun pentru LED-urile interne si se conecteaza la polul negativ al sursei de alimentare. Pinul negativ comun este al doilea dinspre partea plata a LED-ului RGB. Este si cel mai lung dintre cei patru pini ai LED-ului. Acesta va fi conectat la masa.

Pentru a fi alimentat, fiecare dintre LED-urile interne are nevoie de un rezistor de limitare a curentului, de  $270\Omega$ . Cele trei terminale pozitive ale LED-ului RGB vor fi conectate la dispozitivul Arduino prin intermediul a trei rezistoare de  $270\Omega$ .



### Culori

Motivul pentru care putem mixa oricare culoare prin modificarea procentului de culoare rosie, verde si albastra este acela ca ochiul uman posedea trei tipuri de receptori ai luminii (rosu, verde si albastru). Ochiul uman percepe lumina si creierul o proceseaza analizand procentul de lumina rosie, verde si albastra si astfel rezulta o culoare din spectrul pe care il putem noi percepe. Principiul pe care il vom utiliza noi este acelasi ca in cazul televizoarelor LCD.



Daca setam aceeasi luminozitate pentru fiecare dintre cele trei LED-uri, culoare pe care o vom obtine va fi alb. Daca nu vom mai ilumina LED-ul albastru si vom ilumina LED-urile rosu si verde cu aceeasi intensitate, vom obtine culoarea galbena.

### **Codul sursa**

Schema pe care o vom utiliza va controla LED-ul RGB astfel incat acesta sa emita, pe rand culorile: rosu, verde, albastru, galben, violet si turcoaz deschis.

Pentru inceput vor fi configurati pinii de iesire pentru fiecare culoare:

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;
void setup()
{
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop()
{
  setColor(255, 0, 0); // rosu
  delay(1000);
  setColor(0, 255, 0); // verde
  delay(1000);
  setColor(0, 0, 255); // albastru
  delay(1000);
  setColor(255, 255, 0); // galben
  delay(1000);
  setColor(80, 0, 80); // violet
  delay(1000);
  setColor(0, 255, 255); // turcoaz deschis
  delay(1000);
}

void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Programul incepe prin definirea pinilor ce vor fi utilizati pentru fiecare culoare:

```
int redPin = 11;  
int greenPin = 10;  
int bluePin = 9;
```

Pasul urmator il reprezinta scriere functiei de initializare: 'setup'. Aceasta ruleaza o singura data dupa ce dispozitivul Arduino a fost resetat. In cazul de fata, aceasta functie trebuie doar sa defineasca pinii de iesire.

```
void setup()  
{  
  pinMode(redPin, OUTPUT);  
  pinMode(greenPin, OUTPUT);  
  pinMode(bluePin, OUTPUT);  
}
```

Inainte de a examina bucla de rulare vom analiza si ultima functie:

```
void setColor(int red, int green, int blue)  
{  
  analogWrite(redPin, red);  
  analogWrite(greenPin, green);  
  analogWrite(bluePin, blue);  
}
```

Aceasta are trei parametri, cate unul pentru intensitatea luminoasa a fiecarui LED intern: rosu, verde si albastru. In fiecare dintre cazuri, valoare acestor parametri va fi cuprinsa intre 0 si 255, unde 0 este utilizat pentru LED stins si 255 pentru LED aprins la intensitatea maxima. Functia va apela apoi 'analogWrite' pentru a seta intensitatea luminoasa a fiecarui LED.

Daca veti analiza bucla de rulare, veti constata ca, mai intai stabilim intensitatea luminoasa a fiecarui LED intern: rosu, verde si albastru, apoi iluminam LED-ul in culoarea dorita, facem o pauza si trecem la culoarea urmatoare.

Incercati sa adaugati alte culori si observati efectul asupra LED-ului RGB.

```
void setup()  
{  
  pinMode(redPin, OUTPUT);  
  pinMode(greenPin, OUTPUT);  
  pinMode(bluePin, OUTPUT);  
}
```

```
void setColor(int red, int green, int blue){
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
void loop(){

  setColor(255, 0, 0); // rosu
  delay(1000);
  setColor(0, 255, 0); // verde
  delay(1000);
  setColor(0, 0, 255); // albastru
  delay(1000);
  setColor(255, 255, 0); // galben
  delay(1000);
  setColor(80, 0, 80); // indigo
  delay(1000);
  setColor(0, 255, 255); // turcoaz deschis

  delay(1000);
}
```

### **Utilizarea codurilor de culoare**

Daca veti analiza schema de definire a culorilor pentru oricare sistem software veti observa ca acestea sunt reprezentate prin intermediul unui numar hexazecimal. De exemplu, culoarea rosie este reprezentata de numarul #FF0000. Puteti gasi o astfel de *harta* a culorilor pe site-ul:

<http://www.devguru.com/features/colors>

Cele sase cifre din cadrul acestor numere sunt, de fapt, trei perechi de numere; prima pereche reprezinta concentratia culorii rosii, a doua concentratiei culorii verzi si a treia concentratia culorii albastre. Culoarea rosie este reprezentata de numarul #FF0000 deoarece intensitatea culorii rosii este maxima (FF este 255 in sistem zecimal), iar verdele si albastrul nu sunt prezente, adica au valoarea minima.

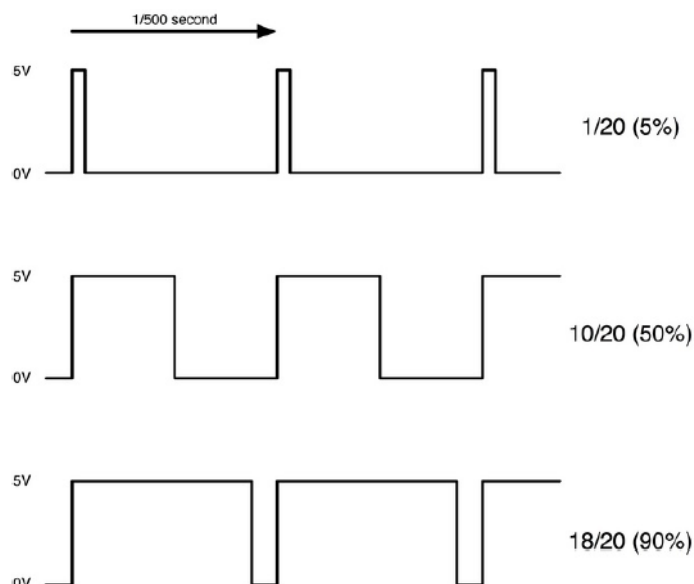
Sa incercam sa reprezentam culoare indigo (#4B0082). Concentratiile pentru rosu, verde si albastru vor fi (in sistem hexazecimal) 4B, 00 si 82. Vom insera aceste valori in functia 'setColor' astfel:

```
setColor(0x4B, 0x0, 0x82); // indigo
```

### PWM – modulatia in frecventa a impulsurilor

PWM este una dintre modalitatile de a controla puterea emisa de o sursa. In situatia de fata, o utilizam pentru a controla luminozitatea fiecarui LED intern al LED-ului RGB.

Diagrama de mai jos, prezinta semnalul emis de dispozitivul Arduino pe unul dintre pinii PWM.



Practic, o data la fiecare 1/500 dintr-o secunda, dispozitivul va produce un impuls energetic. Durata acestui impuls este controlata prin intermediul functiei

'analogWrite'. Drept urmare, functia 'analogWrite(0)' nu va produce nici un fel de impuls, iar 'analogWrite(255)' va genera energie permanent. Daca iesirea este setata sa emita un impuls cu durata de 5% dintr-o perioada, atunci dispozitivul pe care il alimentam va primi doar 5% din puterea maxima posibila. Totusi, noi nu putem observa faptul ca LED-ul se aprinde si se stinge ci doar ca luminozitatea acestuia se modifica.

1. Construiti montajul conform schemei care v-a fost prezentata. Copiati codul sursa intr-o schita noua sistemului Arduino IDE sau deschideti fisierul exemplu3.ino din folderul Experimente. Compilati si incarcati fisierul in dispozitivul Arduino.
2. Accesati site-ul <http://www.devguru.com/features/colors> si definiti noi culori pe care sa le afiseze LED-ul dvs. Verificati rezultatele obtinute.

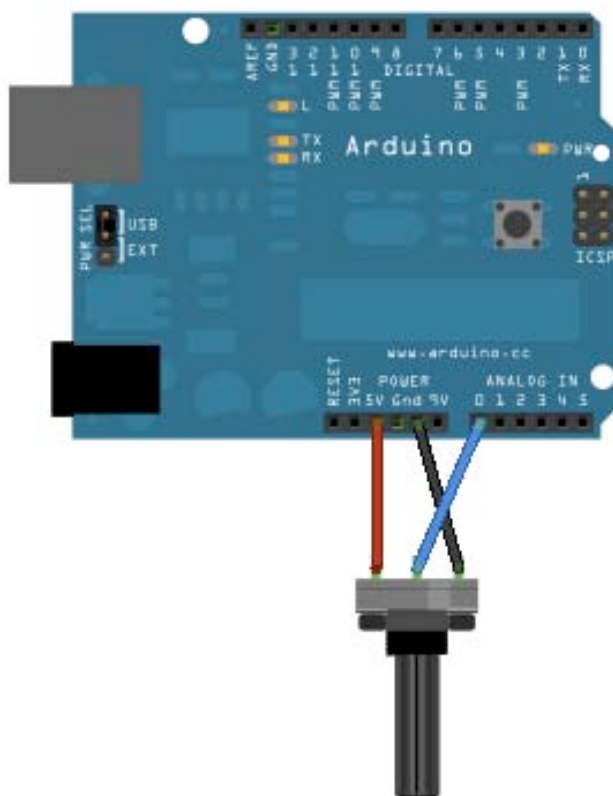


## Experimentul 4. Utilizarea intrarilor analogice. Conectarea unui potentiometru la dispozitivul Arduino.

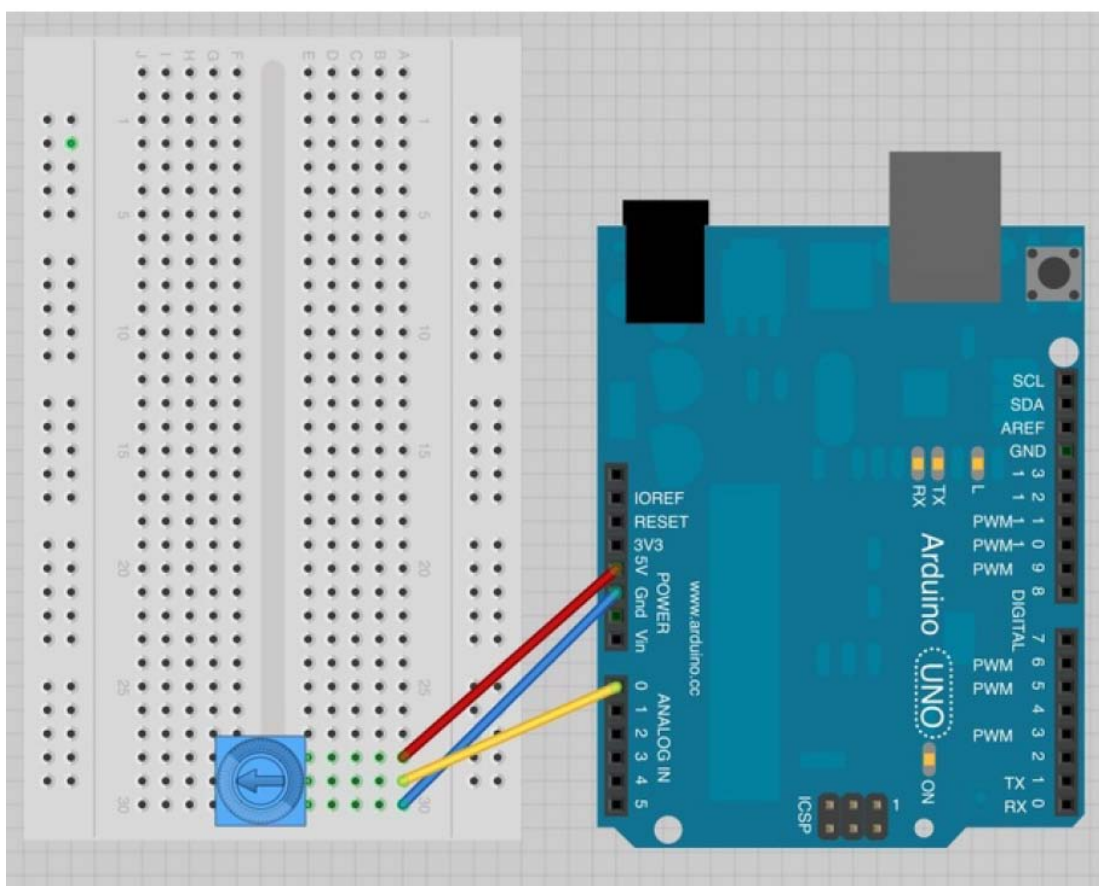
Acest exemplu prezinta modalitatea in care poate fi preluata si interpretata o marime fizica. Un potentiometru va fi conectat la intrarea analogica a dispozitivului Arduino iar valoarea rezistentei rezultate prin rotirea axului va fi convertita, utilizand o anumita logica, intr-un numar. Asa cum a fost prezentat anterior, un potentiometru este un dispozitiv mecanic ce poate varia, prin rotirea axului, rezistenta masurata la bornele sale. Daca inseriem potentiometrul cu o sursa de tensiune si il conectam la intrarea analogica a dispozitivului Arduino putem masura rezistenta instantanee a potentiometrului si o putem converti intr-o valoare analogica. In cadrul acestui experiment vom monitoriza starea potentiometrului dupa stabilirea unei comunicatii seriale intre computer si dispozitivul Arduino.

### Circuitul asociat

Conectati cele trei terminale ale potentiometrului de  $20k\Omega$  din kitul dvs. la dispozitivul Arduino. Primul (una dintre extremitati) va fi conectat la masa Arduino, adica GND. Al treilea terminal (cealalta extremitate) se va conecta la sursa de 5V, iar cursorul (terminalul central) la intrarea analogica 0.



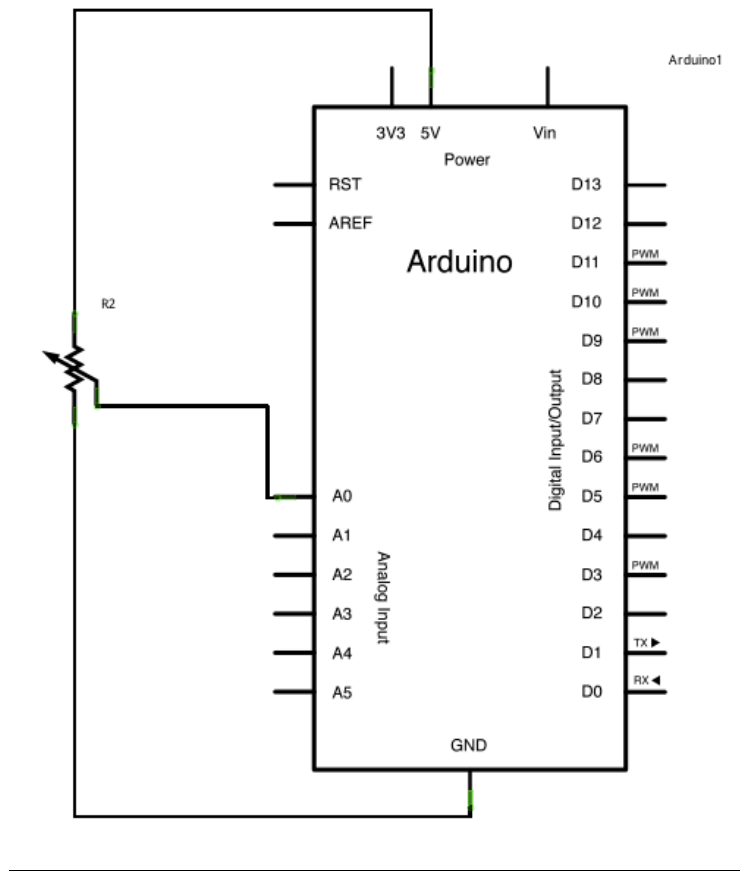
Puteti monta potentiometru direct pe placa breadboard sau puteti utiliza un rezistor semireglabil.



Prin rotirea axului potentiometrului, rezistenta fata de oricare dintre extremitati se va modifica. Aceasta va conduce la schimbarea tensiunii masurata fata de pinul central. Atunci cand rezistenta dintre cursor si extremitatea conectata la +5V este apropiata de zero (iar rezistenta fata de cealalta extremitate este apropiata de 20k $\Omega$ ), tensiunea masurata fata de cursor este de 5V. In situatia in care cursorul se va roti catre maxim in partea opusa, tensiunea masurata fata de cursor este 0V, adica acesta este conectat la masa. Aceasta tensiune este valoarea analogica pe care o citim ca intrare.

Dispozitivul Arduino are integrat un circuit denumit convertor analog-digital care poate citi aceasta tensiune si o poate converti intr-o valoare numerica in intervalul 0 ~ 1023. Atunci cand axul este rotit pana la capat intr-o directie si tensiunea masurata este 0V, valoarea de intrare este 0. Daca axul va fi rotit pana la capat in directia opusa, tensiunea masurata va fi 5V si valoare de intrare, 1023. Functia `analogRead()` are ca rezultat un numar intre 0 si 1023, proportional cu tensiunea aplicata cursorului.

### Schema electrica



### Codul sursa

Programul va defini in cadrul functiei *setup* parametrii pentru inceperea comunicatiei seriale, cu o viteza de transfer a datelor de 9600 biti pe secunda:

```
Serial.begin(9600);
```

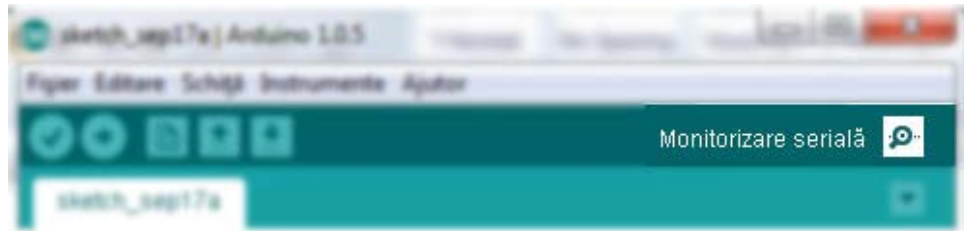
Apoi, in interiorul buclei principale, vom defini variabila care stocheaza informatia despre rezistenta potentiometrului (un numar in intervalul 0 ~1023):

```
int sensorValue = analogRead(A0);
```

In cele din urma, trebuie sa afisam informatia ca numar in sistemul zecimal. Afisarea se va realiza prin intermediul comenzii `Serial.println()`:

```
Serial.println(sensorValue, DEC)
```

Dupa ce am realizat montajul si am transferat programul in dispozitivul Arduino, deschidem *Serial Monitor* (accesandu-l direct precum in imagine sau din meniu **Instruments** -> **Serial Monitor**) si vom observa un sir de numere care se modifica pe masura ce rotim axul potentiometrului.



```
/*  
  Conectarea unui potentiometru la intrarea analogica a dispozitivului Arduino  
*/  
  
//setarea incepe de indata ce veti apasa butonul de reset  
void setup() {  
  
  //initializarea comunicatiei seriale la 9600 de biti pe secunda  
  Serial.begin(9600);  
}  
  
// bucla ruleaza la infinit  
void loop() {  
  
  //citim valoarea pinului analogic 0:  
  int sensorValue = analogRead(A0);  
  
  //afisam valoarea citita:  
  Serial.println(sensorValue);  
  
  delay(1);    //o mica intarziere pentru o citire cat mai corecta  
}
```

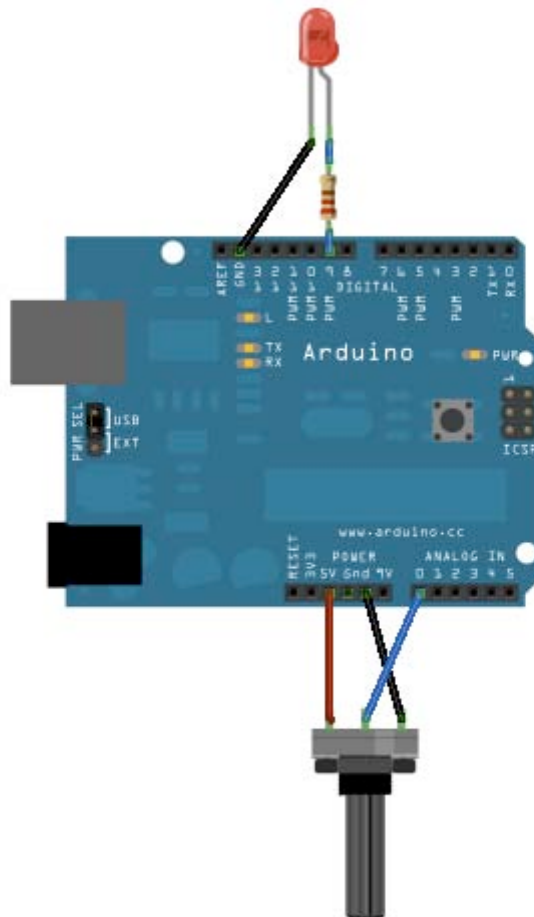
1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File** -> **Examples** -> **Basics** -> **ReadAnalogVoltage**.

## Experimentul 5. Utilizarea unui potentiometru rotativ liniar pentru a controla intensitatea luminoasa.

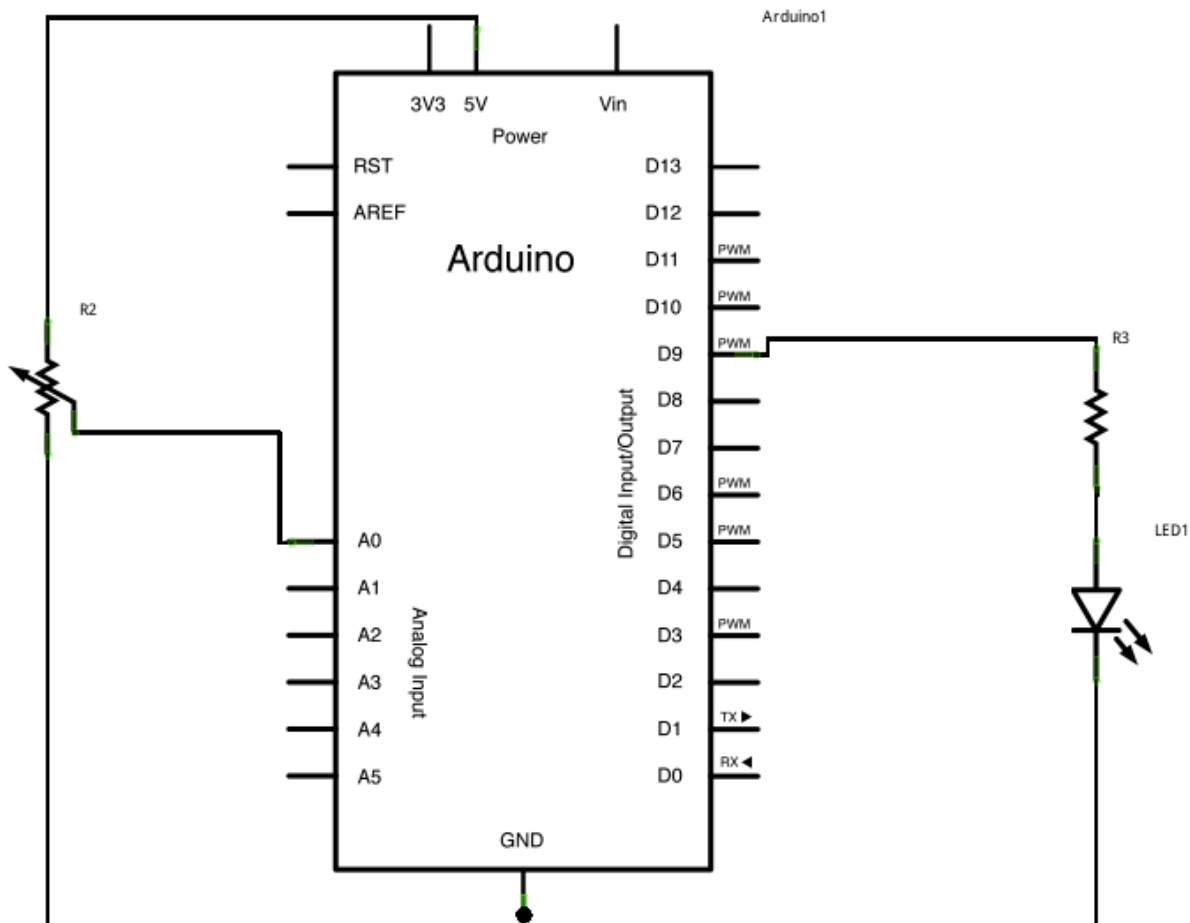
Acest exemplu extinde experimentul precedent, prezentand modalitatea prin care putem citi valoarea analogica a unui senzor, converti aceasta valoare in intervalul 0 ~255 si utiliza valoarea obtinuta pentru a controla intensitatea luminoasa a unui LED conectat la montajul nostru.

### Circuitul asociat

Schema de conectare a potentiometrului va fi la fel cu aceea din exemplul precedent. In plus, vom atasa un rezistor de 220 $\Omega$  (rosu – rosu – maro –auriu) pinului mai lung (pozitiv) al unui LED din kit-ul dvs. Terminalul opus al rezistorului il veti conecta la pinul 9 al dispozitivului Arduino. Apoi, veti conecta pinul mai scurt (negativ) al LED-ului la masa dispozitivului Arduino (pinul GND).



### Schema electrica



### Codul sursa

Dupa definirea celor doi pini (pinul analog 0 pentru potentiometru si digital 9 pentru LED) le vom asocia doua variabile *sensorValue* si *outputValue*, si vom stabili comunicatia seriala, la fel ca in exemplul precedent.

Apoi, in bucla principala, functia *loop*, variabila *sensorValue* este utilizata pentru a stoca valoarea analogica rezultata prin rotirea axului potentiometrului. Intrucat dispozitivul Arduino are o rezolutie intre 0 si 1023 si rezolutia functiei *analogWrite* este intre 0 si 255, va trebui sa scalam datele receptionate prin citirea valorii analogice inainte de a o utiliza pentru modificarea intensitatii luminoase a LED-ului.

Pentru a scala aceasta valoare,utilizam functia *map()* astfel:

```
outputValue = map(sensorValue, 0, 1023, 0, 255);
```

Varibila *outputValue* este numarul atribuit valorii analogice preluate de dispozitivul Arduino de la potentiometru. Functia *map()* utilizeaza cinci argument: valoarea care urmeaza a fi convertita, valoare minima si valoarea maxima a datelor ce urmeaza a fi convertite, valoare minima si maxima a datelor ce urmeaza a fi scalate. In cazul de fata, datele obtinute de la senzor sunt convertite din intervalul 0 ~1023 in intervalul 0 ~ 255.

Valoare obtinuta prin intermediul functiei *map()* este transmisa prin intermediul variabilei *analogOutPin* dispozitivului Arduino pentru a mari sau diminua intensitatea luminoasa a LED-ului. Apoi, atat valoarea bruta a intrarii analogice, cat si cea procesata sunt transmise pentru a fi afisate prin intermediul *Serial Monitor*.

```
/*
```

```
  Utilizarea unui potentiometru rotativ liniar pentru a varia intensitatea luminoasa a unui LED
  Programul citeste intrarea analogica de la un potentiometru, o converteste intr-o valoare in
  intervalul 0 ~255 si apoi utilizeaza aceasta valoare pentru a controla intensitatea luminoasa a
  LED-ului din montaj.
```

```
  Circuitul contine:
```

```
  * un potentiometru de 20k conectat la pinul analogic 0
```

```
  * un LED conectat la pinul digital 9 si masa – GND.
```

```
*/
```

```
const int analogInPin = A0; // pinul de intrare analogica la care este conectat potentiometrul
```

```
const int analogOutPin = 9; // pinul de iesire digitala la care este conectat LED-ul
```

```
int sensorValue = 0;    // valoarea citita de la potentiometru
```

```
int outputValue = 0;    // valoare iesire PWM
```

```
void setup() {
```

```
  // initializarea procesului de comunicare la o viteza de transfer de 9600 bps:
```

```
  Serial.begin(9600);
```

```
}
```

```
void loop() {
```

```
  // citirea valorii analogice:
```

```
  sensorValue = analogRead(analogInPin);
```

```
  // conversia valorii analogice in intervalul 0~255:
```

```
  outputValue = map(sensorValue, 0, 1023, 0, 255);
```

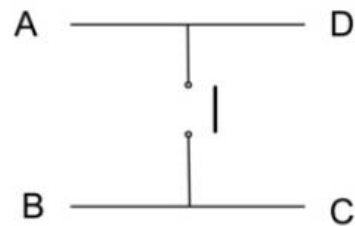
```
// modificare valorii de iesire:  
analogWrite(analogOutPin, outputValue);  
  
// afisarea rezultatelor prin intermediul programului Serial Monitor:  
Serial.print("sensor = " );  
Serial.print(sensorValue);  
Serial.print("\t output = ");  
Serial.println(outputValue);  
  
// o perioada de asteptare de 2 milisecunde pentru stabilizarea iesirii  
delay(2);  
}
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File -> Examples -> Analog -> AnalogInOutSerial**.



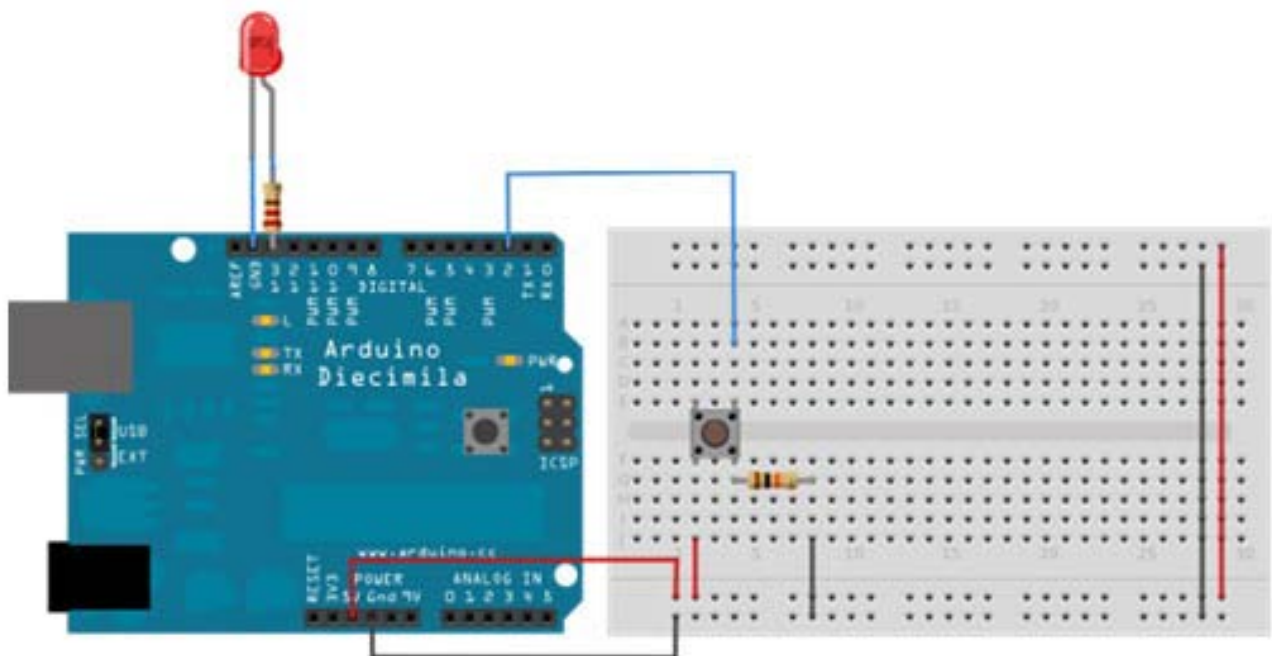
## Experimentul 6. Utilizarea comutatoarelor cu revenire.

Comutatoarele cu revenire au un principiu de functionare foarte simplu. Atunci cand sunt apasate, ele conecteaza doua contacte prin care trece curentul electric, adica, practic, fac legatura intre doua puncte din circuit. Butoanele pe care le utilizam in cadrul acestui experiment au patru pini. De fapt, sunt doar doua conexiuni electrice, intrucat in interiorul capsulei comutatorului pinii B si C sunt conectati intre ei si acelasi lucru este valabil pentru pinii A si D.



In acest exemplu, vom utiliza comutatorul cu revenire pentru a aprinde LED-ul conectat la pinul digital 13 atunci cand se apasa comutatorul.

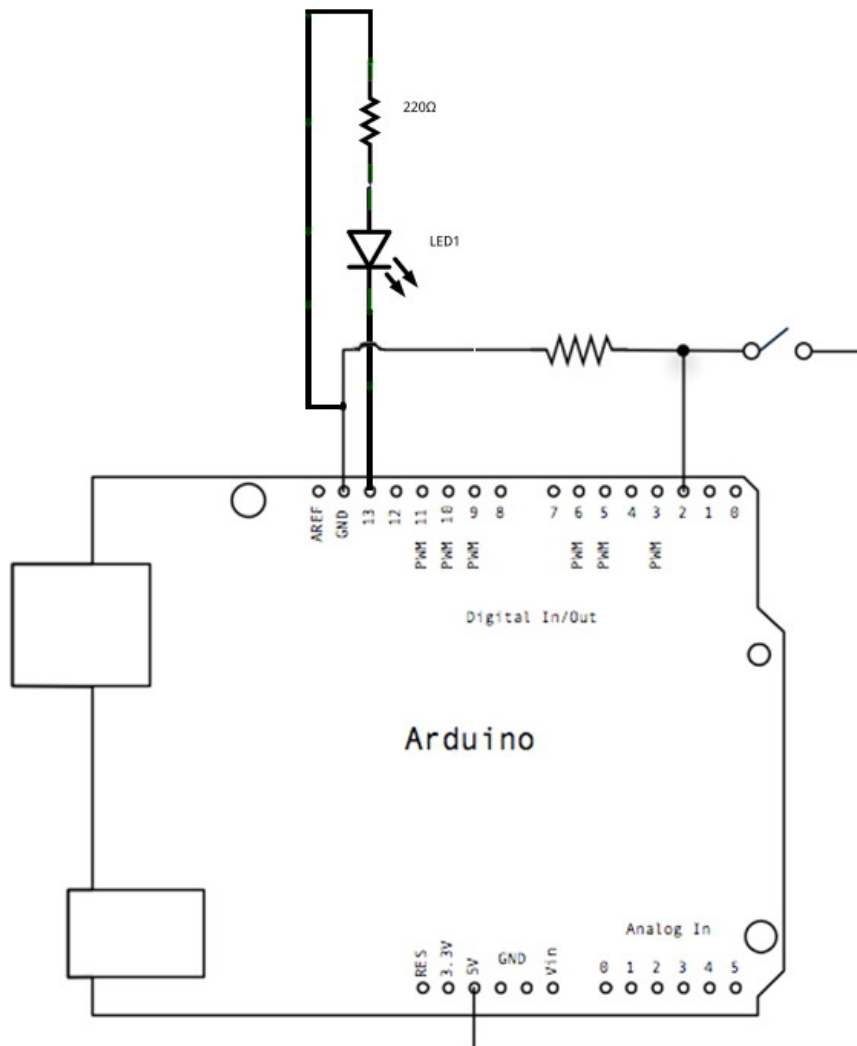
### Circuitul asociat



Veti conecta trei fire la dispozitivul Arduino. Primele doua, rosu si negru din imagine, conecteaza cele doua laturi verticale ale placii breadboard pentru a oferi acces la masa (GND) si la +5V. Al treilea fir va conecta pinul digital 2 al dispozitivului Arduino la unul dintre terminalele comutatorului. Acelasi terminal al comutatorului este conectat prin intermediul unui rezistor de limitare de  $10k\Omega$  la masa. Celalalt pin al comutatorului cu revenire se conecteaza la sursa de 5V.

Atunci cand comutatorul este deschis (neaprasat) nu exista nici o conexiune intre cei doi pini ai sai, astfel ca pinul este conectat la masa (prin intermediul rezistorului de limitare) si citim valoarea LOW, echivalent cu o conectare la masa. Cand comutatorul este inchis (apasat) exista conexiune intre cei doi pini ai sai si acestia sunt conectati la sursa de 5V, deci vom citi valoarea HIGH.

### Schema electrica



**Codul sursa**

/\*

Utilizarea comutatorului cu revenire

Montajul aprinde si stinge un LED, conectat la pinul 13, in functie de stare butonului cu revenire conectat la pinul 2

Circuitul este compus din:

\* LED conectat intre pinul 13 si masa

\* comutatorul cu revenire conectat la pinul 2 si +5V

\* rezistor de limitare de 10K $\Omega$  conectat de la pinul 2 la masa

\*/

```
const int buttonPin = 2;    // pinul de intrare asociat comutatorului cu revenire
```

```
const int ledPin = 13;     // pinul de intrare LED-ului
```

```
int buttonState = 0;       // variabila pentru citirea starii comutatorului
```

```
void setup() {
```

```
    // initializarea pinului de LED ca iesire:
```

```
    pinMode(ledPin, OUTPUT);
```

```
    // initializarea comutatorului cu revenire ca intrare:
```

```
    pinMode(buttonPin, INPUT);
```

```
}
```

```
void loop(){
```

```
    // citirea starii comutatorului cu revenire:
```

```
    buttonState = digitalRead(buttonPin);
```

```
    // verificam daca butonul a fost apasat
```

```
    // daca este apasat, atunci buttonState este HIGH:
```

```
    if (buttonState == HIGH) {
```

```
        // iluminam LED-ul:
```

```
        digitalWrite(ledPin, HIGH);
```

```
    }
```

```
else {  
  
    // oprim iluminarea LED-ului:  
    digitalWrite(ledPin, LOW);  
}  
}
```

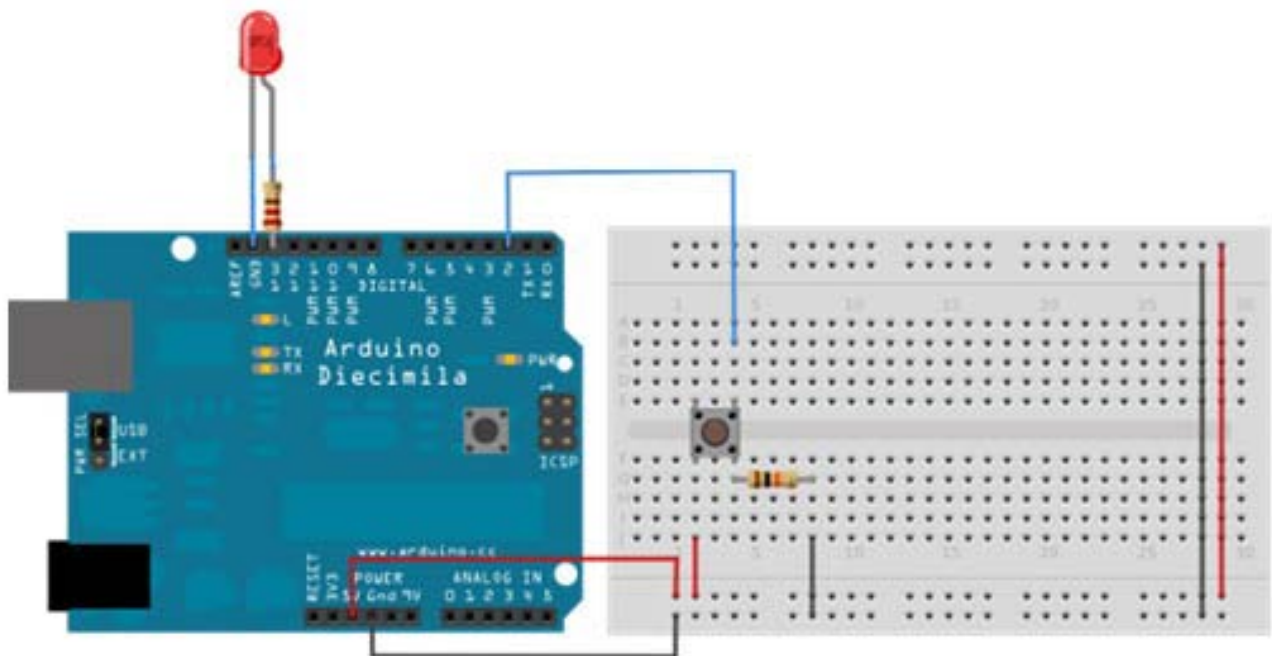
1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File -> Examples -> Digital -> Button**.

## Experimentul 7. Detectarea schimbarii starii comutatorului cu revenire.

Utilizand montajul pe care l-am realizat anterior, vom modifica programul astfel incat starea LED-ului sa depinda de numarul de apasari ale butonului. Pentru a realiza aceasta, trebuie sa determinam exact momentul in care comutatorul isi schimba starea din apasat in neapasat si invers.

### Circuitul asociat

Circuitul este acelasi de la exemplul precedent, deci veti reutiliza montajul realizat anterior.



Programul citește permanent starea comutatorului cu revenire, apoi compară starea curentă cu starea precedentă pentru a identifica dacă a apărut o modificare a acesteia. Dacă starea curentă diferă de cea precedentă și starea butonului este HIGH, atunci înseamnă că starea s-a schimbat din închis în deschis și este incrementat (adică se adună o unitate) un contor.

Programul verifică starea contorului și, dacă acesta este multiplu de patru, aprinde LED-ul. Altfel îl păstrează stins. Deci, la un multiplu de patru apăsări ale butonului LED-ul va începe să lumineze și se va menține în această stare până la o nouă apăsare a butonului cu revenire.

### **Codul sursa**

/\*

Detectarea schimbarii starii unui comutator cu revenire

De cele mai multe ori nu este necesar sa cunoastem permanent starea unei intrari digitale, dar este posibil sa avem nevoie sa cunoastem momentul in care aceasta isi schimba starea.

De exemplu, s-ar putea sa dorim sa cunoastem momentul in care un comutator isi schimba starea din inchis in deschis sau invers.

Acest exemplu prezinta modalitatea de a detecta momentul in care un comutator cu revenire isi schimba starea din inchis in deschis si invers.

Circuitul este compus din:

- \* LED conectat intre pinul 13 si masa

- \* comutatorul cu revenire conectat la pinul 2 si +5V

- \* rezistor de limitare de 10K $\Omega$  conectat de la pinul 2 la masa

\*/

```
const int buttonPin = 2;    // pinul de intrare asociat comutatorului cu revenire
```

```
const int ledPin = 13;     // pinul de intrare LED-ului
```

```
int buttonPushCounter = 0; // contor pentru numarul de apasari ale comutatorului
```

```
int buttonState = 0;        // starea curenta a comutatorului
```

```
int lastButtonState = 0;    // starea precedentaa comutatorului
```

```
void setup() {
```

```
    // initializarea pinului asociat comutatorului ca intrare:
```

```
    pinMode(buttonPin, INPUT);
```

```
    // initializarea pinului asociat LED-ului ca iesire:
```

```
    pinMode(ledPin, OUTPUT);
```

```
    // initializarea procesului de comunicatie seriala:
```

```
    Serial.begin(9600);
```

```
}
```

```
void loop() {
```

```
    // citirea starii comutatorului:
```

```
    buttonState = digitalRead(buttonPin);
```

```
    // este comparata starea curenta cu starea precedenta
```

```
    if (buttonState != lastButtonState) {
```

```
// daca starea s-a schimbat, se incrementeaza contorul
if (buttonState == HIGH) {

    // daca starea curent este HIGH atunci comutatorul si-a modificat starea din oprit in pornit
    buttonPushCounter++;
    Serial.println("on");
    Serial.print("number of button pushes: ");
    Serial.println(buttonPushCounter);
}

else {
    // daca starea curent este LOW atunci comutatorul si-a modificat starea din pornit in oprit
    Serial.println("off");
}
}

// se salveaza starea curenta ca starea cea mai recenta
lastButtonState = buttonState;

// LED-ul este aprins la patru apasari al comutatorului
// verificam ca numarul este multiplu de patru utilizand functia matematica modulo
// adica restul impartirii sa fie zero
if (buttonPushCounter % 4 == 0) {

    digitalWrite(ledPin, HIGH);
}

else {

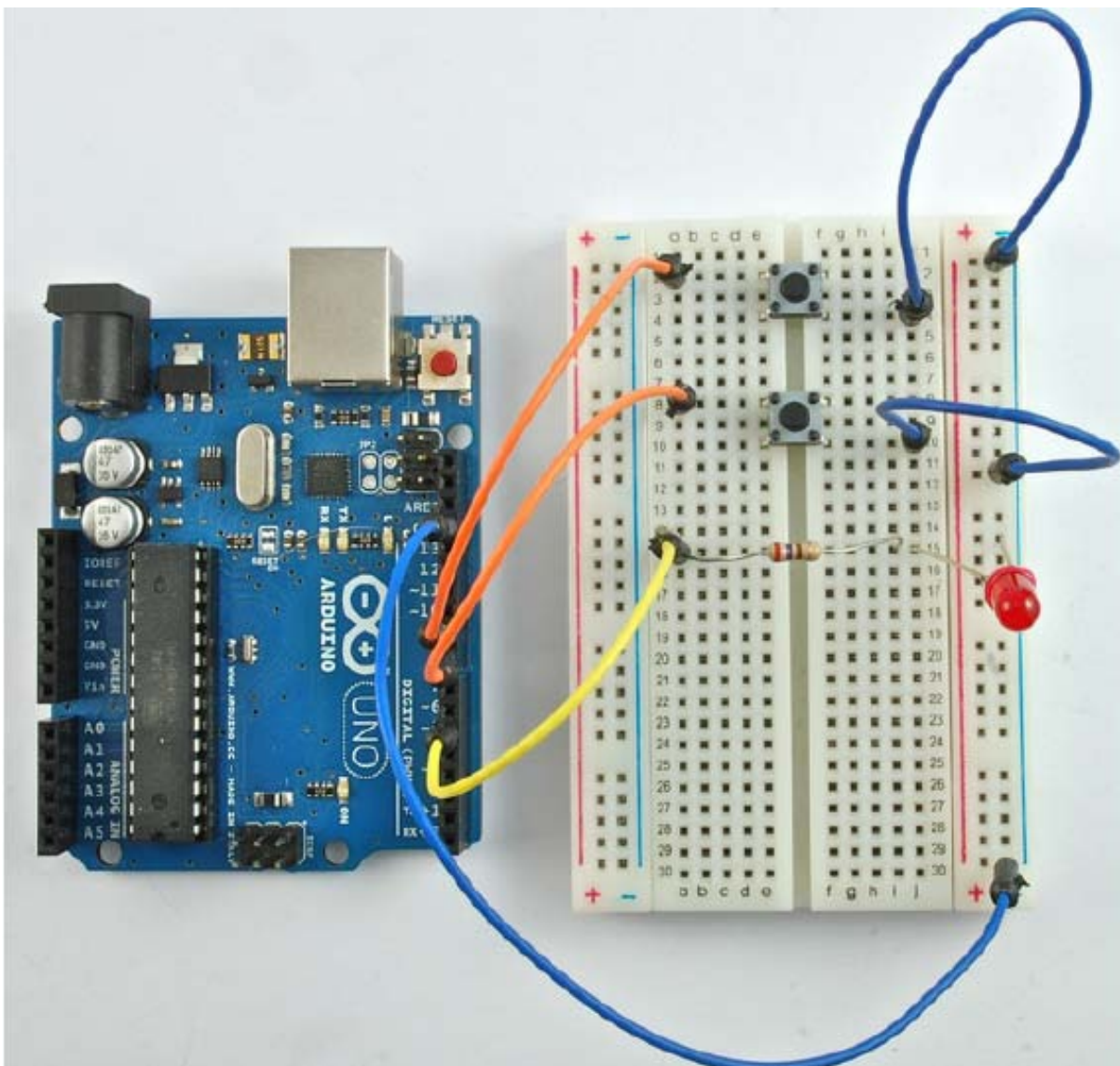
    digitalWrite(ledPin, LOW);
}

}
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File -> Examples -> Digital -> StateChangeDetection.**

## Experimentul 8. Utilizarea a doua comutatoare cu revenire pentru a controla iluminarea unui LED.

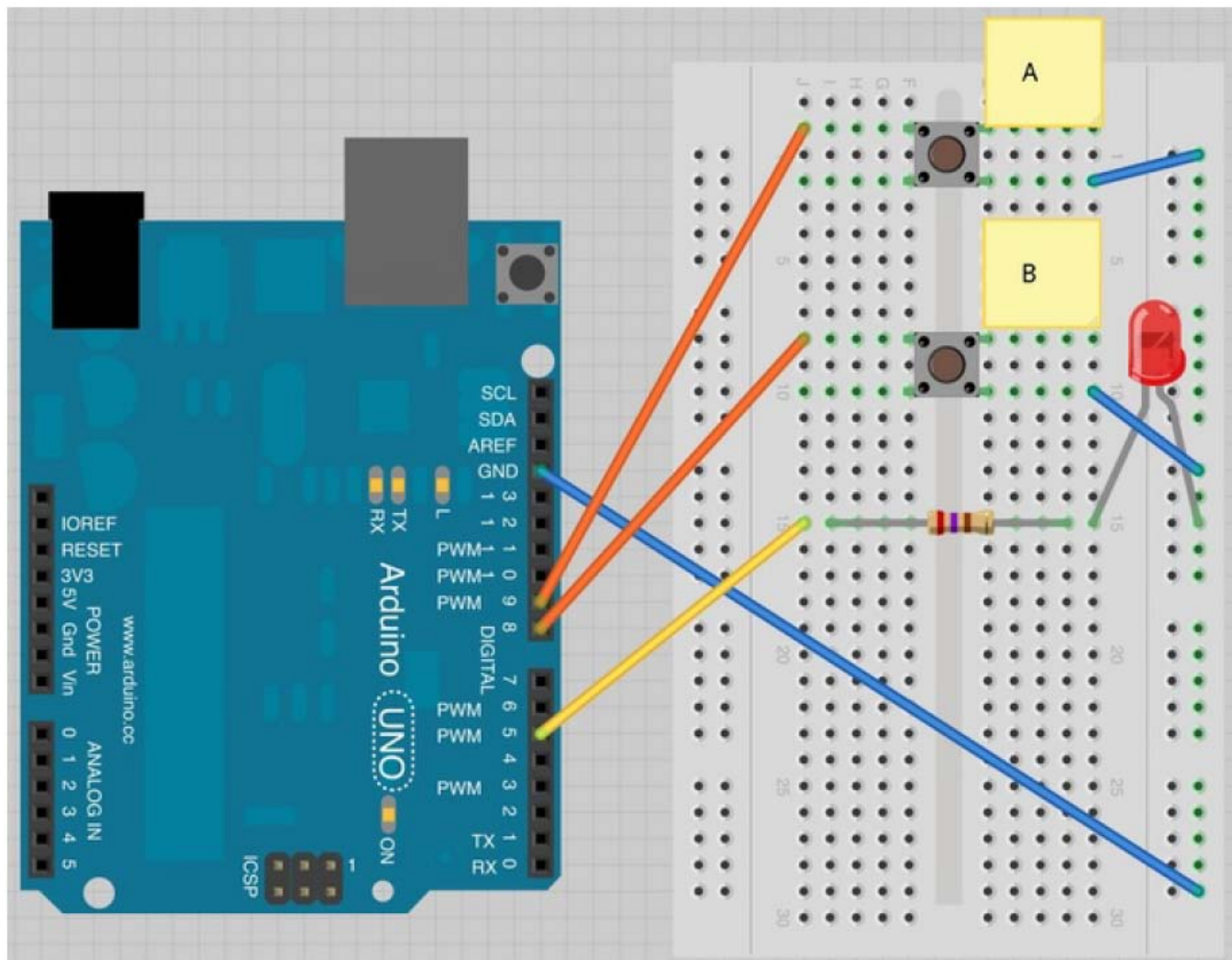
În cadrul acestei lectii, veti invata sa utilizati doua butoane cu revenire impreuna cu intrarile digitale ale dispozitivului Arduino pentru a aprinde sau stinge LED-urile.





### *Schema de montare pe placa Breadboard*

Tineti cont de faptul ca LED-urile au o anumita polaritate si se vor aprinde doar daca sunt alimentate corect; astfel pinul mai scurt, minus, va fi plasat in partea dreapta a placii breadboard asa cum este aceasta prezentata in imaginea de mai jos.



### Codul sursa

Incarcati urmatorul program in dispozitivul Arduino. Atunci cand veti apasa butonul de sus LED-ul va ilumina; daca apasati butonul de jos, LED-ul se va stinge.

```
int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;
byte leds = 0;

void setup(){

  pinMode(ledPin, OUTPUT);
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);

}

void loop(){

  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }

  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

Prima parte a codului sursa defineste cele trei variabile care vor fi utilizate. Variabila *ledPin* reprezinta pinul de iesire iar variabilele *buttonApin* si *buttonBpin* se refera la butonul cu revenire de sus, respectiv la cel de jos. Functia *setup* defineste *ledPin* ca iesire (OUTPUT) precum si modalitatea in care vor fi analizate intrarile *buttonApin* si *buttonBpin*:

```
pinMode(buttonApin, INPUT_PULLUP);
pinMode(buttonBpin, INPUT_PULLUP);
```

Functia *INPUT\_PULLUP* precizeaza modalitatea in care vor fi procesate actiunile butoanelor de impuls: astfel valoarea implicita a intrarii este HIGH, insa este modificata in LOW la apasarea butonului. Tocmai de aceea butoanele sunt conectate la masa (GND). Atunci cand un buton este apasat, el conecteaza pinul de intrare la GND, intrucat nu mai este in starea HIGH.

Logica de functionare a butoanelor o vom defini in cadrul functiei *loop*:

```
void loop(){
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

În cadrul funcției *loop* există două declarații de tip *if*, câte una pentru fiecare dintre butoane. Fiecare dintre acestea utilizează *digitalRead* pentru a verifica starea intrării asociate. Ne amintim că, în condițiile în care butonul este apăsător, starea care îi corespunde va fi LOW, și, dacă starea butonului A este LOW atunci LED-ul trebuie să se aprindă. În mod similar, dacă butonul B este apăsător, LED-ul trebuie să se stingă.

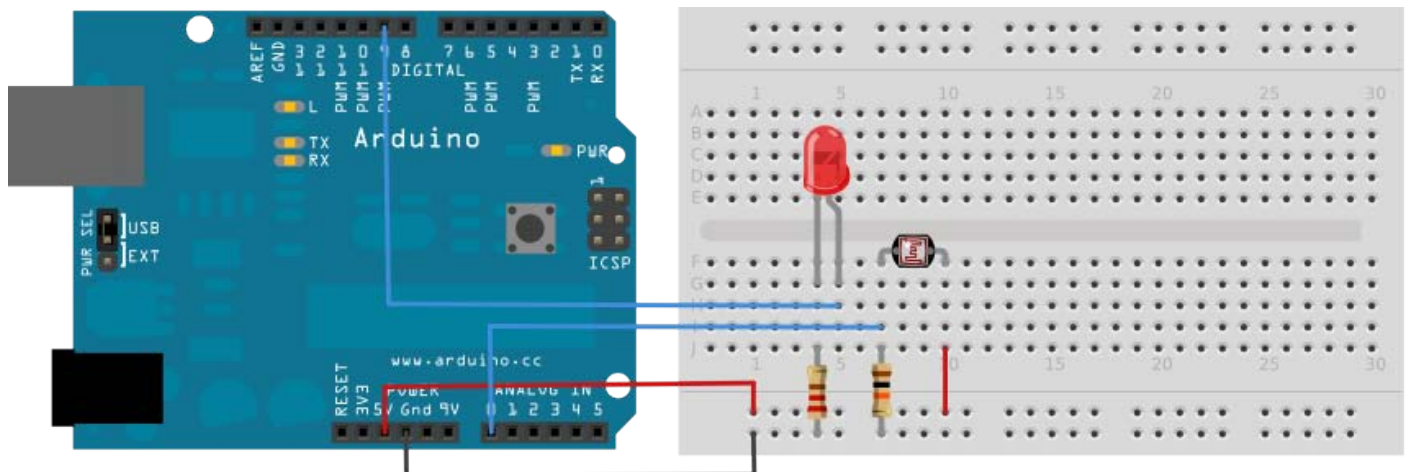
1. Construiți montajul conform schemei care v-a fost prezentată. Copiați codul sursă într-o schiță nouă sistemului Arduino IDE sau deschideți fișierul *exemplu8.ino* din folderul Experimente. Compilați și încarcăți fișierul în dispozitivul Arduino.

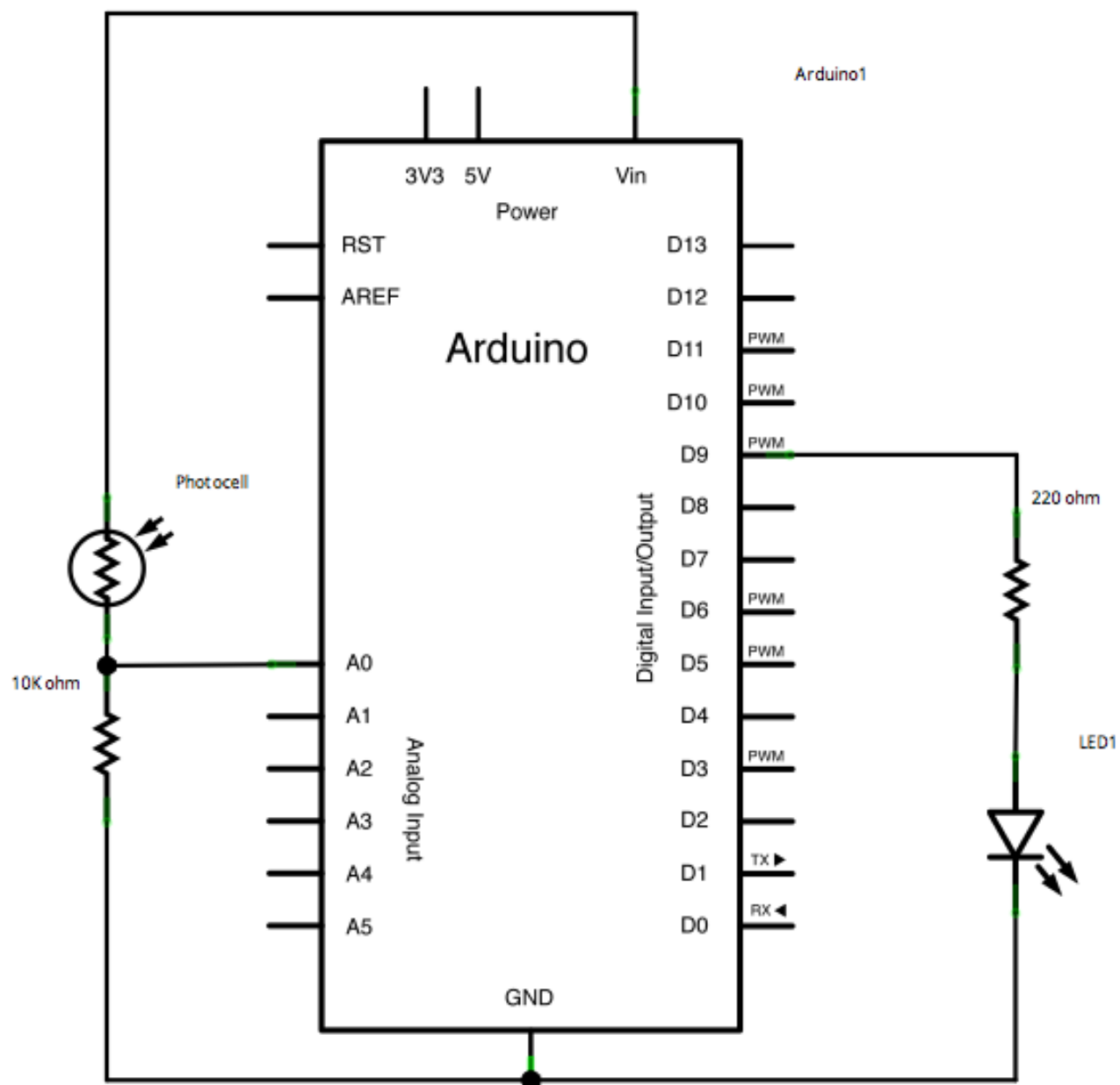
## Experimentul 9. Calibrarea. Utilizarea unei fotocelule pentru a controla intensitatea luminii emise de un LED.

Acest exemplu prezinta una dintre tehnicile utilizate pentru a calibra un senzor. Dispozitivul Arduino citeste valorile senzorului timp de cinci secunde, la initializare, si determina valoarea maxima si minima. Aceste valori citite definesc minimul si maximul asteptate a fi atinse in timpul experimentului.

### Circuit asociat

Conectati un LED la pinul digital 9, utilizand un rezistor de limitare de  $220\Omega$ . Conectati o fotocelula la +5V si la pinul analogic 0 prin intermediul unui rezistor de  $10k\Omega$ , ca referinta, la masa.



Schema electrica

### Codul sursa

Inainte de initializare, se definesc valorile minima si maxima:

```
int sensorMin = 1023;    // valoarea minima a senzorului
int sensorMax = 0;       // valoarea maxima a senzorului
```

Initial, vom seta un minim destul de ridicat si vom citi orice valoare mai mica decat aceasta, inregistrand-o ca noua valoare minima. La fel, setam un maxim destul de scazut si citim orice valoare mai mare decat aceasta, inregistrand-o ca noua valoare maxima.

```
// proces de calibrare in primele cinci secunde dupa initializare:
while (millis() < 5000) {
    sensorValue = analogRead(sensorPin);

    // inregistram valoarea maxima
    if (sensorValue > sensorMax) {
        sensorMax = sensorValue;
    }

    // inregistram valoarea minima
    if (sensorValue < sensorMin) {
        sensorMin = sensorValue;
    }
}
```

Astfel, orice valori preluate ulterior vor fi convertite in intervalul nou definit:

```
// aplicarea calibrarii noilor valori preluate de la senzor
sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
```

Acesta este codul sursa complet:

```
/*
  Calibrarea unui senzor
```

Acest exemplu prezinta una dintre tehnicile utilizate pentru a calibra un senzor. Dispozitivul Arduino citeste valorile senzorului timp de cinci secunde, la initializare, si determina valoarea maxima si minima. Aceste valori citite definesc minimul si maximumul asteptate a fi atinse in timpul experimentului.

Montajul contine:

- \* Un senzor analogic (in aceasta situatie utilizam o fotocelula, dar poate fi folosit si un potentiometru) conectat la pinul analogic 0

\* un LED conectat intre pinul 9 si masa  
\*/

```
const int sensorPin = A0; // pinul analogic de intrare asociat senzorului  
const int ledPin = 9;    // pinul digital de iesire asociat LED-ului
```

```
int sensorValue = 0;      // valoarea curenta a senzorului  
int sensorMin = 1023;     // valoarea minima a senzorului  
int sensorMax = 0;       // valoarea maxima a senzorului
```

```
void setup() {
```

```
// aprindem LED-ul pentru a semnaliza inceputul etapei de calibrare:  
pinMode(13, OUTPUT);  
digitalWrite(13, HIGH);
```

```
// calibrarea dureaza cinci secunde  
while (millis() < 5000) {  
    sensorValue = analogRead(sensorPin);
```

```
    // inregistreaza valoarea maxima  
    if (sensorValue > sensorMax) {  
        sensorMax = sensorValue;  
    }
```

```
    // inregistreaza valoarea minima  
    if (sensorValue < sensorMin) {  
        sensorMin = sensorValue;  
    }  
}
```

```
// semnalizam incheierea etapei de calibrare  
digitalWrite(13, LOW);  
}
```

```
void loop() {
```

```
// citim starea senzoului:  
sensorValue = analogRead(sensorPin);
```

```
// aplicam calibrarea valorii citite  
sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
```

```
// in conditiile in care valoarea este in afara domeniului masurat in etapa de calibrare  
sensorValue = constrain(sensorValue, 0, 255);
```

```
// utilizand valoare calibrata modificam intensitatea luminii emise de LED:  
analogWrite(ledPin, sensorValue);  
}
```

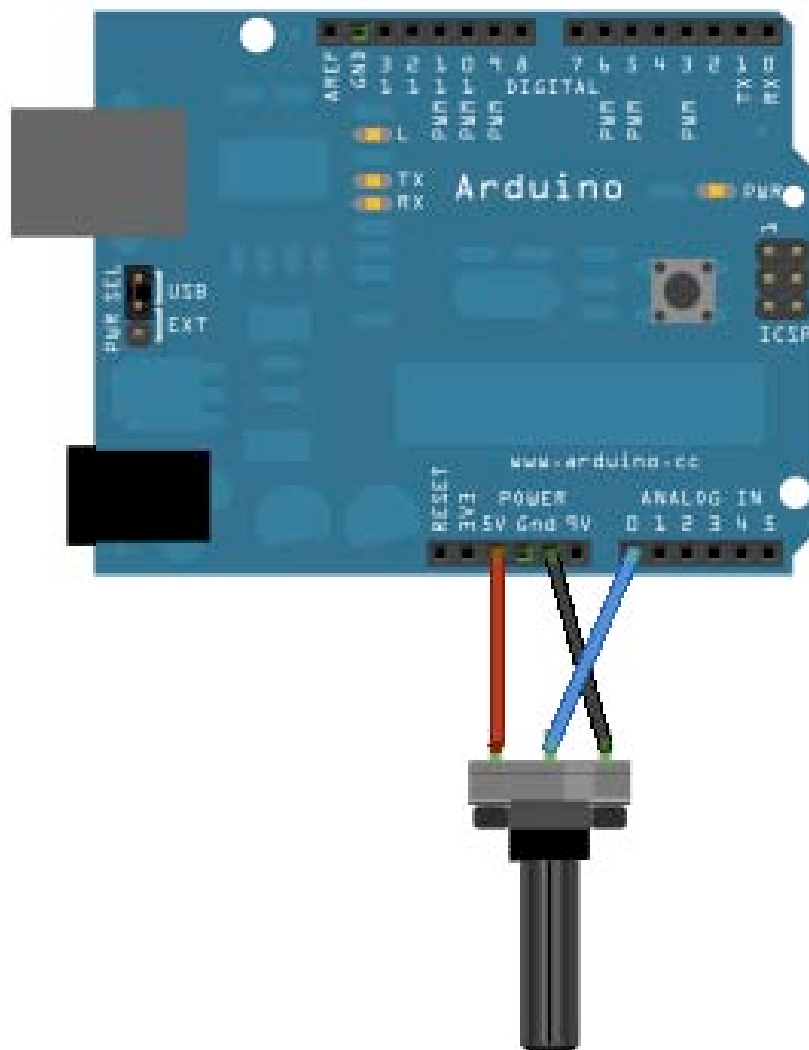
1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File -> Examples -> Analog -> Calibration**. Pentru o calibrare mai precisa, in timpul etapei de calibrare, acoperiti pentru o fractiune de secunda fotocelula astfel incat sa nu mai aiba nici o sursa de lumina.



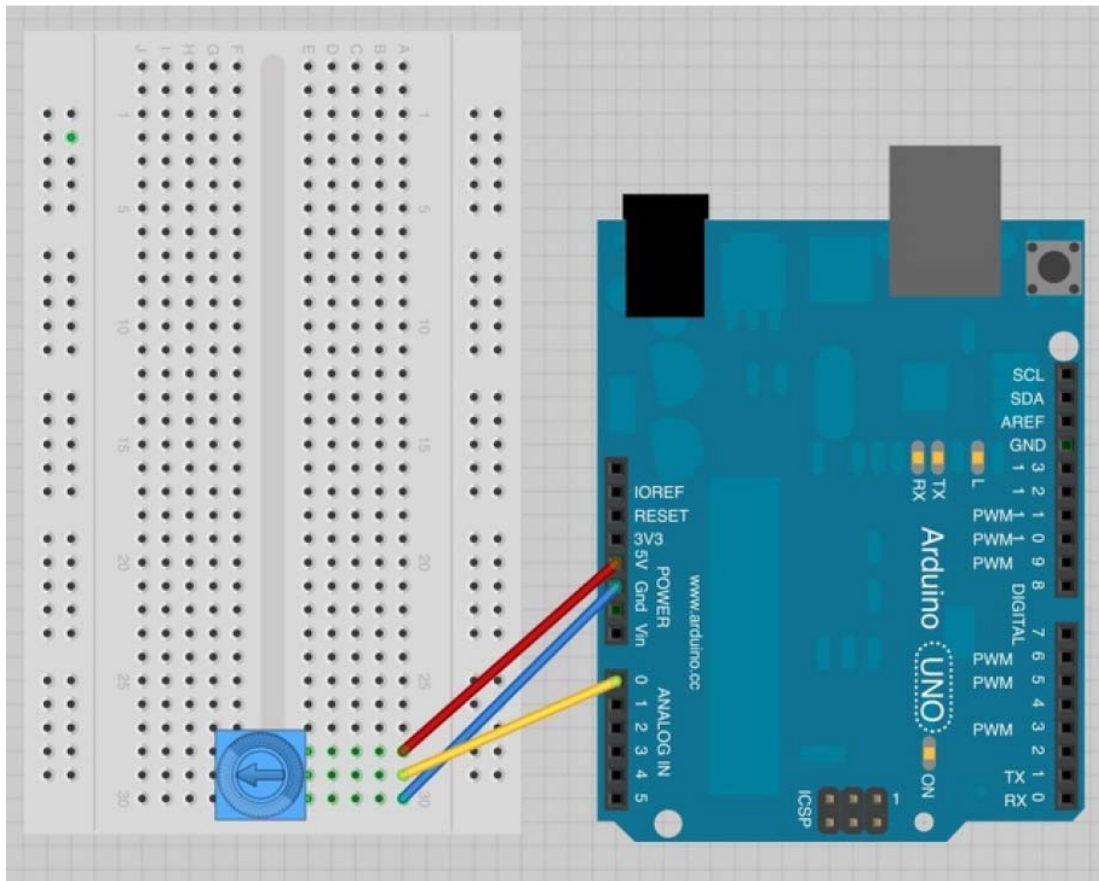
## Experimentul 10. Utilizarea unui potentiometru rotativ liniar pentru a controla o grupare de LED-uri.

In cadrul acestui experiment, vom repeta, pentru inceput experimentul nr. 4. Deschidem initial programul Serial Monitor pentru a afisa valorile citite prin intermediul intrarilor analogice. Mai apoi vom controla numarul LED-urilor ce vor fi aprinse prin intermediul unui potentiometru rotativ liniar.

Inainte de a instala LED-urile in montaj, vom realiza un mic experiment utilizand pontentiometrul de 20k $\Omega$  pe care il avem in kit si programul Serial Monitor. Conectati elementele pe placa breadboard ca in figura:



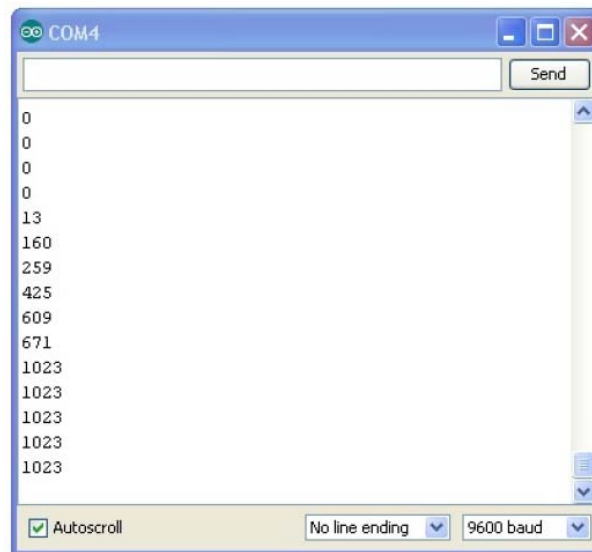
În condițiile în care utilizați un rezistor semireglabil sau va este mai comod să montați potentiometru direct pe placa breadboard puteți folosi următoarele conectare:



Încărcați următorul cod sursă în dispozitivul Arduino:

```
/*  
Intrările analogice ale dispozitivului Arduino  
*/  
int potPin = 0;  
  
void setup(){  
  Serial.begin(9600);  
}  
void loop(){  
  int reading = analogRead(potPin);  
  Serial.println(reading);  
  delay(500);  
}
```

Acum deschideti Serial Monitor, si veti vedea ca apare un sir de numere. Rotiti axul potentiometrului si veti vedea ca numerele din sir se modifica in intervalul 0 ~ 1023.



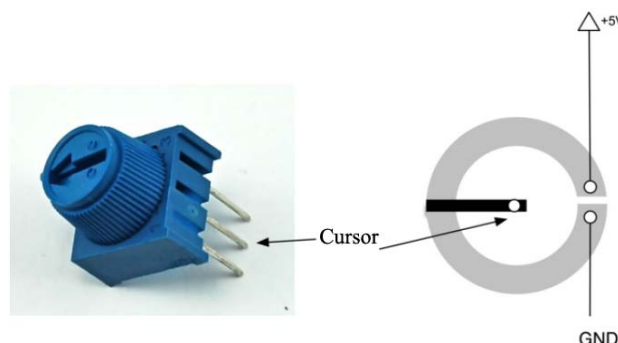
Programul Serial Monitor va afisa valorile analogice citite pe pinul A0, asa cum este definit de linia de cod:

```
int reading = analogRead(potPin);
```

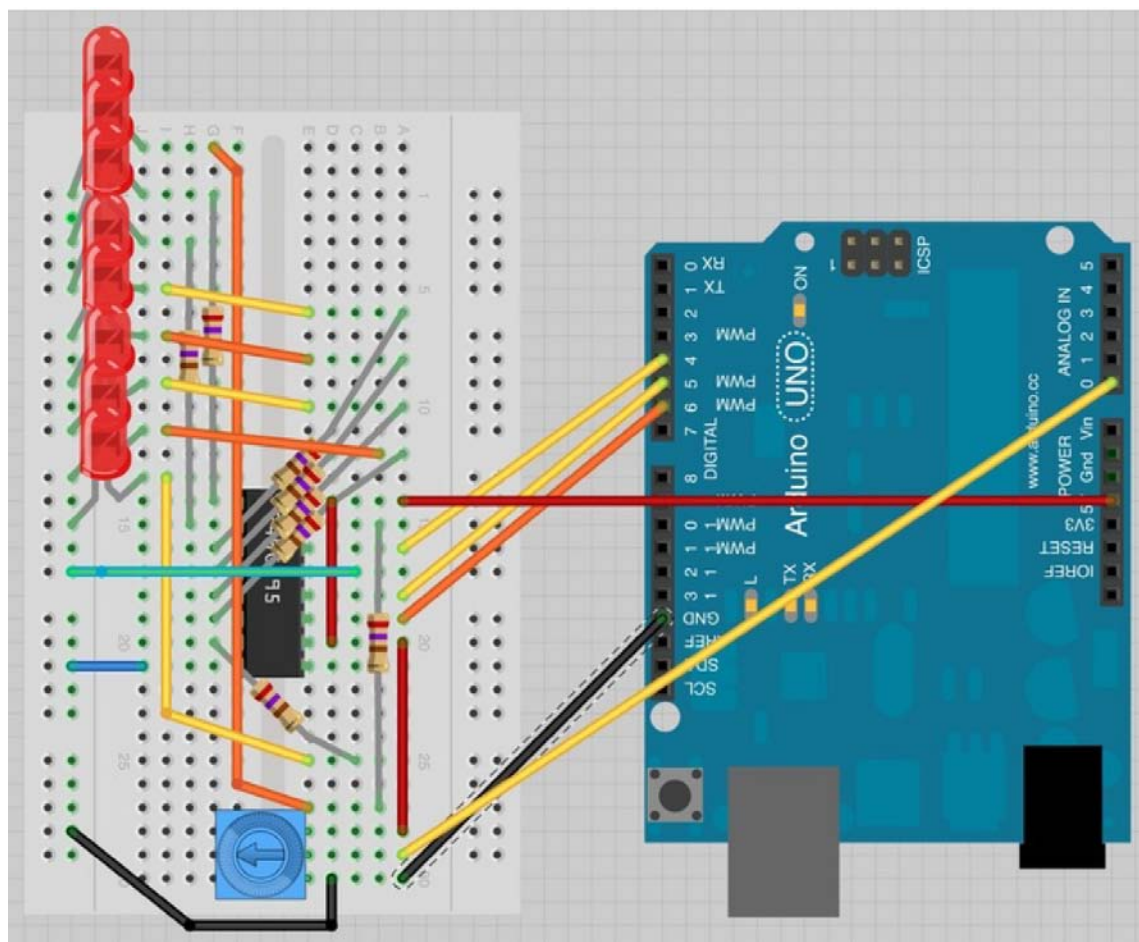
Tensiunea din pinul A0 este convertita intr-un numar intre 0 si 1023. Acelasi experiment il puteti desfasura si utilizand un semireglabil.

### **Rezistoare variabile. Potentiometre.**

In cadrul experimentului nostru, rotirea axului potentiometrului variaza tensiunea din pinul A0, iar codul sursa converteste aceasta tensiune intr-o valoare numerica intre 0 si 1023. Daca vom conecta un terminal de la extremitatea potentiometrului la 5V si cel de la extremitatea opusa la GND, tensiunea masurata intre unul dintre aceste terminale si cursor (pinul central) va varia intre 0 si 5V, in functie de rotirea axului.



*Schema de montare pe placa Breadboard*



**Codul sursa**

Incarcati urmatorul cod sursa in dispozitivul Arduino.

```

/*
Intrările analogice ale dispozitivului Arduino. Controlul unui sir de 8 LED-uri prin intermediul
unui potentiometru rotativ liniar.
*/

int potPin = 0;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
int leds = 0;

void setup(){
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop(){
  int reading = analogRead(potPin);
  int numLEDSLit = reading / 114; //1023 / 9
  leds = 0;
  for (int i = 0; i < numLEDSLit; i++)
  {
    bitSet(leds, i);
  }
  updateShiftRegister();
}

void updateShiftRegister(){
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}

```

Partea esentiala, din punctul de vedere al intrarilor analogice, o reprezinta linia de cod in care definim pinul analogic la care vom conecta cursorul potentiometrului:

```
int potPin = 0;
```

De observat ca nu este nevoie sa realizam o initializare in cadrul functiei *setup* pentru pinul analogic. In bucla principala, functia *loop*, citim valoarea intrarii analogice prin intermediul liniei de cod:

```
int reading = analogRead(potPin);
```

Aceasta valoare va fi între 0 și 1023, iar noi o vom converti într-un număr aflat în intervalul 0 ~ 8, deci va trebui să împărțim valoarea obținută la 114:

```
int numLEDSLit = reading / 114;
```

Pentru a ilumina numărul corect de LED-uri, vom utiliza o buclă de tip *for* numărând de la 0 până la *numLEDSLit* (ce definește numărul de LED-uri care urmează să fie aprinse) și setând bitul la poziția respectivă.

```
leds = 0;
for (int i = 0; i < numLEDSLit; i++)
{
  bitSet(leds, i);
}
```

În cele din urmă actualizăm registrul de deplasare apelând următoarea funcție:

```
updateShiftRegister();
```

1. Puteti modifica programul pentru a-l face să indice poziția pe care o are cursorul potentiometrului, adică să aprindă doar un LED. Pentru aceasta veți modifica buclă:

```
leds = 0;
for (int i = 0; i < numLEDSLit; i++)
{
  bitSet(leds, i);
}
```

cu:

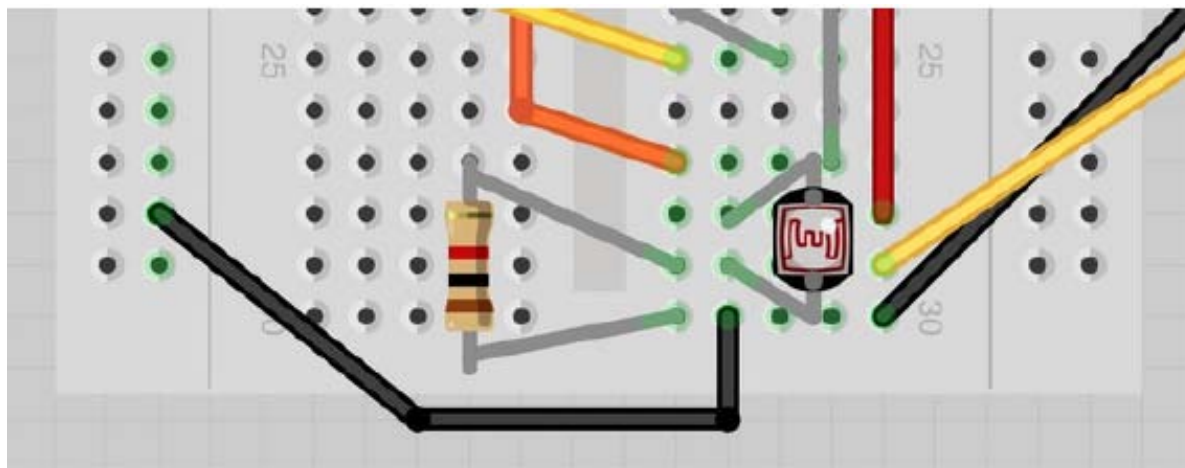
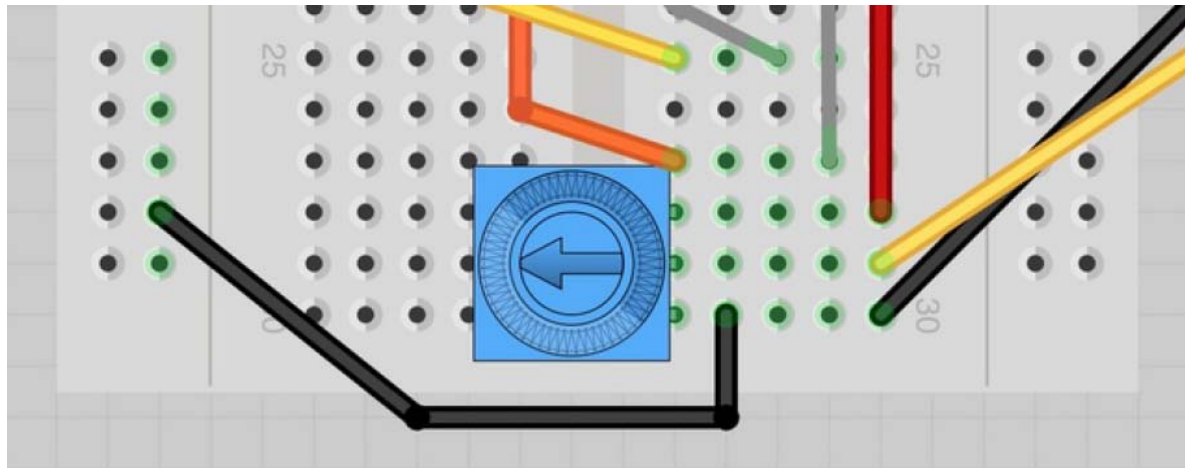
```
leds = 0;
bitSet(leds, numLEDSLit);
```

## Experimentul 11. Sesizarea intensitatii luminii. Utilizarea unei fotocelule pentru a controla o grupare de LED-uri.

În cadrul acestui experiment vom învăța să măsurăm intensitatea luminii utilizând intrarea analogică a dispozitivului Arduino. Vom construi același montaj ca în experimentul precedent și vom utiliza intensitatea luminoasă pentru a controla numărul de LED-uri ce vor fi aprinse. Fotorezistența va înlocui practic potentiometrul sau semireglabilul din experimentul precedent.

### Schema de montare pe placa Breadboard

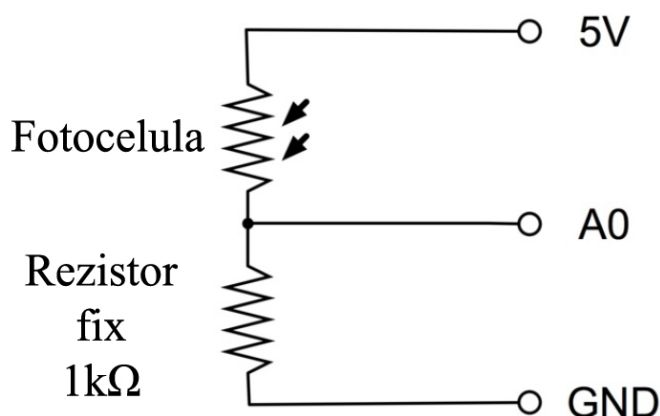
Montajul este similar cu cel precedent cu excepția faptului că potentiometrul a fost înlocuit cu o fotorezistență și un rezistor de 1 k $\Omega$ . Îndepărtați potentiometrul și montați aceste două componente așa cum este prezentat în figura:



### Fotocelula

Fotocelula utilizata in cadrul acestui experiment este denumita fotorezistenta sau rezistor variabil dependent de intensitatea luminoasa (light dependent resistor – LDR). Asa cum sugereaza si denumirea, acest element de circuit se comporta ca un rezistor ce isi schimba rezistenta electrica in functie de lumina primita pe suprafata sa. Aceasta are aproximativ  $50k\Omega$  atunci cand este intuneric si  $500\Omega$  in lumina puternica. Pentru a converti aceasta variatie a rezistentei intr-o unitate care sa poata fi masurata prin intermediul intrarii analogice a dispozitivului Arduino, aceasta trebuie transformata in tensiune.

Cea mai simpla modalitate de a realiza acest lucru este sa o combinam cu un rezistor fix.



Astfel, acest ansamblu se va comporta precum un potentiometru. Atunci cand lumina este foarte puternica, rezistenta fotocelulei va fi foarte mica in comparatie cu aceea a rezistorului fix, situatie comparabila cu atingerea valorii maxime in cazul unui potentiometru. Atunci cand lumina este foarte slaba, aproape de intuneric, rezistenta fotocelulei devine mult mai mare decat aceea a rezistorului fix de  $1k\Omega$ , situatie comparabila cu atingerea valorii minime, sau conectarea la masa a unui potentiometru.

Dupa ce ati realizat montajul, incarcati programul in memoria dispozitivului Arduino si incercati sa acoperiti cu degetul fotocelula, apoi apropiati-o de o sursa de lumina.



### Codul sursa

Utilizand codul sursa din exemplul cu potentiometru, montajul nostru va functiona, insa, nu vom avea niciodata o lumina suficient de puternica pentru a aprinde toate cele opt LED-uri. Acest fenomen se intampla din cauza rezistorului fix si va trebui sa tinem cont sa aprindem toate LED-urile la pragul de  $1k\Omega$ . Iata codul sursa care va ajuta montajul sa functioneze corect:

```
/*
Sesizarea luminii
*/
int lightPin = 0;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
int leds = 0;

void setup(){
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop(){
  int reading = analogRead(lightPin);
  int numLEDSLit = reading / 57; //1023 / 9 / 2
  if (numLEDSLit > 8) numLEDSLit = 8;
  leds = 0;
  for (int i = 0; i < numLEDSLit; i++)
  {
    bitSet(leds, i);
  }
  updateShiftRegister();
}

void updateShiftRegister(){
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}
```

Primul lucru pe care il puteti observa este ca am schimbat denumirea pinului analogic din *potPin* in *lightPin*.

Cealalta modificare notabila este aceea ca modalitatea de a calcula numarul de LED-uri ce urmeaza sa se aprinda a fost schimbata:

```
int numLEDSLit = reading / 57; // toate LED-urile se aprind la 1k $\Omega$ 
```

De data aceasta vom imparti valoarea obtinuta prin intermediul pinului analogic la 57, impartind astfel in noua intervale, de la situatia in care toate LED-urile sunt stinse pana la aceea in care toate LED-uri sunt aprinse. Factorul suplimentar de care a trebuit sa tinem cont a fost rezistorul fix de 1 k $\Omega$ . In momentul in care fotocelula va avea rezistenta de 1k $\Omega$  (adica aceeaasi cu cea a rezistorului fix) valoarea analogica receptionata de dispozitivul Arduino va fi  $1023 / 2 = 511$ . Aceasta situatie va echivala cu aprinderea tuturor LED-urilor (*numLEDSLit* = 9).

1. Pentru a modifica sensibilitatea senzorului la lumina, incercati sa modificati valoarea factorului de demultiplicare (57). Cresterea acestuia va face mai putin sensibilia citirea.
2. Puteti modifica programul astfel incat, la o anumita limita, toate LED-urile sa se stinga sau sa se aprinda.

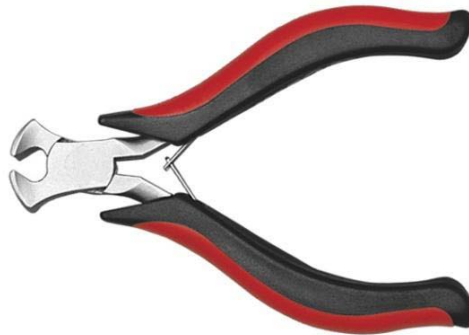
## Fire de legatura

Unul dintre aspectele esentiale ale oricarei activitati pe care o veti intreprinde este sa iti utilizezi eficient si responsabil resursele pe care le ai la dispozitie.

Este posibil ca, o data cu cresterea dificultatii experimentelor pe care le veti desfasura, sa aveti nevoie de fire de legatura suplimentare. Pentru aceasta situatie v-au fost puse la dispozitie inca 8 conexiuni scurte neizolate si 2 metri de cablu din care veti realiza, sub indrumarea coordonatorului dvs., fire de legatura suplimentare. In conditiile in care si aceste fire de legatura vor fi insuficiente, puteti utiliza orice tip de cablu, cu miezul masiv, cu diametru intre 0,5mm si 0,65mm. Recomandat pentru aceasta situatie este conductorul din interiorul oricarui cablu UTP.

Pentru a construi firele de legatura suplimentare veti proceda astfel:

- utilizand clestele sfic pe care il aveti la dispozitie veti taia cablul la dimensiunea dorita; in principiu, veti construi fire de legatura de 10cm si 15cm, la fel ca acelea de dimensiune redusa si medie pe care le aveti in kit; le puteti utiliza pe acestea pentru a va orienta atunci cand veti alege dimensiunea la care sa taiati firele
- dupa ce ati taiat firul de legatura la dimensiunea dorita il veti dezizola la capete; lungimea regiunii dezizolate trebuie sa aiba in jur de 1cm, adica sa fie suficient de lunga pentru a face un contact ferm in placa breadboard dar si destul de scurta pentru a nu atinge alte fire de legatura.



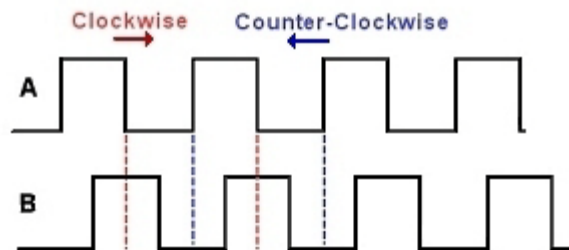
*Figura. Clestele sfic pe care il veti utiliza pentru a construi noi fire de legatura*

## Experimentul 12. Utilizarea unui potentiometru digital.

Utilizand potentiometrul digital avem doua semnale de iesire dreptunghiulare (A si B) defazate cu 90 de grade unul fata de celalalt.



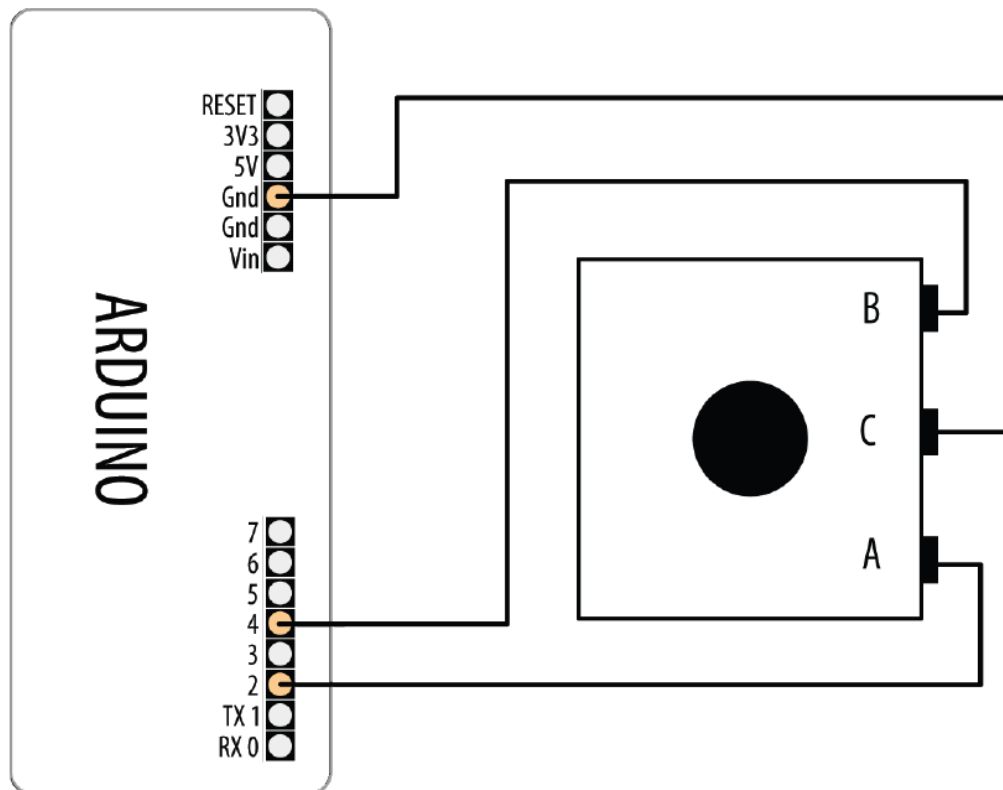
Numarul de impulsuri generate pentru un tur complet poate varia (cel pe care il aveti in kit-ul dvs. genereaza 24 de impulsuri la fiecare tur complet). Diagrama de functionare de mai jos arata relatia dintre fazele A si B atunci cand potentiometrul este intors in sensul acelor de ceasornic sau in sens invers acelor de ceasornic.



De fiecare data cand semnalul A ajunge din valoare pozitiva in zero, vom citi valoarea impulsului B. Vedem ca atunci cand potentiometrul este rotit in sensul acelor de ceasornic impulsul B este intotdeauna pozitiv; daca potentiometrul este rotit in sens invers acelor de ceasornic impulsul B este intotdeauna negativ, Testand ambele iesiri cu un microcontroller putem determina directia in care a fost rotit potentiometrul, si, contorizand numarul de impulsuri A, pozitia unghiulara a acestuia.

### Schema electrica

În cadrul acestui experiment vom învăța să măsurăm viteza de rotație și direcția unui potențiometru digital. Pentru aceasta veți conecta potențiometrul digital din kit-ul dvs. la dispozitivul Arduino ca în schema următoare.



### Codul sursa

/\*

Evaluarea poziției unui potențiometru digital.

Rezultatele vor fi afișate prin intermediul programului Serial Monitor

\*/

```
const int encoderPinA = 4;
const int encoderPinB = 2;
const int encoderStepsPerRevolution=24;
```

```
int angle = 0;
int val;
int encoderPos = 0;
boolean encoderALast = LOW; // retine starea precedenta
```

```

void setup(){

pinMode(encoderPinA, INPUT);
pinMode(encoderPinB, INPUT);
digitalWrite(encoderPinA, HIGH);
digitalWrite(encoderPinB, HIGH);
Serial.begin (9600);
}

void loop(){

boolean encoderA = digitalRead(encoderPinA);

if ((encoderALast == HIGH) && (encoderA == LOW))
{
if (digitalRead(encoderPinB) == LOW)
{
encoderPos--;
}
else
{
encoderPos++;
}
angle=(encoderPos % encoderStepsPerRevolution)*360/encoderStepsPerRevolution;
Serial.print (encoderPos);
Serial.print (" ");
Serial.println (angle);
}

encoderALast = encoderA;
}

```

Asa cum am discutat potentiometrul digital produce doua semnale atunci cand axul sau este rotit. Ambele semnale alterneaza intre HIGH (1 logic) si LOW (0 logic) pe masura ce axul se roteste, dar sunt decalate in timp. Daca vom detecta momentul in care unul dintre semnale se schimba din HIGH in LOW, starea celuilalt pin (fie ca este HIGH sau LOW) delimiteaza directia in care a fost rotit axul.

Astfel, prima linie din cadrul functiei *loop* citeste starea unuia dintre pini:

```
int encoderA = digitalRead(encoderPinA);
```

Apoi verifica valoarea precedenta, pentru a identifica daca valoarea curenta s-a modificat in LOW:

```
if((encoderALast == HIGH) && (encoderA == LOW))
```

Daca valoarea nu s-a modificat, codul ce urmeaza nu este executat, este salvata valoarea care tocmai a fost citita in variabila *encoderALast* si este citita o noua valoare.

Atunci cand conditia:

```
if((encoderALast == HIGH) && (encoderA == LOW))
```

este verificata, programul sursa citeste starea celuiilalt pin si incrementeaza (+1) sau decrementeaza (-1) variabila *encoderPos* in functie de valoarea citita. Programul incepe sa calculeze unghiul axului (luand ca punct de zero momentul in care programul a inceput sa ruleze). Valoarea calculata este transmisa prin intermediul portului serial catre programul *Serial Monitor*.

Potentiometrele digitale pot avea diverse rezolutii, adica numar de impulsuri pentru o rotatie completa. Acest parametru indica numarul de semnale alternand intre HIGH si LOW dupa o rotatie completa a axului. Valoarea poate fi intre 16 si 1000 (asa cum am specificat anterior, cel pe care il aveti in kit-ul dvs. genereaza 24 de impulsuri la fiecare tur complet). Numarul de impulsuri per rotatie completa a fost definit in cadrul programului nostru in linia de cod:

```
const int encoderStepsPerRevolution=16;
```

Daca veti obtine valori care nu cresc si descresc pe masura ce rotiti potentiometru, ci se modifica doar intr-un singur sens, modificati linia de cod:

```
if((encoderALast == HIGH) && (encoderA == LOW))
in
if((encoderALast == LOW) && (encoderA == HIGH))
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Deschideti programul *Serial Monitor* (accesandu-l din meniu **Instruments -> Serial Monitor**) si observati modificarea valorilor unghiului de rotatie al potentiometrului digital pe masura ce rotiti axul sau.

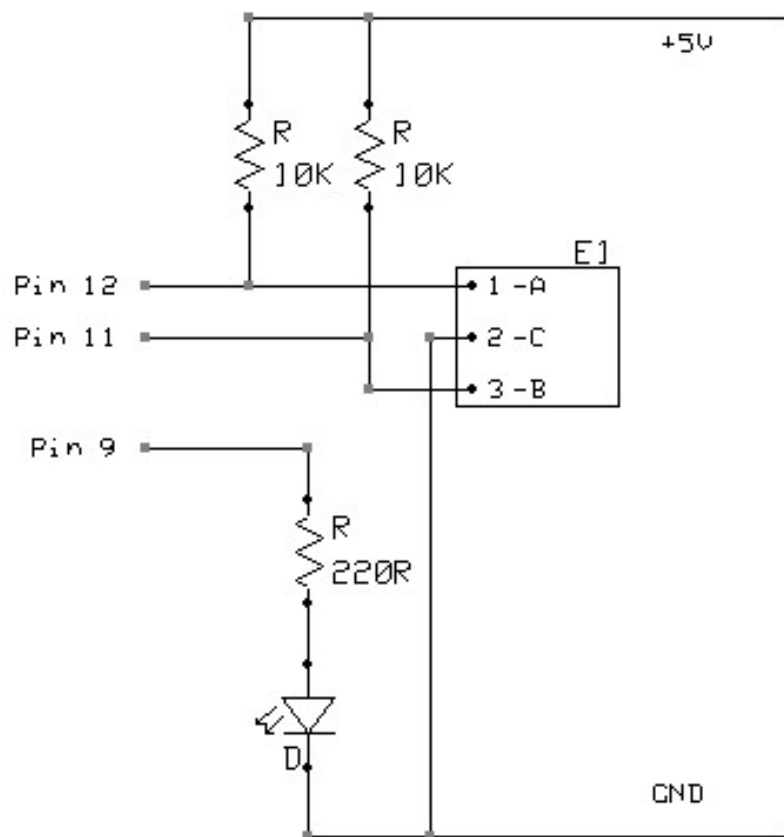
### Experimentul 13. Utilizarea unui potentiometru digital pentru a varia intensitatea luminoasa a unui LED.

Vom utiliza potentiometrul digital pentru o aplicatie simpla, controlul luminozitatii unui LED. Inainte de toate, trebuie sa determinam cat de rapid trebuie sa opereze temporizatorul pe care il vom defini in program. Daca ne imaginam ca putem roti potentiometrul cu 180 de grade intr-o zecime de secunda, aceasta inseamna ca vom avea 12 impulsuri intr-o zecime de secunda, deci 120 de impulsuri intr-o secunda. In realitate, este putin probabil sa reactionam atat de rapid. Intrucat trebuie sa detectam atat valorile minime cat si pe cele maxime, trebuie sa avem o frecventa de lucru de minim 120Hz. Vom utiliza 200Hz pentru a fi siguri.

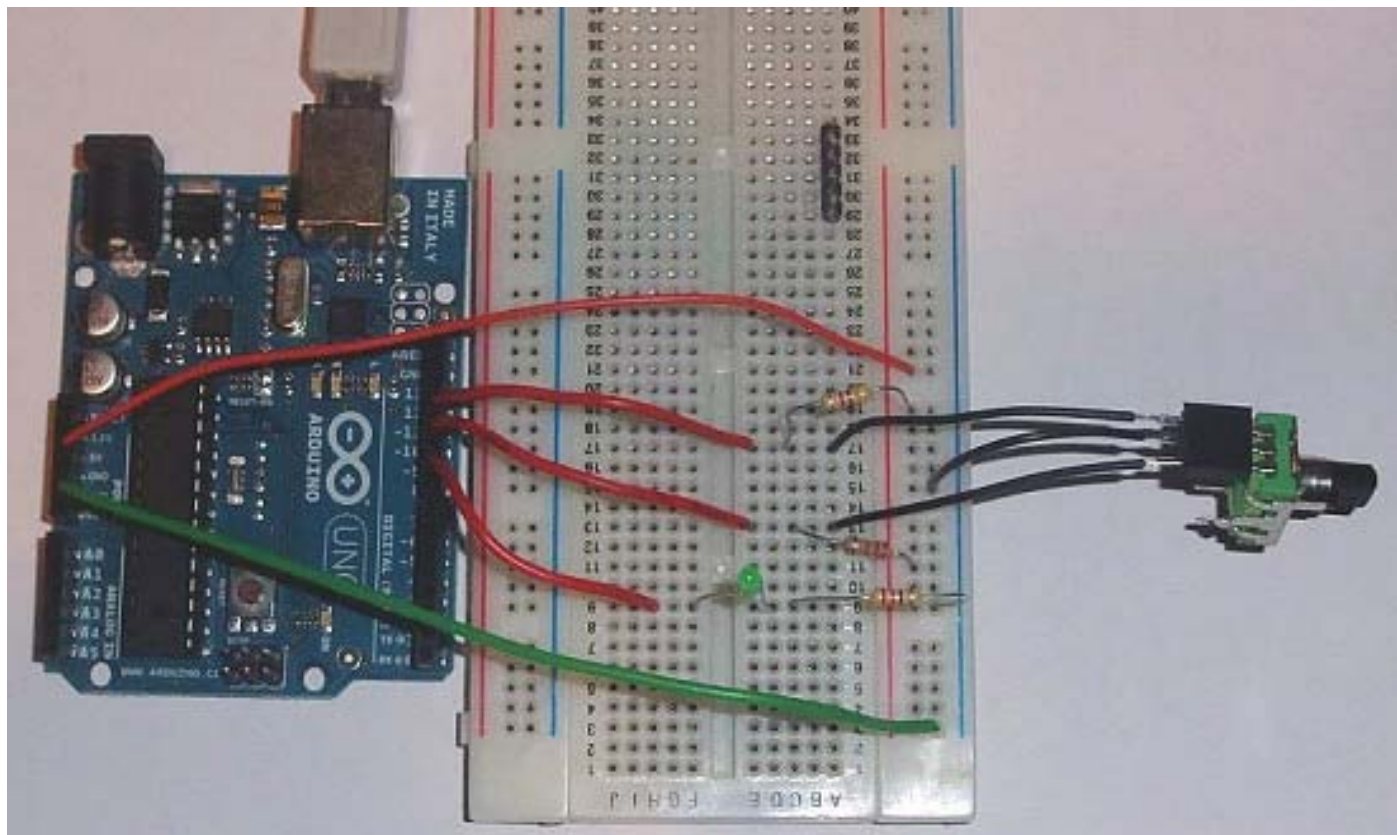
De fiecare data se declanseaza temporizatorul pe care il vom defini, se va efectua o comparatie intre valoarea curenta a pulsului A si cea precedenta. In functie de rezultat putem incrementa sau decrementa un contor. Aceasta valoare o vom utiliza pentru a controla una dintre iesirile PWM ale dispozitivului Arduino pentru a creste sau reduce luminozitatea LED-ului.

#### Schema electrica

Schema de montare este prezentata mai jos:







### Codul sursa

Pentru a testa montajul va trebui sa incarcati programul de mai jos in dispozitivul Arduino:

```

/*
** Utilizarea pontetiometrului digital pentru a varia
** intensitatea luminoasa a unui LED
**
** Rata de esantionare este de 200Hz utilizand functia millis()
*/

int brightness = 120; // la pornire, intensitatea luminoasa este la jumatate
int fadeAmount = 10; // punctele pentru reducerea luminozitatii

unsigned long currentTime;
unsigned long loopTime;

```

```
const int pin_A = 12; // pinul 12
const int pin_B = 11; // pinul 11

unsigned char encoder_A;
unsigned char encoder_B;
unsigned char encoder_A_prev=0;

void setup(){

    // declarare a pinului 9 ca iesire:
    pinMode(9, OUTPUT);

    // declarare a pinilor potentiometrului digital ca intrari:
    pinMode(pin_A, INPUT);
    pinMode(pin_B, INPUT);

    //obtinerea timpului curent
    currentTime = millis();
    loopTime = currentTime;
}

void loop() {

    // obtinem timpul scurs
    currentTime = millis();

    if(currentTime >= (loopTime + 5)){

        // 5ms de la ultima verificare a pozitiei adica o frecventa de 200Hz
        // citirea pinilor pontentiometrului digital
        encoder_A = digitalRead(pin_A);
        encoder_B = digitalRead(pin_B);

        if(!encoder_A && (encoder_A_prev)){

            // A a ajuns din valoarea maxima in minim
            if(encoder_B) {

                // B este in valoarea maxima deci rotirea este in sensul acelor de ceasornic
                // crestem valoarea intensitatii luminoase, dar mai putin de 255
                if(brightness + fadeAmount <= 255) brightness += fadeAmount;
            }
        }
    }
}
```

```

    else {

        // B este minim deci rotirea este in sens invers acelor de ceasornic
        // scade intensitatea luminoasa, dar nu sub 0
        if(brightness - fadeAmount >= 0) brightness -= fadeAmount;
    }

}

encoder_A_prev = encoder_A; // stocam valoarea lui A

// setam intensitatea luminoasa a pinului 9:
analogWrite(9, brightness);

loopTime = currentTime; // actualizam loopTime
}

}

```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Observati modificarea intensitatii luminoase a LED-ului pe masura ce rotiti axul potentiometrului digital.

2. Puteti modifica:

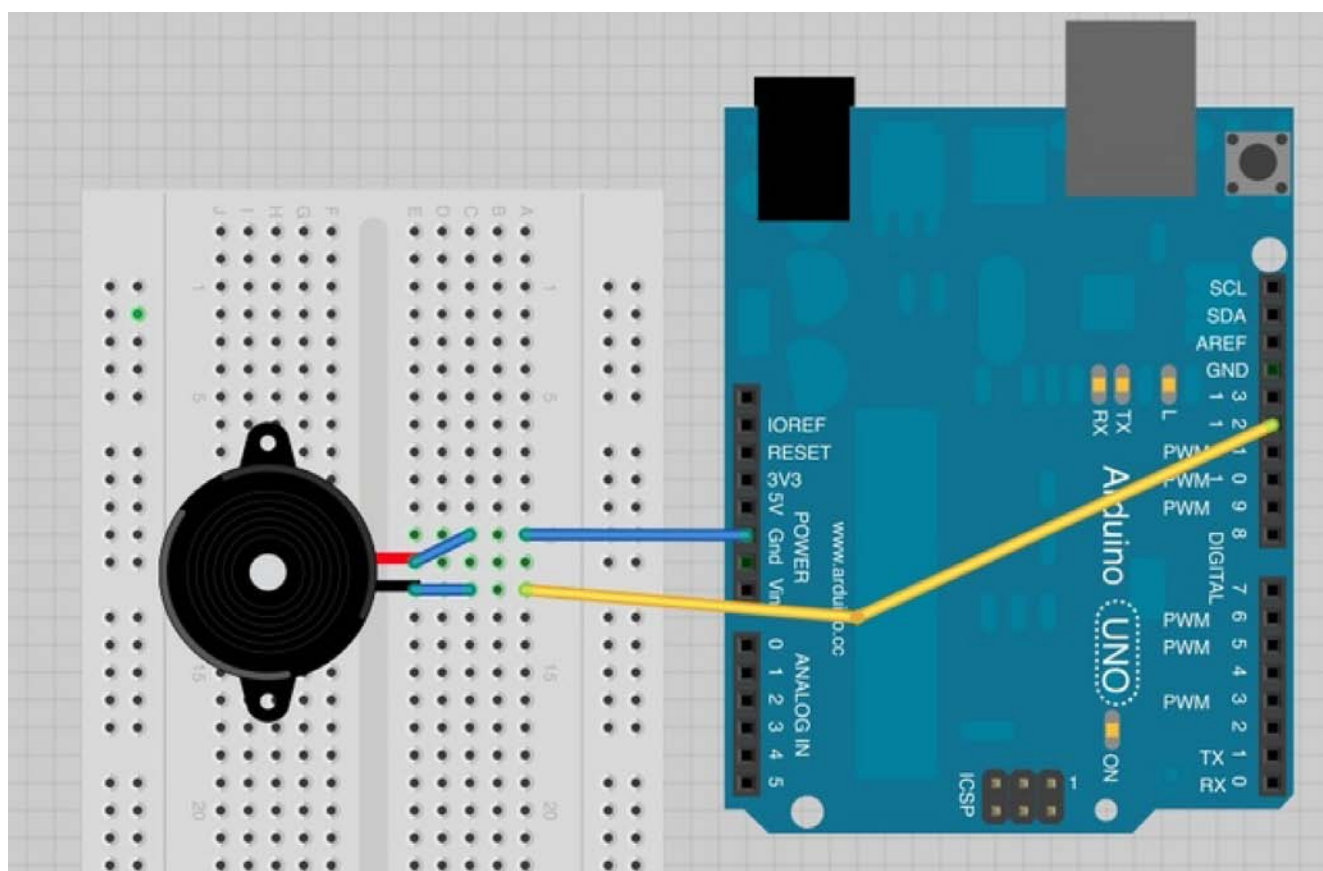
- Intensitatea luminoasa la pornirea experimentului din linia de cod:  
*int brightness = 120; // la pornire, intensitatea luminoasa este la jumatate*
- Numarul de etape in care se reduce intensitatea luminoasa din linia de cod:  
*int fadeAmount = 10; // punctele pentru reducerea luminozitatii*
- Rata de esantionare a semnalului receptionat de la potentiometrul digital din linia de cod:  
*if(currentTime >= (loopTime + 5)){ //o noua evaluare a starii se realizeaza //dupa 5 milisecunde*

## Experimentul 14. Emiterea sunetelor utilizand Arduino si un difuzor. Constructia unui pseudo-theremin.

In cadrul acestei lectii vom invata sa generam sunete cu ajutorul Arduino. Inainte de toate, vom utiliza Arduino pentru a genera o secventa de note muzicale, apoi vom conecta o fotorezistenta in montaj pentru a construi un pseudo-theremin, care va schimba nota muzicala emisa atunci cand trecem mana pe deasupra senzorului.

### Generarea unei secvente de note muzicale

Pentru aceasta prima parte a experimentului, vom conecta pinul pozitiv al difuzorului nostru la pinul 12 al dispozitivului Arduino si pinul negativ la GND.



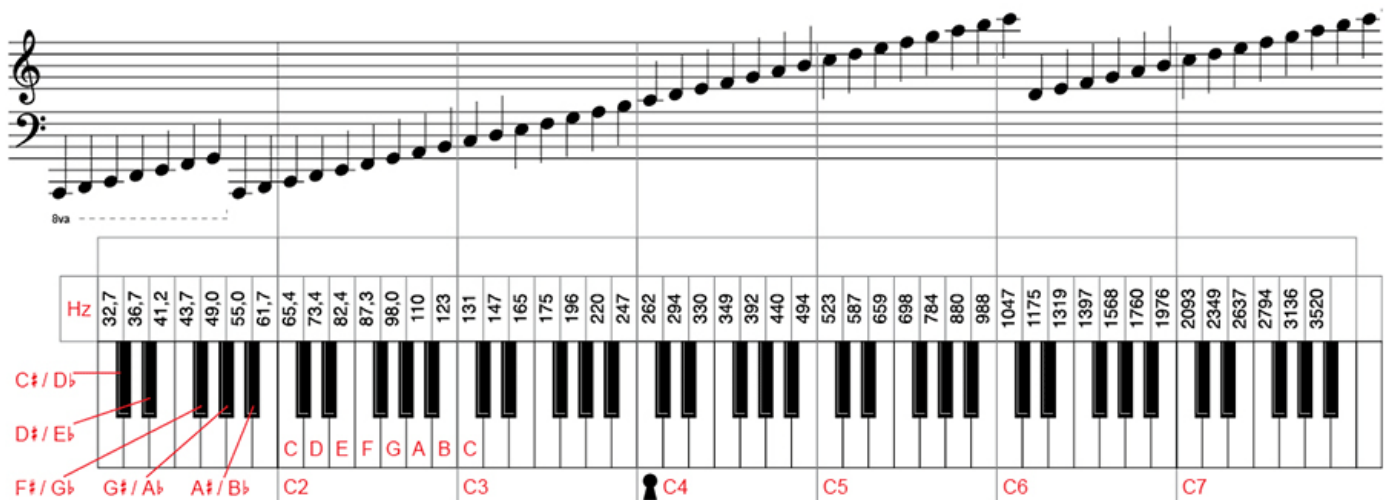
Vom incarca urmatorul program in dispozitivul Arduino:

```
/*
Generarea unei secvente de note muzicale.
*/
int speakerPin = 12;
int numTones = 10;
int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440};
// DO DO# RE RE# MI FA FA# SOL SOL# LA

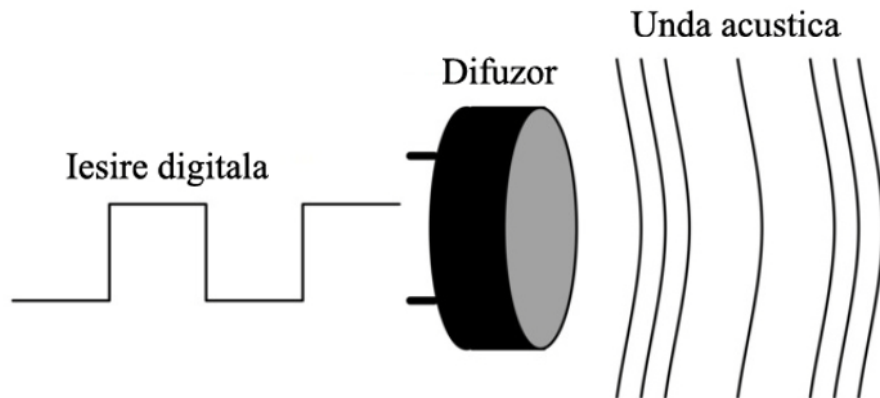
void setup(){
  for (int i = 0; i < numTones; i++)
  {
    tone(speakerPin, tones[i]);
    delay(500);
  }
  noTone(speakerPin);
}

void loop(){ }
```

Pentru a genera un anumit sunet, trebuie specificata frecventa acestuia. Frecventele fiecare note muzicale au fost stocate in sirul *tones*. Bucloa *for* va numara de la 0 pana la 9 utilizand variabila *i*. Pentru a reda frecventa unei note, la fiecare pas vom folosi sirul *tones[i]*. Functia *tone* utilizeaza doi parametri: primul este pinul care isi va modifica valoarea pentru a activa difuzorul si al doilea, frecventa sunetului care va fi rulat. Dupa ce au fost rulate toate sunetele, comanda *noTone* va opri functionare pinului care emite sunetul. Am scris mare parte din codul sursa in cadrul functiei *setup* in loc de *loop* pentru ca acesta sa ruleze o singura data. Daca doriti ca programul sa se repete apasati butonul de reset al dispozitivului Arduino. In imaginea de mai jos, va sunt prezentate frecventele sunetelor emise la apasarea clapelor pianului.

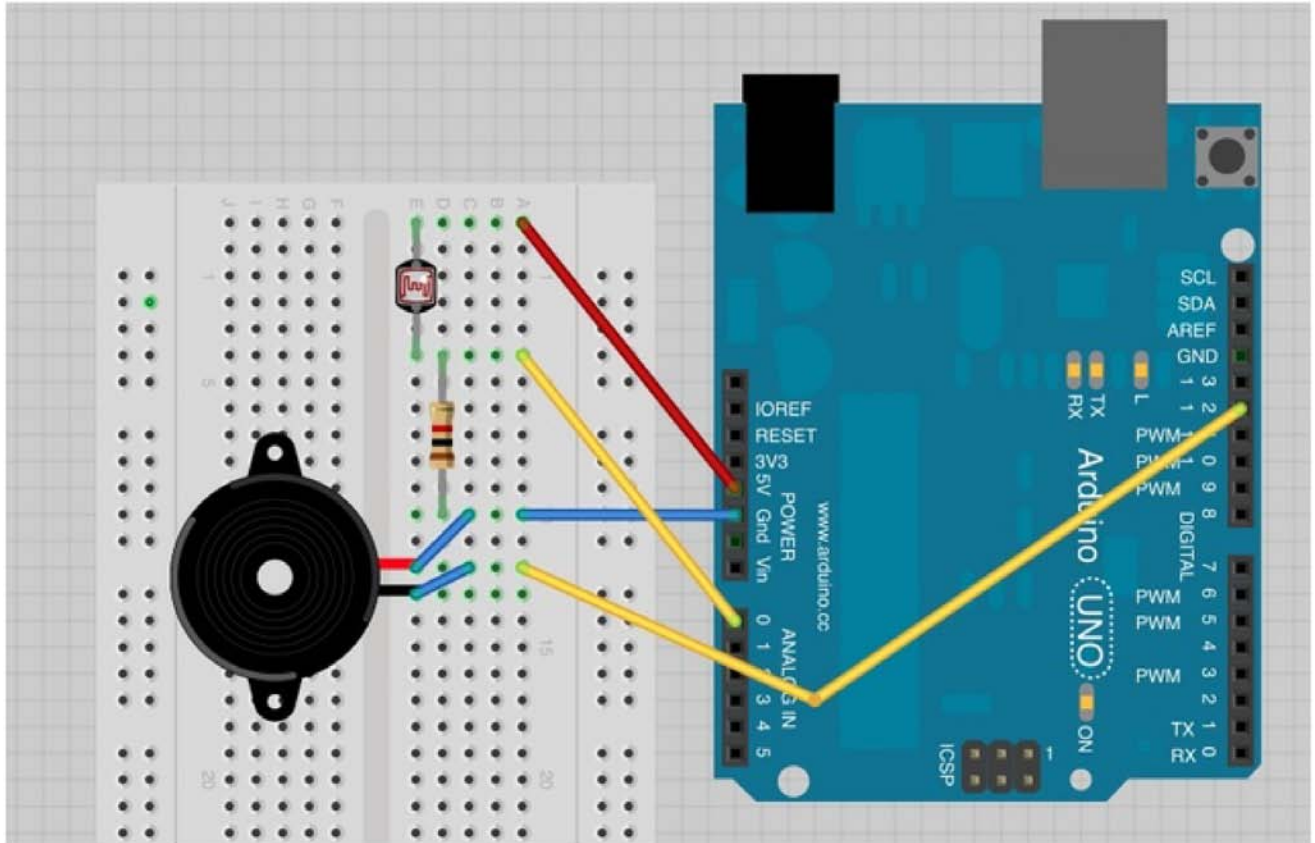


Undele acustice sunt variatii ale presiunii aerului. Viteza acestor variatii (cicluri pe secunda sau Hertzi) este aceea care definește înălțimea sunetului. De exemplu, nota Do este definită la o frecvență de 261 Hz. Dacă vom activa și dezactiva o ieșire digitală de 261 de ori într-o secundă atunci vom emite nota Do. Pentru a auzi această notă muzicală, trebuie să conectăm la această ieșire un dispozitiv care să convertească semnalul electric în unde acustice. Acest lucru este realizat de un difuzor sau, în cazul de față, de un buzzer piezoelectric. Acestea utilizează un cristal special care se dilată și contractă atunci când este străbătut de un semnal electric.



### *Pseudo-Theremin*

Thereminul este un instrument muzical care generează sunete atunci când mâinile unei persoane trec pe deasupra sa. Acesta a fost utilizat pentru muzica din serialul Star Trek. Noi vom construi un instrument similar, care va schimba frecvența notei muzicale atunci când mâna va trece pe deasupra senzorului. Putem lăsa difuzorul în același loc, însă vom utiliza placa breadboard pentru a conecta un fotorezistor și un rezistor.



### Codul sursa

Incarcati codul urmatoar in dispozitivul Arduino:

```

/*
Pseudo-Theremin
*/

int speakerPin = 12;
int photocellPin = 0;
void setup()
{
}

void loop(){

int reading = analogRead(photocellPin);
int pitch = 200 + reading / 4;
tone(speakerPin, pitch);

```



}

Codul sursa este conceput destul de simplu. Citim informatia de pe pinul A0 pentru a masura intensitatea luminoasa, obtinand o valoare cuprinsa intre 0 si 700. Adaugam 200 la aceasta valoare, facand 200Hz cea mai joasa frecventa, si adaugam a patra parte din valoarea obtinuta prin interogarea pinului A0, ceea ce va conduce la obtinerea unui sunet cu o frecventa cuprinsa intre 200Hz si 370Hz.

### **Modificari posibile**

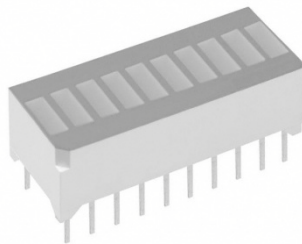
1. Incercati sa modificati valoarea sunetului de baza si a celui care evalueaza intensitatea luminoasa in linia de cod:  
 $\text{int pitch} = 200 + \text{reading} / 4;$   
Aceasta va conduce la extinderea sau restrangerea gamei de frecvente a sunetului emis.
2. Intorcandu-va la prima parte a experimentului, modificati frecventa sunetelor emise. Puteti realiza aceasta, modificand valorile din sirul *tones*. Tineti cont de faptul ca, daca modificati numarul notelor muzicale pe care doriti sa le emita difuzorul, trebuie sa modificati si *numTones*.



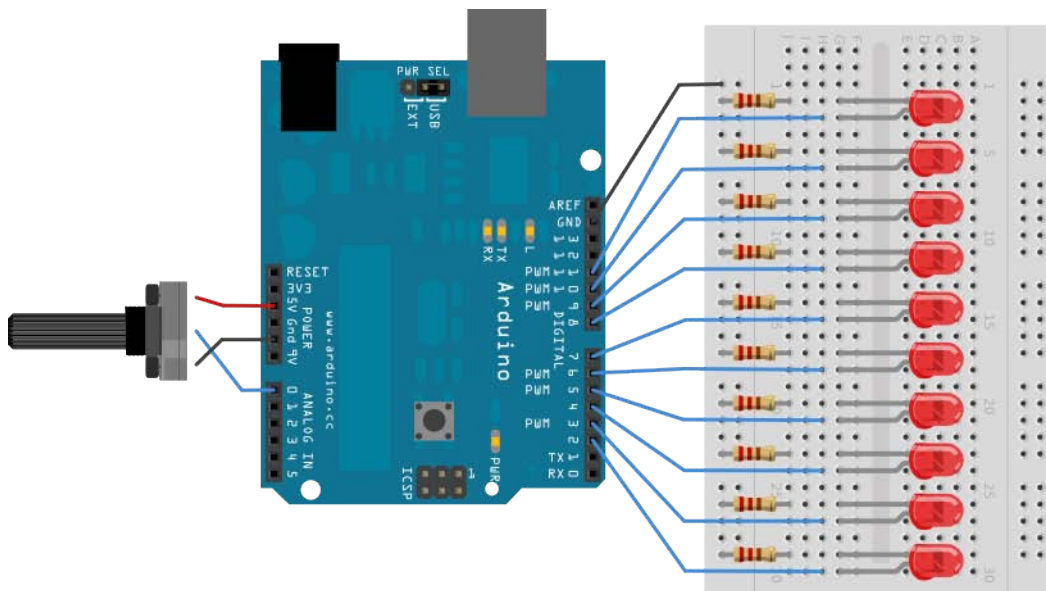
## Experimentul 15. Utilizarea unui bargraf cu LED-uri.

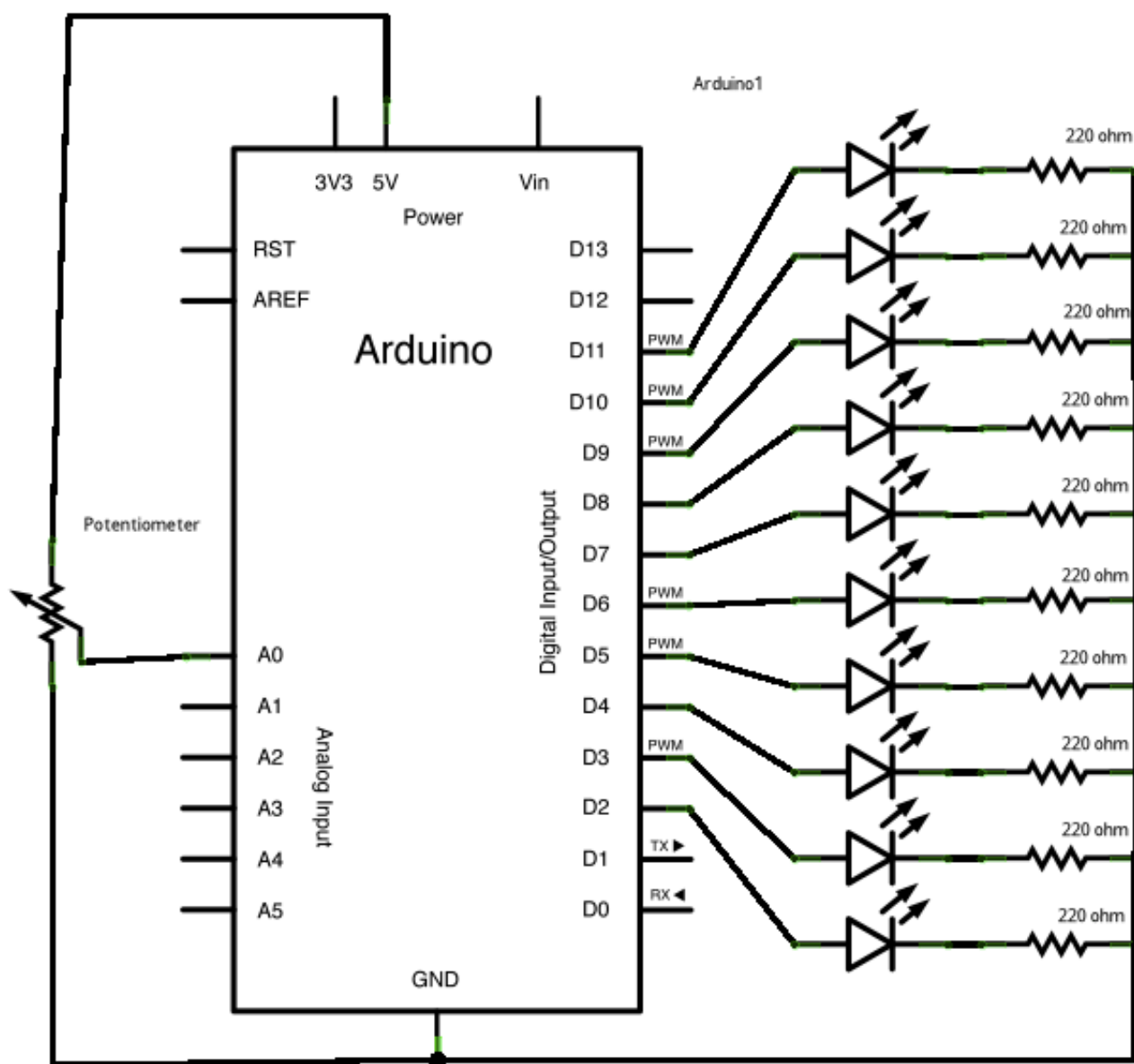
Bargraful, intalnit de cele mai multe ori in componente sistemelor audio pentru afisarea nivelului semnalului, este un dispozitiv utilizat, de cele mai multe ori, pentru a afisa starea senzorilor analogici. Acest exemplu demonstreaza modalitatea in care pot fi controlate LED-urile ce compun bargraful.

Programul functioneaza astfel: mai intai este citita starea intarii; apoi, valoarea obtinuta este convertita intr-un numar, in cazul nostru in intervalul 0 ~10, adica numarul LED-urilor care intra in componenta bargrafului. In bucla principala, functia *loop* sunt aprinse sau stinse LED-urile in functie de pozitia unghiulara a axului potentiometrului.



### Circuitul asociat



Schema electrica

**Codul sursa**

/\*

Utilizarea unui bargraf

Programul va aprinde sau stinge o serie de LED-uri in functie de valoarea intrarii analogice.

Aceasta este cea mai simpla varianta de a afisa starea unui senzor analogic prin utilizarea unui bargraf.

Metoda poate fi utilizata pentru a controla orice tip de iesiri digitale a caror functionare depinde de o intrare analogica.

Circuitul este compus din 10 LED-uri legate intre pinii digitali de la 2 la 11 si masa.

\*/

```
const int analogPin = A0; // pinul analogic asociat potentiometrului rotativ liniar
```

```
const int ledCount = 10; // numarul de LED-uri continute de bargraf
```

```
int ledPins[] = { 2, 3, 4, 5, 6, 7,8,9,10,11 }; //sir cu pinii asociati LED-urilor din bargraf
```

```
void setup() {
```

```
// setarea ca iesire a fiecaruia dintre pinii asociati LED-urilor
```

```
  for (int thisLed = 0; thisLed < ledCount; thisLed++) {
```

```
    pinMode(ledPins[thisLed], OUTPUT);
```

```
  }
```

```
}
```

```
void loop() {
```

```
//citirea valorii obtinute prin rotirea potentiometrului
```

```
  int sensorReading = analogRead(analogPin);
```

```
// conversia numarului rezultat intr-un numar in intervalul 0 ~numar de LED-uri din circuit
```

```
  int ledLevel = map(sensorReading, 0, 1023, 0, ledCount);
```

```
// o bucla pentru selectarea fiecarui LED al bargrafului
```

```
  for (int thisLed = 0; thisLed < ledCount; thisLed++) {
```

```
    // daca elementul sirului ocupa o pozitie in sir mai mica decat ledLevel,
```

```
    // LED-ul asociat acelui element va fi aprins:
```

```
    if (thisLed < ledLevel) {
```

```
      digitalWrite(ledPins[thisLed], HIGH);
```

```
    }
```

```
// daca elementul sirului ocupa o pozitie in sir mai mare decat ledLevel
```

```
// LED-ul asociat acelui element va fi stins
else {
    digitalWrite(ledPins[thisLed], LOW);
}
}
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa. Puteti copia codul sursa intr-un fisier nou sau il puteti accesa din **File -> Examples -> Display -> barGraph**.

## Experimentul 16. Bargraf controlat prin intermediul dispozitivului Arduino si a unui contor CD4017

În cadrul acestui experiment vom învăța să cream o animație simplă utilizând un bargraf cu LED-uri și dispozitivul Arduino. Pentru a reduce numărul de conexiuni vom utiliza un contor CD4017.

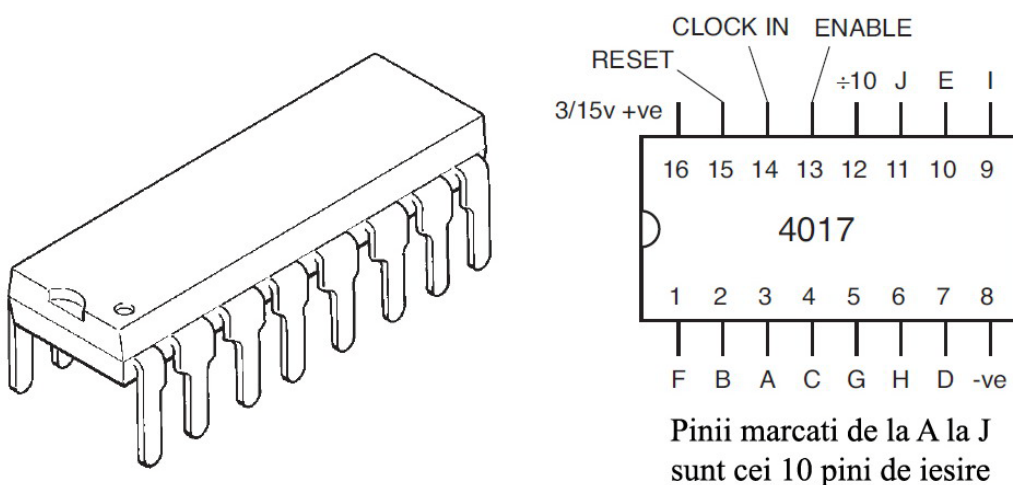
### Circuitul integrat CD4017

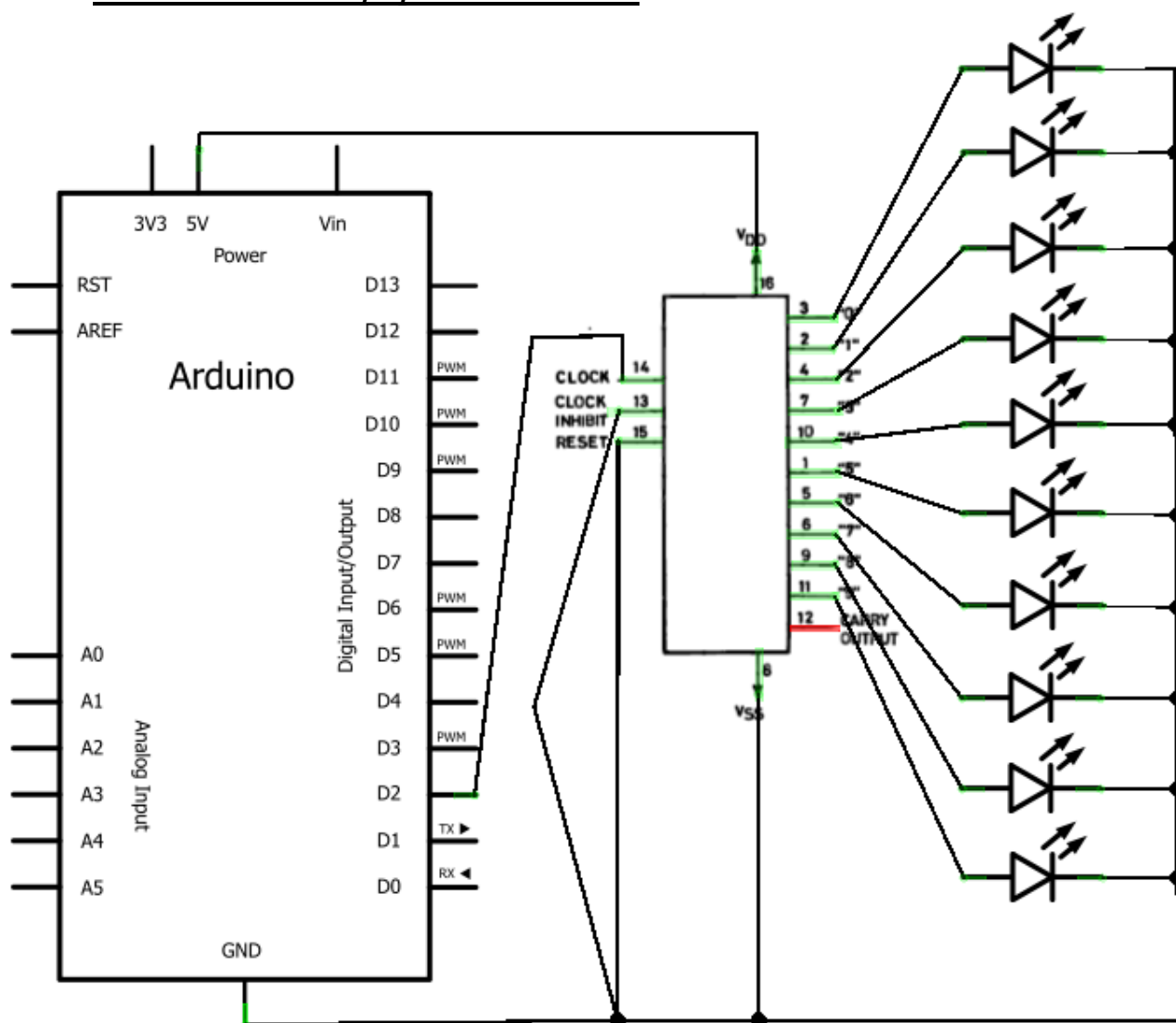
În general, contoarele digitale sunt utilizate pentru a esantiona un semnal, pentru a-l multiplexa și într-o gamă variată de situații care presupun numărarea și afișarea.

Circuitul integrat are 10 ieșiri, o intrare de ceas și pini de activare și resetare. Pini de activare și resetare sunt controlați prin modificarea valorii lor logice, în 0 sau 1.

Principiul de funcționare al contorului CD4017 este următorul: în momentul în care este recepționat un semnal de ceas (semnal dreptunghiular) în pinul 14 (pinul de intrare pentru ceas), fiecare dintre cele 10 ieșiri își schimbă, pe rând, valoarea logică din 0 în 1. În fiecare moment de timp, doar o singură ieșire din cele 10 poate avea valoarea logică 1, celelalte sunt 0. Dacă la fiecare ieșire a acestui circuit integrat se conectează câte un LED, atunci acestea se vor aprinde pe rând, cu o perioadă dată de trenul de impulsuri de ceas.

În condiții normale de utilizare, pinii 13 (de activare) și 15 (de resetare) sunt conectați la masă (GND sau valoarea logică 0). Dacă pinul 13 se conectează la valoarea logică 1 (plusul sursei de alimentare) circuitul integrat se va opri din numărarea impulsurilor. Dacă pinul 15 se conectează la valoarea logică 1 (plusul sursei de alimentare) circuitul integrat va reveni la starea inițială, numărând de la începutul secvenței.



*Schema de montare pe placa Breadboard*

Dupa cum observati, utilizand circuitul integrat CD4017, vom folosi doar un pin digital al dispozitivului Arduino. Altfel, am fi utilizat 10 pini digitali precum in experimentul precedent.

Montajul nostru functioneaza astfel: cele 10 LED-uri ale bargrafului sunt conectate la iesirile circuitului integrat CD4017. Pinul de ceas este conectat la Arduino. Atunci cand dispozitivul Arduino este alimentat, trimite impulsuri catre contor, incrementandu-l. O data cu incrementarea contorului, LED-ul curent este stins si urmatorul este aprins.

### **Codul sursa**

```
/*
 * Animatie simpla a unui bargraf utilizand CD4017 si Arduino
 */

/* Pinul utilizat pentru a trimite semnalul de ceas
 */

int clockPin = 2;

void setup() {

  pinMode(clockPin,OUTPUT);
}

/*
 * Trimitem un semnal de ceas pentru a incrementa contorul
 */

void clock() {

  digitalWrite(clockPin,HIGH);
  delay(1);
  digitalWrite(clockPin,LOW);

}

void loop() {

  /* Bucla principala trimite semnal de ceas contorului. Aceasta se incrementeaza si, ca urmare,
  LED-ul curent se va stinge si LED-ul urmator se va aprinde.
  */

  clock();

  /* Asteapta 300 ms. */

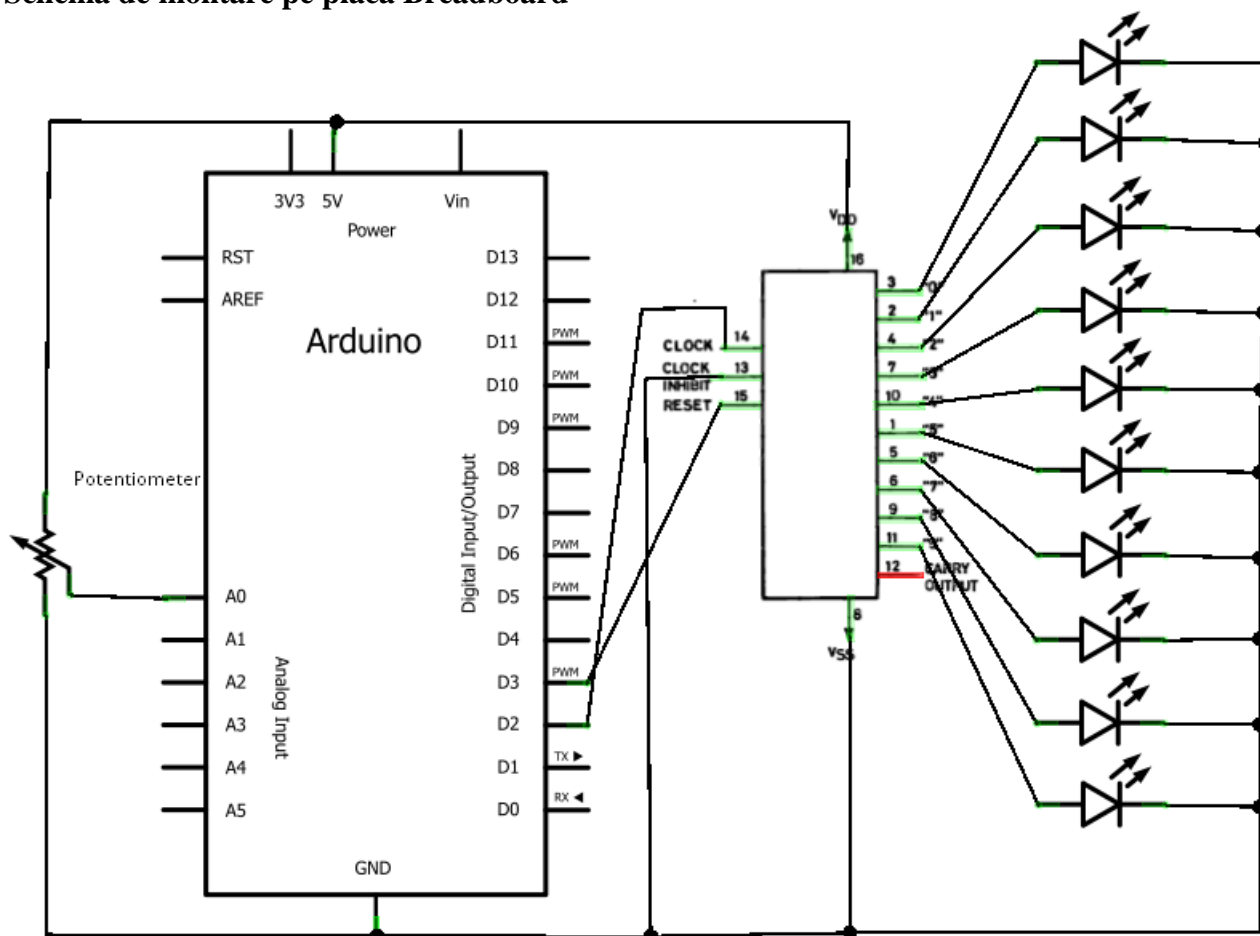
  delay(300);
}
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa.

## Experimentul 17. Bargraf controlat prin intermediul dispozitivului Arduino, contorului CD4017 si a unui potentiometru liniar

În cadrul acestui experiment vom învăța să cream o animație simplă utilizând un bargraf cu LED-uri, dispozitivul Arduino și un potentiometru liniar. Vom demonstra și conceptul de *persistentă a vederii*, adică tendința ochiului uman de a continua să vadă o imagine pentru încă un scurt interval de timp după ce aceasta a dispărut.

### Schema de montare pe placa Breadboard



Montajul nostru funcționează astfel: cele 10 LED-uri ale bargrafului sunt conectate la ieșirile circuitului integrat CD4017. Nu există necesitatea de a le proteja prin limitarea curentului cu rezistoare, întrucât curentul maxim pentru fiecare pin al circuitului integrat este de 10mA. Dispozitivul Arduino controlează circuitul integrat CD4017 prin intermediul a doi pini digitali: pinul D2 este conectat la ceasul contorului iar pinul D3 la pinul de resetare. Dispozitivul Arduino rulează un program care citește intrarea analogică obținută de la potentiometru (returnând o



valoare între 0 și 1023). Valoarea citită este convertită într-un număr aflat în intervalul 0 ~ 9, acesta reprezentând numărul LED-urilor care se vor aprinde. Utilizând contorul CD4017 nu vom putea ilumina mai multe LED-uri simultan. Pentru a rezolva această situație, bazându-ne pe principiul persistenței vederii, vom aprinde și stinge LED-urile pe rând, într-un timp foarte scurt, astfel încât să pară că se aprind simultan.

### **Codul sursa**

```

/*
 * Persistența vederii – controlul unui bargraf cu LED-uri utilizând un potentiometru extern
 */

/*Pinul digital conectat la pinul de ceas al contorului */
int clockPin = 2;

/* Pinul digital conectat la pinul de reset al contorului */
int resetPin = 3;

/* Pinul analogic asociat potentiometrului */
int potPin = 0;

void setup() {

  pinMode(clockPin,OUTPUT);
  pinMode(resetPin,OUTPUT);
  reset();

}

void loop() {

  /*Citirea valorii analogice a potentiometrului */
  int potValue = analogRead(potPin);

  /* Conversia în valoare de la 0 la 9*/
  int n = potValue * 10 / 1024;

  /*Aprinderea și stingerea rapidă a primelor n LED-uri. Datorită efectului de persistență a
  vederii, va părea că LED-urile se aprind simultan.
  */

  for( int i = 0; i < n; i++ ) {
    clock();
  }
}

```

```
    reset();  
}  
  
/*  
 *Trimitem semnal de ceas controlui pentru a-l incrementa  
 */  
void clock() {  
  
    digitalWrite(clockPin,HIGH);  
    delay(1);  
    digitalWrite(clockPin,LOW);  
  
}  
  
/*  
 *Resetarea contorului.  
 */  
void reset() {  
  
    digitalWrite(resetPin,HIGH);  
    delay(1);  
    digitalWrite(resetPin,LOW);  
  
}
```

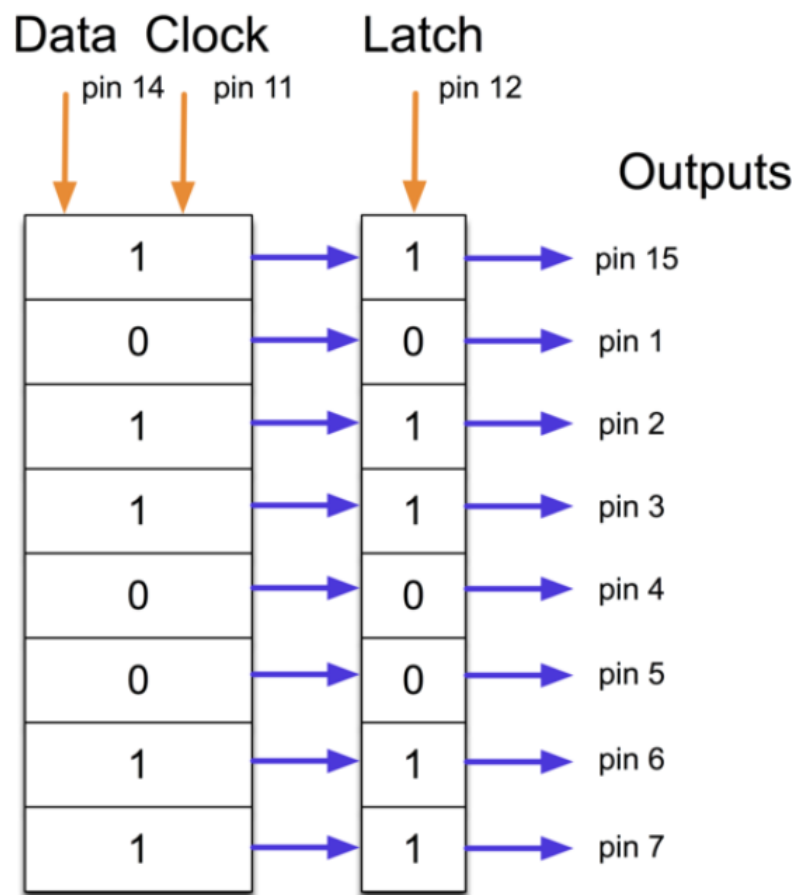
1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa.



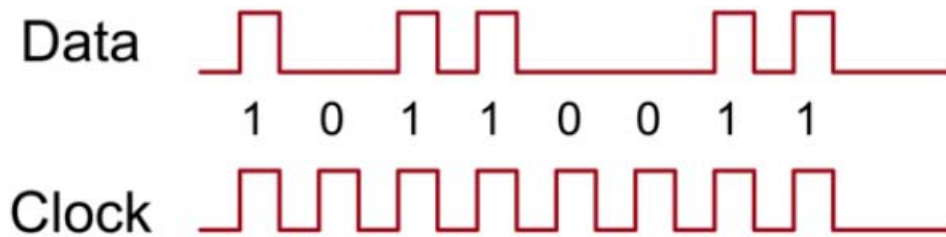
Intrucat avem de conectat 8 LED-uri si 8 rezistoare, va trebui sa facem destul de multe conexiuni. Cel mai usor, este sa plasam mai intai circuit integrat 74HC595, intrucat toate celelalte componente se vor conecta la aceasta. Sapte dintre cele opt iesiri ale acestui circuit se regasesc pe o parte a sa, astfel ca, veti plasa aceasta parte catre zona in care montati LED-urile. Dupa instalarea circuitului integrat, montati si rezistoarele. Trebuie sa aveti grija ca terminalele rezistoarelor sa nu se atinge intre ele si sa verificati acest lucru inca o data inainte de a alimenta dispozitivul Arduino. Apoi, plasati LED-urile pe placa breadboard. Terminalul pozitiv, mai lung, trebuie sa fie indreptat catre circuitul integrat. Urmeaza sa plasati firele de legatura asa cum este prezentat in imagine. Nu uitati ca unul dintre pinii circuitului integrat, numarul 8, se conecteaza la pinul GND al placii breadboard. Dupa ce ati verificat corectitudinea conexiunilor incarcati programul in memoria dispozitivului Arduino si rulati-l. Fiecare LED trebuie sa ilumineze pe rand, pana cand se vor aprinde toate, apoi LED-urile se vor stinge si ciclul va fi reluat.

### Registrul de deplasare 74HC595

Inainte de a analiza codul sursa, vom incerca sa intelegem principiul de functionare al registrului de deplasare. La nivel logic, acesta contine opt locatii de memorie, cu valoarea 1 sau 0. Pentru a modifica aceste valori, trebuie sa ii furnizam date utilizand pinii *Data* si *Clock*.



Pinul de ceas trebuie sa receptioneze opt impulsuri; daca in momentul receptionarii impulsului pinul de date este activat (HIGH) atunci in registrul de deplasare este inregistrata valoarea 1; altfel, este inregistrata valoarea 0. Dupa ce ai fost receptionate toate impulsurile, activarea pinului *Latch* (de blocare) copiaza cele opt valori intr-un registru de date. Fara aceasta operatie de copiere, LED-urile care trebuie sa fie dezactivate ar clipi intermitent cat timp se incarca datele in registrul de deplasare. Circuitul integrat are si un pin de activare a iesirii (OE), utilizat pentru a activa sau dezactiva toate cele opt iesiri simultan.



Acest poate fi conectat la un pin PWM al dispozitivului Arduino si prin intermediul functiei *analogWrite* poate fi folosit pentru a controla luminozitatea LED-urilor. Pinul este activ cand nu receptioneaza semnal, deci il vom conecta la GND.

### Codul sursa

Arduino include o functie speciala denumita *shiftOut* construita special pentru a reimita date catre registrele de deplasare. Codul sursa este cel prezentat in continuare:

```
/*
Utilizarea a 8 LED-uri cu registru de deplasare
*/
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
byte leds = 0;

void setup(){

pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
}
```

```
void loop(){  
  
  leds = 0;  
  updateShiftRegister();  
  delay(500);  
  for (int i = 0; i < 8; i++)  
  {  
    bitSet(leds, i);  
    updateShiftRegister();  
    delay(500);  
  }  
}
```

```
void updateShiftRegister(){  
  
  digitalWrite(latchPin, LOW);  
  shiftOut(dataPin, clockPin, LSBFIRST, leds);  
  digitalWrite(latchPin, HIGH);  
}
```

Înainte de toate, vom defini cei trei pini pe care îi utilizăm. Aceștia vor fi ieșiri digitale ale dispozitivului Arduino ce se conectează la pinii de blocare, ceas și date ai circuitului integrat 74HC595.

```
int latchPin = 5;  
int clockPin = 6;  
int dataPin = 4;
```

Apoi, definim variabila *leds*. Aceasta va fi utilizată pentru a memora tiparul după care LED-urile se aprind sau sting. O structură de date de tip *byte* (număr binar) formată din 8 biți (valori 0 sau 1) reprezintă această variabilă. Fiecare dintre aceste valori reprezintă starea aprins (1) sau stins (0) pentru LED-urile conectate în circuit.

```
byte leds = 0;
```

Funcția *setup* setează cei trei pini ca ieșiri.

```
void setup(){  
  
  pinMode(latchPin, OUTPUT);  
  pinMode(dataPin, OUTPUT);  
  pinMode(clockPin, OUTPUT);
```

```
}
```

Functia *loop* stinge initial toate LED-urile, prin modificare variabile *leds*. Apoi apeleaza functia *updateShiftRegister* care va transmite tiparul continut in variabila *leds* catre registrul de deplasare, astfel incat LED-urile sa se stinga. Se face o pauza de o jumatate de secunda, apoi, prin intermediul unei rutine de tip *for* este analizat fiecare element al sirului *leds*. Functia *bitSet* seteaza pe 1 fiecare element al sirului *leds* apoi este apelata functia *updateShiftRegister* pentru a se aprinde LED-urile. Intre fiecare dintre aceste modificari ale variabilei *leds* este facuta o pauza de o jumatate de secunda, pana cand urmatorul LED este aprins.

```
void loop(){  
  
  leds = 0;  
  updateShiftRegister();  
  delay(500);  
  for (int i = 0; i < 8; i++)  
  {  
    bitSet(leds, i);  
    updateShiftRegister();  
    delay(500);  
  }  
}
```

Functia *updateShiftRegister*, seteaza pinul de blocare pe LOW, apoi apeleaza functia *shiftOut* inainte de a activa pinul de blocare.

Functia *shiftOut* are patru parametri: doi dintre acestia sunt pinii utilizat pentru data si ceas; al treilea parametru specifica bitul de la care se va porni evaluarea. In cazul de fata vom porni de la bitul din partea dreapta, Least Significant Bit (LSB) – adica bitul cu valoarea cea mai mica. Ultimul parametru este reprezentat de datele ce vor fi inscise in registrul de deplasare, in cazul de fata vectorul *leds*.

```
void updateShiftRegister(){  
  
  digitalWrite(latchPin, LOW);  
  shiftOut(dataPin, clockPin, LSBFIRST, leds);  
  digitalWrite(latchPin, HIGH);  
  
}
```

### Controlul intensitatii luminoase

Unul dintre pinii circuitului integrat 74HC595 despre care am discutat este cel de activare a iesirii (Output Enable - OE). Acesta este pinul 13 al circuitului integrat si, in montajul nostru, a fost montat permanent la GND. Pinul se comporta precum un comutator care poate activa sau dezactiva iesirile. Astfel, daca va fi conectat la 5V, iesirile vor fi dezactivate. Daca va fi conectat la masa (GND), LED-urile care trebuie sa ilumineze vor fi aprinse, iar celelalte, stinse.

Utilizand acest pin impreuna cu functia *analogWrite* va controla intensitatea luminoasa a LED-urilor. Pentru a realiza acest montaj, vom deconecta pinul 13 al circuitului integrat 74HC595 de la masa si il vom conecta la pinul 3 al dispozitivului Arduino. Codul sursa care urmeaza va face LED-urile sa se aprinda gradual pentru ca apoi sa se stinga.

```
/*  
Modificarea intensitatii luminoase a unui sir de 8 LED-uri utilizand circuitul integrat 74HC595  
*/  
  
int latchPin = 5;  
int clockPin = 6;  
int dataPin = 4;  
int outputEnablePin = 3;  
byte leds = 0;  
  
void setup(){  
  
  pinMode(latchPin, OUTPUT);  
  pinMode(dataPin, OUTPUT);  
  pinMode(clockPin, OUTPUT);  
  pinMode(outputEnablePin, OUTPUT);  
}  
  
void loop(){  
  
  setBrightness(255);  
  leds = 0;  
  updateShiftRegister();  
  delay(500);  
  for (int i = 0; i < 8; i++)  
  {  
    bitSet(leds, i);  
    updateShiftRegister();  
    delay(500);  
  }  
  for (byte b = 255; b > 0; b--)  
  {  
    setBrightness(b);
```



```
delay(50);
}
}

void updateShiftRegister(){

digitalWrite(latchPin, LOW);
shiftOut(dataPin, clockPin, LSBFIRST, leds);
digitalWrite(latchPin, HIGH);

}

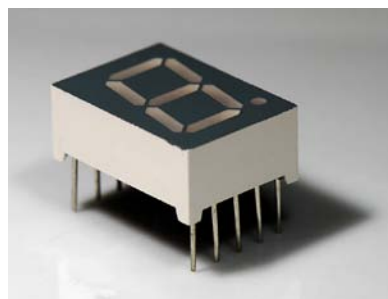
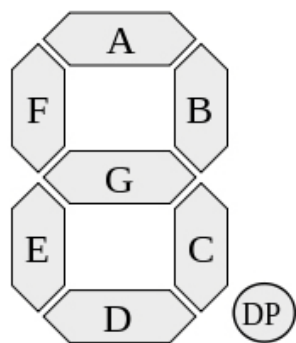
void setBrightness(byte brightness){ // setare luminozitate de la 0 la 255
analogWrite(outputEnablePin, 255-brightness);
}
```

1. Realizati montajul prezentat si incarcati in dispozitivul dvs. Arduino codul sursa pentru fiecare dintre cele doua experimente prezentate.

## Afisaje cu 7 segmente

Unul dintre cele mai raspandite tipuri de afisoare digitale este cel cu diode LED dispuse pe 7 segmente, acesta fiind numarul minim de segmente pe care se pot afisa cifre de la 0 la 9. Aceste dispozitive sunt utilizate pe scară largă în ceasuri digitale, contoare electronice și alte aparate, pentru afișarea informațiilor numerice. Cele sapte segmente au fost notate cu literele *a*, *b*, *c*, *d*, *e*, *f*, *g*, după modelul din figura.

Segmentele sunt aranjate sub forma unui dreptunghi format din cate doua segmente verticale pe latura mai mare si un segment orizontal pe latura mai mica. In pus, al saptelea segment imparte dreptunghiul in doua parti egale. Exista si versiuni cu 14 segmente si cu 16 segmente (pentru afisarea simbolurilor alfa-numerice); acestea au fost insa inlocuite de matricele cu LED-uri. Fiecare afisaj cu 7 segmente materializeaza un digit al unui afisaj complet. Fiecare digit are 8 terminale, cate unul pentru fiecare segment plus unul pentru conexiunea comuna. In unele cazuri in afisaj se introduce si un punct zecimal denumit p (sau dp), in acest caz fiind 9 terminale.

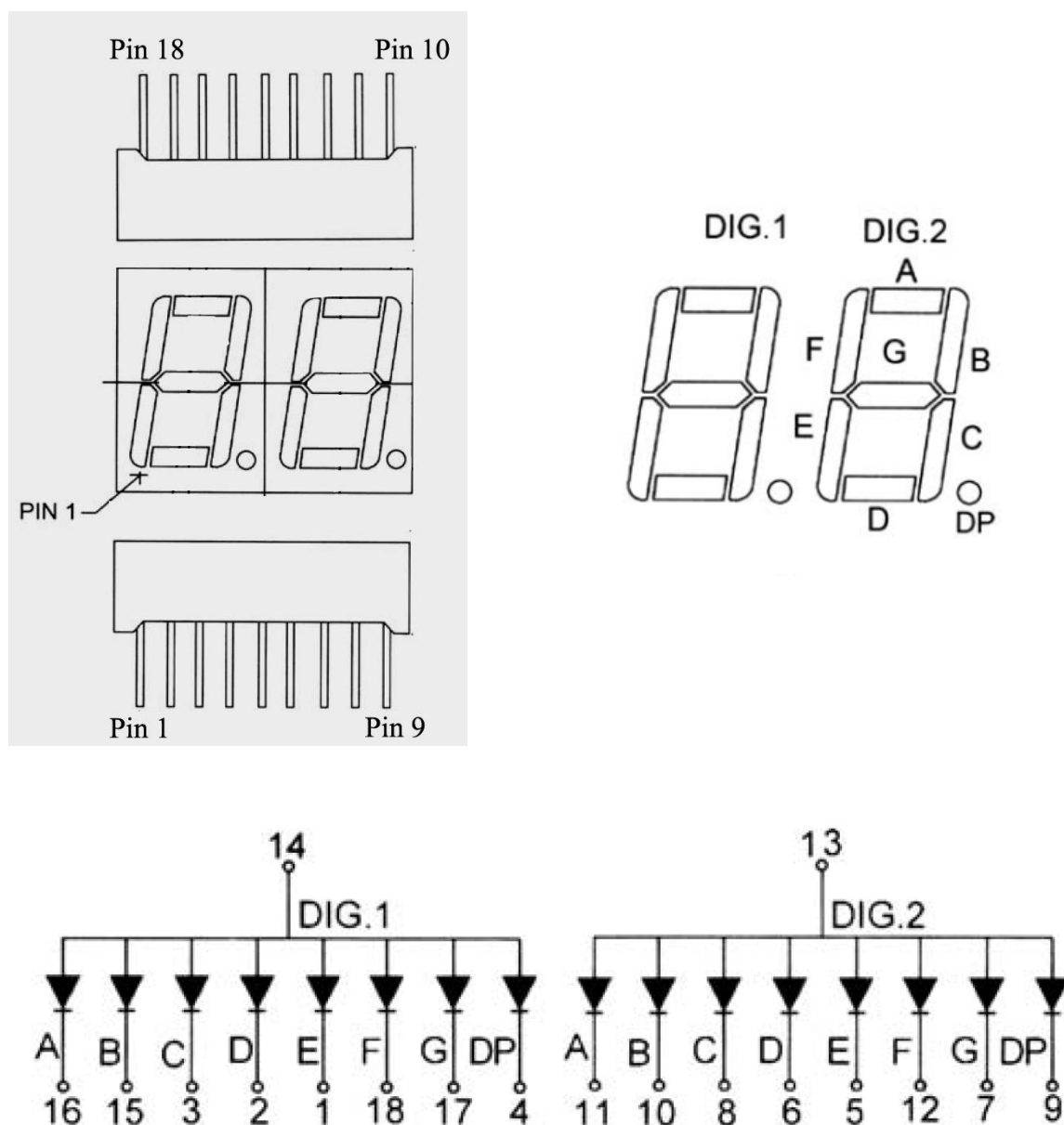


**Figura. Afisaj cu 7 segmente.**

Afișarea uneia din cifrele de la 1 la 9 constă în aprinderea unor anumite segmente din cele 7, după cum urmează:

Cifra	0	1	2	3	4
Segmentele aprinse	a,b,c,d,e,f	b, c	a,b, d,e,g	a,b,c, d, g	b,c,f,g
Cifra	5	6	7	8	9
Segmentele aprinse	a,c, d,f,g	a,c, d,e,f,g	a,b, c	a,b,c, d,e,f,g	a,b,c, d,e,g

Afisajul pe care il veti utiliza in experimentele urmatoare este format din doi digiti cu 7 segmente, cu anod comun. Schema electrica si configuratia pinilor acestuia sunt prezentate in figura de mai jos. Configuratia acestui afisaj permite ca digitii care il compun sa poata fi controlati individual. In conditiile in care sistemul software poate controla digitii simultan, pinul corespunzator segmentului A al digitului 1 se conecteaza la pinul corespunzator segmentului A al digitului 2, pinul corespunzator segmentului B al digitului 1 se conecteaza la pinul corespunzator segmentului B al digitului 2, s.a.m.d. pana la pinul DP. Pinii de anod ai celor doi digiti **nu** se vor conecta intre ei.



*Figura. Afisaj cu 2 digiti compusi din 7 segmente. Configuratia pinilor si diagrama electrica.*

## Experiment 19. Rularea cifrelor de la 0 la 9 pe un afisaj cu 7 segmente utilizand dispozitivul Arduino

Experimentul pe care il vom prezenta ne va ajuta sa intelegem modalitatea de functionare a afisajelor cu 7 segmente. Cu ajutorul dispozitivului Arduino si a unui afisaj cu 7 segmente vom afisa cifrele de la 0 la 9.

### Schema de montare pe placa Breadboard

Cifrele vor fi afisate pe afisajul cu 7 segmente prin activarea unor combinatii de segmente. Pentru a proteja afisajul este, de preferat, sa conectam in serie cu fiecare segment cate un rezistor de  $270\Omega$  asa cum este prezentat in schema de mai jos.

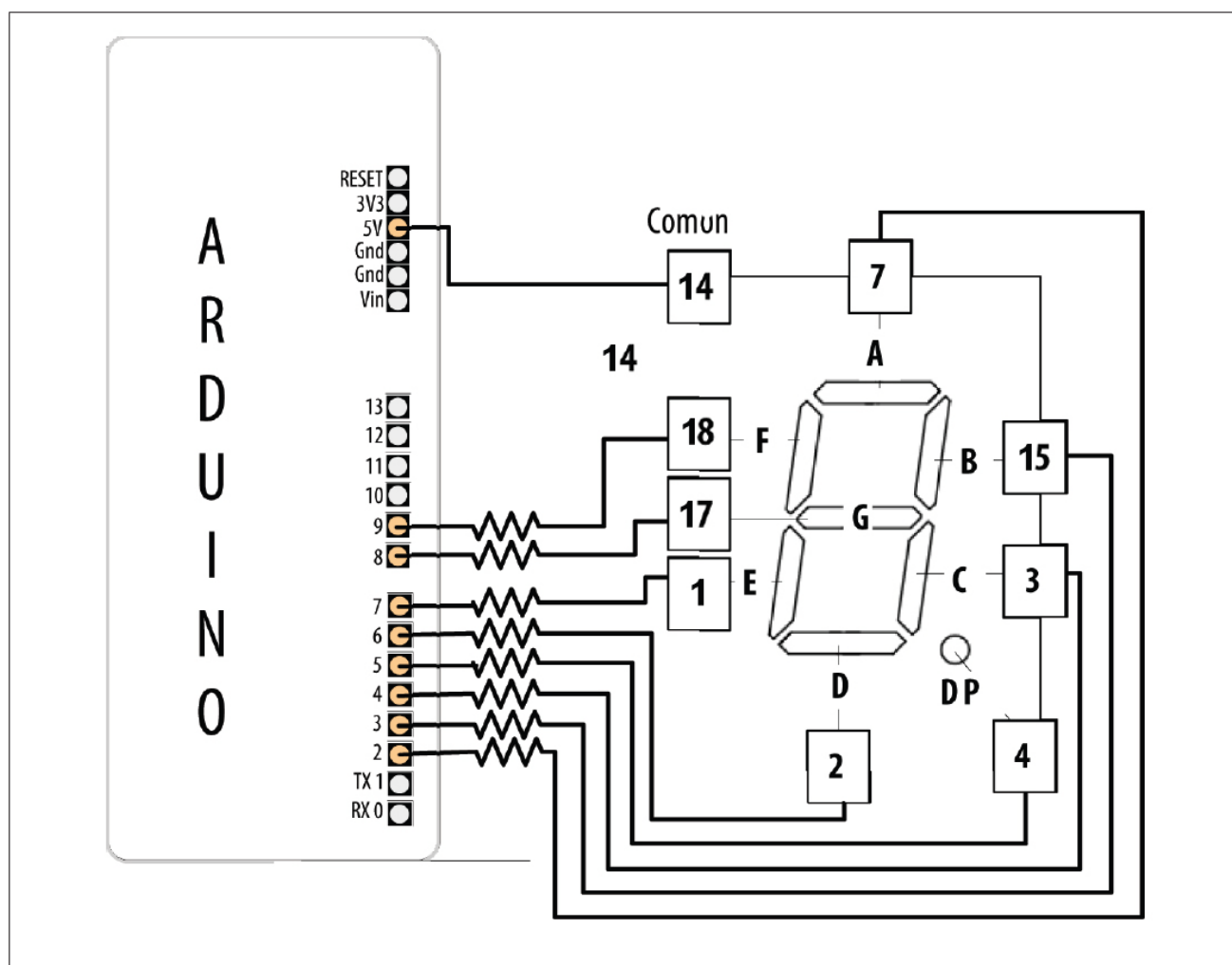


Figura 19.1. Conectarea afisajului cu 7 segmente la dispozitivul Arduino

**Codul sursa**

```
/*
* Prezentarea cifrelor de la 0 la 9 pe un singur digit al afisajului cu 7 segmente
* Acest exemplu numara secunde de la 0 la 9
*/

//fiecare bit reprezinta segmentele de la A la G (si punctul zecimal) pentru cifre de la 0 //pana la
9

const byte numeral[10] = {
//ABCDEFG /dp
B11111100, // 0
B01100000, // 1
B11011010, // 2
B11110010, // 3
B01100110, // 4
B10110110, // 5
B00111110, // 6
B11100000, // 7
B11111110, // 8
B11100110, // 9
};
// pinii pentru punctul decimal si fiecare dintre segmente
// dp,G,F,E,D,C,B,A
const int segmentPins[8] = {5,9,8,7,6,4,3,2};

void setup(){

for(int i=0; i < 8; i++)
{
pinMode(segmentPins[i], OUTPUT); // setare segmentelor si a lui DP ca pini de iesire
}
}

void loop(){

for(int i=0; i <= 10; i++)
{
showDigit(i);
delay(1000);
}

// ultima valoare, pentru i = 10 va stinge afisajul
delay(2000); // o pauza de 2 secunde
```

```

}

// Afisarea cifrelor
// orice valoare in afara intervalului 0-9 stinge afisajul

void showDigit( int number)
{
  boolean isBitSet;
  for(int segment = 1; segment < 8; segment++)
  {
    if( number < 0 || number > 9){
      isBitSet = 0; // stinge toate segmentele
    }

    else{

      // isBitSet va fi adevarat daca bitul dat este 1

      isBitSet = bitRead(numeral[number], segment); //bitRead(numar, pozitie_bit) evalueaza
      //valoarea bitului pozitie_bit din cadrul unui numar binar
    }
    isBitSet = ! isBitSet;
    digitalWrite( segmentPins[segment], isBitSet);
  }
}

```

Segmentele care urmeaza a fi aprinse sunt stocate intr-un sir denumit *numeral*. Fiecare dintre bitii acestei secvente reprezinta unul dintre cele sapte segmente si punctul zecimal, care nu va fi iluminat in aceasta situatie.

Sirul denumit *segmentPins* contine informatii despre pinii asociati fiecarui segment.

Functia *showDigit* verifica daca numerele sunt in intervalul 0 ~ 9, si, daca este respectata aceasta conditie, verifica fiecare bit si activeaza pinul corespunzator daca bitul este setat (adica egal cu 1).

Pinul va fi setat pe HIGH daca segmentul are catod comun, si pe LOW (in cazul afisajului pe care il aveti la dispozitie dvs.) daca segmentul are anod comun.

Intrucat afisajul pe care il utilizam este cu anod comun, vom inversa valoarea (adica transformam 0 in 1 si 1 in 0) pe care o obtinem prin intermediul liniei de cod:

```
isBitSet = ! isBitSet;
```

Ultima linie de cod seteaza starea fiecarui pin de iesire al dispozitivul Arduino conectat la afisajul cu 7 segmente.

## Experiment 20. Construirea unui zar electronic utilizand afisaj compus din doi digiti cu 7 segmente

Componente utilizate	
Component	Descriere
	Arduino Uno sau dispozitiv echivalent cu acesta
D1	Afisaj format din doi digiti cu 7 segmente
R1	Rezistor 100 K $\Omega$
R4-11	Rezistor 270 $\Omega$
R2, R3	Rezistor 1 K $\Omega$
T1, T2	Tranzistor PNP BC557
S1	Comutator cu revenire

### Schema de montare pe placa Breadboard

Schema de asamblare a proiectului este prezentata in figura urmatoare. Modulul de afisare cu 7 segmente pe care il utilizam pentru acest experiment este cu anod comun. De aceea, pentru a controla fiecare dintre digiti este necesar sa conectam, pe rand, din codul sursa, pinul pozitiv al sursei de alimentare la anodul fiecarui digit.

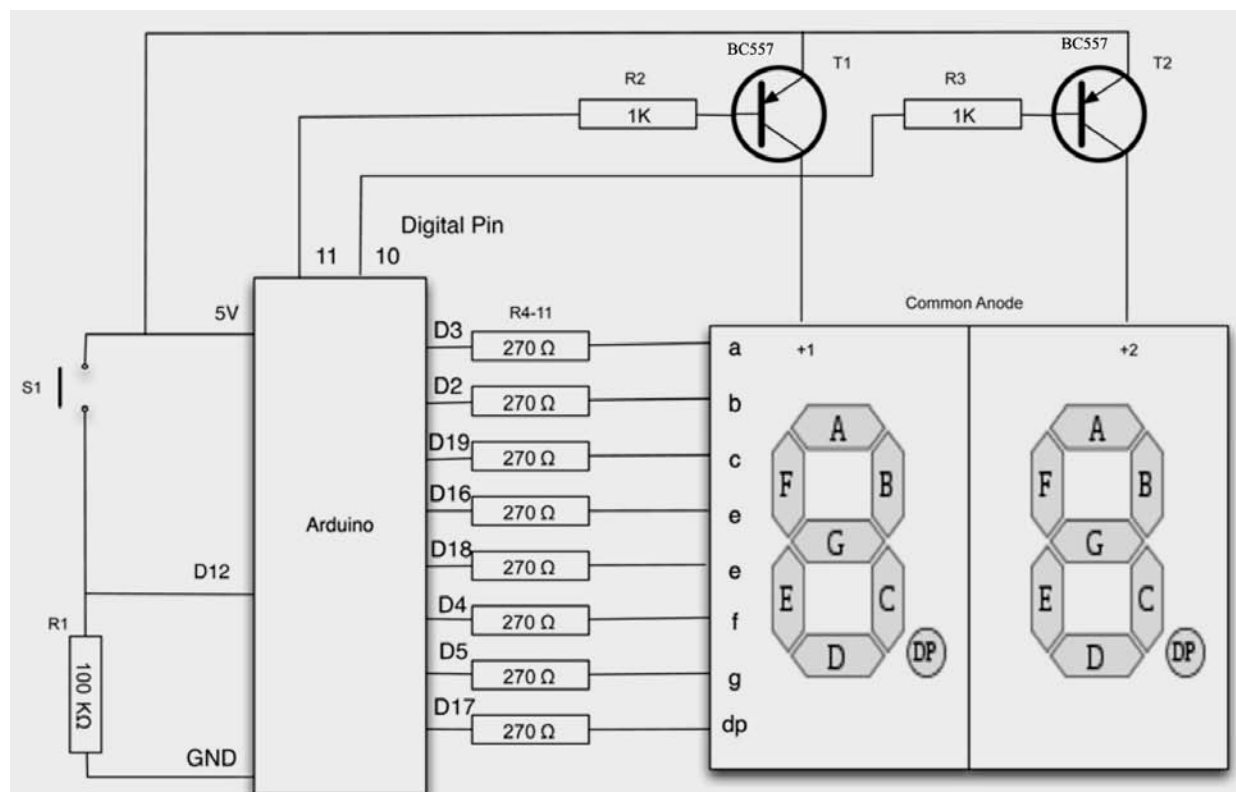
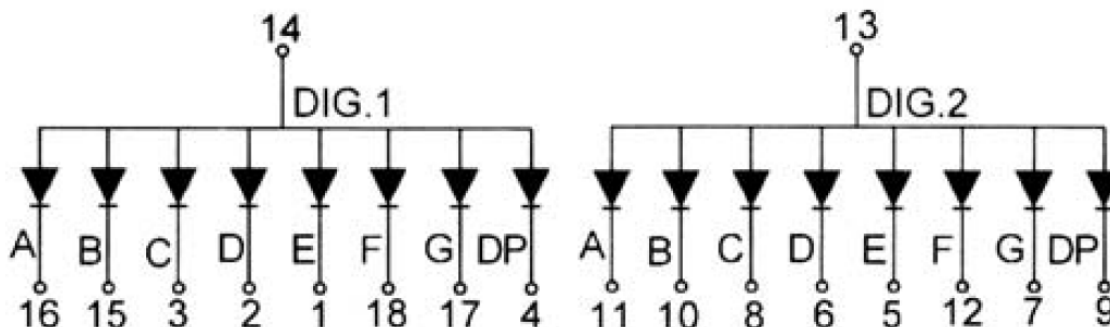


Figura 20.1. Schema de montare a zarului electronic

Pentru a conecta ambii digiti ai afisajului nostru la dispozitivul Arduino va fi nevoie sa realizam cateva conexiuni suplimentare. Vom analiza din nou schema afisajului pe care il avem la dispozitie.



Conexiunile suplimentare vor fi: terminalul 16 al afisajului se va conecta la terminalul 11, terminalul 15 la terminalul 10, si asa mai departe conform tabelului.

Conexiuni intre pinii afisajului dublu cu 7 segmente		
Semnificatie	Conexiune intre pinii afisajului	Conectare la Arduino
Pin A	16 conectat la 11	Pin D3
Pin B	15 conectat la 10	Pin D2
Pin C	3 conectat la 8	Pin D19
Pin D	2 conectat la 6	Pin D16
Pin E	1 conectat la 5	Pin D18
Pin F	18 conectat la 12	Pin D4
Pin G	17 conectat la 7	Pin D5
Pin DP	4 conectat la 9	Pin D17

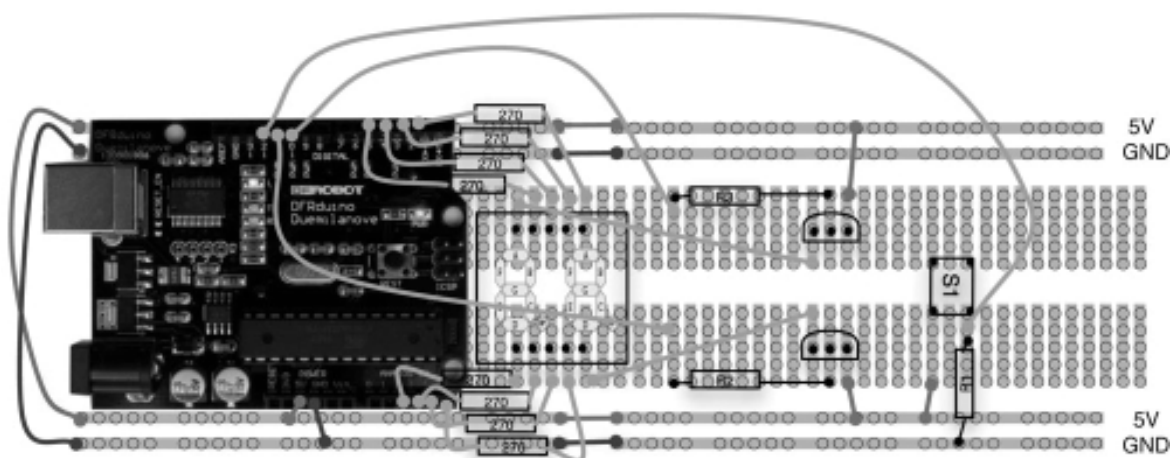
Avand in vedere ca vom controla pinul pozitiv de alimentare, vom utiliza tranzistoare de tip PNP, asa cum este prezentat si in schema. In cazul in care am utiliza un afisaj cu catod comun ar trebui sa utilizam tranzistoare NPN dar am controla pinul negativ al sursei de alimentare.

Disponerea componentelor pe placa breadboard este prezentata in imaginile care urmeaza. Pentru a reduce pe cat posibil numarul de fire de legatura utilizate este de preferat ca dispozitivul Arduino sa fie plasat cat mai aproape de afisajul cu 7 segmente. Aceasta inseamna ca rezistoarele vor fi destul de aproape unul de celalalta si, va trebuie sa avem grija ca acestea sa nu se atinga intre ele. Daca nu sunteti sigur ca rezistoarele nu vor intra in contact unul cu celalalt, nu efectuati experimentul asa cum este prezentat in figura, ci utilizati fire de legatura.

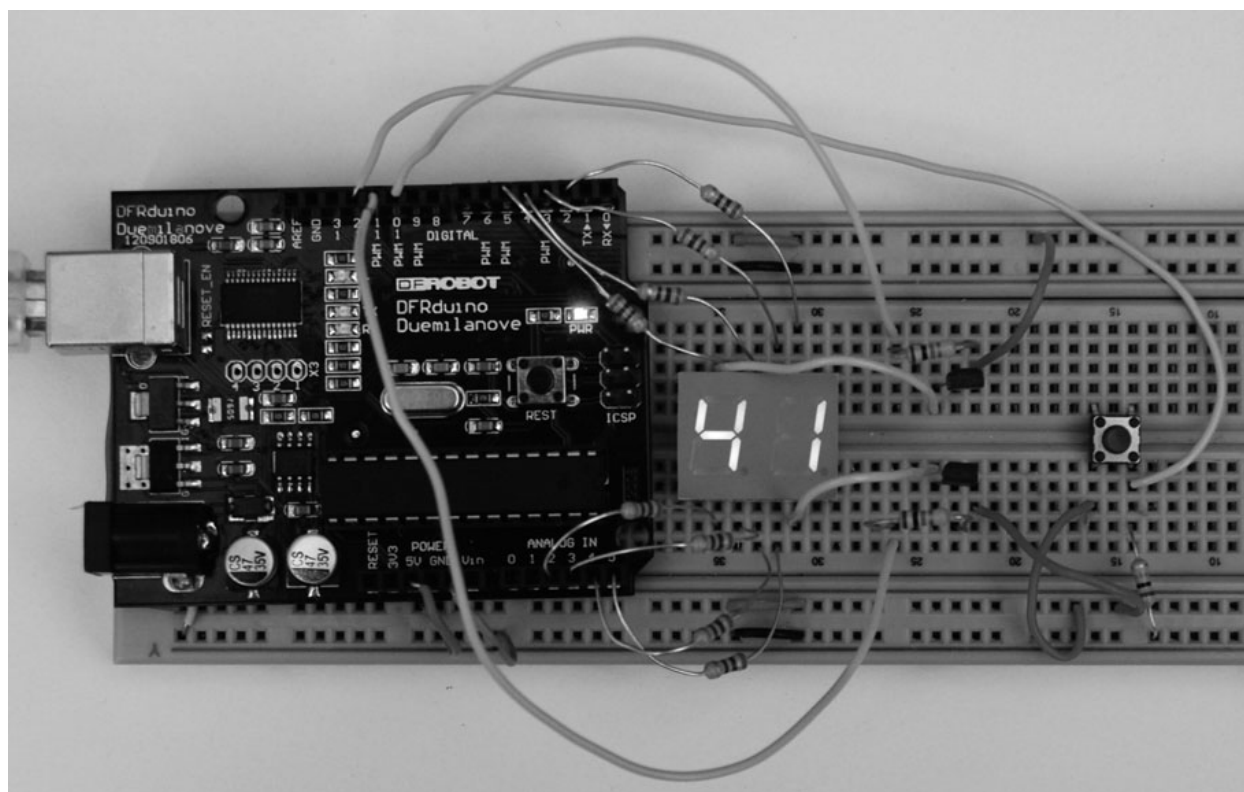


### Codul sursa

Vom utiliza un sir care contine pinii conectati la fiecare dintre segmentele afisajului cu 7 segmente si la punctul zecimal. Vom utiliza si o matrice care determina ce segmente vor fi aprinse pentru a afisa o anumita cifra. Fiecare linie a matricei reprezinta cifra si fiecare coloana segmentul care va fi aprins sau stins.



*Figura 20.2. Model de montare pe breadboard*



*Figura 20.3. Imagine a montajului pentru zar electronic*

```
int segmentPins[] = {3, 2, 19, 16, 18, 4, 5, 17};
int displayPins[] = {10, 11};
int buttonPin = 12;
byte digits[10][8] = {
// a b c d e f g .
{ 1, 1, 1, 1, 1, 1, 0, 0}, // 0
{ 0, 1, 1, 0, 0, 0, 0, 0}, // 1
{ 1, 1, 0, 1, 1, 0, 1, 0}, // 2
{ 1, 1, 1, 1, 0, 0, 1, 0}, // 3
{ 0, 1, 1, 0, 0, 1, 1, 0}, // 4
{ 1, 0, 1, 1, 0, 1, 1, 0}, // 5
{ 1, 0, 1, 1, 1, 1, 1, 0}, // 6
{ 1, 1, 1, 0, 0, 0, 0, 0}, // 7
{ 1, 1, 1, 1, 1, 1, 1, 0}, // 8
{ 1, 1, 1, 1, 0, 1, 1, 0} // 9
};
void setup()
{
for (int i=0; i < 8; i++)
{
pinMode(segmentPins[i], OUTPUT);
}
pinMode(displayPins[0], OUTPUT);
pinMode(displayPins[1], OUTPUT);
pinMode(buttonPin, INPUT);
}

void loop(){

static int dice1;
static int dice2;

if (digitalRead(buttonPin))

{
dice1 = random(1,7);
dice2 = random(1,7);
}

updateDisplay(dice1, dice2);
}
```

```
void updateDisplay(int value1, int value2){

digitalWrite(displayPins[0], HIGH);
digitalWrite(displayPins[1], LOW);
setSegments(value1);
delay(5);
digitalWrite(displayPins[0], LOW);
digitalWrite(displayPins[1], HIGH);
setSegments(value2);
delay(5);
}

void setSegments(int n){

for (int i=0; i < 8; i++)
{
digitalWrite(segmentPins[i], ! digits[n][i]);
}

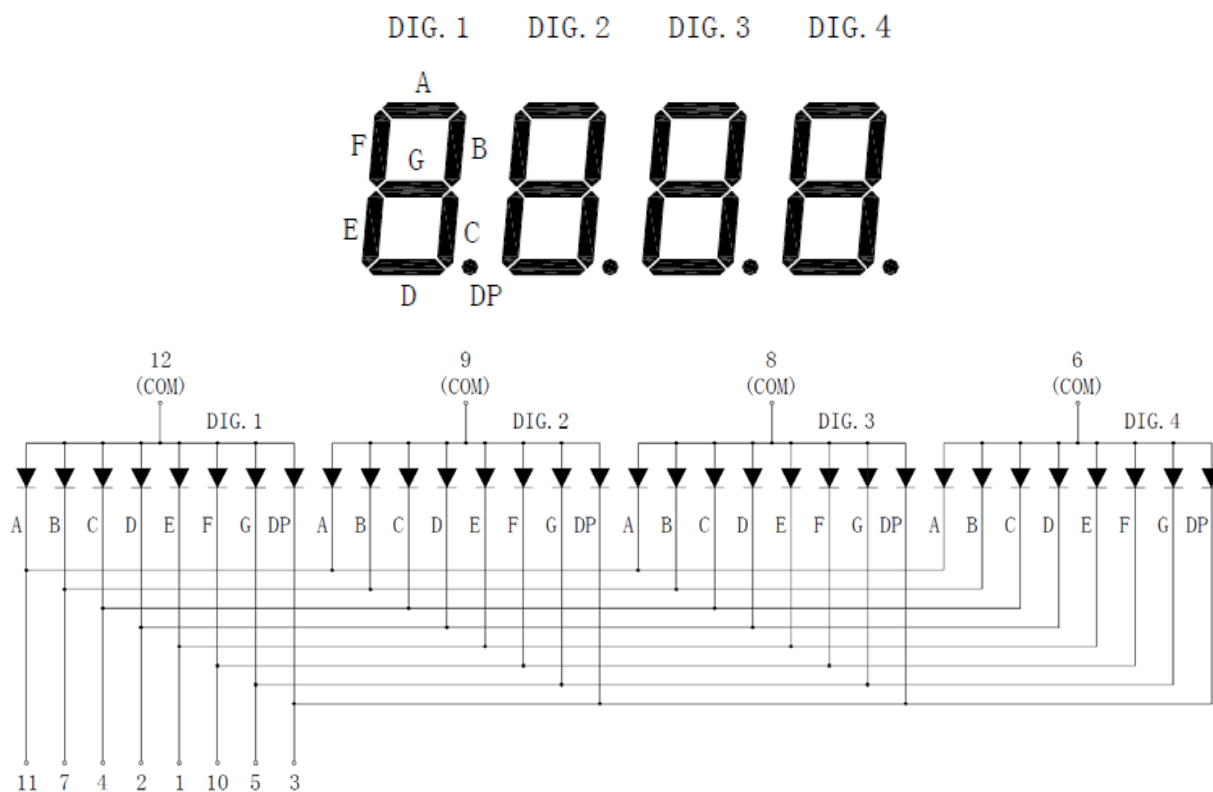
}
```

Pentru a controla ambii digiti, este necesar sa le activam pe rand si sa setam care dintre segmente sa fie aprinse. Astfel, functia noastra principala trebuie sa retina valorile care sunt afisate pe fiecare dintre digiti: *dice1* si *dice2*.

Pentru a simula aruncarea zarului, utilizam o functie de generare a unor numere aleatoare, si atunci cand butonul este apasat, generam numerele *dice1* si *dice2*.

## Experiment 21. Afisarea unui numar pe un afisaj cu 4 digiti formati din 7 segmente

Pentru a utiliza afisaje cu 4 digiti vom utiliza multiplexarea. Schema de conexiune a afisajului cu 4 digiti este cea prezentata in figura de mai jos. Observati ca pinii de la *a* la *g* si *dp* ai celor patru digiti care formeaza afisajul sunt conectati impreuna. Astfel, va trebui sa activam pe rand fiecare digit in parte pentru a-l putea controla.

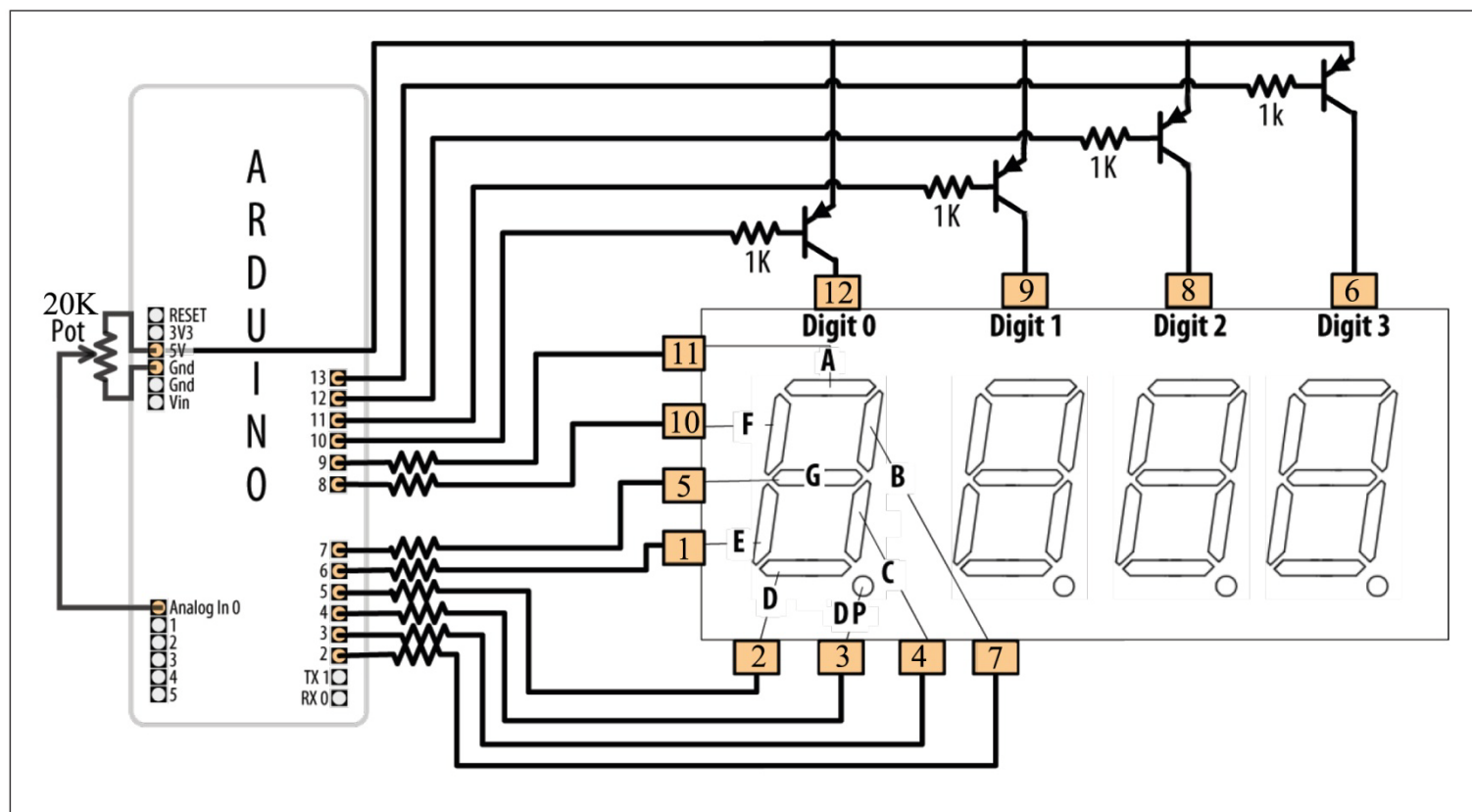


Componente utilizate	
Component	Descriere
	Arduino Uno sau dispozitiv echivalent cu acesta
D1	Afisaj format din patru digiti cu 7 segmente, anod comun
R5-12	Rezistor 270 $\Omega$
R1, R2, R3, R4	Rezistor 1 K $\Omega$
T1, T2, T3, T4	Tranzistor PNP BC557
POT1	Potentiometru liniar 20k $\Omega$

### Schema de montare pe placa Breadboard

Schema montajului pe care urmeaza sa il contruiti este prezentata in figura urmatoare. Informatia referitoare la numarul ce va fi afisat prin intermediul afisajului cu patru digiti este preluata de la un potentiometru de 20k $\Omega$ . Dispozitivul Arduino preia informatia referitoare la pozitia unghiulara a acestuia si o converteste intr-un numar pe care il transmite afisajului. Vor fi utilizate patru tranzistoare pentru a activa, pe rand, fiecare dintre digiti. Dupa selectarea unui digit, sunt iluminate segmentele necesare afisarii cifrei corespunzatoare si se trece la urmatorul digit.

Tranzistoarele utilizate vor fi tot de tip PNP, BC557, ca in experimentul precedent. Valoarea rezistoarelor utilizate pentru a conecta Arduino la segmente este 270  $\Omega$ , iar cea a potentiometrului de 20k $\Omega$ .



**Codul sursa**

```

/*
 * Programul va afisa un numar intre 0 si 9999 pe un afisaj cu 4 digiti
 * In cadrul acestui exemplu afisajul este conectat la o intrare analogica
 * si preia informatia de la un potentiometru liniar
 */
// bitii reprezentand segmentele A ~ G si punctul zecimal pentru cifrele de la 0 la 9

const int numeral[10] = {

//ABCDEFG /dp
B11111100, // 0
B01100000, // 1
B11011010, // 2
B11110010, // 3
B01100110, // 4
B10110110, // 5
B00111110, // 6
B11100000, // 7
B11111110, // 8
B11100110, // 9
};

//pinii pentru fiecare segment si punctul zecimal
// dp,G,F,E,D,C,B,A
const int segmentPins[] = { 4,7,8,6,5,3,2,9};
const int nbrDigits= 4; // the number of digits in the LED display

//dig 1 2 3 4
const int digitPins[nbrDigits] = { 10,11,12,13};

void setup(){

for(int i=0; i < 8; i++)
pinMode(segmentPins[i], OUTPUT); // set segment and DP pins to output
for(int i=0; i < nbrDigits; i++)
pinMode(digitPins[i], OUTPUT);

}

```

```
void loop(){

int value = analogRead(0);
showNumber(value);
}

void showNumber( int number){

if(number == 0)
showDigit( 0, nbrDigits-1) ; // afisare 0 in digitul din dreapta

else
{

// afisarea valorii corespunzatoare fiecarui digit

for( int digit = nbrDigits-1; digit >= 0; digit-- )
{
if(number > 0)
{
showDigit( number % 10, digit) ;
number = number / 10;
}
}
}

// afisare a numarului dat pe afisajul cu sapte segmente, in functie de pozitia digitului

void showDigit( int number, int digit){

digitalWrite( digitPins[digit], HIGH );
for(int segment = 1; segment < 8; segment++)

{
boolean isBitSet = bitRead(numeral[number], segment);
// isBitSet este setat pentru a fi HIGH daca valoarea data este 1

isBitSet = ! isBitSet; // pentru anod comun
```

```
digitalWrite( segmentPins[segment], isBitSet);  
}  
delay(5);  
  
digitalWrite( digitPins[digit], LOW );  
}
```

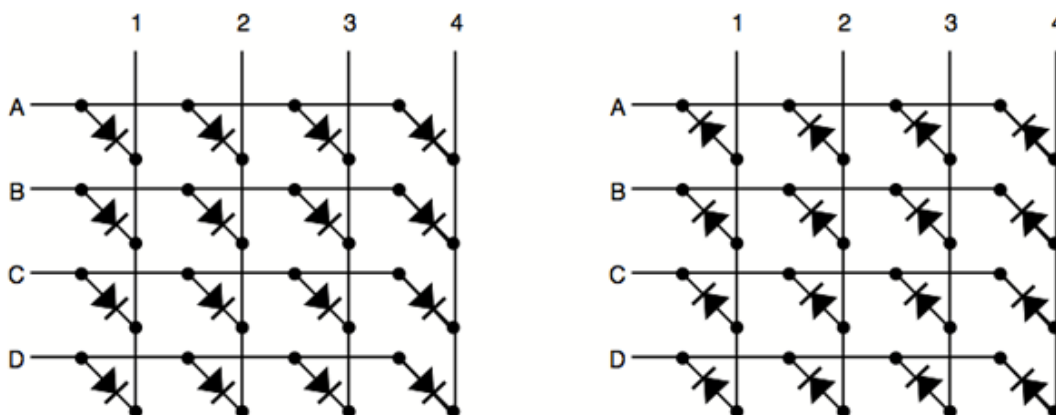
Codul sursa contine functia *showDigit* similara cu aceea din exemplele precedente. Aici, functia are doua argumente, numarul pe care urmeaza sa il afiseze si pozitia digirului in cadrul afisajului. Logica de iluminare a segmentelor este aceeaasi, dar functia seteaza bitul corespunzator anodului unui digit pe HIGH pentru a-l selecta si modifica doar pe acela, aceasta fiind functia de multiplexare.



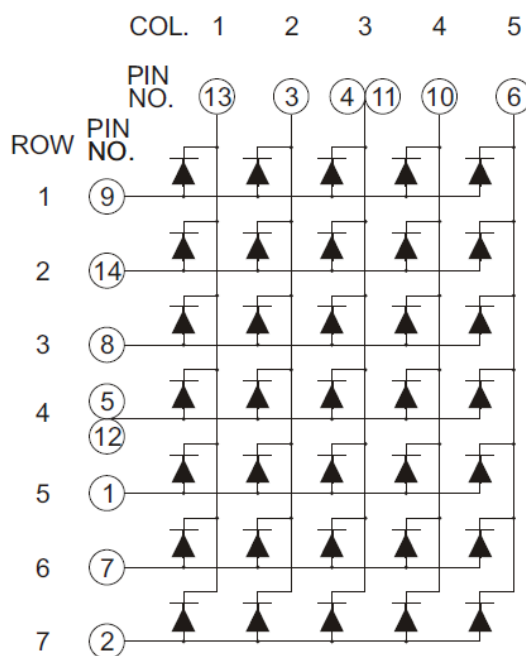
## Matrice cu LED-uri

### Structura unei matrice cu LED-uri

In aceasta structura LED-urile sunt aranjate sub forma de linii si coloane. Astfel, oricare dintre LED-uri poate fi vazut ca un set de coordonate (x,y). Sa analizam matricea de 5 x 7 LED-uri; coloanele vor fi marcate de la 1 la 5 si liniile de la A la G. Putem identifica oricare LED prin coordonatele sale: de exemplu, cel din stanga sus va fi (A,1), iar cel din dreapta jos (G,5). Exista doua tipuri de matrice cu LED-uri: unele au anod comun (stanga), iar altele catod comun (dreapta).



Diferenta intre cele doua metode de configurare a matricelor cu LED-uri o reprezinta modalitatea in care va fi aprins un LED. De exemplu, pentru a ilumina LED-ul (D,4) dintr-o matrice cu catod comun trebuie sa alimentam cu tensiune pozitiva coloana 4 si sa conectam la masa linia D. Matricea pe care o veti regasi in kit are catodul comun si este prezentata in figura de mai jos.



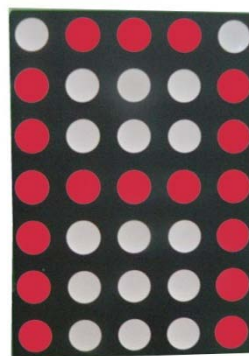
### **Generarea unor simboluri cu ajutorul Arduino si a matricei cu LED-uri**

In cadrul acestei lectii vom invata sa utilizam matricea cu LED-uri pentru a afisa simboluri, si, mai apoi, litere care pot forma un text. Utilizand notiunile invatate vom putea afisa orice figura se incadreaza in dimensiunea de 5 x 7 pixeli.

La nivelul codului sursa, matricea cu LED-uri va fi reprezentata de o matrice ale carei elemente sunt 0 si 1 (0, daca LED-ul asociat coordonatei din matrice este stins si 1, daca LED-ul este aprins). La nivel hardware, LED-ul este stins daca anodul (polul pozitiv) este alimentat de la polul negativ al unei surse de alimentare (un pin al dispozitivului Arduino setat pe LOW) si catodul (polul negativ) este alimentat de la polul pozitiv al sursei de alimentare (un pin al dispozitivului Arduino setat pe HIGH). In caz contrar, daca anodul (polul pozitiv) este alimentat de la polul pozitiv al unei surse de alimentare (un pin al dispozitivului Arduino setat pe HIGH) si catodul (polul negativ) este alimentat de la polul negativ al sursei de alimentare (un pin al dispozitivului Arduino setat pe LOW), LED-ul va ilumina. Astfel, iluminarea elementelor dorite din matricea cu LED-uri se realizeaza prin setarea pe HIGH sau LOW a pinilor de iesire ai dispozitivului Arduino.

In imaginile de mai jos, prezentam cateva exemple de reprezentare a unor litere din alfabet cu ajutorul matricei cu LED-uri.

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1



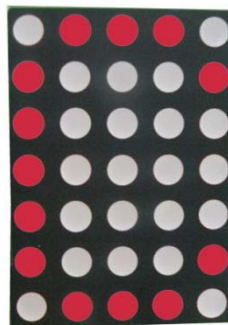
***Litera A. Reprezentare in cod binar pentru matrice 5 x 7 si reprezentare pe matricea cu LED-uri***

1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0



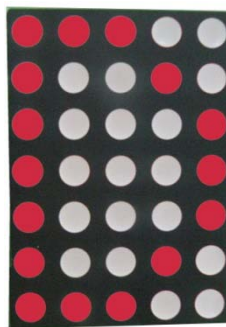
**Litera B. Reprezentare in cod binar pentru matrice 5 x 7 si reprezentare pe matricea cu LED-uri**

0	1	1	1	0
1	0	0	0	1
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	1
0	1	1	1	0



**Litera C. Reprezentare in cod binar pentru matrice 5 x 7 si reprezentare pe matricea cu LED-uri**

1	1	1	0	0
1	0	0	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	1	0
1	1	1	0	0



**Litera D. Reprezentare in cod binar pentru matrice 5 x 7 si reprezentare pe matricea cu LED-uri**

1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1

1	0	0	0	1
1	1	0	1	1
1	0	1	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1

*Literele K si M. Reprezentare in cod binar pentru matrice 5 x 7*

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1

1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1

*Literele Q si R. Reprezentare in cod binar pentru matrice 5 x 7*

0	0	0	0	0
0	0	0	0	0
0	1	1	1	0
0	0	0	0	1
0	1	1	1	1
1	0	0	0	1
0	1	1	1	1

1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0

*Literele a si b. Reprezentare in cod binar pentru matrice 5 x 7*

0	0	0	0	1
0	0	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	1

0	0	0	0	0
0	0	0	0	0
0	1	1	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	0
0	1	1	1	0

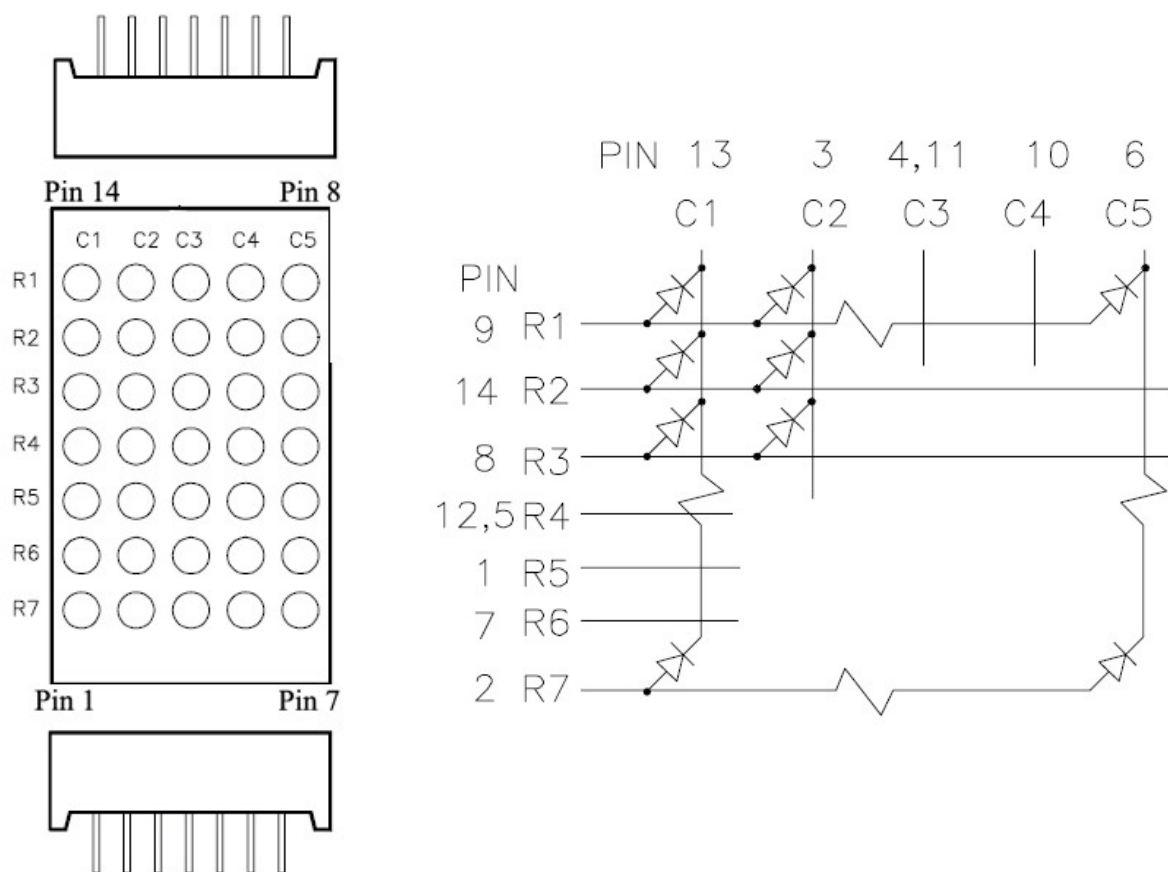
*Literele d si e. Reprezentare in cod binar pentru matrice 5 x 7*

## Experiment 22. Reprezentarea unor simboluri pe matricea cu LED-uri

Cu ajutorul dispozitivului Arduino si a matricei cu LED-uri pe care o avem la dispozitie vom ilumina pe rand, cateva dintre LED-uri pentru a obtine un anumit efect luminos.

### Schema de montare pe placa Breadboard

Pinii matricei cu LED-uri sunt numerotati asa cum este evidentiat in imaginea urmatoare. Numerotarea incepe din partea stanga a regiunii pe care este marcat codul de fabricatie al matricei.



Conectarea pinilor dispozitivului Arduino se realizeaza conform tabelului de mai jos. Pinul 0 al dispozitivului Arduino se conecteaza la pinul 2 al matricei, pinul 1 la pinul 7, s.a.m.d.

Pin de iesire Arduino	Terminal al matricei de LED-uri
0	2
1	7
2	1
3	5
4	8
5	14
6	9
8	13
9	3
10	4
11	10
12	6

**Tabel 22.1. Conectarea matricei cu LED-uri la dispozitivul Arduino**

### Codul sursa

In cadrul codului sursa, sunt definiti initial pinii dispozitivului Arduino conectati la terminalele de tip catod si anod ai matricei cu LED-uri. In etapa urmatoare, este definit un sir care va contine reprezentarea fiecarui simbol pe care urmeaza sa il afisam pe matricea cu LED-uri (sunt delimitate LED-urile care sunt aprinse si stinse). De data aceasta, am utilizat sistemul binar, evidentiat prin simbolul 0B pe care il veti observa inaintea fiecarui numar din cadrul sirului (0 pentru LED-urile stinse, 1 pentru LED-urile stinse) pentru a urmari mai usor codul sursa si efectul acestuia asupra matricei. Chiar daca o coloana a matricei cu LED-uri pe care o vom utiliza are 7 elemente, noi o vom defini in cadrul sirului nostru ca fiind din 8 elemente, ultimul dintre acestea fiind nesemnificativ si putand avea valoarea 0 sau 1. Am ales sa utilizam o reprezentare logica din 8 elemente binare intrucat in exemplul urimator vom folosi reprezentare in sistem hexazecimal, si, un numar hexazecimal format din 2 simboluri este echivalent cu un numar binar format din 8 simboluri.

Functia setup() defineste terminalele dispozitivului Arduino ca pini de iesire. Functia clear() este utilizata a efectua o resetare, adica pentru a stinge toate LED-urile matricei. Functia afisare() va afisa caracterul dorit. In bucla principala, vom afisa, pe rand toate caracterele definite in sirul nostru.

/\*

Cod sursa pentru reprezentarea unor simboluri pe o matrice cu LED-uri utilizand dispozitivul Arduino. Pentru a controla matricea cu LED-uri nu este utilizat nici un tranzistor sau registru de deplasare.

Pentru a proteja matricea cu LED-uri si dispozitivul Arduino, se poate conecta cate un rezistor de 270Ohm intre pinii 0 ~6 ai dispozitivului Arduino si cei care le corespund din matricea de LED-uri.

Conexiuni:

Pin Arduino	Pin matrice
0	2 Linia 7
1	7 Linia 6
2	1 Linia 5
3	5 Linia 4
4	8 Linia 3
5	14 Linia 2
6	9 Linia 1
8	13 Coloana 1
9	3 Coloana 2
10	4 Coloana 3
11	10 Coloana 4
12	6 Coloana 5

\*/

```
const byte xPin[] = {8,9,10,11,12}; // Pinii de catod, coloanele matricei cu LED-uri
const byte yPin[] = {0,1,2,3,4,5,6}; //Pinii de anod, liniile matricei
```

```
const byte xLedNumber = 5; // numarul de LED-uri pe directia x (numarul de coloane)
const byte yLedNumber = 7; // numarul de LED-uri pe directia y (numarul de linii)
```

```
byte matriceLED[21][5] = {
  {0B11111111,
   0B11111111,
   0B11111111,
   0B11111111,
   0B11111111}, // matrice plina
  {0B11111100,
   0B11111100,
   0B11111100,
   0B11111100,
   0B11111100}, // stingem o linie
  {0B111111000,
   0B111111000,
```

```

0B111111000,
0B111111000,
0B111111000}, // stingem si a doua linie
{0B11110000,
0B11110000,
0B11110000,
0B11110000,
0B11110000}, // stingem a treia linie
{0B11100000,
0B11100000,
0B11100000,
0B11100000,
0B11100000}, // stingem a patra linie
{0B11000000,
0B11000000,
0B11000000,
0B11000000,
0B11000000}, // stingem a cincea linie
{0B10000000,
0B10000000,
0B10000000,
0B10000000,
0B10000000}, // stingem a sasea linie si revenim pana vom ilumina din nou complet //matricea
{0B11000000,
0B11000000,
0B11000000,
0B11000000,
0B11000000},
{0B11100000,
0B11100000,
0B11100000,
0B11100000,
0B11100000},
{0B11110000,
0B11110000,
0B11110000,
0B11110000,
0B11110000},
{0B11111000,
0B11111000,
0B11111000,
0B11111000,
0B11111000},
{0B11111100,
0B11111100,
0B11111100,

```



```
0B11111100,  
0B11111100},  
{0B11111111,  
0B11111111,  
0B11111111,  
0B11111111,  
0B11111111,  
0B11111111},  
{0B00000000,  
0B01111100,  
0B01111100,  
0B01111100,  
0B00000000},  
{0B00000000,  
0B00111000,  
0B00111000,  
0B00111000,  
0B00000000},  
{0B00000000,  
0B00010000,  
0B00010000,  
0B00010000,  
0B00000000},  
{0B01000100,  
0B00101000,  
0B00010000,  
0B00101000,  
0B01000100},  
{0B01010100,  
0B00111000,  
0B00010000,  
0B00111000,  
0B01010100},  
{0B11000010,  
0B11000110,  
0B00110110,  
0B11000110,  
0B11000010},  
{0B11000100,  
0B11000100,  
0B00110100,  
0B11000100,  
0B11000100},  
{0B11001000,  
0B11000110,  
0B00110010,  
0B11000110,
```

```
    0B11001000},
};

void setup() {
    // Configurarea pinilor de iesire
    for (int i = 0; i < xLedNumber; i++) {
        pinMode (xPin[i], OUTPUT);
    }

    for (int i = 0; i < yLedNumber; i++) {
        pinMode (yPin[i], OUTPUT);
    }
}

void loop() {
    clear();

    int elemente = 21; //numarul de elemente afisate

    for (int j = 0; j<elemente;j++){
        for (int i = 1; i<90;i++){
            afisare(j);
        }
        delay(15);
    }
    delay(100); // Pauza pana la o noua rulare a programului
}

// Functia clear() este utilizata pentru a stinge toate LED-urile matricei

void clear() {
    for (int i = 0; i < xLedNumber; i++) {
        digitalWrite(xPin[i], HIGH); // Pinii de catod vor fi setati pe HIGH, astfel incat sa // nu
        circule curent intre Arduino si LED-urile matricei
    }
    for (int i = 0; i < yLedNumber; i++) {
        digitalWrite(yPin[i], LOW);
    }
}

// Functia afisare() este utilizata pentru a afisa caracterul dorit

void afisare(int numarsimbol) {
```

```
for (int xCounter = 0; xCounter < xLedNumber; xCounter++) { // Rularea coloanelor din
//matrice
    digitalWrite(xPin[xCounter], LOW); // Pinii de catod trebuie sa fie setati pe LOW //astfel
    incat curentul electric sa circule dispre anod prin LED catre catod

for (int yCounter = 0; yCounter < yLedNumber; yCounter++) { // Rularea liniilor din //matrice
    byte yRow = (matriceLED[numarsimbol][xCounter] >> (yCounter + 1)) & 1;
    digitalWrite(yPin[yCounter], yRow);

}

delay(1); //O mica intarziere pentru ca multiplexarea sa functioneze normal.
clear();
}
}
```

1. Rulati programul desen\_matrice\_led.ino
2. Introduceti noi simboluri in sirul matriceLED, insa tineti cont sa modificati si dimensiunea acestuia (daca introduceti un element veti modifica matriceLED[18][5] in matriceLED[19][5] ). Pentru a afisa toate elementele pe care le-ati introdus, modificati si numarul acestora, adica numarul *elemente* (la fel, daca ati introdus un element suplimentare *elemente* = 18 va fi inlocuit cu *elemente* = 19)

## Experimentul 23. Reprezentarea literelor din alfabet pe matricea cu LED-uri

Cu ajutorul dispozitivului Arduino si a matricei cu LED-uri pe care o avem la dispozitie vom afisa, pe rand, caracterele din alfabet.

### Schema de montare pe placa Breadboard

Pentru a rula acest exemplu, veti utiliza schema de conectare de la exemplul precedent.

### Codul sursa

In cadrul codului sursa, sunt definiti initial pinii dispozitivului Arduino conectati la terminalele de tip catod si anod ai matricei cu LED-uri. In etapa urmatoare, este definit un sir care va contine reprezentarea fiecărei litere pe matricea cu LED-uri (LED-urile care sunt aprinse si stinse). Intrucat este mai simplu de utilizat, pentru reprezentarea literelor a fost folosit sistemul hexazecimal (orice numar hexazecimal format din doua simboluri poate fi reprezentat printr-o succesiune de 8 simboluri binare; intrucat numarul va fi convertit in 8 simboluri si noi avem o matrice cu doar 7 elemente pe o coloana vom programul va retine si va reprezenta doar primele 7 simboluri ale numarului convertit in sistem binar).

In tabelul de mai jos este prezentata corespondenta intre sistemul hexazecimal si cel binar.

Hex	Bin	Hex	Bin	Hex	Bin	Hex	Bin
00	00000000	10	00010000	20	00100000	30	00110000
01	00000001	11	00010001	21	00100001	31	00110001
02	00000010	12	00010010	22	00100010	32	00110010
03	00000011	13	00010011	23	00100011	33	00110011
04	00000100	14	00010100	24	00100100	34	00110100
05	00000101	15	00010101	25	00100101	35	00110101
06	00000110	16	00010110	26	00100110	36	00110110
07	00000111	17	00010111	27	00100111	37	00110111
08	00001000	18	00011000	28	00101000	38	00111000
09	00001001	19	00011001	29	00101001	39	00111001
0A	00001010	1A	00011010	2A	00101010	3A	00111010
0B	00001011	1B	00011011	2B	00101011	3B	00111011
0C	00001100	1C	00011100	2C	00101100	3C	00111100
0D	00001101	1D	00011101	2D	00101101	3D	00111101
0E	00001110	1E	00011110	2E	00101110	3E	00111110
0F	00001111	1F	00011111	2F	00101111	3F	00111111

Hex	Bin	Hex	Bin	Hex	Bin	Hex	Bin
40	01000000	50	01010000	60	01100000	70	01110000
41	01000001	51	01010001	61	01100001	71	01110001
42	01000010	52	01010010	62	01100010	72	01110010
43	01000011	53	01010011	63	01100011	73	01110011
44	01000100	54	01010100	64	01100100	74	01110100
45	01000101	55	01010101	65	01100101	75	01110101
46	01000110	56	01010110	66	01100110	76	01110110
47	01000111	57	01010111	67	01100111	77	01110111
48	01001000	58	01011000	68	01101000	78	01111000
49	01001001	59	01011001	69	01101001	79	01111001
4A	01001010	5A	01011010	6A	01101010	7A	01111010
4B	01001011	5B	01011011	6B	01101011	7B	01111011
4C	01001100	5C	01011100	6C	01101100	7C	01111100
4D	01001101	5D	01011101	6D	01101101	7D	01111101
4E	01001110	5E	01011110	6E	01101110	7E	01111110
4F	01001111	5F	01011111	6F	01101111	7F	01111111
Hex	Bin	Hex	Bin	Hex	Bin	Hex	Bin
80	10000000	90	10010000	A0	10100000	B0	10110000
81	10000001	91	10010001	A1	10100001	B1	10110001
82	10000010	92	10010010	A2	10100010	B2	10110010
83	10000011	93	10010011	A3	10100011	B3	10110011
84	10000100	94	10010100	A4	10100100	B4	10110100
85	10000101	95	10010101	A5	10100101	B5	10110101
86	10000110	96	10010110	A6	10100110	B6	10110110
87	10000111	97	10010111	A7	10100111	B7	10110111
88	10001000	98	10011000	A8	10101000	B8	10111000
89	10001001	99	10011001	A9	10101001	B9	10111001
8A	10001010	9A	10011010	AA	10101010	BA	10111010
8B	10001011	9B	10011011	AB	10101011	BB	10111011
8C	10001100	9C	10011100	AC	10101100	BC	10111100
8D	10001101	9D	10011101	AD	10101101	BD	10111101
8E	10001110	9E	10011110	AE	10101110	BE	10111110
8F	10001111	9F	10011111	AF	10101111	BF	10111111
Hex	Bin	Hex	Bin	Hex	Bin	Hex	Bin
C0	11000000	D0	11010000	E0	11100000	F0	11110000
C1	11000001	D1	11010001	E1	11100001	F1	11110001
C2	11000010	D2	11010010	E2	11100010	F2	11110010
C3	11000011	D3	11010011	E3	11100011	F3	11110011

C4	11000100	D4	11010100	E4	11100100	F4	11110100
C5	11000101	D5	11010101	E5	11100101	F5	11110101
C6	11000110	D6	11010110	E6	11100110	F6	11110110
C7	11000111	D7	11010111	E7	11100111	F7	11110111
C8	11001000	D8	11011000	E8	11101000	F8	11111000
C9	11001001	D9	11011001	E9	11101001	F9	11111001
CA	11001010	DA	11011010	EA	11101010	FA	11111010
CB	11001011	DB	11011011	EB	11101011	FB	11111011
CC	11001100	DC	11011100	EC	11101100	FC	11111100
CD	11001101	DD	11011101	ED	11101101	FD	11111101
CE	11001110	DE	11011110	EE	11101110	FE	11111110
CF	11001111	DF	11011111	EF	11101111	FF	11111111

Functia *setup()* defineste terminalele dispozitivului Arduino ca pini de iesire. Functia *clear()* este utilizata a efectua o resetare, adica pentru a stinge toate LED-urile matricei. Functia *displayCharacter()* va afisa caracterul dorit. In bucla principala, vom afisa, pe rand toate caracterele definite in sirul nostru.

/\*

Cod sursa pentru reprezentarea literelor din alfabet pe o matrice cu LED-uri utilizand dispozitivul Arduino. Pentru a controla matricea cu LED-uri nu este utilizat nici un tranzistor sau registru de deplasare.

Pentru a proteja matricea cu LED-uri si dispozitivul Arduino, se poate conecta cate un rezistor de 270Ohm intre pinii 0 ~6 ai dispozitivului Arduino si cei care le corespund din matricea de LED-uri.

Conexiuni:

Pin Arduino	Pin matrice
0	2 Linia 7
1	7 Linia 6
2	1 Linia 5
3	5 Linia 4
4	8 Linia 3
5	14 Linia 2
6	9 Linia 1
8	13 Coloana 1
9	3 Coloana 2
10	4 Coloana 3

```

11          10 Coloana 4
12          6  Coloana 5
*/

```

```

const byte xPin[] = {8,9,10,11,12}; // Pinii de catod, coloanele matricei cu LED-uri
const byte yPin[] = {0,1,2,3,4,5,6}; //Pinii de anod, liniile matricei

```

```

const byte xLedNumber = 5; // numarul de LED-uri pe directia x (numarul de coloane)

```

```

const byte yLedNumber = 7; // numarul de LED-uri pe directia y (numarul de linii)

```

```

const byte numberOfCharacters = 26; // numarul total de caractere pe care le vom defini

```

```

byte font_5x7w[26][5] = {
  {0x7e,0x88,0x88,0x88,0x7e}, // A 0 <- numerele utilizate ne ajuta sa identificam mai
// usor elementele matricei
  {0xfe,0x92,0x92,0x92,0x6c}, // B 1
  {0x7c,0x82,0x82,0x82,0x44}, // C 2
  {0xfe,0x82,0x82,0x44,0x38}, // D 3
  {0xfe,0x92,0x92,0x92,0x82}, // E 4
  {0xfe,0x90,0x90,0x80,0x80}, // F 5
  {0x7c,0x82,0x82,0x8a,0x4c}, // G 6
  {0xfe,0x10,0x10,0x10,0xfe}, // H 7
  {0x00,0x82,0xfe,0x82,0x00}, // I 8
  {0x04,0x02,0x82,0xfc,0x80}, // J 9
  {0xfe,0x10,0x28,0x44,0x82}, // K 10
  {0xfe,0x02,0x02,0x02,0x02}, // L 11
  {0xfe,0x40,0x20,0x40,0xfe}, // M 12
  {0xfe,0x20,0x10,0x08,0xfe}, // N 13
  {0x7c,0x82,0x82,0x82,0x7c}, // O 14
  {0xfe,0x90,0x90,0x90,0x60}, // P 15
  {0x7c,0x82,0x8a,0x84,0x7a}, // Q 16
  {0xfe,0x90,0x98,0x94,0x62}, // R 17
  {0x62,0x92,0x92,0x92,0x8c}, // S 18
  {0x80,0x80,0xfe,0x80,0x80}, // T 19
  {0xfc,0x02,0x02,0x02,0xfc}, // U 20
  {0xf8,0x04,0x02,0x04,0xf8}, // V 21
  {0xfe,0x04,0x18,0x04,0xfe}, // W 22
  {0xc6,0x28,0x10,0x28,0xc6}, // X 23
  {0xc0,0x20,0x1e,0x20,0xc0}, // Y 24
  {0x86,0x8a,0x92,0xa2,0xc2}, // Z 25
};

```

```
void setup() {
  // Configurarea pinilor de iesire
  for (int i = 0; i < xLedNumber; i++) {
    pinMode (xPin[i], OUTPUT);
  }

  for (int i = 0; i < yLedNumber; i++) {
    pinMode (yPin[i], OUTPUT);
  }
}

void loop() {
  clear();
  for (int j = 0; j<26;j++){
    for (int i = 1; i<260;i++){
      displayCharacter(j);
    }
    delay(500);
  }
  delay(1000); // Pauza pana la o noua rulare a programului
}

// Functia clear() este utilizata pentru a stinge toate LED-urile matricei

void clear() {
  for (int i = 0; i < xLedNumber; i++) {
    digitalWrite(xPin[i], HIGH); // Pinii de catod vor fi setati pe HIGH, astfel incat sa // nu
    circule curent  intre Arduino si LED-urile matricei
  }
  for (int i = 0; i < yLedNumber; i++) {
    digitalWrite(yPin[i], LOW);
  }
}

// Functia displayCharacter() este utilizata pentru a afisa caracterul dorit

void displayCharacter(int characterNumber) {

  for (int xCounter = 0; xCounter < xLedNumber; xCounter++) { // Rularea coloanelor din
  //matrice
    digitalWrite(xPin[xCounter], LOW); // Pinii de catod trebuie sa fie setati pe LOW //astfel
    incat curentul electric sa circule dispre anod prin LED catre catod

  for (int yCounter = 0; yCounter < yLedNumber; yCounter++) { // Rularea liniilor din //matrice
    byte yRow = (font_5x7w[characterNumber][xCounter] >> (yCounter + 1)) & 1;
```



```
    digitalWrite(yPin[yCounter], yRow);  
  }  
  
  delay(1); //O mica intarziere pentru ca multiplexarea sa functioneze normal.  
  clear();  
}  
}
```

1. Rulati programul litere\_alfabet.ino

## Experimentul 24. Rularea unui text pe matricea cu LED-uri

Cu ajutorul dispozitivului Arduino si a matricei cu LED-uri pe care o avem la dispozitie vompe rand literele ce compun un text.

### Schema de montare pe placa Breadboard

Pentru a rula acest program, veti utiliza schema de conectare de la exemplele precedente.

### Codul sursa

Codul sursa este similar cu acela de la exemplul precedent cu anumite modificari: am extins gama caracterelor ce pot fi afisate si am introdus posibilitatea de a rula caracterele in ordinea dorita de noi.

/\*

Cod sursa pentru generarea unui text ce va fi reprezentat pe matricea cu LED-uri. Acesta este un text demonstrativ va scrie matricea cu LED-uri. Pentru a controla matricea cu LED-uri nu este utilizat nici un tranzistor sau registru de deplasare.

Pentru a proteja matricea cu LED-uri si dispozitivul Arduino, se poate conecta cate un rezistor de 270Ohm intre pinii 0 ~6 ai dispozitivului Arduino si cei care le corespund din matricea de LED-uri.

Conexiuni:

Pin Arduino	Pin matrice
0	2 Linia 7
1	7 Linia 6
2	1 Linia 5
3	5 Linia 4
4	8 Linia 3
5	14 Linia 2
6	9 Linia 1
8	13 Coloana 1
9	3 Coloana 2
10	4 Coloana 3
11	10 Coloana 4
12	6 Coloana 5

\*/

```
const byte xPin[] = {8,9,10,11,12}; // Pinii de catod, coloanele matricei cu LED-uri
const byte yPin[] = {0,1,2,3,4,5,6}; //Pinii de anod, liniile matricei
```

```
const byte xLedNumber = 5; // numarul de LED-uri pe directia x (numarul de coloane)
const byte yLedNumber = 7; // numarul de LED-uri pe directia y (numarul de linii)
```

```
byte font_matrice[96][5] = {
    {0x00,0x00,0x00,0x00,0x00}, // 0
    {0x00,0x00,0xfa,0x00,0x00}, // ! 1
    {0x00,0xe0,0x00,0xe0,0x00}, // " 2
    {0x28,0xfe,0x28,0xfe,0x28}, // # 3
    {0x24,0x54,0xfe,0x54,0x48}, // $ 4
    {0xc4,0xc8,0x10,0x26,0x46}, // % 5
    {0x6c,0x92,0xaa,0x44,0x0a}, // & 6
    {0x00,0xa0,0xc0,0x00,0x00}, // ' 7
    {0x00,0x38,0x44,0x82,0x00}, // ( 8
    {0x00,0x82,0x44,0x38,0x00}, // ) 9
    {0x10,0x54,0x38,0x54,0x10}, // * 10
    {0x10,0x10,0x7c,0x10,0x10}, // + 11
    {0x00,0x0a,0x0c,0x00,0x00}, // , 12
    {0x10,0x10,0x10,0x10,0x10}, // - 13
    {0x00,0x06,0x06,0x00,0x00}, // . 14
    {0x04,0x08,0x10,0x20,0x40}, // / 15
    {0x7c,0x8a,0x92,0xa2,0x7c}, // 0 16
    {0x00,0x42,0xfe,0x02,0x00}, // 1 17
    {0x42,0x86,0x8a,0x92,0x62}, // 2 18
    {0x84,0x82,0xa2,0xd2,0x8c}, // 3 19
    {0x18,0x28,0x48,0xfe,0x08}, // 4 20
    {0xe4,0xa2,0xa2,0xa2,0x9c}, // 5 21
    {0x3c,0x52,0x92,0x92,0x0c}, // 6 22
    {0x80,0x8e,0x90,0xa0,0xc0}, // 7 23
    {0x6c,0x92,0x92,0x92,0x6c}, // 8 24
    {0x60,0x92,0x92,0x94,0x78}, // 9 25
    {0x00,0x6c,0x6c,0x00,0x00}, // : 26
    {0x00,0x6a,0x6c,0x00,0x00}, // ; 27
    {0x00,0x10,0x28,0x44,0x82}, // < 28
    {0x28,0x28,0x28,0x28,0x28}, // = 29
    {0x82,0x44,0x28,0x10,0x00}, // > 30
    {0x40,0x80,0x8a,0x90,0x60}, // ? 31
    {0x4c,0x92,0x9e,0x82,0x7c}, // @ 32
    {0x7e,0x88,0x88,0x88,0x7e}, // A 33
    {0xfe,0x92,0x92,0x92,0x6c}, // B 34
    {0x7c,0x82,0x82,0x82,0x44}, // C 35
    {0xfe,0x82,0x82,0x44,0x38}, // D 36
    {0xfe,0x92,0x92,0x92,0x82}, // E 37
    {0xfe,0x90,0x90,0x80,0x80}, // F 38
    {0x7c,0x82,0x82,0x8a,0x4c}, // G 39
    {0xfe,0x10,0x10,0x10,0xfe}, // H 40
```

{0x00,0x82,0xfe,0x82,0x00}, // I	41
{0x04,0x02,0x82,0xfc,0x80}, // J	42
{0xfe,0x10,0x28,0x44,0x82}, // K	43
{0xfe,0x02,0x02,0x02,0x02}, // L	44
{0xfe,0x40,0x20,0x40,0xfe}, // M	45
{0xfe,0x20,0x10,0x08,0xfe}, // N	46
{0x7c,0x82,0x82,0x82,0x7c}, // O	47
{0xfe,0x90,0x90,0x90,0x60}, // P	48
{0x7c,0x82,0x8a,0x84,0x7a}, // Q	49
{0xfe,0x90,0x98,0x94,0x62}, // R	50
{0x62,0x92,0x92,0x92,0x8c}, // S	51
{0x80,0x80,0xfe,0x80,0x80}, // T	52
{0xfc,0x02,0x02,0x02,0xfc}, // U	53
{0xf8,0x04,0x02,0x04,0xf8}, // V	54
{0xfe,0x04,0x18,0x04,0xfe}, // W	55
{0xc6,0x28,0x10,0x28,0xc6}, // X	56
{0xc0,0x20,0x1e,0x20,0xc0}, // Y	57
{0x86,0x8a,0x92,0xa2,0xc2}, // Z	58
{0x00,0x00,0xfe,0x82,0x82}, // [	59
{0x40,0x20,0x10,0x08,0x04}, // "\"	60
{0x82,0x82,0xfe,0x00,0x00}, // ]	61
{0x20,0x40,0x80,0x40,0x20}, // ^	62
{0x02,0x02,0x02,0x02,0x02}, // _	63
{0x00,0x80,0x40,0x20,0x00}, // `	64
{0x04,0x2a,0x2a,0x2a,0x1e}, // a	65
{0xfe,0x12,0x22,0x22,0x1c}, // b	66
{0x1c,0x22,0x22,0x22,0x04}, // c	67
{0x1c,0x22,0x22,0x12,0xfe}, // d	68
{0x1c,0x2a,0x2a,0x2a,0x18}, // e	69
{0x10,0x7e,0x90,0x80,0x40}, // f	70
{0x10,0x28,0x2a,0x2a,0x3c}, // g	71
{0xfe,0x10,0x20,0x20,0x1e}, // h	72
{0x00,0x22,0xbe,0x02,0x00}, // i	73
{0x04,0x02,0x22,0xbc,0x00}, // j	74
{0x00,0xfe,0x08,0x14,0x22}, // k	75
{0x00,0x82,0xfe,0x02,0x00}, // l	76
{0x3e,0x20,0x18,0x20,0x1e}, // m	77
{0x3e,0x10,0x20,0x20,0x1e}, // n	78
{0x1c,0x22,0x22,0x22,0x1c}, // o	79
{0x3e,0x28,0x28,0x28,0x10}, // p	80
{0x10,0x28,0x28,0x18,0x3e}, // q	81
{0x3e,0x10,0x20,0x20,0x10}, // r	82
{0x12,0x2a,0x2a,0x2a,0x04}, // s	83
{0x20,0xfc,0x22,0x02,0x04}, // t	84
{0x3c,0x02,0x02,0x04,0x3e}, // u	85
{0x38,0x04,0x02,0x04,0x38}, // v	86

```

    {0x3c,0x02,0x0c,0x02,0x3c}, // w    87
    {0x22,0x14,0x08,0x14,0x22}, // x    88
    {0x30,0x0a,0x0a,0x0a,0x3c}, // y    89
    {0x22,0x26,0x2a,0x32,0x22}, // z    90
    {0x00,0x10,0x6c,0x82,0x00}, // {    91
    {0x00,0x00,0xfe,0x00,0x00}, // |    92
    {0x00,0x82,0x6c,0x10,0x00}, // }    93
    {0x40,0x80,0xc0,0x40,0x80}, // ~    94
    {0x00,0x00,0x00,0x00,0x00}, //     95

```

```
};
```

```
void setup() {
```

```
    // Configurarea pinilor de iesire
```

```
    for (int i = 0; i < xLedNumber; i++) {
```

```
        pinMode (xPin[i], OUTPUT);
```

```
    }
```

```
    for (int i = 0; i < yLedNumber; i++) {
```

```
        pinMode (yPin[i], OUTPUT);
```

```
    }
```

```
}
```

```
void loop() {
```

```
    clear();
```

```
    int text[]={33, 67, 69, 83, 84, 65, 0, 69, 83, 84, 69, 0, 85, 78, 0, 84, 69, 88, 84, 0, 68, 69, 77, 79,
78, 83, 84, 82, 65, 84, 73, 86};
```

```
    int n = 32;
```

```
    for (int k = 0 ; k<n; k++){
```

```
        for (int j = 1; j<200;j++){ //aici puteti modifica durata afisarii fiecarui caracter
```

```
            displayCharacter(text[k]);
```

```
        }
```

```
        delay(100);           //pauza intre afisarea a doua caractere consecutive (milisecunde)
```

```
    }
```

```
    delay(1000); // Pauza pana la o noua rulare a programului
```

```
}
```

```
// Functia clear() este utilizata pentru a stinge toate LED-urile matricei
```

```

void clear() {
  for (int i = 0; i < xLedNumber; i++) {
    digitalWrite(xPin[i], HIGH); // Pinii de catod vor fi setati pe HIGH, astfel incat sa
// nu circule curent intre Arduino si LED-urile matricei
  }
  for (int i = 0; i < yLedNumber; i++) {
    digitalWrite(yPin[i], LOW);
  }
}

// Functia displayCharacter() este utilizata pentru a afisa caracterul dorit
void displayCharacter(int characterNumber) {
  for (int xCounter = 0; xCounter < xLedNumber; xCounter++) { // Rularea coloanelor din
//matrice
    digitalWrite(xPin[xCounter], LOW); // Pinii de catod trebuie sa fie setati pe LOW //astfel
incat curentul electric sa circule dinspre anod prin LED catre catod

    for (int yCounter = 0; yCounter < yLedNumber; yCounter++) { // Rularea liniilor din //matrice
      byte yRow = (font_matrice[characterNumber][xCounter] >> (yCounter + 1)) & 1;
      digitalWrite(yPin[yCounter], yRow);

    }

    delay(1); //O mica intarziere pentru ca multiplexarea sa functioneze normal.
    clear();
  }
}

```

1. Rulati programul `generare_text_demonstrativ.ino`
2. Puteti modifica acest program pentru a afisa textul dorit de dvs. Pentru aceasta, identificati in sirul `font_matrice` elementele pe care doriti sa le utilizati (in dreptul fiecarui element este scris simbolul pe care il reprezinta si numarul de ordine din cadrul sirului: numarul de ordine 33 ii corespunde lui *A*, 34 lui *B*, 65 lui *a*, 65 lui *b*, s.a.m.d.), apoi scrieti numarul de ordine al elementelor pe care le-ati identificat, in ordinea lor logica, in sirul `text[]` si modificati numarul `n` astfel incat sa fie egal cu numarul caracterelor pe care le-ati utilizat.

Daca doriti sa modificati durata pentru care va fi afisat un caracter si durata pauzei dintre doua caractere consecutive veti modifica bucla:

```

for (int k = 0 ; k<n; k++){
  for (int j = 1; j<200;j++){ //aici puteti modifica durata afisarii fiecarui caracter
    displayCharacter(text[k]);
  }
  delay(100); //pauza intre afisarea a doua caractere consecutive (milisecunde)
}

```

## Experimentul 25. Proiectarea unui dispozitiv de inregistrare a temperaturii, utilizand un termistor si dispozitivul Arduino.

Masurarea temperaturii este o problema similara masurarii intensitatii luminii. In loc sa utilizam o fotocelula, folosim un dispozitiv denumit termistor. O data cu cresterea temperaturii, va creste si rezistenta termistorului. Unul dintre parametrii care caracterizeaza un termistor este rezistenta sa electrica la temperatura de 25°C; in cazul termistorului nostru aceasta este de 33 KΩ.

Formula de calcul a rezistentei in functie de temperatura este urmatoarea:

$$R = R_0 \exp(-\text{Beta}/(T + 273) - \text{Beta}/(T_0 + 273))$$

In cazul de fata,  $R_0$  este rezistenta la 25°C (33K) si beta este o valoare constanta, care pentru termistorul din kit-ul dvs. este egala cu 4350.

Deci,

$$R = R_0 \exp(\text{Beta}/(T + 273) - \text{Beta}/298)$$

Utilizand formula pentru un rezistor fix de 33 KΩ, putem calcula tensiunea din intrarea analogica:

$$V = 5 * 33K / (R + 33K)$$

deci

$$R = ((5 * 33K) / V) - 33K$$

Din moment ce valoarea analogica  $a$  este data de formula:

$$a = V * (1023/5)$$

atunci

$$V = a/205$$

si

$$R = ((5 * 33K * 205) / a) - 33K$$

$$R = (1025 * 33K / a) - 33K$$

Rearanjand formula vom avea urmatoarea expresie pentru temperatura:

$$T = \text{Beta} / (\ln(R/33) + (\text{Beta}/298)) - 273$$

Si vom putea calcula temperatura in functie de intrarea analogica  $a$ :

$$T = \text{Beta} / (\ln(((1025 * 33 / a) - 33) / 33) + (\text{Beta}/298)) - 273$$

Aceasta formula o vom utiliza pentru codul sursa.

Programul este controlat de computer, insa dupa ce programul a transmis toate datele necesare dispozitivul Arduino poate fi utilizat independent de computer daca este alimentat de la o sursa externa, de exemplu, de la o baterie de 9V. Acesta stocheaza datele si, in momentul in

care este reconectat la computer, le transfera pe harddisk prin intermediul conexiunii USB, de unde pot fi importate intr-un tabel. In mod implicit, programul inregistreaza temperatura o data la fiecare cinci minute si poate stoca pana la 255 de inregistrari.

Pentru a programa sistemul de inregistrare vom defini cateva comenzi pe care sa le transmita computerul. Acestea se regasesc in urmatorul tabel.

<b>Lista comenzilor ce pot fi transmise catre sistemul de inregistrare a temperaturii</b>	
R	Citirea datelor stocate sub forma de text in forma CSV
X	Stergerea datelor memorate
C	Prezentarea datelor in grade Celsius
F	Prezentarea datelor in grade Fahrenheit
1-9	Setarea perioadei de esantionare de la 1 minut la 9 minute
G	Pornirea procesului de inregistrare
?	Raportarea starii dispozitivului, a numarului de mostre preluate, etc.

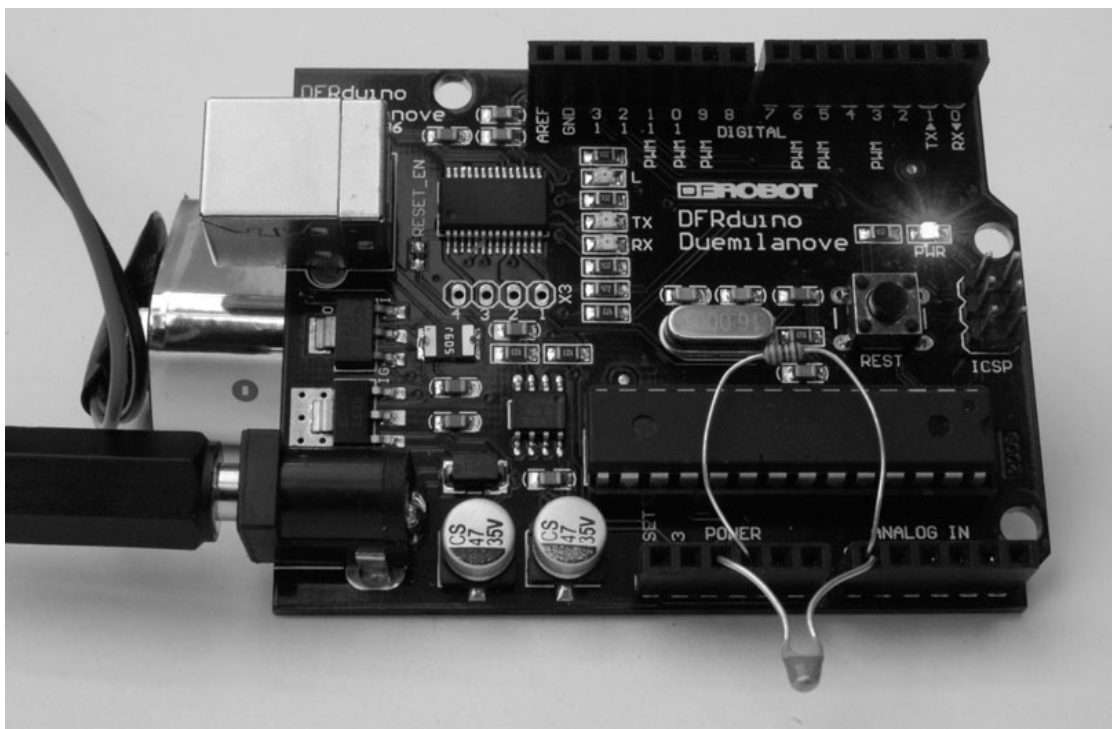
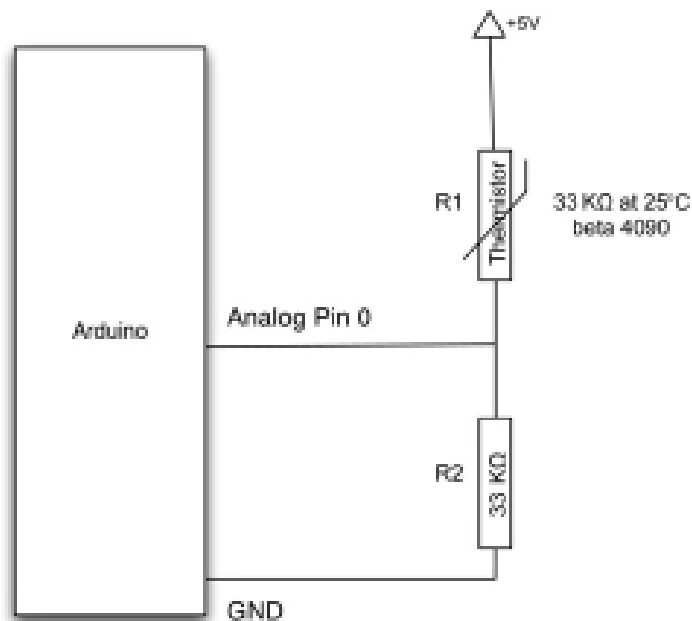
Acest proiect utilizeaza doar un termistor si un rezistor.

	<b>Componente necesare</b>
	Descriere
	Arduino Uno
R1	Termistor, 33K $\Omega$ la 25°C, beta 4350
R2	Rezistor de 33K $\Omega$



### Schema electrica

Schema electrica este prezentata in figura.



### Codul sursa

Codul sursa este un pic mai complex decat majoritatea celor pe care le-am studiat pana acum. Toate variabilele pe care le-am utilizat pana in prezent nu erau memorate de dispozitivul nostru dupa deconectarea alimentarii. Uneori este necesar sa putem stoca datele astfel incat sa poata fi accesate la reconectarea surse de alimentare. Astfel, va trebui sa accesam o anumita zona de memorie a dispozitivului Arduino denumita EEPROM - electrically erasable programmable read-only memory, memorie nevolatila folosita pentru a stoca date ce trebuie sa persiste si dupa intreruperea alimentarii cu curent. Dispozitivul nostru are 1024 de octeti la dispozitie pentru a stoca date in memoria EEPROM. Stocand datele in memoria EEPROM nu le vom mai pierde in momentul in care deconectam dispozitivul de la computer sau sursa de alimentare.

Veti observa ca variabilele noastre vor fi initializate, de data aceasta, utilizand comanda *#define*. Aceasta este o metoda mult mai eficienta de a defini constante, adica elemente a caror valoare nu se va modifica pe durata rularii programului. Comanda *#define* este o denumita directiva pre-procesare, intrucat, inainte de a fi procesat codul sursa toate variabilele cu acelasi nume sunt inlocuite de valoarea sa.

Citirea sau scrierea in memoria EEPROM se realizeaza octet cu octet. Astfel, daca dorim sa memoram un caracter sau un byte in memoria EEPROM vom utiliza functiile *EEPROM.write* si *EEPROM.read*, ca in exemplul de mai jos:

```
char letterToWrite = 'A';
EEPROM.write(0, myLetter);
char letterToRead;
letterToRead = EEPROM.read(0);
```

Cifra 0 din lista de parametri reprezinta locatia de memorie in care a fost scrisa si de unde va fi citita informatia. Aceasta este reprezentata de un numar intre 0 si 1023, fiecare dintre aceste numere reprezentand locatiile de memorie in care este stocat fiecare octet. In cadrul acestui program vom stoca atat locatiile de memorie ale celei mai recente valori masurate, cat si ale celorlalte valori masurate.

Fiecare temperatura este citita si stocata intr-o variabila de tip float (numar real cu virgula mobila) insa este convertita intr-o structura care ocupa doar un octet in memorie. Convertind insa temperatura intr-o variabila care ocupa doar un octet, facem cateva presupuneri. Prima este ca temperatura va varia in gama -20°C ~ +40°C. Oricare temperatura in afara acestui interval, ar distruge dispozitivul Arduino. In al doilea rand, vom presupune ca termometrul are o precizie de 0,25°C.

Avand in vedere aceste doua ipoteze, vom prelua valoarea intrarii analogice, vom adauga 20 la aceasta valoare, o vom inmulti cu 4, fiind siguri ca valoarea pe care urmeaza sa o obtinem se va incadra in intervalul 0 ~ 240 (un octet poate retine numere in intervalul 0~255). Dupa ce

vom prelua informatiile din memoria EEPROM va trebui sa le convertim, inversand formula pe care am utilizat-o.

Funcțiile de codare și decodare a valorilor sunt integrate în *storeReading* și *getReading*.

```
#include <EEPROM.h>

#define ledPin 13 //utilizam comanda #define pentru variabilele care nu se vor modifica
#define analogPin 0
#define maxReadings 255
#define beta 4350 // din datele tehnice ale termistorului
#define resistance 33

float readings[maxReadings];
int lastReading = EEPROM.read(0);
boolean loggingOn = false;
long period = 300;
long count = 0;

char mode = 'C'; //grade Celsius
void setup(){
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Gata de lucru");
}

void loop(){
  if (Serial.available())
  {
    char ch = Serial.read();

    if (ch == 'r' || ch == 'R') //se verifica daca a fost solicitata prezentarea rezultatelor
    {
      sendBackdata();
    }

    else if (ch == 'x' || ch == 'X') //stergere a valorilor memorate
    {
      lastReading = 0;
      EEPROM.write(0, 0);
      Serial.println("Datele au fost sterse din memorie");
    }

    else if (ch == 'g' || ch == 'G') //incepe preluarea informatiilor
    {
      loggingOn = true;
    }
  }
}
```

```
Serial.println("Incepe preluarea informatiilor ");
}

else if (ch > '0' && ch <= '9') //perioada de esantionare in minute
{
setPeriod(ch);
}

else if (ch == 'c' or ch == 'C') //Celsius
{
Serial.println("Mod setat pe grade Celsius");
mode = 'C';
}

else if (ch == 'f' or ch == 'F') // Fahrenheit
{
Serial.println("Mod setat pe grade Fahrenheit ");
mode = 'F';
}

else if (ch == '?')//interogarea starii
{
reportStatus();
}
}

if (loggingOn && count > period)
{
logReading();
count = 0;
}
count++;
delay(1000);
}

void sendBackdata(){
loggingOn = false;
Serial.println("Inregistrarea a fost oprita ");
Serial.println("-----");
Serial.print("Timp (min)\tTemp (");
Serial.print(mode);
Serial.println(")");
for (int i = 0; i < lastReading; i++)
{
Serial.print((period * i) / 60);
Serial.print("\t");
```

```
float temp = getReading(i);

if (mode == 'F')
{
temp = (temp * 9) / 5 + 32;
}
Serial.println(temp);
}
Serial.println("-----");
}

void setPeriod(char ch){
int periodMins = ch - '0';
Serial.print("Perioada de esantionare setata la: ");
Serial.print(periodMins);
Serial.println(" minute");
period = periodMins * 60;
}

void logReading(){
if (lastReading < maxReadings)
{
long a = analogRead(analogPin);
float temp = beta / (log(((1025.0 * resistance / a) - 33.0) / 33.0) +
(beta / 298.0)) - 273.0;
storeReading(temp, lastReading);
lastReading++;
}

else
{
Serial.println("Memorie plina! Inregistrarea a fost oprita");
loggingOn = false;
}
}

void storeReading(float reading, int index){
EEPROM.write(0, (byte)index); // stocarea valorilor obtinute
byte compressedReading = (byte)((reading + 20.0) * 4);
EEPROM.write(index + 1, compressedReading);
}

float getReading(int index){
lastReading = EEPROM.read(0);
byte compressedReading = EEPROM.read(index + 1);
float uncompressesReading = (compressedReading / 4.0) - 20.0;
```

```
return uncompressesReading;
}

void reportStatus(){
Serial.println("-----");
Serial.println("Stare");
Serial.print("Perioada de esantionare \t");
Serial.println(period / 60);
Serial.print("Numarul de citiri \t");
Serial.println(lastReading);
Serial.print("Mod temperatura\t");
Serial.println(mode);
Serial.println("-----");
}
```

## Asamblare

Realizati montajul si incarcati codul sursa in dispozitivul Arduino. Deschideti Serial Monitor si, pentru a testa circuitul setati-l astfel incat sa preia informatii despre temperatura o data pe minut, tastand **1**. Veti receptiona mesajul *Perioada de esantionare setata la: 1 minute*. Daca doriti, puteti schimba standardul in Fahrenheit tastand **F** in programul Serial Monitor. Interogati dispozitivul apasand **?**.

Pentru a putea deconecta dispozitivul Arduino de la USB, trebuie utilizata o sursa auxiliara de tensiune, precum bateria de 9V. Dispozitivul trebuie alimentat de la sursa auxiliara in timp ce cablul USB este conectat la computer, pentru a prelua in continuare date.

Apasati tasta **G** pentru a incepe preluarea independenta a informatiilor. Acum puteti deconecta montajul de la USB si il veti lasa sa opereze fiind alimentat de bateria de 9V. Dupa 10 ~ 15 minute de la decuplare inserati cablul USB in computer, deschideti Serial Monitor din nou si apasat tasta **R** pentru a verifica rezultatele obtinute. Copiati datele obtinute si inseratil-le intr-un program de calcul tabelar, precum Microsoft Excel, pentru a vedea modalitatea in care a variat temperatura.

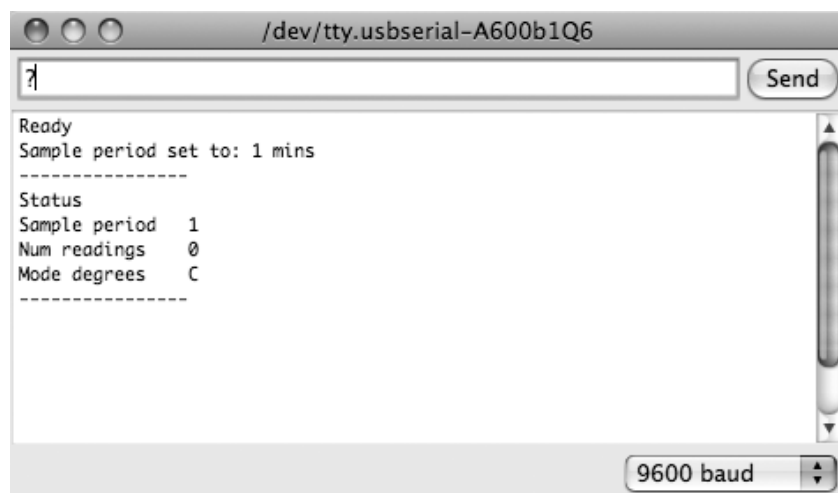


Figura 25.1. Transmiterea comenzilor catre sistemul software

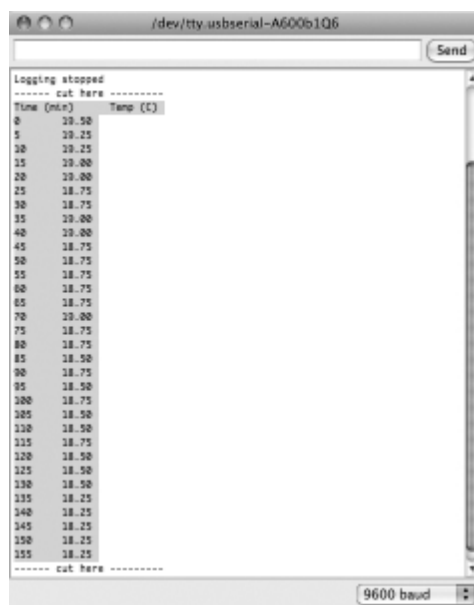


Figura 25.2. Datele care pot fi copiate intr-un tabel.

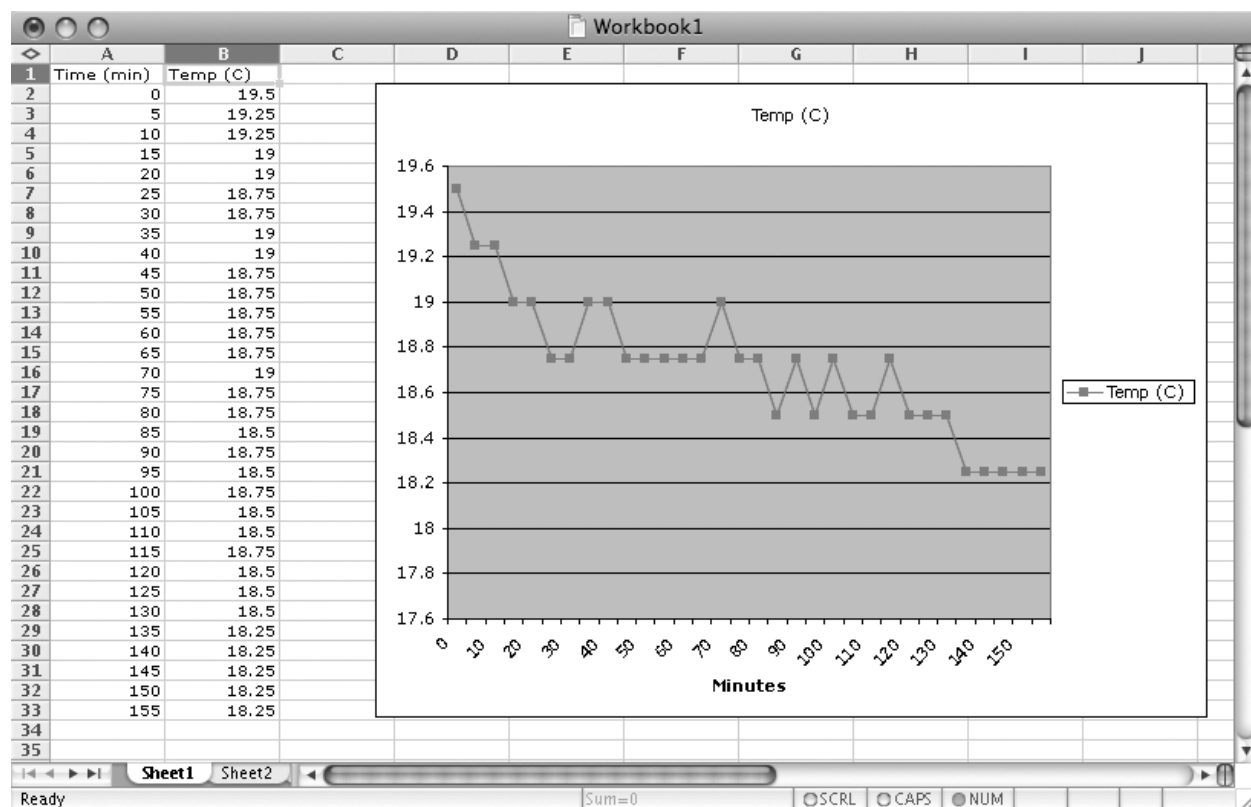


Figura 25.1. Datele importate intr-un tabel.



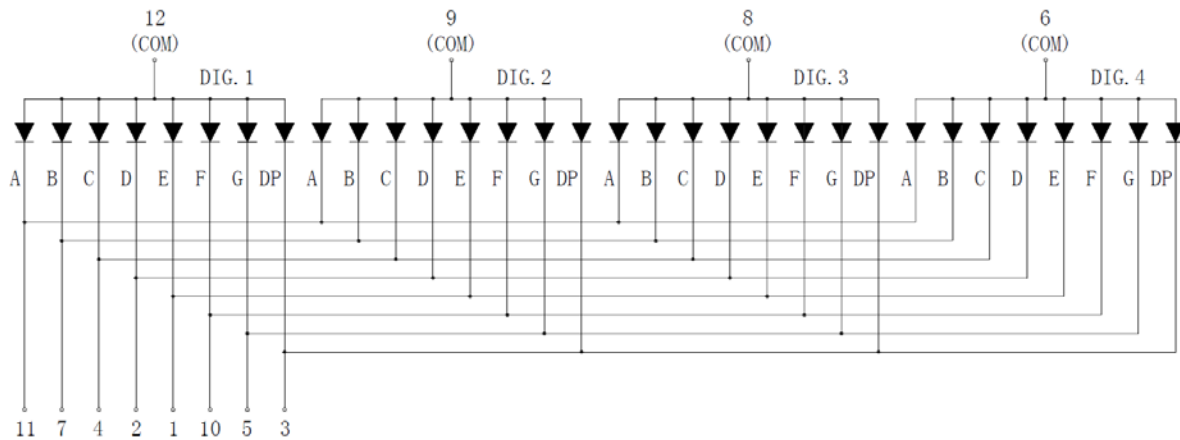
## Experimentul 26. Proiectarea unui temporizator electronic utilizand un afisaj cu patru digiti si dispozitivul Arduino

Aceasta este cea mai simpla metoda de a conecta un afisaj cu patru digiti la dispozitivul Arduino pentru a realiza un temporizator digital. Temporizarea va porni de la o valoare prestabilita in cadrul codului sursa si va avea rezolutia de o secunda, adica, pornind de la valoarea afisata initial, va numara din secunda in secunda, decrementand valoarea cu o unitate pana in momentul in care va la ajunge la zero.

### Schema electrica

Modalitatea de conectare este prezentata in codul sursa si in tabelul urmator:

Conexiuni intre pinii afisajului cu patru digiti si dispozitivul Arduino		
Semnificatie	Pinul afisajului	Conectare la Arduino
Pin A	11	A1
Pin B	7	D3
Pin C	4	D4
Pin D	2	D5
Pin E	1	A0
Pin F	10	D7
Pin G	5	D8
+1	12	D11
+2	9	D10
+3	8	D9
+4	6	D6
Pin DP	3	neconectat



### Codul sursa

/\*

Temporizator digital cu afisare pe patru digiti. Este recomandat sa nu modificati codul sursa in anumite sectiuni, intrucat puteti modifica valorile curentilor care strabat segmentele afisajului si riscati sa il deteriorati.

\*/

```
int digit1 = 11; //PWM pinul 12 al afisajului (+1)
int digit2 = 10; // PWM pinul 9 al afisajului(+2)
int digit3 = 9; // PWM pinul 8 al afisajului(+3)
int digit4 = 6; // PWM pinul 6 al afisajului(+4)
```

```
int segA = A1; //pinul 11 al afisajului
int segB = 3; // pinul 7 al afisajului
int segC = 4; // pinul 4 al afisajului
int segD = 5; // pinul 2 al afisajului
int segE = A0; // pinul 1 al afisajului
int segF = 7; // pinul 10 al afisajului
int segG = 8; // pinul 5 al afisajului
```

```
int start_num=30; // numarul de la care va porni temporizarea
unsigned long time;
```

```
void setup() {
  pinMode(segA, OUTPUT);
  pinMode(segB, OUTPUT);
  pinMode(segC, OUTPUT);
  pinMode(segD, OUTPUT);
  pinMode(segE, OUTPUT);
  pinMode(segF, OUTPUT);
  pinMode(segG, OUTPUT);
}
```

```

pinMode(segG, OUTPUT);

pinMode(digit1, OUTPUT);
pinMode(digit2, OUTPUT);
pinMode(digit3, OUTPUT);
pinMode(digit4, OUTPUT);

pinMode(13, OUTPUT);
}

void loop() {

  //long startTime = millis();
  if((millis()/1000) < start_num){
    displayNumber(start_num -(millis()/1000));
  }
  else {

    // in momentul in care s-a ajuns la zero, afisajul lumineaza intermitent
    time=millis();
    while(millis() < time+200) {
      displayNumber(0); //afisare 0 pentru 200 de milisecunde
    }
    time=millis();
    while(millis() < time+200) {
      lightNumber(10); // stingerea afisajului pentru 200 de milisecunde
    }
  }

  //while( (millis() - startTime) < 2000) {
  //displayNumber(1217);
  //}
  //delay(1000);

}

//fiind dat un numar, afisam 10:22
//dupa ce am rulat cele patru numere, afisajul este stins

/*
Iluminam afisajul
Fiecare digit va sta aprins cateva microsecunde apoi se va stinge pana cand se ajunge la un total
de 20milisecunde pentru a apela functia. Sa presupunem ca fiecare digit va ilumina timp de o
milisecunda; exista patru digiti si afisajul va fi stins pentru 16ms. Raportul este de 6.25%
(PWM). Vom defini o variabila pentru setarea luminozitatii, brightness, care variaza de la:
5000 lumina intensa (15.7mA pentru fiecare digit)

```

2000 lumina puternica (11.4mA pentru fiecare digit)  
 1000 lumina relativ puternica (5.9mA pentru fiecare digit)  
 500 normal (3mA pentru fiecare digit)  
 200 slab iluminat dar cat sa poata fi citit (1.4mA pentru fiecare digit)  
 50 slab iluminat dar cat sa poata fi citit (0.56mA pentru fiecare digit)  
 5 slab iluminat dar cat sa poata fi citit (0.31mA pentru fiecare digit)  
 1 slab iluminat dar cat sa poata fi citit in intuneric (0.28mA pentru fiecare digit)  
 \*/

```
void displayNumber(int toDisplay) {
#define DISPLAY_BRIGHTNESS 500

#define DIGIT_ON HIGH
#define DIGIT_OFF LOW

  long beginTime = millis();

  for(int digit = 4 ; digit > 0 ; digit--) {

    //aprindem un digit pentru o perioada scurta de timp
    switch(digit) {
    case 1:
      digitalWrite(digit1, DIGIT_ON);
      break;
    case 2:
      digitalWrite(digit2, DIGIT_ON);
      break;
    case 3:
      digitalWrite(digit3, DIGIT_ON);
      break;
    case 4:
      digitalWrite(digit4, DIGIT_ON);
      break;
    }

    //aprindem segmentele corespunzatoare
    lightNumber(toDisplay % 10);
    toDisplay /= 10;

    delayMicroseconds(DISPLAY_BRIGHTNESS);
    //afisam digitul pentru o fractiune de secunda

    //stingem toate segmentele
    lightNumber(10);
```

```
//stingem toti digitii
digitalWrite(digit1, DIGIT_OFF);
digitalWrite(digit2, DIGIT_OFF);
digitalWrite(digit3, DIGIT_OFF);
digitalWrite(digit4, DIGIT_OFF);
}

while( (millis() - beginTime) < 10) ;
//asteptam 20ms inainte sa afisam altceva
}

//fiind dat un numar, aprindem segmentele corespunzatoare afisarii acestuia
//daca numarul este egal cu 10, atunci il stingem
void lightNumber(int numberToDisplay) {

#define SEGMENT_ON LOW
#define SEGMENT_OFF HIGH

    switch (numberToDisplay){

    case 0:
        digitalWrite(segA, SEGMENT_ON);
        digitalWrite(segB, SEGMENT_ON);
        digitalWrite(segC, SEGMENT_ON);
        digitalWrite(segD, SEGMENT_ON);
        digitalWrite(segE, SEGMENT_ON);
        digitalWrite(segF, SEGMENT_ON);
        digitalWrite(segG, SEGMENT_OFF);
        break;

    case 1:
        digitalWrite(segA, SEGMENT_OFF);
        digitalWrite(segB, SEGMENT_ON);
        digitalWrite(segC, SEGMENT_ON);
        digitalWrite(segD, SEGMENT_OFF);
        digitalWrite(segE, SEGMENT_OFF);
        digitalWrite(segF, SEGMENT_OFF);
        digitalWrite(segG, SEGMENT_OFF);
        break;

    case 2:
        digitalWrite(segA, SEGMENT_ON);
        digitalWrite(segB, SEGMENT_ON);
        digitalWrite(segC, SEGMENT_OFF);
        digitalWrite(segD, SEGMENT_ON);
        digitalWrite(segE, SEGMENT_ON);
```

```
digitalWrite(segF, SEGMENT_OFF);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 3:

```
digitalWrite(segA, SEGMENT_ON);  
digitalWrite(segB, SEGMENT_ON);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_ON);  
digitalWrite(segE, SEGMENT_OFF);  
digitalWrite(segF, SEGMENT_OFF);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 4:

```
digitalWrite(segA, SEGMENT_OFF);  
digitalWrite(segB, SEGMENT_ON);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_OFF);  
digitalWrite(segE, SEGMENT_OFF);  
digitalWrite(segF, SEGMENT_ON);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 5:

```
digitalWrite(segA, SEGMENT_ON);  
digitalWrite(segB, SEGMENT_OFF);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_ON);  
digitalWrite(segE, SEGMENT_OFF);  
digitalWrite(segF, SEGMENT_ON);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 6:

```
digitalWrite(segA, SEGMENT_ON);  
digitalWrite(segB, SEGMENT_OFF);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_ON);  
digitalWrite(segE, SEGMENT_ON);  
digitalWrite(segF, SEGMENT_ON);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 7:

```
digitalWrite(segA, SEGMENT_ON);
```

```
digitalWrite(segB, SEGMENT_ON);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_OFF);  
digitalWrite(segE, SEGMENT_OFF);  
digitalWrite(segF, SEGMENT_OFF);  
digitalWrite(segG, SEGMENT_OFF);  
break;
```

case 8:

```
digitalWrite(segA, SEGMENT_ON);  
digitalWrite(segB, SEGMENT_ON);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_ON);  
digitalWrite(segE, SEGMENT_ON);  
digitalWrite(segF, SEGMENT_ON);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 9:

```
digitalWrite(segA, SEGMENT_ON);  
digitalWrite(segB, SEGMENT_ON);  
digitalWrite(segC, SEGMENT_ON);  
digitalWrite(segD, SEGMENT_ON);  
digitalWrite(segE, SEGMENT_OFF);  
digitalWrite(segF, SEGMENT_ON);  
digitalWrite(segG, SEGMENT_ON);  
break;
```

case 10:

```
digitalWrite(segA, SEGMENT_OFF);  
digitalWrite(segB, SEGMENT_OFF);  
digitalWrite(segC, SEGMENT_OFF);  
digitalWrite(segD, SEGMENT_OFF);  
digitalWrite(segE, SEGMENT_OFF);  
digitalWrite(segF, SEGMENT_OFF);  
digitalWrite(segG, SEGMENT_OFF);  
break;
```

```
}  
}
```

## Experimentul 27. Temporizator electronic cu potentiometru digital si buzzer

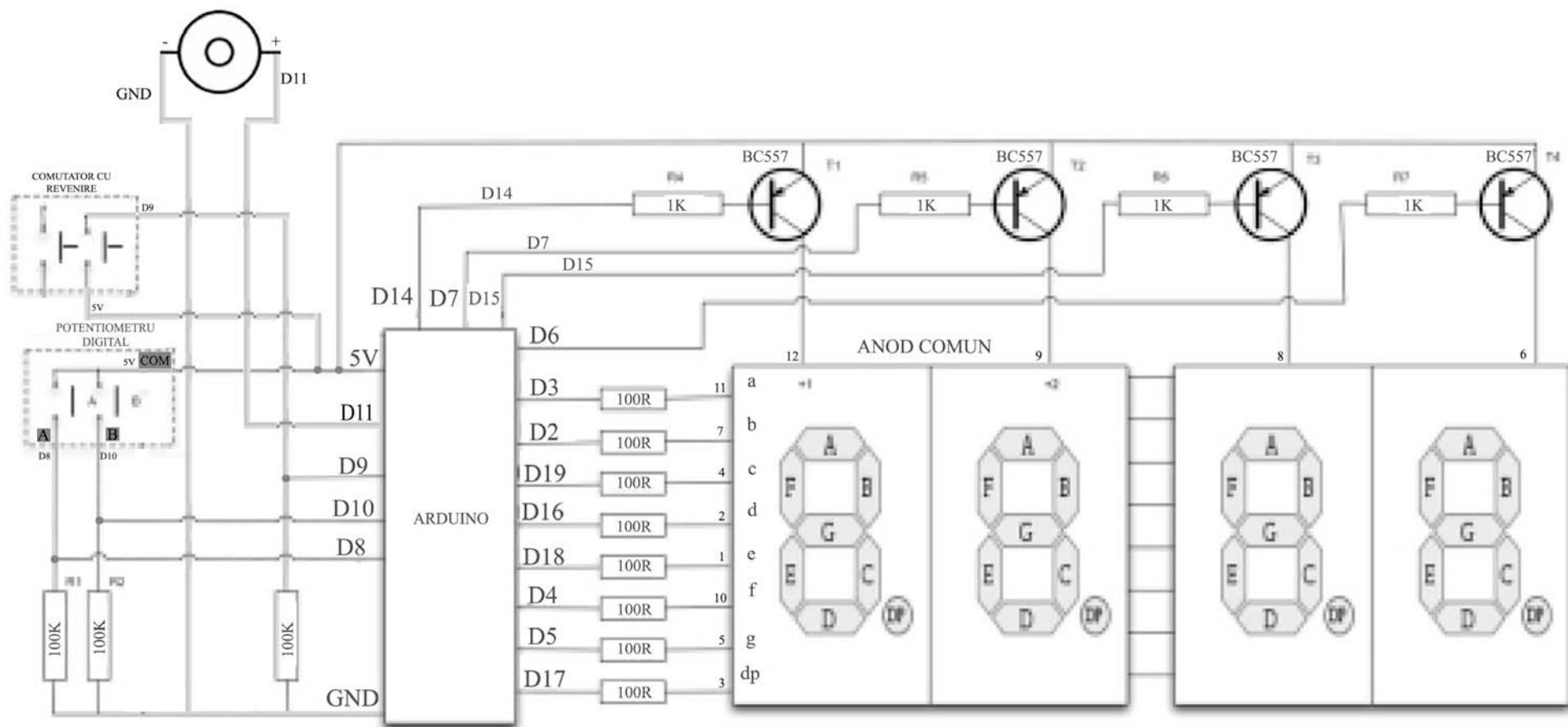
Acest montaj poate fi programat prin intermediul unui potentiometru digital, cu rezolutie de o secunda, pentru a asigura temporizarea oricarui proces. In montaj va fi integrat si un buzzer piezoelectric alimentat in 5V pentru a anunta momentul in care a expirat timpul setat.

Lista componentelor necesare	
Reprezentare in circuit	Descriere
	Arduino UNO
D1-4	Afisaj cu 4 digiti formati din 7 segmente (anod comun)
R1-3	Rezistor 100k $\Omega$
R4-7	Rezistor 1k $\Omega$
R8-15	Rezistor 100 $\Omega$
T1-4	Tranzistor PNP, BC557
	Comutator cu revenire
	Potentiometru digital
	Buzzer piezoelectric

### Schema electrica

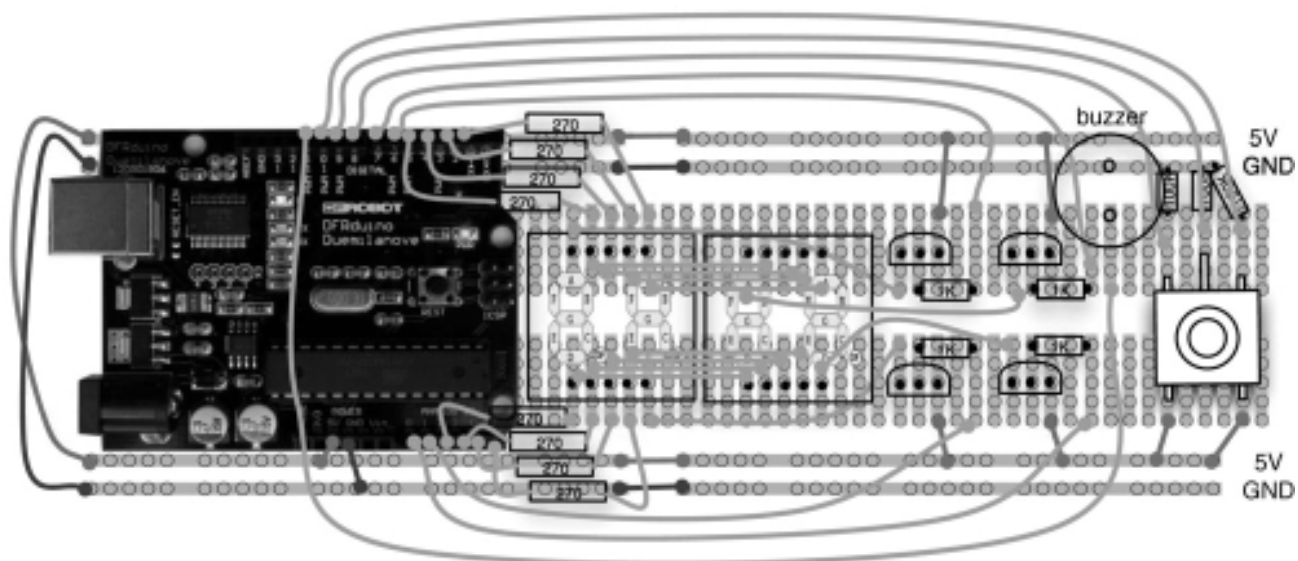
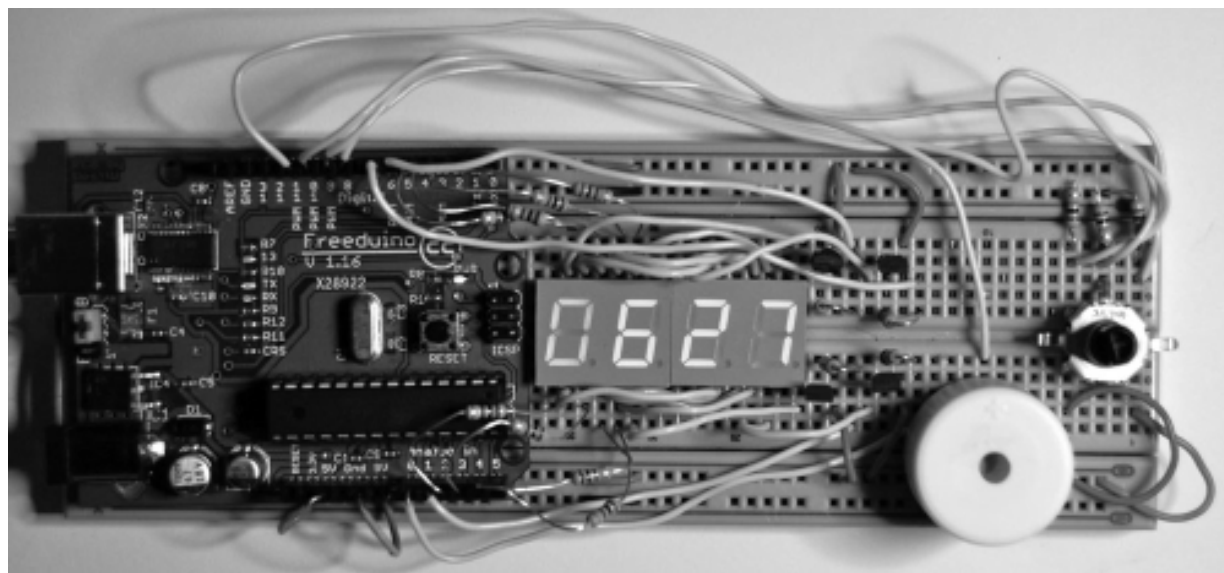
Proiectul este similar cu acela de zar electronic, avand, de data aceasta, un afisaj cu patru digiti si tranzistorii asociati. Vom utiliza si un potentiometru digital pentru a seta timpul de pornire al temporizatorului.





### Codul sursa

În cadrul codului sursa ne concentrăm pe actualizarea afisajului și crearea iluziei ca afisajele sunt aprinse simultan, deși numai unul dintre ele va fi aprins.



```
#include <EEPROM.h>

int segmentPins[] = {3, 2, 19, 16, 18, 4, 5, 17};
int displayPins[] = {14, 7, 15, 6};

int times[] = {5, 10, 15, 20, 30, 45, 100, 130, 200, 230, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 3000};

int numTimes = 19;
byte selectedTimeIndex;

int timerMinute;
int timerSecond;

int buzzerPin = 11;
int aPin = 8;
int bPin = 10;
int buttonPin = 9;

boolean stopped = true;

byte digits[10][8] = { // a b c d e f g .
  { 1, 1, 1, 1, 1, 1, 0, 0}, // 0
  { 0, 1, 1, 0, 0, 0, 0, 0}, // 1
  { 1, 1, 0, 1, 1, 0, 1, 0}, // 2
  { 1, 1, 1, 1, 0, 0, 1, 0}, // 3
  { 0, 1, 1, 0, 0, 1, 1, 0}, // 4
  { 1, 0, 1, 1, 0, 1, 1, 0}, // 5
  { 1, 0, 1, 1, 1, 1, 1, 0}, // 6
  { 1, 1, 1, 0, 0, 0, 0, 0}, // 7
  { 1, 1, 1, 1, 1, 1, 1, 0}, // 8
  { 1, 1, 1, 1, 0, 1, 1, 0} // 9
};

void setup(){

  for (int i=0; i < 8; i++){
    {
      pinMode(segmentPins[i], OUTPUT);
    }
  }

  for (int i=0; i < 4; i++){
    {
      pinMode(displayPins[i], OUTPUT);
    }
  }
}
```

```
pinMode(buzzerPin, OUTPUT);

pinMode(buttonPin, INPUT);

pinMode(aPin, INPUT);
pinMode(bPin, INPUT);

selectedTimeIndex = EEPROM.read(0);
timerMinute = times[selectedTimeIndex] / 100;
timerSecond = times[selectedTimeIndex] % 100;
}

void loop(){

if (digitalRead(buttonPin))
{
stopped = ! stopped;
digitalWrite(buzzerPin, LOW);
while (digitalRead(buttonPin)) { };
EEPROM.write(0, selectedTimeIndex);
}

updateDisplay();
}

void updateDisplay(){ // minut minut secunda secunda

int minsecs = timerMinute * 100 + timerSecond;
int v = minsecs;

for (int i = 0; i < 4; i ++){
{
int digit = v % 10;
setDigit(i);
setSegments(digit);
v = v / 10;
process();
}

setDigit(5); // vom opri toti digitii pentru a preveni iluminare inegala
}
```

```
void process(){

for (int i = 0; i < 100; i++)
{
int change = getEncoderTurn();
if (stopped)
{
changeSetTime(change);
}
else
{
updateCountingTime();
}
}

if (timerMinute == 0 && timerSecond == 0)
{
digitalWrite(buzzerPin, HIGH);
}
}

void changeSetTime(int change){

selectedTimeIndex += change;

if (selectedTimeIndex < 0)
{
selectedTimeIndex = numTimes;
}

else if (selectedTimeIndex > numTimes)
{
selectedTimeIndex = 0;
}

timerMinute = times[selectedTimeIndex] / 100;
timerSecond = times[selectedTimeIndex] % 100;
}

void updateCountingTime(){
static unsigned long lastMillis;
unsigned long m = millis();
if (m > (lastMillis + 1000) && (timerSecond > 0 || timerMinute > 0))
{
digitalWrite(buzzerPin, HIGH);
}
```

```
delay(10);
digitalWrite(buzzerPin, LOW);

if (timerSecond == 0)
{
    timerSecond = 59;
    timerMinute --;
}
else
{
    timerSecond --;
}
lastMillis = m;
}
}

void setDigit(int digit){
for (int i = 0; i < 4; i++)
{
    digitalWrite(displayPins[i], (digit != i));
}
}

void setSegments(int n){
for (int i = 0; i < 8; i++)
{
    digitalWrite(segmentPins[i], ! digits[n][i]);
}
}

int getEncoderTurn(){

// returneaza valorile -1, 0, or +1
static int oldA = LOW;
static int oldB = LOW;
int result = 0;
int newA = digitalRead(aPin);
int newB = digitalRead(bPin);
if (newA != oldA || newB != oldB)
{

// s-a efectuat o modificare
if (oldA == LOW && newA == HIGH)
{
    result = -(oldB * 2 - 1);
}
```

```
}  
  
oldA = newA;  
oldB = newB;  
return result;  
}
```

Temporizatorul va avea in orice moment doua stari. Acesta va fi oprit, caz in care, rotirea potentiometrului digital va avea drept rezultat modificarea timpului sau va fi pornit, situatie in care va numara descrescator la interval de o secunda. Butonul cu revenire montat in circuit va realiza trecerea intre cele doua stari.

Biblioteca de date EEPROM este utilizata pentru a stoca valoarea cea mai recent folosita, astfel incat, la alimentare sa dispozitivul va retine valoarea setata.

1. Copiati codul sursa si incarcati-l in dispozitivul Arduino.

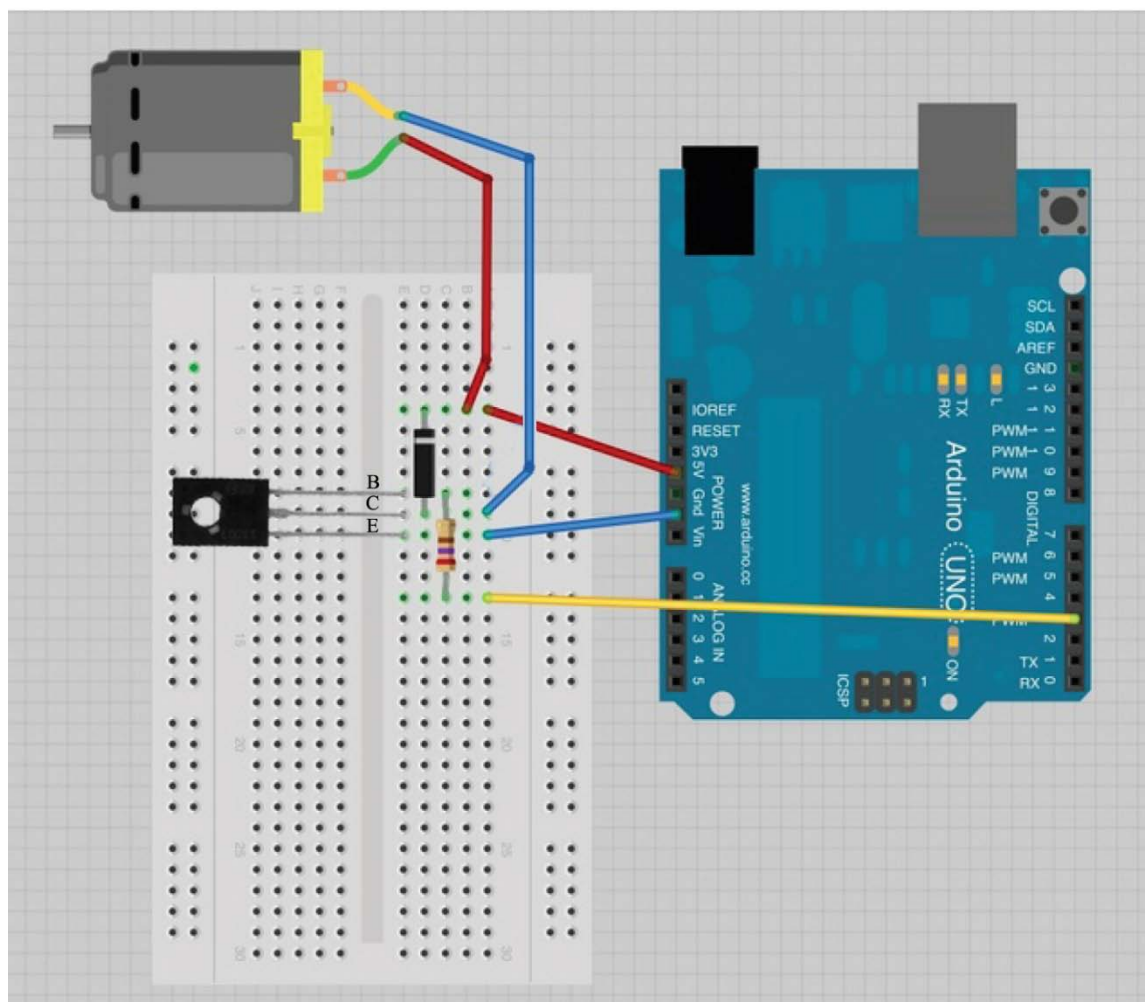
## Experiementul 28. Motorul in curent continuu

### Schema de montare pe placa Breadboard

In momentul cand veti construi montajul trebuie sa fiti atenti la doua aspecte: ca veti instala corect tranzistorul si ca terminalul marcat al diodei este indreptat cate linia de alimentare de +5V.

Atentie! In sarcina, motorul consuma, in medie 200mA si maxim 500mA, insa este posibil ca, daca il blocati sa consume o sarcina mai mare decat acea livrata de port-ul USB al computerului. Este recomandat ca, atunci cand desfasurati experimentul sa alimentati dispozitivul dintr-o sursa externa, de exemplu bateria de 9V din kit.

Este de preferat sa pastrati polaritatea motorului.



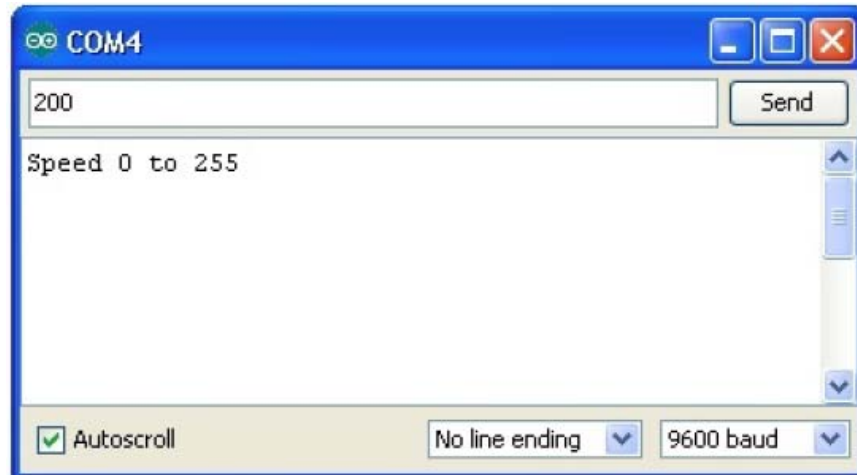


### Codul sursa

Incarcati urmatorul cod sursa in dispozitivul Arduino:

```
/*  
Motor in curent continuu  
*/  
int motorPin = 3;  
  
void setup(){  
  pinMode(motorPin, OUTPUT);  
  Serial.begin(9600);  
  while (! Serial);  
  Serial.println("Speed 0 to 255");  
}  
  
void loop(){  
  if (Serial.available())  
  {  
    int speed = Serial.parseInt();  
    if (speed >= 0 && speed <= 255)  
    {  
      analogWrite(motorPin, speed);  
    }  
  }  
}
```

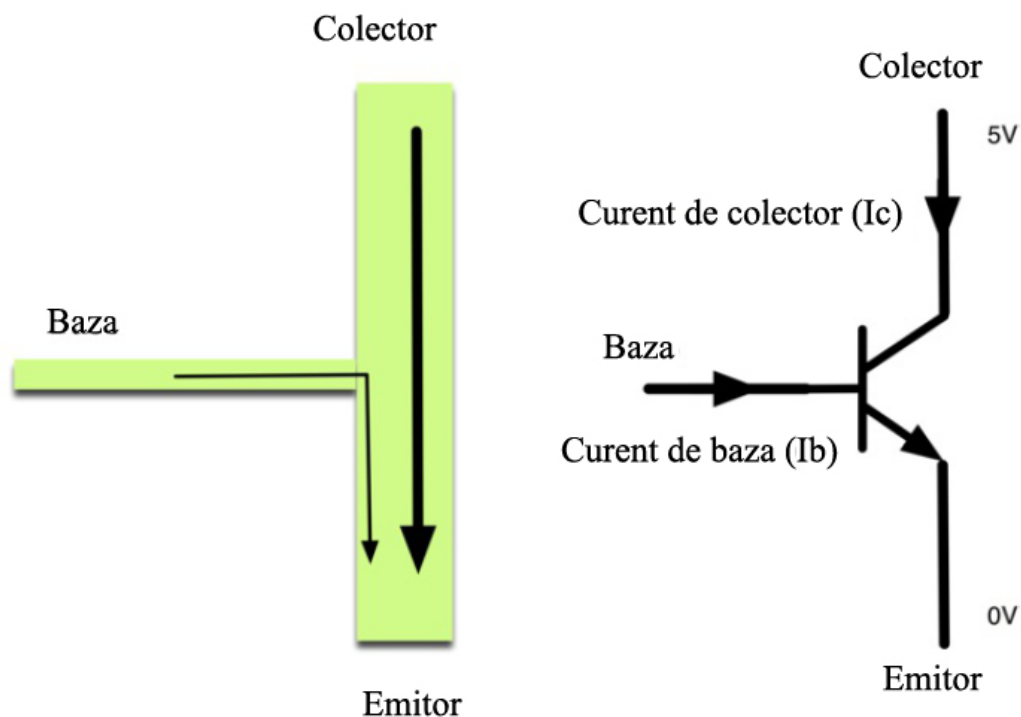
Tranzistorul se comporta precum un comutator, controland curentul pe care il primeste motorul. Pinul 3 al dispozitivului Arduino, a carui denumire in cadrul codului sursa este *motorPin*, va fi utilizat pentru a activa sau dezactiva tranzistorul. La rularea codului sursa, interfata va solicita sa introduceti in Seria Monitor o valoare intre 0 si 255 care va reprezenta viteza motorului.



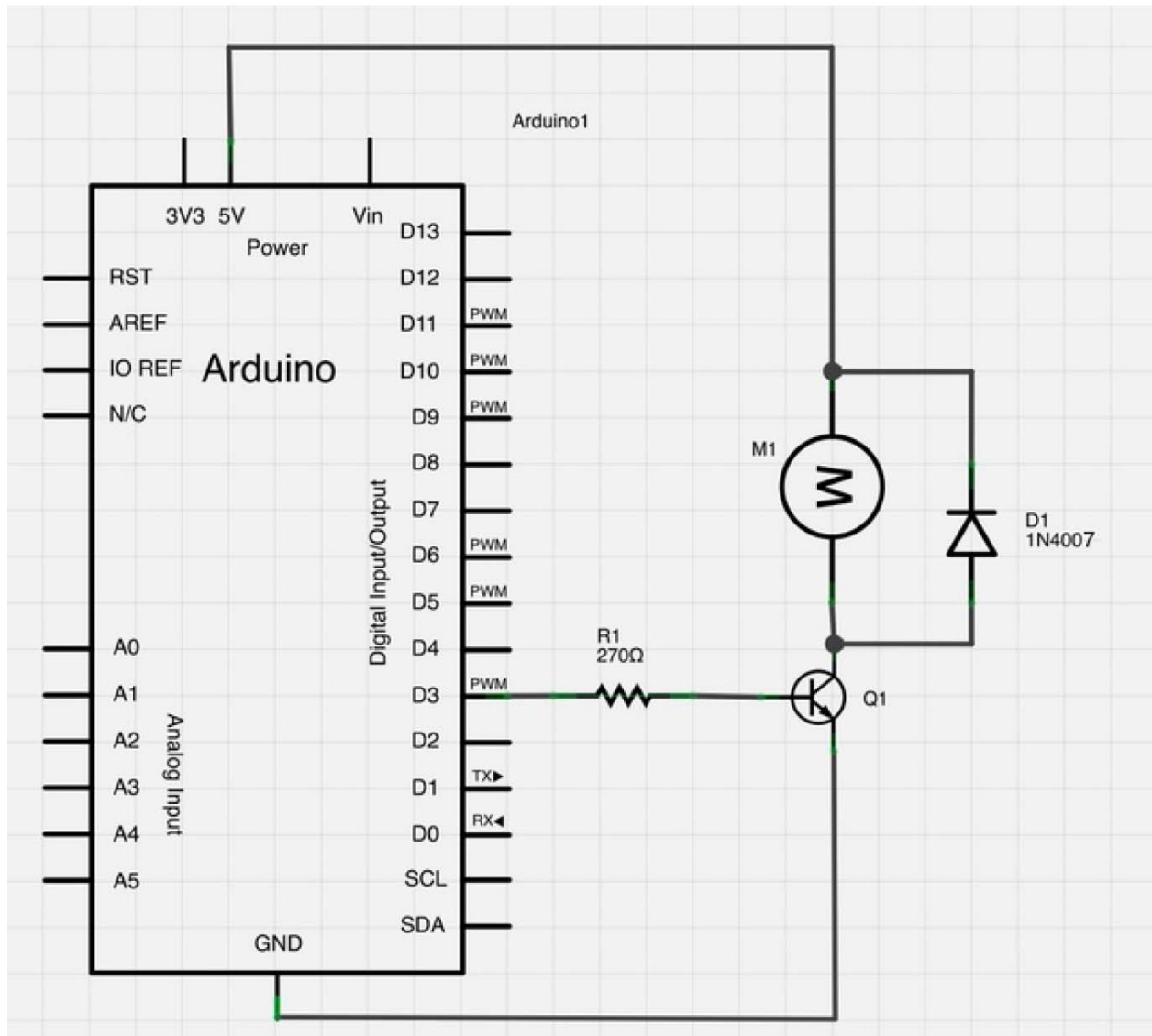
Comanda *Serial.parseInt* din functia *loop* este utilizata pentru a citi valoarea introdusa ca text in Serial Monitor si convertirea acesteia intr-un numar ( variabila de tip *int*, adica numar intreg). Puteti tasta orice valoare, insa codul sursa va verifica daca valoare se regaseste in intervalul 0 ~ 255 si o va transmite pinului analog.

### Tranzistoare

Este foarte probabil ca motorul de curent continuu sa consume un curent mai mare decat cel generat de o iesire digitala a dispozitivului Arduino. Daca l-am conecta direct la dispozitivul Arduino ar exista o probabilitate foarte mare ca acesta sa se defecteze. Un tranzistor precum BD139 poate fi utilizat ca un comutator care consuma un curent foarte mic de la dispozitivul Arduino dar poate controla un curent mai mare pe care il solicita motorul nostru in curent continuu.



Tranzistorul are trei terminale. Mare parte a curentului electric circula de la colector catre emitor, dar acest fenomen se intampla doar atunci cand un curent mic circula catre baza. Acest mic curent este furnizat de catre pinul de iesire digitala al dispozitivului Arduino. Mai jos este prezentat schema de principiu a montajului pe care urmeaza sa il construiti. La fel ca aranjarea pe placa breadboard, aceasta este o modalitate de a prezenta amplasarea componentelor in interiorul montajului pe care il construim.

Schema electrica

Pinul D3 al dispozitivului Arduino este conectat la un rezistor. La fel ca și în cazul utilizării unui LED, acesta limitează curentul din baza tranzistorului. În circuit este montată și o diodă conectată la motor. Diodele permit curentului să circule într-un singur sens. Atunci când se va opri motorul, va rezulta un impuls tensiune inversă care poate avaria tranzistorul sau dispozitivul Arduino. Tocmai de aceea este montată dioda, pentru a proteja împotriva tensiunii inverse.

Desigur lista experimentelor prezentate in cadrul acestui ghid, ce pot fi realizate utilizand componentele kitului si dispozitivul Arduino, este doar o mica parte din ceea ce puteti realiza documentandu-va suplimentar. Limita o constituie doar imaginatia voastra si disponibilitatea de a invata sa infaptuiti ceea ce va imaginati. <http://www.arduino.cc/> va constitui un punct de plecare pentru dvs.