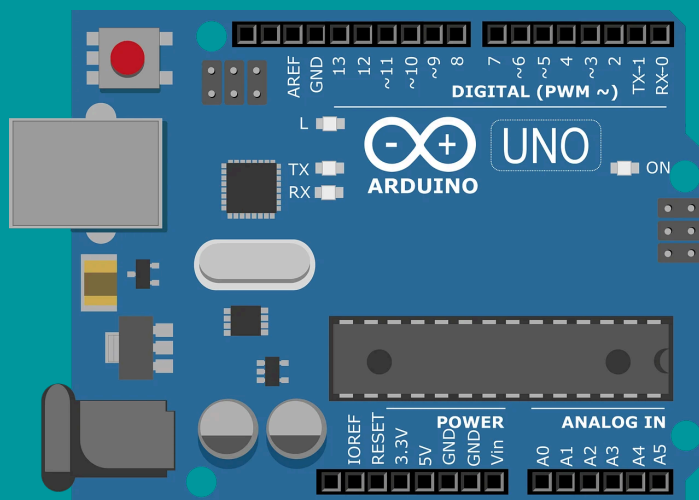


Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-00340573

PROIECT ERASMUS+

Introducere în Arduino

Ghid practic pentru începători



Curs practic · 10 ore · 10 proiecte · 2026

Curs realizat de

Manea-Ivan Ștefan



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-00340573

Disclaimer

Finanțat de Uniunea Europeană. Punctele de vedere și opiniile exprimate aparțin, însă, exclusiv autorului (autorilor) și nu reflectă neapărat punctele de vedere și opiniile Uniunii Europene sau ale Agenției Naționale pentru Programe Comunitare în Domeniul Educației și Formării Profesionale (ANPCDEFP). Nici Uniunea Europeană și nici ANPCDEFP nu pot fi ținute răspunzătoare pentru acestea.

Experiența mea în proiect

Participarea la mobilitatea Erasmus+ din Cipru a reprezentat pentru mine o experiență importantă de dezvoltare personală și profesională. Încă din etapa de selecție am învățat să redactez un CV și un dosar de candidatură, precum și să susțin un interviu. Pe parcursul mobilității am devenit mai deschis, mai încrezător în propriile abilități și mai dispus să îmi exprim ideile și să colaborez cu ceilalți.

Din punct de vedere educațional, proiectul mi-a descoperit pasiunea pentru robotică. Am dobândit cunoștințe practice despre Arduino, principiile de bază ale roboticii și modurile în care aceste tehnologii pot fi folosite în viața de zi cu zi. Experiența m-a motivat să continui să învăț, să îmi dezvolt competențele și să transform ceea ce am acumulat într-un material util și altor elevi.

Despre acest ghid

Curs realizat de **Manea-Ivan Ștefan**.

Acest material a fost realizat în cadrul unui proiect Erasmus+ și are scop educativ. Poate fi utilizat, distribuit și adaptat liber în context didactic, cu menționarea sursei. Toate mărcile și logo-urile aparțin deținătorilor lor.

Cuprins

Capitolul 0. Introducere în Arduino	4
Capitolul 1. Prezentarea echipamentelor	11
Capitolul 2. Siguranță și bune practici	14
Proiectul 1. Primul LED și semaforul	15
Proiectul 2. Buton și LED	21
Proiectul 3. Lampă RGB de dispoziție	25
Proiectul 4. Dimmer cu potențiometrul	29
Proiectul 5. Mini-pian cu buzzer	34
Proiectul 6. Lumină automată de noapte	38
Proiectul 7. Stație meteo cu ecran LCD	42
Proiectul 8. Senzor de parcare cu ultrasunete	48
Proiectul 9. Seif cu cod PIN	53
Proiectul 10. Proiect final: Sistem inteligent de alarmă	58
Glosar de termeni	66
Anexe	68

Capitolul 0. Introducere în Arduino

Bine ai venit! Acest ghid te învață electronică și programare de la zero, folosind Arduino. Nu ai nevoie de cunoștințe anterioare: dacă știi să apeși pe un buton și să urmezi niște pași, poți construi toate cele 10 proiecte din acest curs. La final vei înțelege cum „gândesc” dispozitivele din jurul tău și vei putea crea propriile invenții.

I Ce este Arduino?

Arduino este o **platformă electronică open-source** formată din două lucruri care lucrează împreună:

- **O plăcuță** (hardware): un mic computer pe care poți conecta LED-uri, butoane, senzori și motoare;
- **Un program** (software): aplicația **Arduino IDE**, în care scrii instrucțiuni pe calculator și le trimiți plăcii.

Ideea centrală: placa **citește** semnale de la lumea reală (un buton apăsat, temperatura din cameră, distanța până la un obiect), le **procesează** după regulile pe care le scrii tu și apoi **acționează** asupra lumii reale (aprinde un LED, pornește un motor, emite un sunet). Acest ciclu (**citește** → **decide** → **acționează**) stă la baza tuturor proiectelor.

i REȚINE

„Arduino” a apărut în 2005 în Italia, ca instrument prin care artiștii și studenții fără studii de inginerie să poată construi obiecte interactive. Astăzi este folosit în școli, universități și de milioane de pasionați din întreaga lume.

I Ce este un microcontroler?

Inima plăcii Arduino UNO este un **microcontroler**: un cip care conține un procesor, memorie și pini de intrare/ieșire, toate într-o singură componentă. Este ca un calculator foarte mic și simplu, dedicat unei singure sarcini: să ruleze la nesfârșit programul tău.

Spre deosebire de un laptop, microcontrolerul nu are ecran, tastatură sau sistem de operare. Pornește instant, execută exact ce i-ai spus și consumă foarte puțină energie. Microcontrolerul de pe UNO se numește **ATmega328P**.

I Anatomia plăcii Arduino UNO

Înainte să conectăm ceva, să ne familiarizăm cu placa. Iată elementele pe care le vei folosi cel mai des:

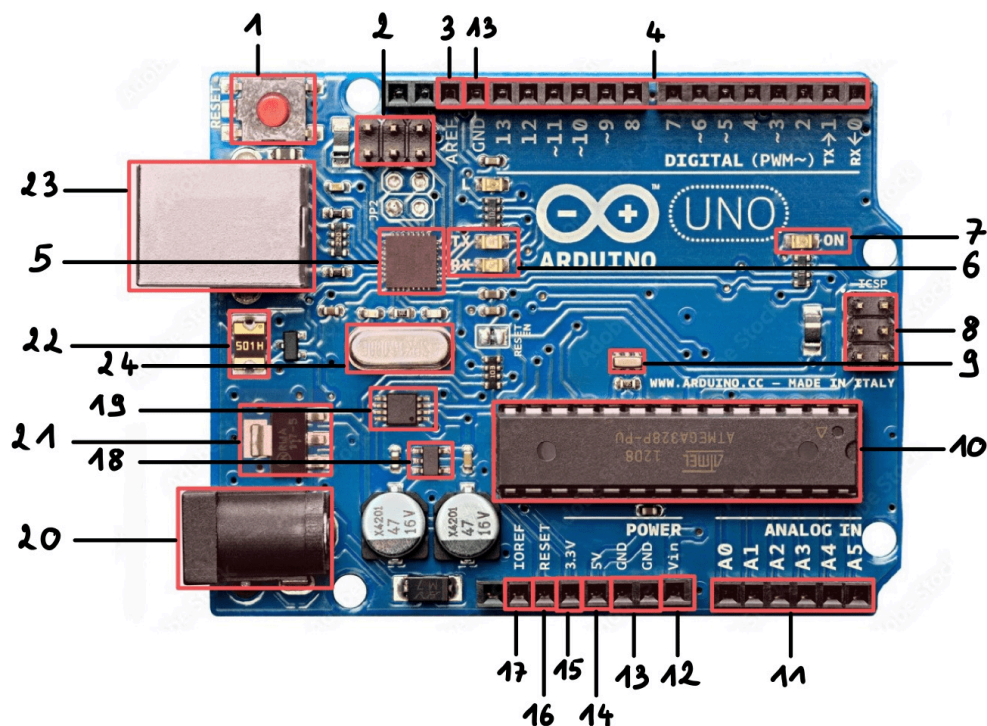


Figura 1: Placa Arduino UNO cu toate componentele numerotate

Legenda completă a plăcii

1	Butonul RESET	13	Pinii GND (masă)
2	Conector ICSP al cipului USB	14	Pinul 5V
3	Pinul AREF	15	Pinul 3.3V
4	Pinii digitali 0-13	16	Pinul RESET
5	Cip USB-serial (ATmega16U2)	17	Pinul IOREF
6	LED-urile TX și RX	18	Tranzistor MOSFET
7	LED-ul ON (alimentare)	19	Comparator (comutare alimentare)
8	Conector ICSP	20	Mufa de alimentare (jack)
9	Rezonator ceramic 16 MHz	21	Regulator de tensiune 5 V
10	Microcontrolerul ATmega328P	22	Siguranță resetabilă 500 mA
11	Pinii analogici A0-A5	23	Mufa USB tip B
12	Pinul Vin	24	Cristal de cuarț 16 MHz

Ce contează cu adevărat la început

Nu trebuie să reții toate cele 24 de elemente. Acestea sunt cele pe care le vei folosi efectiv în proiecte:

- **Mufa USB** 23: conectează placa la calculator; prin ea încarci programul și alimentezi placa.
- **Mufa de alimentare** 20: alimentare externă (7-12 V), de exemplu de la o baterie.

- **Pinii digitali 0-13** (4): pot fi intrări sau ieșiri cu doar două stări, pornit (HIGH , 5 V) sau oprit (LOW , 0 V). Pinii marcați cu ~ pot simula valori intermediare (PWM).
- **Pinii analogici A0-A5** (11): citesc valori variabile (de exemplu poziția unui potențiometrului sau intensitatea luminii). Pot fi folosiți și ca **pinii digitali** (digitalWrite , digitalWrite), atunci când ai nevoie de mai mulți pini decât cei 14 digitali.
- **Pinii de alimentare** (14) (15) (13): 5V și 3.3V furnizează tensiune, iar GND (masă) este „minusul” comun al oricărui circuit. Observă că există **mai multe pini GND** pe placă, toți echivalenți.
- **Microcontrolerul** (10): cipul mare, „creierul” care rulează programul tău.
- **Butonul RESET** (1): repornește programul de la început.
- **LED-ul ON** (7): confirmă că placa este alimentată.
- **LED-ul „L”**: un LED integrat, legat la pinul 13, folosit pentru primele teste. Nu este numerotat în imagine, dar îl găsești lângă pinii digitali.

i REȚINE

Restul componentelor (21, 22, 18, 19) formează **sistemul de alimentare**: iau tensiunea de la USB sau de la mufa jack, o coboară la 5 V, o comută automat între cele două surse și protejează placa la scurtcircuit. Lucrează în fundal, fără să te ocupe tu de ele. Cele două „ceasuri” (9 și 24) dau ritmul de lucru al cipurilor: 16 milioane de pași pe secundă.

✓ SFAT

Reține doar trei tipuri de pini pentru început: **alimentare** (5V , GND), **digitali** (0-13) și **analogici** (A0-A5). Restul le vei învăța pe parcurs, proiect cu proiect.

I Arduino IDE: programul cu care lucrăm

Arduino IDE este aplicația în care scriem codul (numit **sketch**) și îl trimitem plăcii.

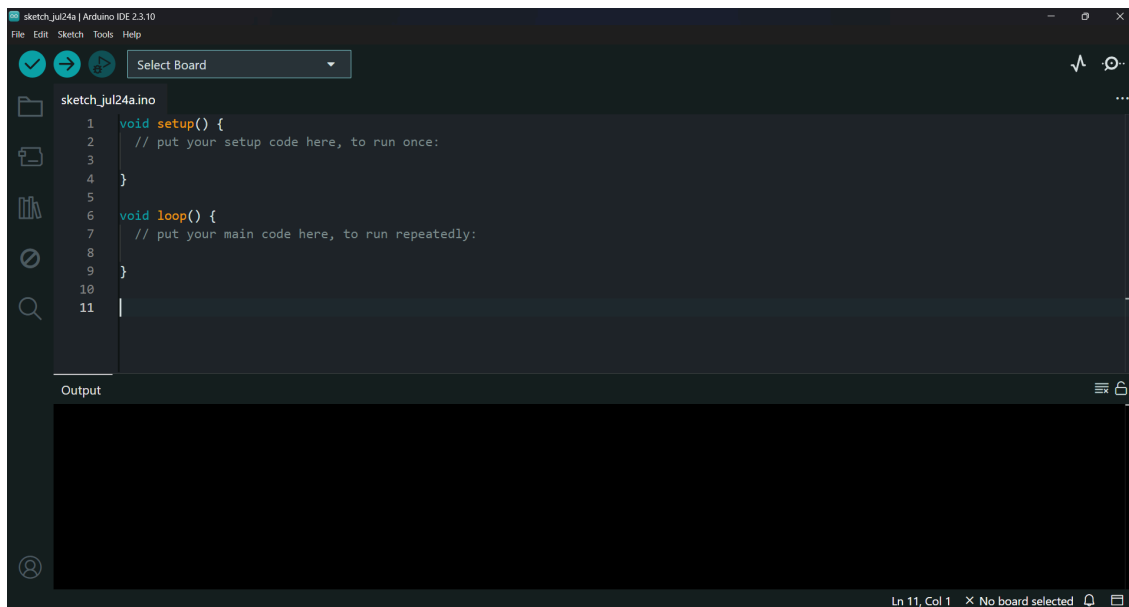


Figura 2: Interfața Arduino IDE: editorul de cod și butoanele Verify / Upload

Instalarea (o singură dată):

1. Intră pe arduino.cc/en/software și descarcă „Arduino IDE” pentru sistemul tău de operare (Windows, macOS sau Linux).
2. Instalează aplicația urmând pașii asistentului (acceptă instalarea driverelor dacă ți se cere).
3. Deschide Arduino IDE.

Primul upload (Blink):

1. Conectează placa la calculator cu cablul USB.
2. Din meniul **Tools** → **Board**, alege „Arduino Uno”.
3. Din **Tools** → **Port**, alege portul pe care apare placa (de ex. COM3 pe Windows).
4. Deschide un exemplu gata făcut: **File** → **Examples** → **01.Basics** → **Blink**.
5. Apasă butonul **Upload** (săgeata →). După câteva secunde, LED-ul „L” de pe placă ar trebui să clipească.

Fiecare sketch are două funcții obligatorii, pe care le vei vedea în toate proiectele:

```
void setup() {  
  // Rulează O SINGURĂ DATĂ, la pornire.  
  // Aici pregătim pinii și pornim comunicarea.  
}  
  
void loop() {  
  // Rulează LA NESFÂRȘIT, de sus în jos, apoi de la capăt.  
  // Aici punem comportamentul repetat al proiectului.  
}
```

! ATENȚIE / DEPANARE

Dacă la **Upload** apare o eroare de tip „port not found” sau „programmer is not responding”: verifică **cablul USB** (unele cabluri sunt doar de încărcare, fără date), reselectează **portul** corect din **Tools** → **Port** și asigură-te că ai ales placa **Arduino Uno**.



► Video: instalarea Arduino IDE

Scanează codul QR cu telefonul sau accesează linkul:

<https://youtu.be/kvppSa9MDQE>

I Bibliotecile: cod scris de alții

O **bibliotecă** este un pachet de cod pe care altcineva l-a scris deja și l-a făcut public, ca tu să nu reinventezi roata. Ca să afișezi un text pe un ecran LCD sau să citești un senzor de temperatură ar trebui zeci de linii complicate; cu o bibliotecă potrivită, scrii două.

Instalarea se face o singură dată, direct din Arduino IDE:

1. Deschide **Sketch** → **Include Library** → **Manage Libraries...**
2. Scrie numele bibliotecii în caseta de căutare.
3. Alege versiunea cea mai nouă și apasă **Install**.
4. Dacă IDE-ul te întreabă despre dependențe, răspunde **Install all**.

Bibliotecile de care vei avea nevoie în acest curs:

Bibliotecă	Autor	Unde o folosim
DHT sensor library	Adafruit	Proiectul 7, senzorul de temperatură
LiquidCrystal	inclusă în Arduino IDE	Proiectele 7, 9 și 10, ecranul LCD
Keypad	Mark Stanley, A. Brevig	Proiectul 9, tastatura matriceală
Servo	inclusă în Arduino IDE	Proiectele 9 și 10, servomotorul

! ATENȚIE / DEPANARE

Dacă după instalare IDE-ul tot se plânge cu o eroare de tipul `DHT.h: No such file or directory`, închide complet Arduino IDE și redeschide-l. Bibliotecile noi sunt recunoscute abia la repornire.



► Video: cum instalezi o bibliotecă

Scanează codul QR cu telefonul sau accesează linkul:

<https://youtu.be/Iym8vwXINuA>

I Breadboard-ul: laboratorul tău fără lipituri

Breadboard-ul (placă de test) ne permite să conectăm componente fără a le lipi. Are găuri legate electric între ele după un tipar fix:

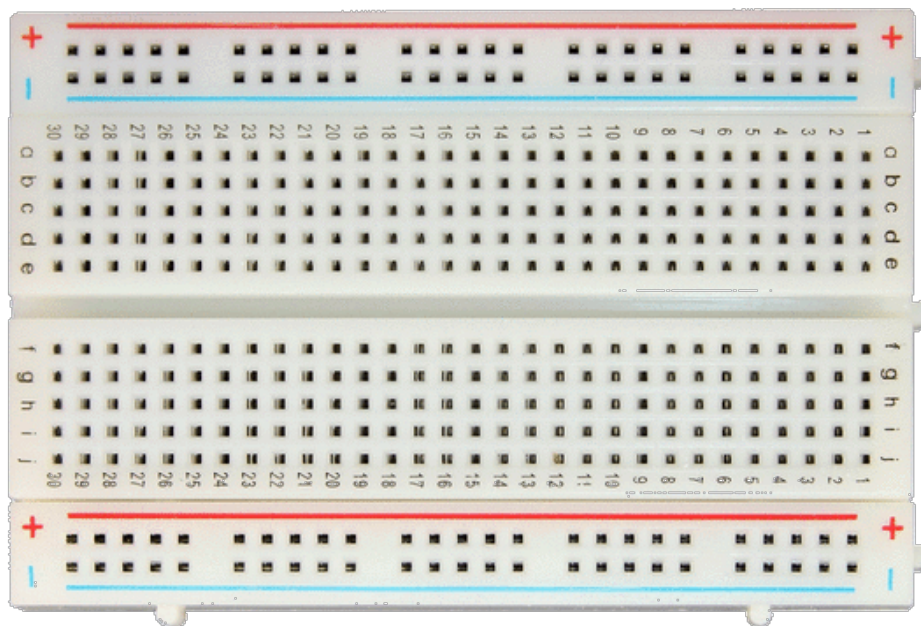


Figura 3: Conexiunile interne ale unui breadboard: liniile roșu/albastru și coloanele centrale

- **Liniile de alimentare** (marcate + roșu și - albastru), pe margini, sunt legate pe **orizontală**, pe toată lungimea. Aici aducem de obicei 5V și GND.
- **Coloanele centrale** sunt legate pe **verticală**, în grupuri de câte 5 găuri, separate de canalul din mijloc.
- **Canalul central** desparte cele două jumătăți, perfect pentru a așeza cipuri.

✓ SFAT

Obişnuieşte-te să conectezi mereu, la început, două fire de la placă la breadboard: unul de la 5V la linia roşie + şi unul de la GND la linia albastră -. Aşa vei avea alimentarea „la îndemână” pe toată lungimea plăcii de test.

I Cum citeşti schemele de conectare

În fiecare proiect vei găsi o **schemă pe breadboard**, care arată exact unde se pun firele. Ca s-o urmăreşti uşor, firele sunt colorate după o convenţie simplă:

Codul culorilor firelor

- Roşu** — 5V (alimentare, plus)
- Albastru sau negru** — GND (masă, minus)
- Galben** — semnal (date de la sau către un pin)

Pentru **LED-uri**, firul are **culoarea LED-ului** pe care îl reprezintă: spre LED-ul roşu merge un fir roşu, spre cel verde unul verde şi aşa mai departe.

Dacă într-o schemă culorile diferă de convenţia de mai sus, înseamnă că le-am schimbat pentru o **vizibilitate mai bună**.

I Puțină electronică: tensiune, curent, rezistență

Nu ai nevoie de formule complicate, dar trei noțiuni te ajută enorm:

- **Tensiunea** (V, volți): „presiunea” care împinge electricitatea. Arduino lucrează la 5 V.
- **Curentul** (I, amperi): „debitul”, cât de multă electricitate curge.
- **Rezistența** (R, ohmi Ω): cât de mult se opune un material trecerii curentului.

Ele sunt legate prin **Legea lui Ohm**: $V = I \times R$. În practică, reținem o regulă de aur: componentele sensibile (precum LED-urile) au nevoie de o **rezistență de limitare** în serie, altfel prin ele trece prea mult curent și se ard.

i REȚINE

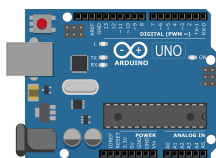
De aceea în aproape fiecare proiect cu LED vei folosi o rezistență de **220 Ω** sau **330 Ω** . Rolul ei este să protejeze LED-ul, limitând curentul la o valoare sigură.

Cu aceste noțiuni ești pregătit. În capitolul următor cunoaștem, pe rând, fiecare componentă din trusă.

Capitolul 1. Prezentarea echipamentelor

Aceasta este „trusa” cu care vei construi toate proiectele. Nu trebuie să le memorezi acum. Revino la acest capitol ori de câte ori întâlnești o componentă nouă. Fiecare card arată **ce este** componenta și **la ce folosește**.

I Placa, alimentarea și zona de lucru



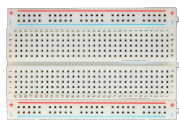
Placă Arduino UNO

Creierul proiectelor. Rulează programul și comandă toate celelalte componente.



Cablu USB (tip B)

Conectează placa la calculator: prin el încarci programul și alimentezi placa.



Breadboard

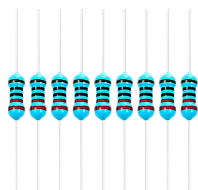
Placă de test cu găuri conectate electric. Permite montaje rapide, fără lipituri.



Fire jumper

Fire cu capete rigide (tată-tată, tată-mamă) pentru a lega componentele între ele.

I Componente de bază



Rezistență

Limitează curentul care trece printr-un circuit. Protejează LED-urile și alte componente.



LED

Diodă care emite lumină când curentul trece prin ea în sensul corect. Are un plus și un minus.



LED RGB

Trei LED-uri (roșu, verde, albastru) într-o singură capsulă. Prin amestec obții orice culoare.



Buton (push-button)

Înterupător momentan: conduce curentul doar cât timp este apăsat.



Potențiometru

Rezistență reglabilă prin rotire. Îl folosim pentru a citi o valoare variabilă (0-1023).



Buzzer piezoelectric

Emite sunete atunci când îi trimitem un semnal. Îl folosim pentru melodii și alarme.

I Senzori, module și actuatore



Tastatură matriceală 4×4

16 taste pe doar 8 pini. Permite introducerea de cifre și litere, ca la un bancomat.



Fotorezistor (LDR)

Rezistență care se modifică în funcție de lumină. Detectează întunericul sau lumina.



Senzor DHT11

Măsoară temperatura și umiditatea aerului și le trimite plăcii ca valori digitale.



Senzor ultrasonic HC-SR04

Măsoară distanța până la un obiect trimițând ultrasunete, ca un sonar.



Servomotor SG90

Motor care se rotește la un unghi precis (0-180°). Mișcă bariere, brațe, capace.



Ecran LCD 16×2

Afișează 2 rânduri a câte 16 caractere. Perfect pentru a arăta valori măsurate.

I Cum citim o rezistență (codul culorilor)

Rezistențele nu au valoarea scrisă pe ele, ci codată prin **inele colorate**. Primele două inele sunt cifre, al treilea este multiplicatorul (numărul de zerouri), iar ultimul (auriu/argintiu) indică toleranța.

Inel	Culoare	Cifră	Multiplicator
	Negru	0	×1
	Maro	1	×10
	Roșu	2	×100
	Portocaliu	3	×1 000
	Galben	4	×10 000
	Verde	5	×100 000
	Albastru	6	×1 000 000
	Violet	7	×10 000 000
	Gri	8	
	Alb	9	

Ultimul inel, puțin mai depărtat de celelalte, arată **toleranța** (cât de exactă este valoarea): auriu = ±5%, iar argintiu = ±10%. Citește inelele **dinspre capătul opus** celui cu inelul de toleranță.

✓ SFAT

Rezistența de **220 Ω** , cea mai folosită în acest curs, are inelele    **roșu, roșu, maro** (2, 2, $\times 10 \rightarrow 220 \Omega$). Cea de **10 k Ω** are    **maro, negru, portocaliu** (1, 0, $\times 1\,000 \rightarrow 10\,000 \Omega$).



Calculator online pentru rezistențe

Nu vrei să calculezi de fiecare dată? Alege culorile inelelor pe site și îți spune instant valoarea în ohmi. Funcționează și invers: introduci valoarea dorită și îți arată ce culori să cauți.

<https://www.calculator.net/resistor-calculator.html>

i REȚINE

Rezistențele **nu au polaritate**, deci le poți conecta în orice sens. LED-urile, în schimb, **au polaritate**: piciorul lung este **plusul (+, anod)**, iar cel scurt este **minusul (–, catod)**, dinspre partea teșită a capsulei.

Capitolul 2. Siguranță și bune practici

Arduino funcționează la **5 volți**, o tensiune joasă, care **nu prezintă pericol de electrocutare**. Totuși, câteva reguli simple îți protejează placa și componentele și te ajută să eviți frustrarea unui montaj care „nu merge”.

I Reguli de aur

1. **Conectează întâi, alimentează după.** Realizează complet montajul cu placa **deconectată** de la USB. Abia la final conectezi cablul. Așa eviți scurtcircuiturile accidentale în timp ce muți firele.
2. **Respectă polaritatea.** GND merge la GND, 5V la +. Inversarea alimentării la senzori sau la ecran le poate distruge.
3. **Folosește mereu rezistență de limitare la LED-uri.** Un LED conectat direct la 5 V, fără rezistență, se arde în câteva secunde.
4. **Nu lega niciodată direct 5V la GND.** Este un scurtcircuit: placa se încinge și intră în protecție (sau se poate deteriora).
5. **Un singur pin, o singură funcție.** Nu conecta două ieșiri care se „bat” pe același fir.

! ATENȚIE / DEPANARE

Nu alimenta motoare mari direct din pinul de 5 V al plăcii! Servomotorul mic SG90 din trusă este acceptabil pentru teste scurte, dar motoarele mai puternice trag prea mult curent și pot reseta sau arde placa. Pentru ele se folosește o **sursă externă** și, eventual, un **driver de motor**.

I Bune practici care îți salvează timp

- **Cod de culori la fire.** Folosește **roșu pentru 5V** și **negru/albastru pentru GND**. Când toate montajele respectă aceeași convenție, greșelile se văd imediat.
- **Verifică de două ori înainte de alimentare.** Urmărește fiecare fir de la un capăt la celălalt și compară-l cu schema.
- **Deconectează când modifici.** Scoate USB-ul ori de câte ori rearanjezi componente pe breadboard.
- **Lucrează ordonat.** Fire scurte și aranjate înseamnă montaje ușor de depanat.
- **Salvează des sketch-urile** și dă-le nume clare (de ex. proiect_02_buton).

✓ SFAT

Dacă un proiect nu funcționează, verifică în această ordine: (1) alimentarea (5V și GND ajung unde trebuie?), (2) conexiunile (vreun fir ieșit din breadboard?), (3) polaritatea LED-urilor și a senzorilor, (4) pinii din cod corespund cu cei din montaj. În **90% din cazuri** problema este una dintre acestea patru.

i REȚINE

Componentele din această trusă sunt reutilizabile. Le poți demonta după fiecare proiect și folosi din nou; nimic nu se consumă. Singura piesă care se poate „arde” prin utilizare greșită este LED-ul fără rezistență.

Proiectul 1. Primul LED și semaforul

Dificultate ●○○

Durată 60 min

Componente 5

Începem cu „Salut, lume!”, din electronică: un LED care clipește. Apoi, cu exact aceleași cunoștințe, dar cu trei LED-uri, construim un semafor adevărat. Vei vedea cât de puțin e nevoie ca un montaj simplu să devină un obiect care chiar face ceva.

I Ce vei învăța

- Ce este un **pin digital de ieșire** și cum îl pornești sau îl oprești;
- Funcțiile `pinMode()`, `digitalWrite()` și `delay()`;
- Cum se conectează un LED cu rezistență de protecție pe breadboard;
- Cum construiești o **secvență temporizată** și cum repeți acțiuni cu o buclă `for`.

I Componente necesare

Cant.	Componentă
1×	Placă Arduino UNO
1×	Breadboard
1×	LED roșu
1×	LED galben (pentru partea a doua)
1×	LED verde (pentru partea a doua)
3×	Rezistență 220 Ω
5×	Fire jumper (tată-tată)

I Partea 1: un singur LED care clipește

Schema de conectare

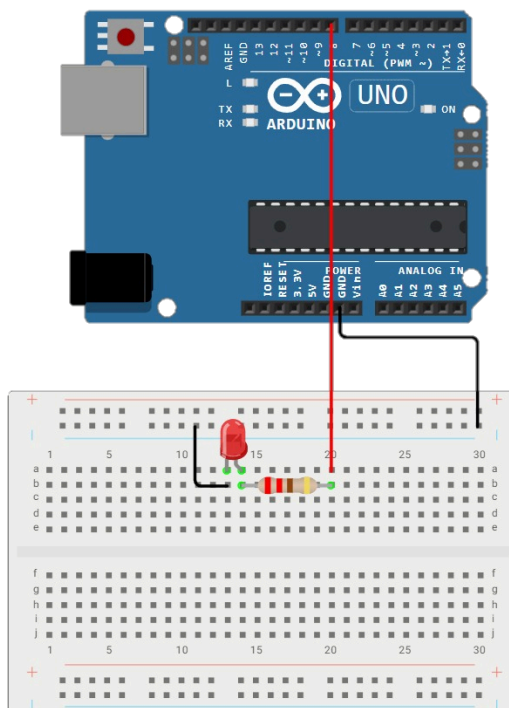


Figura 4: Montajul Blink, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- ▶ Construiește și rulează circuitul în browser, fără componente fizice:
<https://wokwi.com/projects/470424206084711425>

Cum funcționează

Pinul 8 al plăcii poate scoate două tensiuni: **5 V** (stare HIGH) sau **0 V** (stare LOW). LED-ul se aprinde când prin el trece curent, deci când pinul e HIGH. Rezistența de **220 Ω** **limitează curentul** ca LED-ul să nu se ardă. Programul aprinde LED-ul o secundă, îl stinge o secundă și repetă la nesfârșit.

Montaj pas cu pas

1. Cu placa **deconectată** de la USB, așază LED-ul pe breadboard: piciorul **lung (+)** într-o coloană, piciorul **scurt (-)** în altă coloană.
2. Conectează o **rezistență de 220 Ω** în serie cu piciorul **lung (+)**: un capăt în coloana piciorului lung, celălalt într-o coloană liberă.
3. Leagă **pinul 8** de pe placă la celălalt capăt al rezistenței.
4. Leagă piciorul **scurt (-)** al LED-ului la linia albastră **-** (masă), apoi **GND** de pe placă la aceeași linie **-**.
5. Conectează placa la USB.

Codul

```
/*  
  Proiectul 1 - Blink (LED care clipește)  
  Aprinde și stinge un LED o dată pe secundă.  
  Curs Arduino Erasmus+  
*/
```

```
const int PIN_LED = 8;           // pinul digital la care e conectat LED-ul

void setup() {
    pinMode(PIN_LED, OUTPUT);    // pinul LED-ului devine IESIRE
}

void loop() {
    digitalWrite(PIN_LED, HIGH); // aprinde LED-ul (trimite 5V)
    delay(1000);                 // asteapta 1000 ms = 1 secunda
    digitalWrite(PIN_LED, LOW);  // stinge LED-ul (0V)
    delay(1000);                 // asteapta inca o secunda
}
```

Explicația codului

- `const int PIN_LED = 8;` : dăm un **nume** pinului. Dacă mutăm LED-ul pe alt pin, schimbăm o singură linie.
- `pinMode(PIN_LED, OUTPUT);` : în `setup()` spunem că pinul 8 este **ieșire** (comandă, nu citește).
- `digitalWrite(PIN_LED, HIGH);` : pune 5 V pe pin → LED aprins. `LOW` pune 0 V → stins.
- `delay(1000);` : oprește programul **1000 ms** (1 secundă). Fără el, LED-ul ar clipi atât de repede încât ar părea aprins continuu.

✓ SFAT

Vrei să clipească mai repede? Micșorează valorile din `delay()`, de exemplu `200`. Valoarea este în **milisecunde**: 1000 ms = 1 s.

I Partea 2: de la un LED la semafor

Acum adăugăm încă două LED-uri. Nu ai nevoie de nicio funcție nouă: doar de aceleași `digitalWrite` și `delay`, folosite într-o **ordine** care imită un semafor real.

Schema de conectare

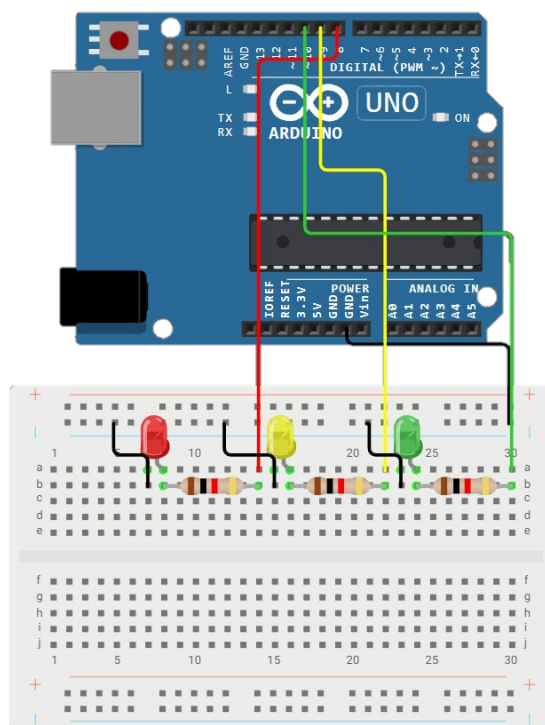


Figura 5: Semafor, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470425772806316033>

Montaj pas cu pas

1. Lasă LED-ul roșu pe **pinul 8** și adaugă alte două LED-uri lângă el.
2. Pune câte o rezistență de $220\ \Omega$ în serie cu piciorul **lung (+)** al fiecărui LED nou.
3. Conectează, prin rezistență, piciorul **lung (+)** al LED-ului galben la **pinul 9** și pe cel verde la **pinul 10**. Picioarele **scurte (-)** merg la linia **-**.
4. Verifică din nou firul de **GND**, apoi conectează placa la USB.

Codul

```
/*  
  Proiectul 1 - Semafor (partea a doua)  
  Simuleaza un semafor cu trei LED-uri: roșu, galben, verde.  
  Curs Arduino Erasmus+  
*/  
  
const int PIN_ROSU    = 8;  
const int PIN_GALBEN  = 9;  
const int PIN_VERDE   = 10;  
  
void setup() {  
  pinMode(PIN_ROSU,    OUTPUT);  
  pinMode(PIN_GALBEN,  OUTPUT);  
  pinMode(PIN_VERDE,   OUTPUT);  
}
```

```
}  
  
void loop() {  
  // 1. ROSU aprins - stop (5 secunde)  
  digitalWrite(PIN_ROSU, HIGH);  
  delay(5000);  
  
  // 2. ROSU + GALBEN - pregătire de plecare (2 secunde)  
  digitalWrite(PIN_GALBEN, HIGH);  
  delay(2000);  
  
  // 3. VERDE aprins - liber (5 secunde)  
  digitalWrite(PIN_ROSU, LOW);  
  digitalWrite(PIN_GALBEN, LOW);  
  digitalWrite(PIN_VERDE, HIGH);  
  delay(5000);  
  
  // 4. GALBEN clipind - atenție (3 clipiri)  
  digitalWrite(PIN_VERDE, LOW);  
  for (int i = 0; i < 3; i++) {  
    digitalWrite(PIN_GALBEN, HIGH);  
    delay(400);  
    digitalWrite(PIN_GALBEN, LOW);  
    delay(400);  
  }  
  // ciclul se reia de la început (rosu)  
}
```

Explicația codului

- În `setup()` declarăm toți cei trei pini ca `OUTPUT`.
- În `loop()`, fiecare fază aprinde sau stinge anumite LED-uri și așteaptă cu `delay()`.
- Bucla `for (int i = 0; i < 3; i++) { ... }` execută de **3 ori** codul dintre acolade: aici, o clipire a galbenului. Este mai scurt și mai clar decât să scriem de trei ori aceleași linii.

Sfaturi și depanare

✓ SFAT

Vrei un semafor cu ritm real? Mărește durata verdelui și a roșului la 10 secunde (10000). Vrei să testezi rapid? Micșorează toate valorile la câteva sute de milisecunde.

! ATENȚIE / DEPANARE

LED-ul nu se aprinde? (1) Verifică **polaritatea**: piciorul lung merge spre pin (prin rezistență), cel scurt spre `GND`. (2) Rezistența și LED-ul trebuie să fie pe **aceeași coloană** pentru a fi conectate. (3) Confirmă că firul e pe pinul corect.

Toate rămân stinse? Verifică firul de `GND`: dacă lipsește, niciun LED nu are circuit închis.

I Provocare

★ PROVOCARE

Adaugă un **semafor pentru pietoni** cu încă două LED-uri (roșu și verde) care funcționează invers față de cel auto: când mașinile au verde, pietonii au roșu. Bonus: fă LED-ul verde al pietonilor să clipească înainte de a se face roșu.

Proiectul 2. Buton și LED

Dificultate ●●○

Durată 45 min

Componente 4

Până acum placa doar **comanda**. Acum învață să **asculte**: un buton devine intrarea care decide când se aprinde LED-ul.

Ce vei învăța

- Ce este un **pin digital de intrare**;
- Funcția `digitalRead()` și decizia cu `if / else`;
- Rezistența internă `INPUT_PULLUP` și de ce logica butonului este „inversată”.

Componente necesare

Cant.	Componentă
1x	Placă Arduino UNO
1x	Breadboard
1x	Buton (push-button)
1x	LED
1x	Rezistență 220 Ω (pentru LED)

Schema de conectare

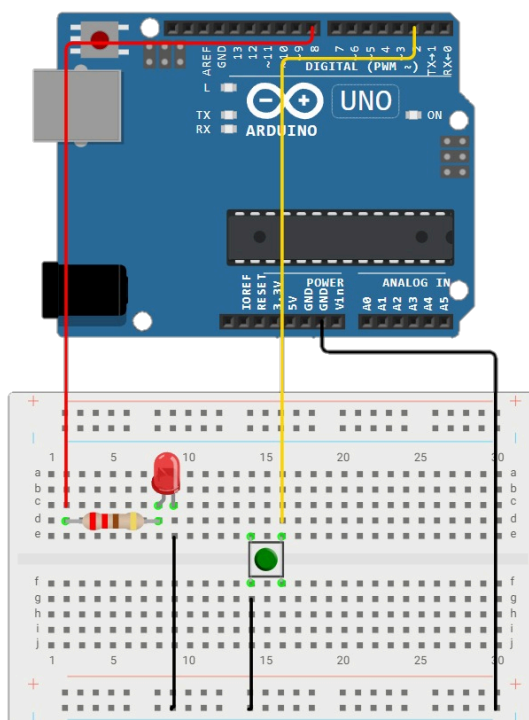


Figura 6: Buton și LED, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470606328669932545>

I Cum funcționează

Un pin de **intrare** citește dacă la el ajunge 5 V (HIGH) sau 0 V (LOW). Ca să nu „plutească” într-o valoare nesigură când butonul e liber, folosim rezistența internă INPUT_PULLUP , care ține pinul la HIGH . Când apeși butonul, îl legi la GND , iar pinul devine LOW . De aceea: **neapăsăat = HIGH, apăsăat = LOW**.

Montaj pas cu pas

1. Așază butonul **peste canalul central** al breadboard-ului (picioarele de o parte și de alta).
2. Leagă un picior al butonului la GND .
3. Leagă celălalt picior al butonului la **pinul 2**.
4. Montează LED-ul cu rezistența de 220 Ω: piciorul lung prin rezistență la **pinul 8**, cel scurt direct la GND .
5. Conectează placa la USB.

I Codul

```
/*
  Proiectul 2 - Buton si LED
  Aprinde LED-ul doar cat timp butonul este apasat.
  Folosim rezistenta internă PULL-UP (nu ne trebuie rezistenta externa).
  Curs Arduino Erasmus+
*/

const int PIN_BUTON = 2; // butonul intre pinul 2 si GND
const int PIN_LED = 8; // LED-ul cu rezistenta de 220 ohmi

void setup() {
  pinMode(PIN_BUTON, INPUT_PULLUP); // activeaza rezistenta internă pull-up
  pinMode(PIN_LED, OUTPUT);
}

void loop() {
  // Cu INPUT_PULLUP: neapasat = HIGH, apasat = LOW (logica inversata)
  int stare = digitalRead(PIN_BUTON);

  if (stare == LOW) { // butonul este apasat
    digitalWrite(PIN_LED, HIGH);
  } else { // butonul este liber
    digitalWrite(PIN_LED, LOW);
  }
}
```

I Explicația codului

- pinMode(PIN_BUTON, INPUT_PULLUP); : pinul 2 devine **intrare** cu rezistență pull-up internă.
- int stare = digitalRead(PIN_BUTON); : citim starea butonului (HIGH sau LOW).
- if (stare == LOW) { ... } else { ... } : dacă e apăsăat (LOW) aprindem LED-ul, altfel îl stingem. == înseamnă „este egal cu” (comparație), diferit de = (atribuire).

i REȚINE

De ce nu folosim o rezistență externă la buton? INPUT_PULLUP activează una **în interiorul** microcontrolerului. Mai puține piese, montaj mai simplu, aceeași funcție.

I Sfaturi și depanare

✓ SFAT

Vrei ca butonul să **comute** LED-ul (apeși o dată → aprins, apeși din nou → stins)? Vei avea nevoie să reții starea anterioară a butonului și să detectezi **momentul apăsării**. Aceasta e „provocarea” de mai jos.

! ATENȚIE / DEPANARE

LED-ul e mereu aprins sau pâlpâie singur? Probabil ai uitat `INPUT_PULLUP` și pinul „plutește”. **Nu reacționează la buton?** Verifică dacă un picior chiar ajunge la `GND` și celălalt la pinul 2 (butoanele au picioarele legate în perechi, încearcă altă diagonală).

I Provocare

Montaj pas cu pas

1. Așază butonul **peste canalul central** al breadboard-ului (picioarele de o parte și de alta).
2. Leagă un picior al butonului la `GND`.
3. Leagă celălalt picior al butonului la **pinul 2**.
4. Montează LED-ul cu rezistența de 220 Ω : piciorul lung prin rezistență la **pinul 8**, cel scurt direct la `GND`.
5. Conectează placa la USB.

I Codul

```
/*
  Proiectul 2 - Buton si LED
  Aprinde LED-ul doar cat timp butonul este apasat.
  Folosim rezistenta interna PULL-UP (nu ne trebuie rezistenta externa).
  Curs Arduino Erasmus+
*/

const int PIN_BUTON = 2; // butonul intre pinul 2 si GND
const int PIN_LED = 8; // LED-ul cu rezistenta de 220 ohmi

void setup() {
  pinMode(PIN_BUTON, INPUT_PULLUP); // activeaza rezistenta interna pull-up
  pinMode(PIN_LED, OUTPUT);
}

void loop() {
  // Cu INPUT_PULLUP: neapasat = HIGH, apasat = LOW (logica inversata)
  int stare = digitalRead(PIN_BUTON);

  if (stare == LOW) { // butonul este apasat
    digitalWrite(PIN_LED, HIGH);
  } else { // butonul este liber
    digitalWrite(PIN_LED, LOW);
  }
}
```

I Explicația codului

- `pinMode(PIN_BUTON, INPUT_PULLUP);` : pinul 2 devine **intrare** cu rezistență pull-up internă.
- `int stare = digitalRead(PIN_BUTON);` : citim starea butonului (HIGH sau LOW).
- `if (stare == LOW) { ... } else { ... }` : dacă e apăsat (LOW) aprindem LED-ul, altfel îl stingem. `==` înseamnă „este egal cu” (comparație), diferit de `=` (atribuire).

i REȚINE

De ce nu folosim o rezistență externă la buton? `INPUT_PULLUP` activează una **în interiorul** microcontrolerului. Mai puține piese, montaj mai simplu, aceeași funcție.

I Sfaturi și depanare

✓ SFAT

Vrei ca butonul să **comute** LED-ul (apeși o dată → aprins, apeși din nou → stins)? Vei avea nevoie să reții starea anterioară a butonului și să detectezi **momentul apăsării**. Aceasta e „provocarea” de mai jos.

! ATENȚIE / DEPANARE

LED-ul e mereu aprins sau pâlpâie singur? Probabil ai uitat `INPUT_PULLUP` și pinul „plutește”. **Nu reacționează la buton?** Verifică dacă un picior chiar ajunge la `GND` și celălalt la pinul 2 (butoanele au picioarele legate în perechi, încearcă altă diagonală).

I Provocare

★ PROVOCARE

Transformă butonul într-un **întrerupător cu memorie**: o apăsare aprinde LED-ul, următoarea îl stinge. Indiciu: reține starea LED-ului într-o variabilă și schimb-o doar când butonul **trece** din neapăsat în apăsat.

Proiectul 3. Lampă RGB de dispoziție

Dificultate ●●○

Durată 50 min

Componente 4

Leșirile digitale știu doar „pornit” sau „oprit”. Dar cum obținem **intensități** și **culori**? Cu ajutorul PWM și al unui LED RGB.

Ce vei învăța

- Ce este **PWM** (modularea în lățime a impulsului) și funcția `analogWrite()` ;
- Cum se amestecă **roșu, verde și albastru** pentru a obține orice culoare;
- Să îți scrii propria **funcție** (`setCuloare`).

Componente necesare

Cant.	Componentă
1×	Placă Arduino UNO
1×	Breadboard
1×	LED RGB (catod comun)
3×	Rezistență 220 Ω
4×	Fire jumper

Schema de conectare

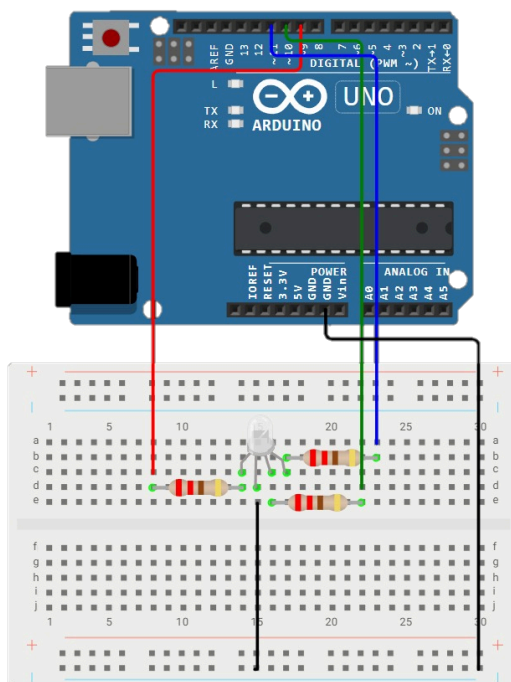


Figura 7: Lampă RGB, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470607277317772289>

I Cum funcționează

Pinii marcați cu ~ (3, 5, 6, 9, 10, 11) pot folosi **PWM**: pornesc și opresc foarte repede, iar dacă stau „pornit” 50% din timp, LED-ul pare aprins la jumătate. `analogWrite()` primește o valoare de la **0** (stins) la **255** (maxim). Un LED RGB are trei mici LED-uri; dozând intensitatea fiecăruia, obținem milioane de culori.

Montaj pas cu pas

1. Identifică cele 4 picioare ale LED-ului RGB. Cel mai lung este **catodul comun (-)**.
2. Leagă catodul comun la **GND**.
3. Pune câte o rezistență de $220\ \Omega$ pe fiecare dintre celelalte trei picioare (R, G, B).
4. Conectează, prin rezistențe, picioarele la pinii **9 (R)**, **10 (G)**, **11 (B)**.
5. Conectează placa la USB.

I Codul

```
/*
  Proiectul 3 - Lampa RGB de dispozitie
  Amesteca rosu, verde si albastru pentru a obtine culori care se schimba lin.
  Folosim analogWrite (PWM) pe pinii ~9, ~10, ~11.
  Curs Arduino Erasmus+ (LED RGB cu catod comun)
*/

const int PIN_R = 9;
const int PIN_G = 10;
const int PIN_B = 11;

// Trimite o culoare data prin cele trei valori (0-255)
void setCuloare(int rosu, int verde, int albastru) {
  analogWrite(PIN_R, rosu);
  analogWrite(PIN_G, verde);
  analogWrite(PIN_B, albastru);
}

void setup() {
  pinMode(PIN_R, OUTPUT);
  pinMode(PIN_G, OUTPUT);
  pinMode(PIN_B, OUTPUT);
}

void loop() {
  setCuloare(255, 0, 0); delay(1000); // rosu
  setCuloare(0, 255, 0); delay(1000); // verde
  setCuloare(0, 0, 255); delay(1000); // albastru
  setCuloare(255, 255, 0); delay(1000); // galben (rosu + verde)
  setCuloare(0, 255, 255); delay(1000); // cyan (verde + albastru)
  setCuloare(255, 0, 255); delay(1000); // magenta
  setCuloare(255, 255, 255); delay(1000); // alb (toate)
}
```

I Explicația codului

- `void setCuloare(int rosu, int verde, int albastru) { ... }`: ne definim o **funcție proprie** care primește trei valori și le trimite pinilor. Astfel, în `loop()` scriem scurt `setCuloare(255, 0, 0)` în loc de trei linii de fiecare dată.

- `analogWrite(PIN_R, rosu);` : reglează intensitatea (0-255) pe pinul PWM.
- În `loop()` parcurgem culori: roșu, verde, albastru, apoi combinații (galben, cyan, magenta, alb).

! REȚINE

Ai un LED RGB cu **anod comun**? Atunci logica se inversează: 0 = maxim, 255 = stins. Leagă piciorul comun la 5V în loc de GND.

I Sfaturi și depanare

✓ SFAT

Pentru o tranziție **lină** între culori (efect „respirație”), folosește o buclă `for` care crește treptat valoarea de la 0 la 255 cu un mic `delay(10)` la fiecare pas.

! ATENȚIE / DEPANARE

O culoare lipsește? Verifică rezistența și firul pinului respectiv. **Culorile par greșite?** Probabil ai inversat două picioare (R cu B): schimbă firele între pini.

I Provocare

Montaj pas cu pas

1. Identifică cele 4 picioare ale LED-ului RGB. Cel mai lung este **catodul comun (-)**.
2. Leagă catodul comun la GND.
3. Pune câte o rezistență de 220 Ω pe fiecare dintre celelalte trei picioare (R, G, B).
4. Conectează, prin rezistențe, picioarele la pinii **9 (R)**, **10 (G)**, **11 (B)**.
5. Conectează placa la USB.

I Codul

```
/*
  Proiectul 3 - Lampa RGB de dispozitie
  Amesteca rosu, verde si albastru pentru a obtine culori care se schimba lin.
  Folosim analogWrite (PWM) pe pinii ~9, ~10, ~11.
  Curs Arduino Erasmus+ (LED RGB cu catod comun)
*/

const int PIN_R = 9;
const int PIN_G = 10;
const int PIN_B = 11;

// Trimite o culoare data prin cele trei valori (0-255)
void setCuloare(int rosu, int verde, int albastru) {
  analogWrite(PIN_R, rosu);
  analogWrite(PIN_G, verde);
  analogWrite(PIN_B, albastru);
}
```

```
void setup() {  
  pinMode(PIN_R, OUTPUT);  
  pinMode(PIN_G, OUTPUT);  
  pinMode(PIN_B, OUTPUT);  
}  
  
void loop() {  
  setCuloare(255, 0, 0); delay(1000); // rosu  
  setCuloare( 0, 255, 0); delay(1000); // verde  
  setCuloare( 0, 0, 255); delay(1000); // albastru  
  setCuloare(255, 255, 0); delay(1000); // galben (rosu + verde)  
  setCuloare( 0, 255, 255); delay(1000); // cyan (verde + albastru)  
  setCuloare(255, 0, 255); delay(1000); // magenta  
  setCuloare(255, 255, 255); delay(1000); // alb (toate)  
}
```

I Explicația codului

- `void setCuloare(int rosu, int verde, int albastru) { ... }`: ne definim o **funcție proprie** care primește trei valori și le trimite pinilor. Astfel, în `loop()` scriem scurt `setCuloare(255, 0, 0)` în loc de trei linii de fiecare dată.
- `analogWrite(PIN_R, rosu);`: reglează intensitatea (0-255) pe pinul PWM.
- În `loop()` parcurgem culori: roșu, verde, albastru, apoi combinații (galben, cyan, magenta, alb).

i REȚINE

Ai un LED RGB cu **anod comun**? Atunci logica se inversează: 0 = maxim, 255 = stins. Leagă piciorul comun la 5V în loc de GND.

I Sfaturi și depanare

✓ SFAT

Pentru o tranziție **lină** între culori (efect „respirație”), folosește o buclă `for` care crește treptat valoarea de la 0 la 255 cu un mic `delay(10)` la fiecare pas.

! ATENȚIE / DEPANARE

O culoare lipsește? Verifică rezistența și firul pinului respectiv. **Culorile par greșite?** Probabil ai inversat două picioare (R cu B): schimbă firele între pini.

I Provocare

★ PROVOCARE

Realizează un **curcubeu automat**: trece lin prin toate culorile spectrului, fără trepte vizibile. Indiciu: crește o culoare în timp ce o scazi pe alta, folosind bucle `for`.

Proiectul 4. Dimmer cu potențiometrul

Dificultate ●●○

Durată 50 min

Componente 4

Butonul avea doar două stări: aprins sau stins. Acum construim un **variator (dimmer)**: rotești un buton și LED-ul se aprinde **lin**, de la stins la maxim, exact ca un întrerupător cu reglaj din casă. Iar pe Monitorul Serial vom vedea în timp real numerele din spatele efectului.

Ce vei învăța

- Ce este o **intrare analogică** și funcția `analogRead()` (valori 0-1023);
- Cum reglezi **luminozitatea** unui LED cu `analogWrite()` (PWM);
- Cum transformi un interval în altul cu `map()`;
- Cum „spionezi” valorile în timp real pe calculator cu `Serial`.

Componente necesare

Cant.	Componentă
1x	Placă Arduino UNO
1x	Breadboard
1x	Potențiometrul
1x	LED
1x	Rezistență 220 Ω

Schema de conectare

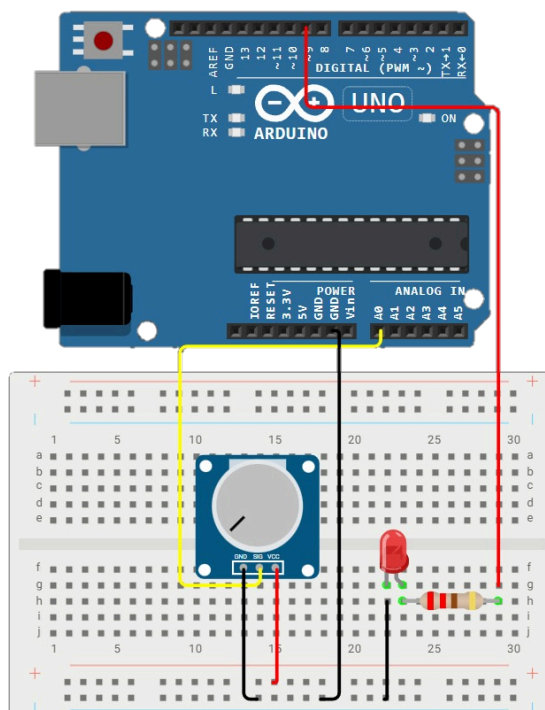


Figura 8: Potențiometrul + LED, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470608615203352577>

I Cum funcționează

Potențiometrul este o rezistență reglabilă cu 3 picioare. Cele laterale merg la 5V și GND, iar cel din mijloc (cursorul) dă o tensiune între 0 și 5 V, în funcție de rotire. `analogRead()` transformă această tensiune într-un număr de la **0 la 1023**.

Dar `analogWrite()` (PWM), cu care reglăm LED-ul, acceptă doar valori de la **0 la 255**. Aici intervine `map()`: „traduce” numărul din intervalul potențiometrului (0-1023) în intervalul LED-ului (0-255). Rezultatul: rotești, iar becul urmează lin poziția butonului. În paralel, trimitem ambele numere pe **Monitorul Serial**, ca să vezi corespondența dintre poziția butonului și luminozitate.

! IMPORTANT

Există multe tipuri de potențiometre; în acest curs vom folosi un trimmer (trim pot). În schemă este desenat unul **rotativ** (cu ax de răsucit), dar componenta din trusă este cea mică, albastră, pătrată. O reglezi fie cu o **șurubelniță**, fie cu **cilindrul de plastic** care vine în starter kit. Electric funcționează la fel, doar montajul diferă.

Trimmer-ul se pune **ca butonul**, călare peste canalul central al breadboard-ului: cele **două picioare** de pe o parte, iar **piciorul singur** (cursorul) pe cealaltă parte. Îl legi așa:

- piciorul **singur** (cursorul) la **A0** — ieșirea pe care o citim;
- din cele două picioare, cel din **stânga la GND**, cel din **dreapta la 5V**.

Dacă încurci GND cu 5V nu strici nimic: se inversează doar sensul (valoarea scade când te aștepti să crească).

Montaj pas cu pas

1. Așază potențiometrul pe breadboard (dacă ai **trimmer**, vezi nota de mai sus).
2. Leagă cele două picioare **laterale** la 5V și GND (sensul lor nu contează).
3. Leagă piciorul din **mijloc (cursorul)** la A0.
4. Montează LED-ul cu rezistență: piciorul lung prin rezistență la **pinul 9**, cel scurt direct la GND.
5. Conectează placa la USB și deschide **Tools** → **Serial Monitor** (viteză 9600).

I Codul

```
/*
  Proiectul 4 - Dimmer cu potentiometru
  Rotești potentiometrul și reglezi lin luminozitatea unui LED, ca un
  întrerupător cu variator. În paralel, trimitem valorile pe Monitorul
  Serial ca să vedem ce se întâmplă "în spate".
  Curs Arduino Erasmus+
*/

const int PIN_POT = A0;    // potentiometrul pe intrare analogică
const int PIN_LED = 9;     // LED pe pin PWM (~9)
```

```
void setup() {  
  pinMode(PIN_LED, OUTPUT);  
  Serial.begin(9600);    // porneste comunicarea cu calculatorul  
}  
  
void loop() {  
  int valoare = analogRead(PIN_POT);    // citește 0...1023  
  
  // Transforma intervalul 0-1023 în 0-255 (cat accepta analogWrite)  
  int luminozitate = map(valoare, 0, 1023, 0, 255);  
  
  analogWrite(PIN_LED, luminozitate);    // regleaza luminozitatea LED-ului  
  
  // Afiseaza valorile pe Monitorul Serial (Tools -> Serial Monitor)  
  Serial.print("Potentiometru: ");  
  Serial.print(valoare);  
  Serial.print("  Luminozitate: ");  
  Serial.println(luminozitate);  
  
  delay(50);    // mica pauza ca sa nu inundam ecranul  
}
```

I Explicația codului

- `int valoare = analogRead(PIN_POT);` : citește poziția potențiometrului (0-1023).
- `map(valoare, 0, 1023, 0, 255)` : „traduce” din intervalul 0-1023 în 0-255.
- `analogWrite(PIN_LED, luminozitate);` : **inima dimmer-ului**. Trimite valoarea (0-255) către LED prin PWM, adică îl aprinde și îl stinge de mii de ori pe secundă; cu cât valoarea e mai mare, cu atât stă „aprins” mai mult timp, deci pare mai luminos.
- `Serial.begin(9600);` și `Serial.print/println(...)` : pornesc comunicarea cu calculatorul și afișează valorile. `print` scrie pe același rând, `println` trece pe rândul următor.

i REȚINE

Monitor Serial vs Serial Plotter: primul afișează numerele ca text, al doilea le desenează ca grafic în timp real (**Tools** → **Serial Plotter**). Încearcă-le pe amândouă!

I Sfaturi și depanare

✓ SFAT

`map()` e util în aproape orice proiect cu senzori: transformă orice interval în altul. Exemplu: `map(valoare, 0, 1023, 0, 180)` ar da un unghi pentru un servomotor.

! ATENȚIE / DEPARARE

Valoarea sare haotic sau rămâne la 0/1023? Verifică cele două picioare laterale: dacă 5V și GND sunt inversate sau lipsesc, citirea nu are sens. **Monitorul afișează simboluri ciudate?** Viteza din Monitor trebuie să fie tot **9600**.

I Provocare

! IMPORTANT

Există multe tipuri de potențiometre; în acest curs vom folosi un trimmer (trim pot). În schemă este desenat unul **rotativ** (cu ax de răsucit), dar componenta din trusă este cea mică, albastră, pătrată. O reglezi fie cu o **șurubelniță**, fie cu **cilindrul de plastic** care vine în starter kit. Electric funcționează la fel, doar montajul diferă.

Trimmer-ul se pune **ca butonul**, călare peste canalul central al breadboard-ului: cele **două picioare** de pe o parte, iar **piciorul singur** (cursorul) pe cealaltă parte. Îl legi așa:

- piciorul **singur** (cursorul) la **A0** — ieșirea pe care o citim;
- din cele două picioare, cel din **stânga la GND**, cel din **dreapta la 5V**.

Dacă încurci **GND** cu **5V** nu strici nimic: se inversează doar sensul (valoarea scade când te aștepți să crească).

Montaj pas cu pas

1. Așază potențiometrul pe breadboard (dacă ai **trimmer**, vezi nota de mai sus).
2. Leagă cele două picioare **laterale** la **5V** și **GND** (sensul lor nu contează).
3. Leagă piciorul din **mijloc (cursorul)** la **A0**.
4. Montează LED-ul cu rezistență: piciorul lung prin rezistență la **pinul 9**, cel scurt direct la **GND**.
5. Conectează placa la USB și deschide **Tools** → **Serial Monitor** (viteză 9600).

I Codul

```
/*
  Proiectul 4 - Dimmer cu potentiometru
  Rotesti potentiometrul si reglezi lin luminozitatea unui LED, ca un
  intrerupator cu variator. In paralel, trimitem valorile pe Monitorul
  Serial ca sa vedem ce se intampla "in spate".
  Curs Arduino Erasmus+
*/

const int PIN_POT = A0;    // potentiometrul pe intrare analogica
const int PIN_LED = 9;     // LED pe pin PWM (~9)

void setup() {
  pinMode(PIN_LED, OUTPUT);
  Serial.begin(9600);      // porneste comunicarea cu calculatorul
}

void loop() {
  int valoare = analogRead(PIN_POT);          // citeste 0...1023

  // Transforma intervalul 0-1023 in 0-255 (cat accepta analogWrite)
  int luminozitate = map(valoare, 0, 1023, 0, 255);

  analogWrite(PIN_LED, luminozitate);         // regleaza luminozitatea LED-ului

  // Afiseaza valorile pe Monitorul Serial (Tools -> Serial Monitor)
  Serial.print("Potentiometru: ");
  Serial.print(valoare);
  Serial.print("    Luminozitate: ");
```

```
Serial.println(luminozitate);  
  
delay(50); // mica pauza ca sa nu inundam ecranul  
}
```

I Explicația codului

- `int valoare = analogRead(PIN_POT);` : citește poziția potențiometrului (0-1023).
- `map(valoare, 0, 1023, 0, 255)` : „traduce” din intervalul 0-1023 în 0-255.
- `analogWrite(PIN_LED, luminozitate);` : **inima dimmer-ului**. Trimite valoarea (0-255) către LED prin PWM, adică îl aprinde și îl stinge de mii de ori pe secundă; cu cât valoarea e mai mare, cu atât stă „aprins” mai mult timp, deci pare mai luminos.
- `Serial.begin(9600);` și `Serial.print/println(...)` : pornesc comunicarea cu calculatorul și afișează valorile. `print` scrie pe același rând, `println` trece pe rândul următor.

i REȚINE

Monitor Serial vs Serial Plotter: primul afișează numerele ca text, al doilea le desenează ca grafic în timp real (**Tools** → **Serial Plotter**). Încearcă-le pe amândouă!

I Sfaturi și depanare

✓ SFAT

`map()` e util în aproape orice proiect cu senzori: transformă orice interval în altul. Exemplu: `map(valoare, 0, 1023, 0, 180)` ar da un unghi pentru un servomotor.

! ATENȚIE / DEPANARE

Valoarea sare haotic sau rămâne la 0/1023? Verifică cele două picioare laterale: dacă 5V și GND sunt inversate sau lipsesc, citirea nu are sens. **Monitorul afișează simboluri ciudate?** Viteza din Monitor trebuie să fie tot **9600**.

I Provocare

★ PROVOCARE

Folosește potențiometrul ca **reglaj de viteză**: cu cât îl rotești mai mult, cu atât un LED (separat) clipește mai repede. Indiciu: folosește valoarea citită în `delay()`.

Proiectul 5. Mini-pian cu buzzer

Dificultate ●●○

Durată 50 min

Componente 4

Facem placa să **cânte**! Trei butoane devin clapele unui pian în miniatură.

Ce vei învăța

- Cum se produce sunetul cu `tone()` și `noTone()`;
- Legătura dintre **frecvență** și **înălțimea notei**;
- Cum combini mai multe intrări cu decizii `if / else if`.

Componente necesare

Cant.	Componentă
1x	Placă Arduino UNO
1x	Breadboard
1x	Buzzer piezoelectric
3x	Buton (push-button)
5x	Fire jumper

Schema de conectare

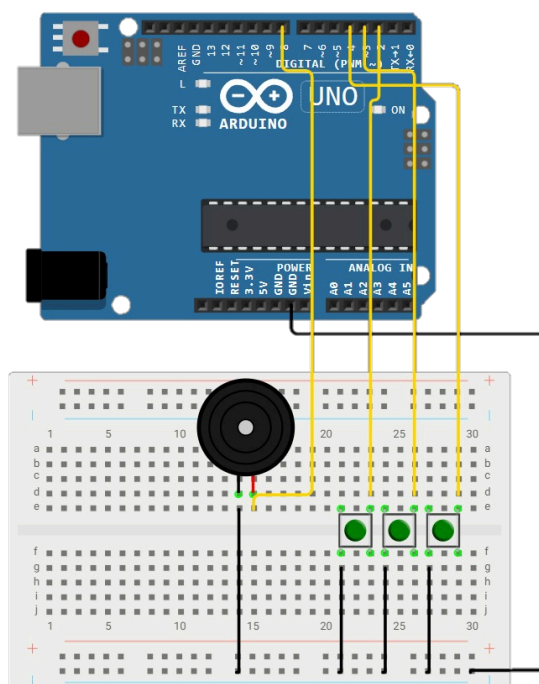


Figura 9: Mini-pian, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470609691701999617>

I Cum funcționează

Un buzzer piezoelectric vibrează cu frecvența pe care i-o dăm. `tone(pin, frecventa)` produce un sunet continuu la acea frecvență (în Hertzi), iar `noTone(pin)` îl oprește. Fiecare notă muzicală are o frecvență: DO $\approx$ 262 Hz, RE $\approx$ 294 Hz, MI $\approx$ 330 Hz. Fiecare buton, când e apăsat, declanșează nota lui.

Montaj pas cu pas

1. Conectează buzzer-ul: piciorul + la **pinul 8**, celălalt la GND.
2. Așază cele 3 butoane peste canalul central.
3. Leagă câte un picior al fiecărui buton la GND.
4. Leagă celălalt picior al butoanelor la pinii **2 (DO)**, **3 (RE)**, **4 (MI)**.
5. Conectează placa la USB.

I Codul

```
/*
  Proiectul 5 - Mini-pian cu buzzer
  Trei butoane canta trei note diferite pe un buzzer piezoelectric.
  Curs Arduino Erasmus+
*/

const int PIN_BUZZER = 8;
const int PIN_DO = 2;    // buton nota DO
const int PIN_RE = 3;    // buton nota RE
const int PIN_MI = 4;    // buton nota MI

// Frecventele notelor, in Hertz
const int NOTA_DO = 262;
const int NOTA_RE = 294;
const int NOTA_MI = 330;

void setup() {
  pinMode(PIN_DO, INPUT_PULLUP);
  pinMode(PIN_RE, INPUT_PULLUP);
  pinMode(PIN_MI, INPUT_PULLUP);
  pinMode(PIN_BUZZER, OUTPUT);
}

void loop() {
  // Cu INPUT_PULLUP butonul apasat da LOW
  if (digitalRead(PIN_DO) == LOW) {
    tone(PIN_BUZZER, NOTA_DO);
  } else if (digitalRead(PIN_RE) == LOW) {
    tone(PIN_BUZZER, NOTA_RE);
  } else if (digitalRead(PIN_MI) == LOW) {
    tone(PIN_BUZZER, NOTA_MI);
  } else {
    noTone(PIN_BUZZER); // niciun buton apasat -> liniste
  }
}
```

I Explicația codului

- Toate butoanele folosesc `INPUT_PULLUP`, deci **apasat = LOW**.
- Lanțul `if / else if / else` verifică butoanele pe rând: primul apăsător își cântă nota.

- `tone(PIN_BUZZER, NOTA_DO)` pornește sunetul; `noTone(PIN_BUZZER)` îl oprește când niciun buton nu e apăsat.

i REȚINE

Modifică valorile frecvenței pentru a obține alte note. Cu cât numărul e mai mare, cu atât sunetul e mai **înalt** (mai ascuțit).

I Sfaturi și depanare

✓ SFAT

Vrei să cânti o **melodie automată**? Pune într-un tablou (array) frecvențele notelor și parcurge-le cu o buclă `for`, folosind `tone(pin, frecventa, durata)` pentru fiecare notă.

! ATENȚIE / DEPANARE

Nu se aude nimic? Verifică polaritatea buzzer-ului (are un `+` marcat) și că e pe pinul 8. **Sună o notă greșită?** Un buton e legat la alt pin decât crezi: verifică ordinea 2-3-4.

I Provocare

Montaj pas cu pas

1. Conectează buzzer-ul: piciorul `+` la **pinul 8**, celălalt la `GND`.
2. Așază cele 3 butoane peste canalul central.
3. Leagă câte un picior al fiecărui buton la `GND`.
4. Leagă celălalt picior al butoanelor la pinii **2 (DO)**, **3 (RE)**, **4 (MI)**.
5. Conectează placa la USB.

I Codul

```
/*
  Proiectul 5 - Mini-pian cu buzzer
  Trei butoane canta trei note diferite pe un buzzer piezoelectric.
  Curs Arduino Erasmus+
*/

const int PIN_BUZZER = 8;
const int PIN_DO = 2;    // buton nota DO
const int PIN_RE = 3;    // buton nota RE
const int PIN_MI = 4;    // buton nota MI

// Frecventele notelor, in Hertz
const int NOTA_DO = 262;
const int NOTA_RE = 294;
const int NOTA_MI = 330;

void setup() {
  pinMode(PIN_DO, INPUT_PULLUP);
```

```
pinMode(PIN_RE, INPUT_PULLUP);
pinMode(PIN_MI, INPUT_PULLUP);
pinMode(PIN_BUZZER, OUTPUT);
}

void loop() {
  // Cu INPUT_PULLUP butonul apasat da LOW
  if (digitalRead(PIN_DO) == LOW) {
    tone(PIN_BUZZER, NOTA_DO);
  } else if (digitalRead(PIN_RE) == LOW) {
    tone(PIN_BUZZER, NOTA_RE);
  } else if (digitalRead(PIN_MI) == LOW) {
    tone(PIN_BUZZER, NOTA_MI);
  } else {
    noTone(PIN_BUZZER); // niciun buton apasat -> liniste
  }
}
```

Explicația codului

- Toate butoanele folosesc INPUT_PULLUP, deci **apasat = LOW**.
- Lanțul if / else if / else verifică butoanele pe rând: primul apăsător își cântă nota.
- tone(PIN_BUZZER, NOTA_DO) pornește sunetul; noTone(PIN_BUZZER) îl oprește când niciun buton nu e apăsător.

REȚINE

Modifică valorile frecvenței pentru a obține alte note. Cu cât numărul e mai mare, cu atât sunetul e mai înalt (mai ascuțit).

Sfaturi și depanare

✓ SFAT

Vrei să cânti o **melodie automată**? Pune într-un tablou (array) frecvențele notelor și parcurge-le cu o buclă for, folosind tone(pin, frecventa, durata) pentru fiecare notă.

! ATENȚIE / DEPANARE

Nu se aude nimic? Verifică polaritatea buzzer-ului (are un + marcat) și că e pe pinul 8. **Sună o notă greșită?** Un buton e legat la alt pin decât crezi: verifică ordinea 2-3-4.

Provocare

★ PROVOCARE

Programează placa să cânte singură „**Odă bucuriei**” sau o melodie simplă la pornire, folosind un tablou de note și durate. Apoi lasă butoanele pentru „cântat live”.

Proiectul 6. Lumină automată de noapte

Dificultate ●●○

Durată 50 min

Componente 3

Primul nostru **senzor de mediu**: placa „vede” cât de întuneric este și aprinde singură lumina, exact ca o lampă cu senzor de pe stradă. Folosim un **modul LDR** cu patru pini (VCC, GND, AO, DO) care are deja rezistența inclusă pe placă. Noi folosim doar trei dintre ei (VCC, GND și AO); pinul DO (ieșire digitală) nu este conectat.

Ce vei învăța

- Cum funcționează un **modul LDR** (fotorezistor cu ieșire analogică);
- Cum iei o decizie pe baza unui **prag** (threshold);
- Cum reglezi pragul observând valorile pe Monitorul Serial.

Componente necesare

Cant.	Componentă
1×	Placă Arduino UNO
1×	Breadboard
1×	Modul LDR (4 pini: VCC, GND, AO, DO)
1×	LED + rezistență 220 Ω

Schema de conectare

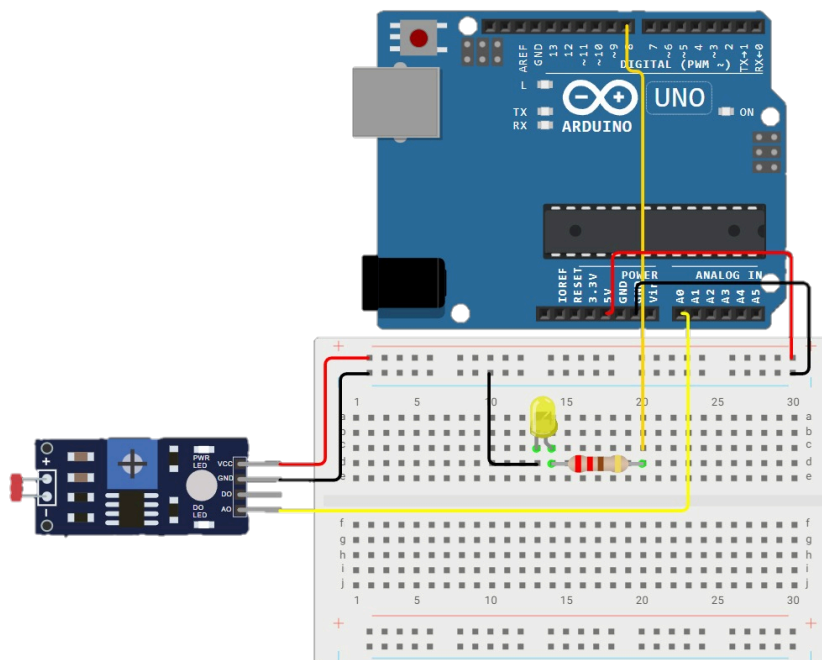


Figura 10: Lumină de noapte, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- Construiește și rulează circuitul în browser, fără componente fizice:
<https://wokwi.com/projects/470640156861217793>

I Cum funcționează

Modulul LDR conține un fotorezistor și o rezistență de referință deja montate pe o placuță mică, formând intern un **divizor de tensiune**. Pinul **AO** (ieșire analogică) ne dă direct o tensiune care variază cu lumina, pe care o citim la **A0**. Nu avem nevoie de rezistență externă. Dacă valoarea scade sub un **prag**, considerăm că e întuneric și aprindem LED-ul.

Montaj pas cu pas

1. Conectează pinul **VCC** al modulului LDR la **5V**.
2. Conectează pinul **GND** al modulului LDR la **GND**.
3. Conectează pinul **AO** al modulului LDR la **A0**.
4. Montează LED-ul cu rezistență: piciorul lung prin rezistență la **pinul 8**, cel scurt direct la **GND**.
5. Conectează placa la USB și deschide **Serial Monitor**.

I Codul

```
/*
  Proiectul 6 - Lumina automata de noapte
  Un modul LDR masoara lumina din camera. Cand se intuneca,
  aprindem automat un LED.
  Curs Arduino Erasmus+
*/

const int PIN_LDR = A0;    // pinul AO al modulului LDR
const int PIN_LED = 8;

const int PRAG = 400;      // sub aceasta valoare consideram ca e intuneric

void setup() {
  Serial.begin(9600);
  pinMode(PIN_LED, OUTPUT);
}

void loop() {
  int lumina = analogRead(PIN_LDR);    // 0 = intuneric, ~1023 = lumina puternica
  Serial.println(lumina);

  if (lumina < PRAG) {
    digitalWrite(PIN_LED, HIGH);    // e intuneric -> aprinde lumina
  } else {
    digitalWrite(PIN_LED, LOW);     // e lumina -> stinge
  }

  delay(200);
}
```

I Explicația codului

- `int lumina = analogRead(PIN_LDR);` : citește tensiunea de pe pinul AO al modulului (0 = întuneric, 1023 = lumină puternică).
- `const int PRAG = 400;` : valoarea de decizie. O ajustezi după camera ta.
- `if (lumina < PRAG) { aprinde } else { stinge }` : logica lămpii automate.
- `Serial.println(lumina);` : ne arată valorile ca să alegem un prag potrivit.

I Sfaturi și depanare

✓ SFAT

Cum alegi pragul? Deschide Monitorul Serial și notează valoarea când e lumină și când acoperi LDR-ul cu degetul. Alege un `PRAG` la mijloc între cele două.

! ATENȚIE / DEPANARE

LED-ul stă mereu aprins sau mereu stins? Pragul nu se potrivește cu lumina din cameră: citește valorile reale și ajustează-l. **Valoarea nu se schimbă deloc?** Verifică conexiunile modulului: VCC la `5V`, GND la GND, AO la `A0`.

I Provocare

Montaj pas cu pas

1. Conectează pinul **VCC** al modulului LDR la `5V`.
2. Conectează pinul **GND** al modulului LDR la `GND`.
3. Conectează pinul **AO** al modulului LDR la `A0`.
4. Montează LED-ul cu rezistență: piciorul lung prin rezistență la **pinul 8**, cel scurt direct la `GND`.
5. Conectează placa la USB și deschide **Serial Monitor**.

I Codul

```
/*
  Proiectul 6 - Lumina automata de noapte
  Un modul LDR masoara lumina din camera. Cand se intuneca,
  aprindem automat un LED.
  Curs Arduino Erasmus+
*/

const int PIN_LDR = A0;    // pinul AO al modulului LDR
const int PIN_LED = 8;

const int PRAG = 400;      // sub aceasta valoare consideram ca e intuneric

void setup() {
  Serial.begin(9600);
  pinMode(PIN_LED, OUTPUT);
}

void loop() {
```

```
int lumina = analogRead(PIN_LDR); // 0 = întuneric, ~1023 = lumina puternică
Serial.println(lumina);

if (lumina < PRAG) {
    digitalWrite(PIN_LED, HIGH); // e întuneric -> aprinde lumina
} else {
    digitalWrite(PIN_LED, LOW); // e lumina -> stinge
}

delay(200);
}
```

Explicația codului

- `int lumina = analogRead(PIN_LDR);` : citește tensiunea de pe pinul AO al modulului (0 = întuneric, 1023 = lumină puternică).
- `const int PRAG = 400;` : valoarea de decizie. O ajustezi după camera ta.
- `if (lumina < PRAG) { aprinde } else { stinge }` : logica lămpii automate.
- `Serial.println(lumina);` : ne arată valorile ca să alegem un prag potrivit.

Sfaturi și depanare

✓ SFAT

Cum alegi pragul? Deschide Monitorul Serial și notează valoarea când e lumină și când acoperi LDR-ul cu degetul. Alege un `PRAG` la mijloc între cele două.

! ATENȚIE / DEPANARE

LED-ul stă mereu aprins sau mereu stins? Pragul nu se potrivește cu lumina din cameră: citește valorile reale și ajustează-l. **Valoarea nu se schimbă deloc?** Verifică conexiunile modulului: VCC la 5V, GND la GND, AO la AO.

Provocare

★ PROVOCARE

În loc să aprinzi/stingi brusc, fă LED-ul să lumineze **proporțional** cu întunericul: cu cât e mai întuneric, cu atât mai puternic. Indiciu: `analogWrite()` + `map()` pe un pin PWM.

Proiectul 7. Stație meteo cu ecran LCD

Dificultate ●●●

Durată 60 min

Componente 5

Facem un instrument adevărat: măsurăm temperatura și umiditatea și le afișăm pe un ecran, fără să mai avem nevoie de calculator.

Ce vei învăța

- Cum instalezi și folosești **biblioteci** externe;
- Cum citești un senzor digital (**DHT11**);
- Cum afișezi text pe un **ecran LCD 16×2** conectat direct (fără modul I2C).

Componente necesare

Cant.	Componentă
1×	Placă Arduino UNO
1×	Senzor DHT11 (temperatură + umiditate)
1×	Ecran LCD 16×2 (conectare directă, fără modul I2C)
1×	Potențiomtru 10 kΩ (pentru reglaj contrast LCD)
1×	Breadboard
16×	Fire jumper

Schema de conectare

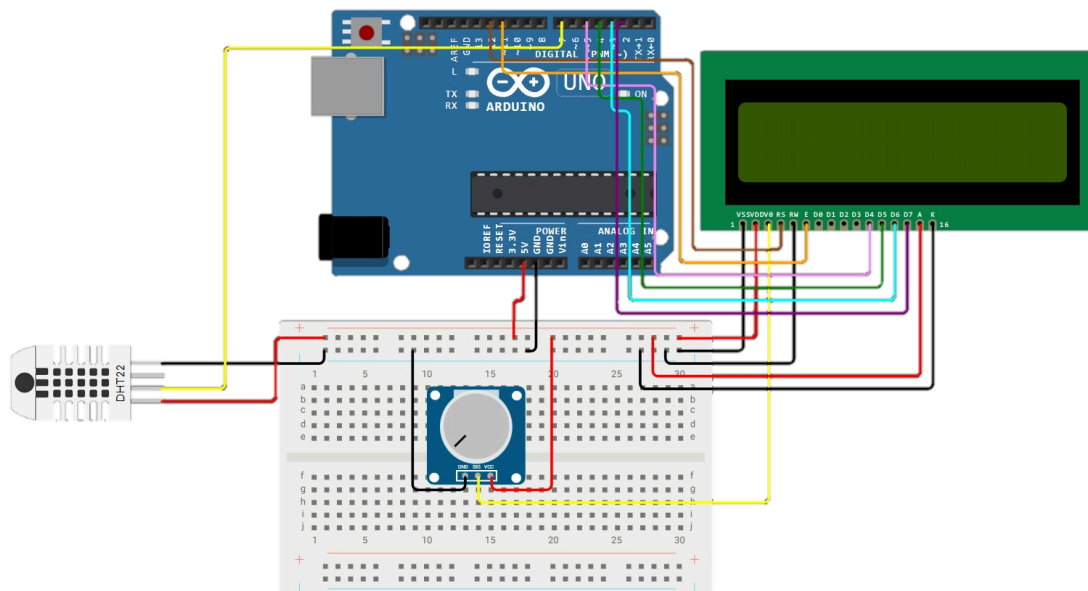


Figura 11: Stație meteo, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- ▶ Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470642307274707969>

I Cum funcționează

Senzorul **DHT11** măsoară singur temperatura și umiditatea și le trimite plăcii pe un singur fir de date. Ecranul **LCD 16×2** este conectat direct la placă, folosind 6 pini de date/control (RS, EN, D4–D7) plus alimentare și un potențiomtru pentru contrast. Ca să nu scriem noi tot codul de comunicare, folosim **biblioteci**: colecții de funcții gata făcute (`dht.readTemperature()` , `lcd.print()`). Biblioteca `LiquidCrystal` vine deja instalată cu Arduino IDE.

i REȚINE

Instalează întâi biblioteka! Caută și instalează „DHT sensor library” (Adafruit). Fără ea, codul nu compilează. Biblioteca `LiquidCrystal` (pentru ecranul LCD) vine inclusă în Arduino IDE și **nu** trebuie instalată separat. Pașii detaliați și un video cu instalarea sunt în **Capitolul 0, secțiunea „Bibliotecile”**.

! IMPORTANT

În schema de mai sus este desenat senzorul DHT22, deoarece Wokwi nu conține o componentă DHT11. În realitate vom folosi tot **senzorul DHT11** din trusa noastră — conexiunile sunt identice (VCC, GND, DATA).

Montaj pas cu pas

1. **DHT11:** VCC → 5V , GND → GND , DATA → pinul 7 .
2. **LCD** VSS (pin 1) → GND .
3. **LCD** VDD (pin 2) → 5V .
4. **LCD** V0 (pin 3) → pinul din mijloc al potențiometrului (contrast).
5. **LCD** RS (pin 4) → **pinul 12**.
6. **LCD** RW (pin 5) → GND .
7. **LCD** EN (pin 6) → **pinul 11**.
8. **LCD** D4 (pin 11) → **pinul 5**.
9. **LCD** D5 (pin 12) → **pinul 4**.
10. **LCD** D6 (pin 13) → **pinul 3**.
11. **LCD** D7 (pin 14) → **pinul 2**.
12. **LCD** A (pin 15) → 5V (backlight +).
13. **LCD** K (pin 16) → GND (backlight –).
14. Potențiometrul de contrast: picioarele laterale la 5V și GND , cursorul (mijloc) la V0 .
15. Verifică de două ori alimentarea (inversarea poate distruge modulele).
16. Conectează placa la USB.

I Codul

```
/*
  Proiectul 7 - Statie meteo (temperatura si umiditate)
  Citim un senzor DHT11 si afisam valorile pe un ecran LCD 16x2
  (conectat direct, fara modul I2C).

  BIBLIOTECI necesare (Sketch -> Include Library -> Manage Libraries):
  - "DHT sensor library" (Adafruit)
  Biblioteca "LiquidCrystal" vine inclusa cu Arduino IDE.
  Curs Arduino Erasmus+
*/

#include <DHT.h>
#include <LiquidCrystal.h>

#define PIN_DHT 7
#define TIP_DHT DHT11
// #define TIP_DHT DHT22 // decommentati daca folositi senzorul DHT22

// LCD conectat direct: RS=12, EN=11, D4=5, D5=4, D6=3, D7=2
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

DHT dht(PIN_DHT, TIP_DHT);

void setup() {
  dht.begin();
  lcd.begin(16, 2); // 16 coloane, 2 randuri
  lcd.print("Statie meteo");
  delay(2000);
  lcd.clear();
}

void loop() {
  float temperatura = dht.readTemperature(); // grade Celsius
  float umiditate = dht.readHumidity(); // procente

  // Daca citirea a esuat, incearca din nou
  if (isnan(temperatura) || isnan(umiditate)) {
    lcd.setCursor(0, 0);
    lcd.print("Eroare senzor ");
    delay(2000);
    return;
  }

  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperatura, 1);
  lcd.print(" C ");

  lcd.setCursor(0, 1);
  lcd.print("Umid: ");
  lcd.print(umiditate, 0);
  lcd.print(" % ");

  delay(2000); // DHT11 ofera o citire noua la ~2 secunde
}
```

I Explicația codului

- `#include <DHT.h>` și `#include <LiquidCrystal.h>` : aducem bibliotecile în sketch.
- `LiquidCrystal lcd(12, 11, 5, 4, 3, 2);` : creăm obiectul LCD indicând pinii RS, EN, D4, D5, D6, D7.
- `DHT dht(PIN_DHT, TIP_DHT);` : creăm obiectul senzorului.
- `lcd.begin(16, 2);` : pornește ecranul cu 16 coloane și 2 rânduri.
- `dht.readTemperature()` / `dht.readHumidity()` : citesc valorile (numere cu zecimale).
- `if (isnan(...))` : verifică dacă citirea a eșuat (`nan` = „not a number”).
- `lcd.setCursor(coloană, rând)` mută cursorul, iar `lcd.print(...)` scrie pe ecran.

I Sfaturi și depanare

✓ SFAT

Nu vezi nimic pe ecran? Rotește potențiometrul de contrast încet până când apar caracterele. Dacă ecranul afișează doar dreptunghiuri pline, mai rotește puțin.

! ATENȚIE / DEPANARE

Ecran complet negru sau complet gol? Reglează potențiometrul de contrast (pinul V0). **Valori „Eroare senzor”?** Verifică firul de date al DHT11 pe pinul 7 și așteaptă 2 secunde între citiri.

I Provocare

! IMPORTANT

În schema de mai sus este desenat senzorul DHT22, deoarece Wokwi nu conține o componentă DHT11. În realitate vom folosi tot **senzorul DHT11** din trusa noastră — conexiunile sunt identice (VCC, GND, DATA).

Montaj pas cu pas

1. **DHT11:** VCC → 5V , GND → GND , DATA → pinul 7 .
2. **LCD** VSS (pin 1) → GND .
3. **LCD** VDD (pin 2) → 5V .
4. **LCD** v0 (pin 3) → pinul din mijloc al potențiometrului (contrast).
5. **LCD** RS (pin 4) → **pinul 12**.
6. **LCD** RW (pin 5) → GND .
7. **LCD** EN (pin 6) → **pinul 11**.
8. **LCD** D4 (pin 11) → **pinul 5**.
9. **LCD** D5 (pin 12) → **pinul 4**.
10. **LCD** D6 (pin 13) → **pinul 3**.
11. **LCD** D7 (pin 14) → **pinul 2**.
12. **LCD** A (pin 15) → 5V (backlight +).
13. **LCD** K (pin 16) → GND (backlight -).
14. Potențiometrul de contrast: picioarele laterale la 5V și GND , cursorul (mijloc) la v0 .
15. Verifică de două ori alimentarea (inversarea poate distruge modulele).

16. Conectează placa la USB.

I Codul

```
/*
Proiectul 7 - Statie meteo (temperatura si umiditate)
Citim un senzor DHT11 si afisam valorile pe un ecran LCD 16x2
(conectat direct, fara modul I2C).

BIBLIOTECI necesare (Sketch -> Include Library -> Manage Libraries):
- "DHT sensor library" (Adafruit)
Biblioteca "LiquidCrystal" vine inclusa cu Arduino IDE.
Curs Arduino Erasmus+
*/

#include <DHT.h>
#include <LiquidCrystal.h>

#define PIN_DHT 7
#define TIP_DHT DHT11
// #define TIP_DHT DHT22 // decommentati daca folositi senzorul DHT22

// LCD conectat direct: RS=12, EN=11, D4=5, D5=4, D6=3, D7=2
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

DHT dht(PIN_DHT, TIP_DHT);

void setup() {
  dht.begin();
  lcd.begin(16, 2); // 16 coloane, 2 randuri
  lcd.print("Statie meteo");
  delay(2000);
  lcd.clear();
}

void loop() {
  float temperatura = dht.readTemperature(); // grade Celsius
  float umiditate = dht.readHumidity(); // procente

  // Daca citirea a esuat, incearca din nou
  if (isnan(temperatura) || isnan(umiditate)) {
    lcd.setCursor(0, 0);
    lcd.print("Eroare senzor ");
    delay(2000);
    return;
  }

  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperatura, 1);
  lcd.print(" C ");

  lcd.setCursor(0, 1);
  lcd.print("Umid: ");
  lcd.print(umiditate, 0);
  lcd.print(" % ");
}
```

```
delay(2000); // DHT11 ofera o citire noua la ~2 secunde  
}
```

I Explicația codului

- `#include <DHT.h>` și `#include <LiquidCrystal.h>` : aducem bibliotecile în sketch.
- `LiquidCrystal lcd(12, 11, 5, 4, 3, 2);` : creăm obiectul LCD indicând pinii RS, EN, D4, D5, D6, D7.
- `DHT dht(PIN_DHT, TIP_DHT);` : creăm obiectul senzorului.
- `lcd.begin(16, 2);` : pornește ecranul cu 16 coloane și 2 rânduri.
- `dht.readTemperature()` / `dht.readHumidity()` : citesc valorile (numere cu zecimale).
- `if (isnan(...))` : verifică dacă citirea a eșuat (`nan` = „not a number”).
- `lcd.setCursor(coloană, rând)` mută cursorul, iar `lcd.print(...)` scrie pe ecran.

I Sfaturi și depanare

✓ SFAT

Nu vezi nimic pe ecran? Rotește potențiometrul de contrast încet până când apar caracterele. Dacă ecranul afișează doar dreptunghiuri pline, mai rotește puțin.

! ATENȚIE / DEPANARE

Ecran complet negru sau complet gol? Reglează potențiometrul de contrast (pinul V0). **Valori „Eroare senzor”?** Verifică firul de date al DHT11 pe pinul 7 și așteaptă 2 secunde între citiri.

I Provocare

★ PROVOCARE

Adaugă un LED „de confort”: verde când temperatura e între 20-25 °C, roșu în rest. Sau afișează pe rând, la câteva secunde, temperatura în °C și în °F ($F = C \times 9 / 5 + 32$).

Proiectul 8. Senzor de parcare cu ultrasunete

Dificultate ●●●

Durată 60 min

Componente 4

Construim un **senzor de parcare** ca la mașini: cu cât te apropii de obstacol, cu atât sună mai des.

Ce vei învăța

- Cum măsoară distanța un senzor **HC-SR04** (ecou de ultrasunete);
- Funcțiile `pulseIn()` și `delayMicroseconds()` ;
- Cum transformi o măsurătoare în feedback util (bipuri mai dese).

Componente necesare

Cant.	Componentă
1x	Placă Arduino UNO
1x	Senzor ultrasonic HC-SR04
1x	Buzzer piezoelectric
1x	Breadboard
6x	Fire jumper

Schema de conectare

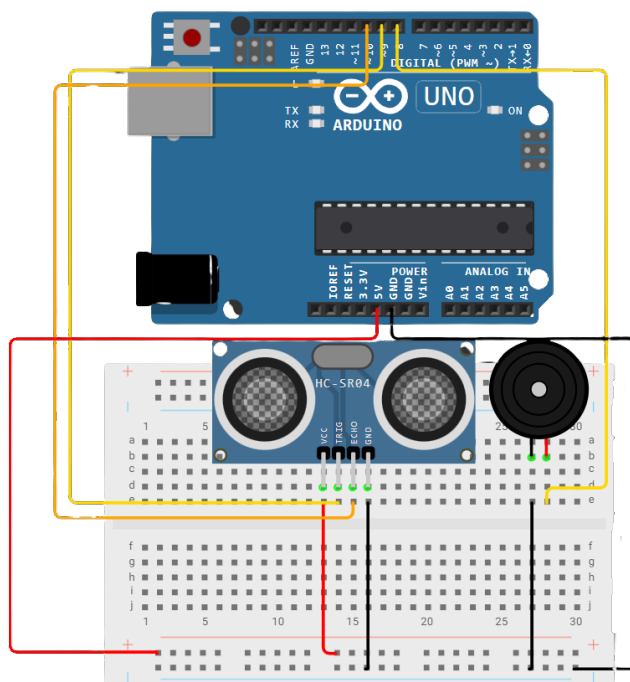


Figura 12: Senzor de parcare, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- ▶ Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470682026879932417>

I Cum funcționează

HC-SR04 are doi „ochi”: unul trimite un impuls de ultrasunete (pinul **TRIG**), celălalt ascultă ecoul întors (pinul **ECHO**). Măsurăm **cât timp** durează dus-întors și, știind viteza sunetului ($\approx 0,034 \text{ cm}/\mu\text{s}$), calculăm distanța. Împărțim la 2 pentru că sunetul parcurge distanța **dus și întors**. Apoi, cu `map()`, transformăm distanța în pauza dintre bipuri.

Montaj pas cu pas

1. **HC-SR04**: VCC $\rightarrow$ 5V, GND $\rightarrow$ GND, TRIG $\rightarrow$ pinul 9, ECHO $\rightarrow$ pinul 10.
2. **Buzzer**: piciorul + la **pinul 8**, celălalt la GND.
3. Conectează placa la USB și deschide **Serial Monitor** ca să vezi distanța.

I Codul

```
/*
  Proiectul 8 - Senzor de parcare cu ultrasunete
  Senzorul HC-SR04 masoara distanta pana la un obstacol. Cu cat obstacolul
  este mai aproape, cu atat buzzer-ul suna mai des (ca la senzorii auto).
  Curs Arduino Erasmus+
*/

const int PIN_TRIG   = 9;
const int PIN_ECHO   = 10;
const int PIN_BUZZER = 8;

void setup() {
  Serial.begin(9600);
  pinMode(PIN_TRIG,  OUTPUT);
  pinMode(PIN_ECHO,  INPUT);
  pinMode(PIN_BUZZER, OUTPUT);
}

// Returneaza distanta in centimetri
long masoaraDistanța() {
  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_TRIG, HIGH);      // impuls scurt de trimitere
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  long durata = pulseIn(PIN_ECHO, HIGH); // cat timp revine ecoul
  long distanta = durata * 0.034 / 2;    // viteza sunetului -> cm
  return distanta;
}

void loop() {
  long distanta = masoaraDistanța();
  Serial.print(distanta);
  Serial.println(" cm");

  if (distanta > 0 && distanta < 30) {
    // Cu cat mai aproape, cu atat pauza mai mica intre bipuri
    int pauza = map(distanta, 0, 30, 40, 500);
    tone(PIN_BUZZER, 1000, 40);
    delay(pauza);
  } else {
```

```
noTone(PIN_BUZZER);  
delay(100);  
}  
}
```

I Explicația codului

- `masoaraDistanța()` : funcția noastră, care trimite impulsul pe `TRIG` și măsoară ecoul.
- `digitalWrite(TRIG, HIGH); delayMicroseconds(10);` : impuls scurt de exact 10 microsecunde.
- `pulseIn(PIN_ECHO, HIGH)` : cronometrează cât timp pinul `ECHO` rămâne `HIGH` (durata ecoului).
- `distanța = durata * 0.034 / 2;` : transformă timpul în centimetri.
- Cu `map(distanța, 0, 30, 40, 500)` calculăm pauza: aproape → pauză mică (bipuri dese).

I Sfaturi și depanare

✓ SFAT

`tone(pin, frecvența, durata)` cu al treilea parametru sună exact atâtea milisecunde și se oprește singur, perfect pentru un „bip” scurt, fără `noTone()`.

! ATENȚIE / DEPANARE

Distanța e mereu 0 sau o valoare uriașă? Verifică să nu fi inversat `TRIG` și `ECHO`. **Citiri instabile?** Sensorul are nevoie de o suprafață care reflectă bine; obiectele moi sau înclinate „împrăștie” ecoul.

I Provocare

Montaj pas cu pas

1. **HC-SR04:** `VCC` → 5V, `GND` → `GND`, `TRIG` → pinul 9, `ECHO` → pinul 10.
2. **Buzzer:** piciorul + la **pinul 8**, celălalt la `GND`.
3. Conectează placa la USB și deschide **Serial Monitor** ca să vezi distanța.

I Codul

```
/*  
Proiectul 8 - Senzor de parcare cu ultrasunete  
Senzorul HC-SR04 masoara distanta pana la un obstacol. Cu cat obstacolul  
este mai aproape, cu atat buzzer-ul suna mai des (ca la senzorii auto).  
Curs Arduino Erasmus+  
*/  
  
const int PIN_TRIG    = 9;  
const int PIN_ECHO     = 10;  
const int PIN_BUZZER  = 8;  
  
void setup() {  
  Serial.begin(9600);  
  pinMode(PIN_TRIG,  OUTPUT);  
}
```

```
pinMode(PIN_ECHO, INPUT);
pinMode(PIN_BUZZER, OUTPUT);
}

// Returneaza distanta in centimetri
long masoaraDistanta() {
    digitalWrite(PIN_TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(PIN_TRIG, HIGH);    // impuls scurt de trimitere
    delayMicroseconds(10);
    digitalWrite(PIN_TRIG, LOW);

    long durata = pulseIn(PIN_ECHO, HIGH); // cat timp revine ecoul
    long distanta = durata * 0.034 / 2;    // viteza sunetului -> cm
    return distanta;
}

void loop() {
    long distanta = masoaraDistanta();
    Serial.print(distanta);
    Serial.println(" cm");

    if (distanta > 0 && distanta < 30) {
        // Cu cat mai aproape, cu atat pauza mai mica intre bipuri
        int pauza = map(distanta, 0, 30, 40, 500);
        tone(PIN_BUZZER, 1000, 40);
        delay(pauza);
    } else {
        noTone(PIN_BUZZER);
        delay(100);
    }
}
```

Explicația codului

- `masoaraDistanta()` : funcția noastră, care trimite impulsul pe `TRIG` și măsoară ecoul.
- `digitalWrite(TRIG, HIGH); delayMicroseconds(10);` : impuls scurt de exact 10 microsecunde.
- `pulseIn(PIN_ECHO, HIGH)` : cronometrează cât timp pinul `ECHO` rămâne `HIGH` (durata ecoului).
- `distanta = durata * 0.034 / 2;` : transformă timpul în centimetri.
- Cu `map(distanta, 0, 30, 40, 500)` calculăm pauza: aproape → pauză mică (bipuri dese).

Sfaturi și depanare

✓ SFAT

`tone(pin, frecventa, durata)` cu al treilea parametru sună exact atâtea milisecunde și se oprește singur, perfect pentru un „bip” scurt, fără `noTone()`.

! ATENȚIE / DEPANARE

Distanța e mereu 0 sau o valoare uriașă? Verifică să nu fi inversat `TRIG` și `ECHO`. **Citiri instabile?** Senzorul are nevoie de o suprafață care reflectă bine; obiectele moi sau înclinate „împrăștie” ecoul.

I Provocare

★ PROVOCARE

Adaugă **trei LED-uri** (verde/galben/roșu) ca la un semafor de parcare: verde la distanță mare, galben la mijloc, roșu foarte aproape, pe lângă bipurile buzzer-ului.

Proiectul 9. Seif cu cod PIN

Dificultate ●●●

Durată 75 min

Componente 5

Până acum, o intrare însemna un buton. Acum folosim **16 butoane pe doar 8 pini**, printr-un truc numit matrice. Construim un seif care se descuie doar cu codul corect.

! Ce vei învăța

- Cum funcționează o **tastatură matriceală** și de ce are nevoie de atât de puțini pini;
- Cum citești caractere, nu doar HIGH și LOW ;
- Cum comanzi un **servomotor** ca să miști ceva în lumea reală;
- Cum construiești o logică de verificare a unui cod.

! Componente necesare

Cant.	Componentă
1×	Placă Arduino UNO
1×	Tastatură matriceală 4×4
1×	Servomotor SG90 (zăvorul)
1×	Ecran LCD 16×2 (conectare directă, fără modul I2C)
1×	Potențiomtru 10 kΩ (pentru reglaj contrast LCD)
1×	Buzzer piezoelectric
1×	Breadboard + fire

! Schema de conectare

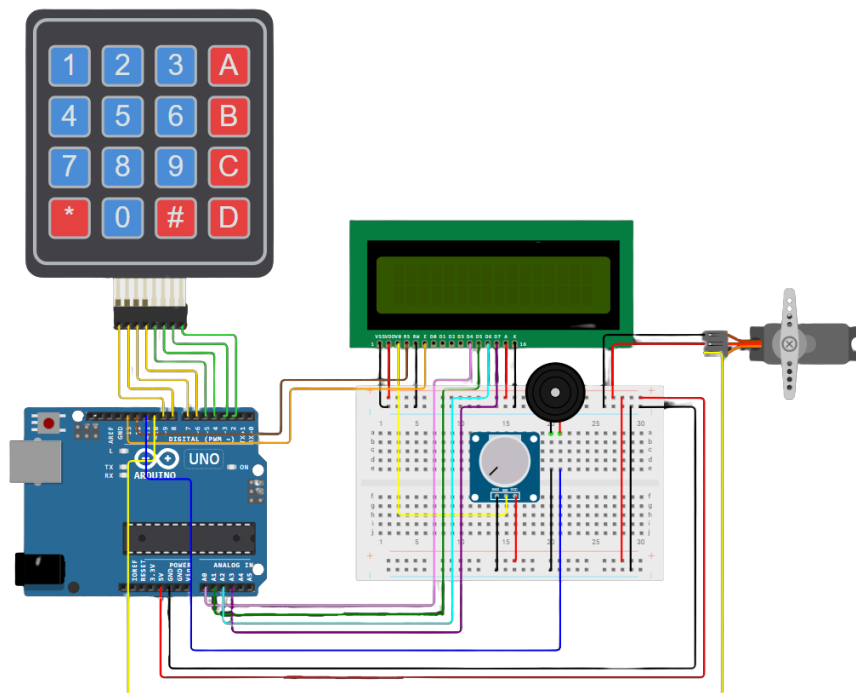


Figura 13: Seif cu cod PIN, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- ▶ Construiește și rulează circuitul în browser, fără componente fizice:

<https://wokwi.com/projects/470683787320102913>

I Cum funcționează

O tastatură cu 16 taste ar avea nevoie, teoretic, de 16 pini. Ar ocupa aproape toată placa. Trucul se numește **matrice**: tastele sunt așezate pe **4 rânduri** și **4 coloane**, iar fiecare tastă leagă un rând de o coloană atunci când o apeși.

Placa verifică rândurile pe rând, foarte repede: activează rândul 1 și se uită pe ce coloană apare semnal, apoi rândul 2, și tot așa. Dacă găsește semnal pe rândul 2, coloana 3, știe că s-a apăsât tastea de la intersecția lor, adică 6. Așa, **4 + 4 = 8 pini** sunt de ajuns pentru 16 taste. Biblioteca `Keypad` face toată această scanare în locul nostru.

i REȚINE

Instalează biblioteca **Keypad** (de Mark Stanley și Alexander Brevig), la fel ca `DHT sensor library` de la Proiectul 7. Pașii și videoul cu instalarea bibliotecilor sunt în **Capitolul 0, secțiunea „Bibliotecile”**. Biblioteca `LiquidCrystal` vine inclusă în Arduino IDE și **nu** trebuie instalată separat.

Montaj pas cu pas

1. **Tastatura** are 8 pini într-un singur șir. Privind-o cu tastele spre tine, primii patru (de la stânga) sunt **rândurile**, ultimii patru sunt **coloanele**.
2. Conectează rândurile la pinii **9, 8, 7, 6** și coloanele la pinii **5, 4, 3, 2**.
3. **Servomotorul**: firul roșu → 5V, maro/negru → GND, portocaliu (semnal) → pinul 10.
4. **Buzzer**: un picior la **pinul 11**, celălalt la GND.
5. **LCD** VSS (pin 1) → GND.
6. **LCD** VDD (pin 2) → 5V.
7. **LCD** V0 (pin 3) → pinul din mijloc al potențiometrului (contrast).
8. **LCD** RS (pin 4) → **pinul 12**.
9. **LCD** RW (pin 5) → GND.
10. **LCD** EN (pin 6) → **pinul 13**.
11. **LCD** D4 (pin 11) → **pinul A0**.
12. **LCD** D5 (pin 12) → **pinul A1**.
13. **LCD** D6 (pin 13) → **pinul A2**.
14. **LCD** D7 (pin 14) → **pinul A3**.
15. **LCD** A (pin 15) → 5V (backlight +).
16. **LCD** K (pin 16) → GND (backlight -).
17. Potențiometrul de contrast: picioarele laterale la 5V și GND, cursorul (mijloc) la V0.
18. Conectează placa la USB. Codul implicit este **1234**.

I Codul

```
/*
  Proiectul 9 - Seif cu cod PIN
  O tastatura matriceala 4x4 cere un cod. Daca e corect, servomotorul
  "descuie" seiful, iar ecranul LCD confirma. Daca e gresit, buzzer-ul protesteaza.

  BIBLIOTECI (Sketch -> Include Library -> Manage Libraries):
  - "Keypad" (Mark Stanley, Alexander Brevig)
  - "LiquidCrystal" (inclusa in Arduino IDE)
  - "Servo" (inclusa in Arduino IDE)
  Curs Arduino Erasmus+
*/

#include <Keypad.h>
#include <Servo.h>
#include <LiquidCrystal.h>

const byte RANDURI = 4;
const byte COLOANE = 4;

// Ce scrie pe fiecare tasta, asezate ca pe tastatura reala
char taste[RANDURI][COLOANE] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte piniRanduri[RANDURI] = {9, 8, 7, 6};
byte piniColoane[COLOANE] = {5, 4, 3, 2};

Keypad tastatura = Keypad(makeKeymap(taste), piniRanduri, piniColoane, RANDURI, COLOANE);

const int PIN_SERVO = 10;
const int PIN_BUZZER = 11;

const String COD_CORECT = "1234"; // schimba-l cu al tau
String codIntrodus = "";

Servo servo;
// LCD conectat direct: RS=12, EN=13, D4=A0, D5=A1, D6=A2, D7=A3
LiquidCrystal lcd(12, 13, A0, A1, A2, A3);

void ecranInitial() {
  lcd.clear();
  lcd.print("Introdu codul:");
  lcd.setCursor(0, 1);
}

void setup() {
  pinMode(PIN_BUZZER, OUTPUT);
  servo.attach(PIN_SERVO);
  servo.write(0); // seiful este incuiat

  lcd.begin(16, 2);
  ecranInitial();
}
```

```
void deschideSeiful() {
    lcd.clear();
    lcd.print("Cod corect!");
    lcd.setCursor(0, 1);
    lcd.print("Seiful e deschis");

    tone(PIN_BUZZER, 1000, 120); // doua bipuri vesele
    delay(160);
    tone(PIN_BUZZER, 1400, 200);

    servo.write(90); // descuie
    delay(4000); // ramane deschis 4 secunde
    servo.write(0); // incuie la loc
}

void codGresit() {
    lcd.clear();
    lcd.print("Cod gresit!");
    tone(PIN_BUZZER, 200, 600); // sunet grav de eroare
    delay(1500);
}

void loop() {
    char tasta = tastatura.getKey(); // returneaza 0 daca nu s-a apasat nimic

    if (tasta) {
        if (tasta == '#') { // # = confirma codul
            if (codIntrodus == COD_CORECT) {
                deschideSeiful();
            } else {
                codGresit();
            }
            codIntrodus = "";
            ecranInitial();
        } else if (tasta == '*') { // * = sterge tot
            codIntrodus = "";
            ecranInitial();
        } else { // orice alta tasta se adauga la cod
            codIntrodus += tasta;
            lcd.print('*'); // ascundem cifrele pe ecran
            tone(PIN_BUZZER, 1200, 40); // click scurt de confirmare
        }
    }
}
```

I Explicația codului

- `char taste[4][4] = {...}` : un **tablou bidimensional**, adică o hartă a tastaturii. Scrie exact ce vezi pe taste, în aceeași așezare.
- `Keypad tastatura = Keypad(makeKeymap(taste), piniRanduri, piniColoane, 4, 4);` : îi spunem bibliotecii cum arată tastatura și pe ce pini e legată.
- `char tasta = tastatura.getKey();` : citește tasta apăsată. Dacă nu s-a apăsă nimic, returnează `0`, iar `if (tasta)` este fals, deci nu se întâmplă nimic.

- `codIntrodus += tasta;` : adaugă caracterul la textul introdus până acum. `+=` înseamnă „adaugă la ce era deja”.
- `if (codIntrodus == COD_CORECT)` : comparăm tot textul dintr-o dată, la apăsarea lui `#`.
- `lcd.print('*');` : afișăm o steluță în loc de cifră, ca la orice tastatură de bancomat.

I Sfaturi și depanare

✓ SFAT

Schimbă-ți codul modificând o singură linie: `const String COD_CORECT = "1234";` . Poate fi oricât de lung și poate conține și litere (A , B , C , D).

! ATENȚIE / DEPANARE

Nu răspunde la nicio tastă? Cel mai probabil rândurile și coloanele sunt inversate. Schimbă între ele valorile din `piniRanduri` și `piniColoane` , este cea mai frecventă greșeală la acest montaj.

Apeși o tastă și apare alta? Ordinea pinilor e amestecată. Verifică firele unul câte unul.

Servomotorul tremură? Trage mult curent; pentru montaje serioase alimentează-l separat.

I Provocare

★ PROVOCARE

Adaugă o **limită de trei încercări**: după trei coduri greșite, seiful se blochează 30 de secunde și afișează pe LCD un numărător invers. Bonus: aprinde un LED roșu cât timp e blocat.

Proiectul 10. Proiect final: Sistem inteligent de alarmă

Dificultate ...

Durată 90 min

Componente 6

Proiectul de absolvire! Combinăm tot ce am învățat (intrări, ieșiri, sunet, mișcare și afișaj) într-un singur sistem coerent: o alarmă antiefracție inteligentă.

Ce vei învăța

- Cum **integrezi** mai multe componente într-un proiect real;
- Cum **refolosești** o funcție scrisă anterior (măsurarea distanței de la Proiectul 8);
- Cum comanzi un **servomotor** și cum organizezi codul în **funcții** pentru claritate.

Componente necesare

Cant.	Componentă
1×	Placă Arduino UNO
1×	Senzor ultrasonic HC-SR04
1×	Servomotor SG90
1×	Buzzer + LED (semnalizare)
1×	Ecran LCD 16×2 (conectare directă, fără modul I2C)
1×	Potențiometru 10 kΩ (pentru reglaj contrast LCD)
1×	Breadboard + fire

Schema de conectare

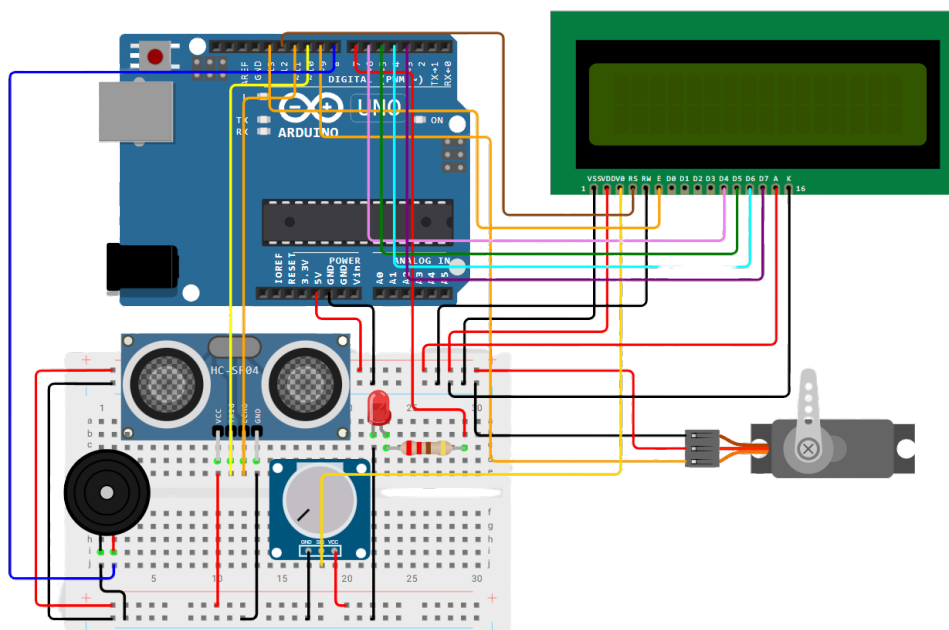


Figura 14: Sistem de alarmă, vedere pe breadboard (Wokwi)

Testează montajul online (Wokwi)

- ▶ Construiește și rulează circuitul în browser, fără componente fizice:
<https://wokwi.com/projects/470693754496131073>

I Cum funcționează

Senzorul **HC-SR04** păzește o zonă: măsoară continuu distanța până la primul obstacol din fața lui. Cât timp zona e liberă, distanța rămâne mare. Dacă cineva intră în zonă, distanța scade brusc sub un **prag** (aici 25 cm), iar sistemul **declanșează alarma**: aprinde LED-ul, sună buzzer-ul, coboară „bariera” cu servomotorul și afișează pe LCD la ce distanță a fost detectat intrusul. Este suma tuturor abilităților din curs, puse să lucreze împreună.

✓ SFAT

Recunoști funcția `masoaraDistanța()` ? Este exact cea pe care ai scris-o la **Proiectul 8**. Așa lucrează programatorii: o funcție care merge bine se refolosește, nu se rescrie.

i REȚINE

Bibliotecile `Servo` și `LiquidCrystal` sunt ambele incluse în Arduino IDE și **nu** trebuie instalate separat.

Montaj pas cu pas

1. **HC-SR04:** VCC → 5V , GND → GND , TRIG → pinul 10 , ECHO → pinul 11 .
2. **Servomotor:** firul roșu → 5V , maro/negru → GND , portocaliu (semnal) → pinul 9 .
3. **LED** cu rezistență la **pinul 7**; **buzzer** la **pinul 8**.
4. **LCD** VSS (pin 1) → GND .
5. **LCD** VDD (pin 2) → 5V .
6. **LCD** V0 (pin 3) → pinul din mijloc al potențiometrului (contrast).
7. **LCD** RS (pin 4) → **pinul 12**.
8. **LCD** RW (pin 5) → GND .
9. **LCD** EN (pin 6) → **pinul 13**.
10. **LCD** D4 (pin 11) → **pinul 6**.
11. **LCD** D5 (pin 12) → **pinul 5**.
12. **LCD** D6 (pin 13) → **pinul 4**.
13. **LCD** D7 (pin 14) → **pinul 3**.
14. **LCD** A (pin 15) → 5V (backlight +).
15. **LCD** K (pin 16) → GND (backlight -).
16. Potențiometrul de contrast: picioarele laterale la 5V și GND , cursorul (mijloc) la V0 .
17. Așază senzorul astfel încât să „privească” spre zona pe care vrei să o supraveghezi.
18. Conectează placa la USB. La pornire, ai 2 secunde să ieși din zona supravegheată.

Codul

```
/*
Proiectul 10 - PROIECT FINAL: Sistem inteligent de alarma
Combina tot ce am invatat:
- senzor ultrasonic HC-SR04 -> pazeste o zona si detecteaza intrusul
- buzzer + LED                -> semnal de alarma
- servomotor                  -> "inchide" o bariera / roteste o camera
- ecran LCD 16x2 (direct)     -> afiseaza starea sistemului

BIBLIOTECI: "Servo" si "LiquidCrystal" (ambele incluse in Arduino IDE).
Curs Arduino Erasmus+
*/

#include <Servo.h>
#include <LiquidCrystal.h>

const int PIN_TRIG   = 10;
const int PIN_ECHO   = 11;
const int PIN_BUZZER = 8;
const int PIN_LED    = 7;
const int PIN_SERVO  = 9;

const int PRAG_CM = 25; // sub aceasta distanta consideram ca cineva a intrat in zona

Servo servo;
// LCD conectat direct: RS=12, EN=13, D4=6, D5=5, D6=4, D7=3
LiquidCrystal lcd(12, 13, 6, 5, 4, 3);

// Aceeasi functie pe care ai scris-o la Proiectul 8
long masoaraDistanta() {
    digitalWrite(PIN_TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(PIN_TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(PIN_TRIG, LOW);

    long durata = pulseIn(PIN_ECHO, HIGH, 30000); // renunta dupa 30 ms
    if (durata == 0) return -1;                  // niciun ecou primit
    return durata * 0.034 / 2;                   // centimetri
}

void setup() {
    pinMode(PIN_TRIG, OUTPUT);
    pinMode(PIN_ECHO, INPUT);
    pinMode(PIN_BUZZER, OUTPUT);
    pinMode(PIN_LED, OUTPUT);
    servo.attach(PIN_SERVO);
    servo.write(0); // bariera ridicata (pozitia de repaus)

    lcd.begin(16, 2);
    lcd.print("Sistem armat");
    delay(2000); // timp sa iesi din zona supravegheata
    lcd.clear();
}

void declanseazaAlarma(long distanta) {
    lcd.clear();
    lcd.print("!! ALARMA !!");
}
```

```
lcd.setCursor(0, 1);  
lcd.print("Intrus la ");  
lcd.print(distanța);  
lcd.print("cm ");  
  
for (int i = 0; i < 5; i++) {  
    digitalWrite(PIN_LED, HIGH);  
    tone(PIN_BUZZER, 1200);  
    servo.write(90);          // coboara bariera  
    delay(300);  
    digitalWrite(PIN_LED, LOW);  
    noTone(PIN_BUZZER);  
    servo.write(0);           // o ridica la loc  
    delay(300);  
}  
lcd.clear();  
}  
  
void loop() {  
    long distanța = masoaraDistanța();  
  
    if (distanța > 0 && distanța < PRAG_CM) {  
        declanseazaAlarma(distanța);  
    } else {  
        lcd.setCursor(0, 0);  
        lcd.print("Supraveghere... ");  
        lcd.setCursor(0, 1);  
        if (distanța > 0) {  
            lcd.print("Zona libera:");  
            lcd.print(distanța);  
            lcd.print("cm ");  
        } else {  
            lcd.print("Zona libera      ");  
        }  
    }  
  
    delay(200);  
}
```

Explicația codului

- `#include <Servo.h>` + `servo.attach(PIN_SERVO)` : pregătim servomotorul; `servo.write(unghi)` îl duce la un unghi (0-180°).
- `declanseazaAlarma()` : **funcția noastră** care rulează toată secvența de alarmă. A o pune separat face `loop()` scurt și ușor de citit.
- `const int PRAG_CM = 25;` : distanța sub care considerăm că cineva a intrat în zonă. Este singura valoare pe care trebuie să o reglezi pentru camera ta.
- În `loop()`, `if (distanța > 0 && distanța < PRAG_CM)` verifică zona; `&&` înseamnă „și”, deci ambele condiții trebuie să fie adevărate în același timp.
- `pulseIn(PIN_ECHO, HIGH, 30000)` : al treilea argument este un **timp maxim de așteptare**. Fără el, programul s-ar bloca atunci când nu revine niciun ecou.
- Bucla `for` din alarmă clipește LED-ul, sună buzzer-ul și mișcă servomotorul de 5 ori.

I Sfaturi și depanare

✓ SFAT

Reglează `PRAG_CM` după locul unde montezi senzorul: pune-l cât mai aproape de distanța până la peretele din față, minus câțiva centimetri. Deschide **Monitorul Serial** și urmărește valorile ca să alegi pragul potrivit.

! ATENȚIE / DEPANARE

Alarma pornește singură la nesfârșit? Pragul e prea mare, iar senzorul „vede” un obiect fix (o cutie, marginea mesei). Micșorează `PRAG_CM` sau îndreaptă senzorul în altă direcție. **Servomotorul tremură sau resetează placa?** Trage mult curent; pentru montaje serioase, alimentează-l dintr-o sursă externă de 5 V.

I Provocare

★ PROVOCARE

Adaugă **tastatura de la Proiectul 9** și transformă montajul într-o alarmă adevărată: la pornire cere un cod pentru a **arma** sistemul, iar când alarma sună, ea se oprește doar dacă introduci codul corect. LCD-ul arată „ARMAT” / „DEZARMAT”.

Bonus: afișează **de câte ori** s-a declanșat alarma de la pornire.

Montaj pas cu pas

1. **HC-SR04:** VCC → 5V, GND → GND, TRIG → pinul 10, ECHO → pinul 11.
2. **Servomotor:** firul roșu → 5V, maro/negru → GND, portocaliu (semnal) → pinul 9.
3. **LED** cu rezistență la **pinul 7**; **buzzer** la **pinul 8**.
4. **LCD** VSS (pin 1) → GND.
5. **LCD** VDD (pin 2) → 5V.
6. **LCD** V0 (pin 3) → pinul din mijloc al potențiometrului (contrast).
7. **LCD** RS (pin 4) → **pinul 12**.
8. **LCD** RW (pin 5) → GND.
9. **LCD** EN (pin 6) → **pinul 13**.
10. **LCD** D4 (pin 11) → **pinul 6**.
11. **LCD** D5 (pin 12) → **pinul 5**.
12. **LCD** D6 (pin 13) → **pinul 4**.
13. **LCD** D7 (pin 14) → **pinul 3**.
14. **LCD** A (pin 15) → 5V (backlight +).
15. **LCD** K (pin 16) → GND (backlight -).
16. Potențiometrul de contrast: picioarele laterale la 5V și GND, cursorul (mijloc) la V0.
17. Așază senzorul astfel încât să „privească” spre zona pe care vrei să o supraveghezi.
18. Conectează placa la USB. La pornire, ai 2 secunde să ieși din zona supravegheată.

Codul

```
/*
Proiectul 10 - PROIECT FINAL: Sistem inteligent de alarma
Combina tot ce am invatat:
- senzor ultrasonice HC-SR04 -> pazeste o zona si detecteaza intrusul
- buzzer + LED -> semnal de alarma
- servomotor -> "inchide" o bariera / roteste o camera
- ecran LCD 16x2 (direct) -> afiseaza starea sistemului

BIBLIOTECI: "Servo" si "LiquidCrystal" (ambele incluse in Arduino IDE).
Curs Arduino Erasmus+
*/

#include <Servo.h>
#include <LiquidCrystal.h>

const int PIN_TRIG = 10;
const int PIN_ECHO = 11;
const int PIN_BUZZER = 8;
const int PIN_LED = 7;
const int PIN_SERVO = 9;

const int PRAG_CM = 25; // sub aceasta distanta consideram ca cineva a intrat in zona

Servo servo;
// LCD conectat direct: RS=12, EN=13, D4=6, D5=5, D6=4, D7=3
LiquidCrystal lcd(12, 13, 6, 5, 4, 3);

// Aceeasi functie pe care ai scris-o la Proiectul 8
long masoaraDistanta() {
    digitalWrite(PIN_TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(PIN_TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(PIN_TRIG, LOW);

    long durata = pulseIn(PIN_ECHO, HIGH, 30000); // renunta dupa 30 ms
    if (durata == 0) return -1; // niciun ecou primit
    return durata * 0.034 / 2; // centimetri
}

void setup() {
    pinMode(PIN_TRIG, OUTPUT);
    pinMode(PIN_ECHO, INPUT);
    pinMode(PIN_BUZZER, OUTPUT);
    pinMode(PIN_LED, OUTPUT);
    servo.attach(PIN_SERVO);
    servo.write(0); // bariera ridicata (pozitia de repaus)

    lcd.begin(16, 2);
    lcd.print("Sistem armat");
    delay(2000); // timp sa iesi din zona supravegheata
    lcd.clear();
}

void declanseazaAlarma(long distanta) {
    lcd.clear();
    lcd.print("!! ALARMA !!");
}
```

```
lcd.setCursor(0, 1);
lcd.print("Intrus la ");
lcd.print(distanța);
lcd.print("cm ");

for (int i = 0; i < 5; i++) {
    digitalWrite(PIN_LED, HIGH);
    tone(PIN_BUZZER, 1200);
    servo.write(90);          // coboara bariera
    delay(300);
    digitalWrite(PIN_LED, LOW);
    noTone(PIN_BUZZER);
    servo.write(0);           // o ridica la loc
    delay(300);
}
lcd.clear();
}

void loop() {
    long distanța = masoaraDistanța();

    if (distanța > 0 && distanța < PRAG_CM) {
        declanseazaAlarma(distanța);
    } else {
        lcd.setCursor(0, 0);
        lcd.print("Supraveghere... ");
        lcd.setCursor(0, 1);
        if (distanța > 0) {
            lcd.print("Zona libera:");
            lcd.print(distanța);
            lcd.print("cm ");
        } else {
            lcd.print("Zona libera      ");
        }
    }

    delay(200);
}
```

Explicația codului

- `#include <Servo.h>` + `servo.attach(PIN_SERVO)`: pregătim servomotorul; `servo.write(unghi)` îl duce la un unghi (0-180°).
- `declanseazaAlarma()`: **funcția noastră** care rulează toată secvența de alarmă. A o pune separat face `loop()` scurt și ușor de citit.
- `const int PRAG_CM = 25;`: distanța sub care considerăm că cineva a intrat în zonă. Este singura valoare pe care trebuie să o reglezi pentru camera ta.
- În `loop()`, `if (distanța > 0 && distanța < PRAG_CM)` verifică zona; `&&` înseamnă „și”, deci ambele condiții trebuie să fie adevărate în același timp.
- `pulseIn(PIN_ECHO, HIGH, 30000)`: al treilea argument este un **timp maxim de așteptare**. Fără el, programul s-ar bloca atunci când nu revine niciun ecou.
- Bucla `for` din alarmă clipește LED-ul, sună buzzer-ul și mișcă servomotorul de 5 ori.

I Sfaturi și depanare

✓ SFAT

Reglează `PRAG_CM` după locul unde montezi senzorul: pune-l cât mai aproape de distanța până la peretele din față, minus câțiva centimetri. Deschide **Monitorul Serial** și urmărește valorile ca să alegi pragul potrivit.

! ATENȚIE / DEPANARE

Alarma pornește singură la nesfârșit? Pragul e prea mare, iar senzorul „vede” un obiect fix (o cutie, marginea mesei). Micșorează `PRAG_CM` sau îndreaptă senzorul în altă direcție. **Servomotorul tremură sau resetează placa?** Trage mult curent; pentru montaje serioase, alimentează-l dintr-o sursă externă de 5 V.

I Provocare

★ PROVOCARE

Adaugă **tastatura de la Proiectul 9** și transformă montajul într-o alarmă adevărată: la pornire cere un cod pentru a **arma** sistemul, iar când alarma sună, ea se oprește doar dacă introduci codul corect. LCD-ul arată „ARMAT” / „DEZARMAT”.

Bonus: afișează **de câte ori** s-a declanșat alarma de la pornire.

i REȚINE

Felicitări! Ai parcurs toate cele 10 proiecte. Ai învățat intrări și ieșiri digitale și analogice, PWM, sunet, senzori, biblioteci, ecrane și motoare, fundamentele cu care poți construi acum propriile invenții. Următorul pas îți aparține.

Glosar de termeni

Termenii de care ai nevoie, explicați pe scurt. Revino aici ori de câte ori întâlnești un cuvânt necunoscut.

Sketch	Programul scris în Arduino IDE și încărcat pe placă.
setup()	Funcția care rulează o singură dată, la pornire. Aici pregătim pinii.
loop()	Funcția care se repetă la nesfârșit. Aici stă comportamentul proiectului.
Pin digital	Pin cu două stări: HIGH (5 V) sau LOW (0 V).
Pin analogic	Pin care citește valori variabile, de la 0 la 1023 (A0-A5).
HIGH / LOW	Cele două stări digitale: pornit (5 V) și oprit (0 V).
INPUT / OUTPUT	Rolul unui pin: intrare (citește) sau ieșire (comandă).
INPUT_PULLUP	Intrare cu rezistență internă care ține pinul la HIGH când e liber.
PWM	Tehnică prin care un pin simulează valori intermediare (intensitate, viteză). Pinii ~.
analogWrite()	Trimite o valoare PWM (0-255) pe un pin ~.
analogRead()	Citește un pin analogic și returnează 0-1023.
digitalWrite()	Pune un pin digital pe HIGH sau LOW.
digitalRead()	Citește starea (HIGH / LOW) a unui pin digital.
delay()	Oprește programul un număr de milisecunde (1000 ms = 1 s).
map()	„Traduce” o valoare dintr-un interval în altul.
Serial	Canalul de comunicare între placă și calculator (Monitor Serial).
Bibliotecă	Colecție de funcții gata scrise pe care le aduci cu #include.
Breadboard	Placă de test cu găuri conectate electric, pentru montaje fără lipituri.

Rezistență	Componentă care limitează curentul; se măsoară în ohmi (Ω).
Divizor de tensiune	Două rezistențe în serie folosite pentru a citi un senzor (ex. LDR).
Polaritate	Orientarea + / - a unei componente (LED-urile o au, rezistențele nu).
GND (masă)	„Minusul” comun al circuitului; punctul de referință 0 V.
Microcontroler	Cipul-computer de pe placă (ATmega328P) care rulează sketch-ul.

Anexe

I A1. Fișă rapidă: funcții Arduino esențiale

Funcție	Ce face
<code>pinMode(pin, MOD)</code>	Setează pinul ca <code>INPUT</code> , <code>OUTPUT</code> sau <code>INPUT_PULLUP</code> .
<code>digitalWrite(pin, val)</code>	Pune pinul pe <code>HIGH</code> (5 V) sau <code>LOW</code> (0 V).
<code>digitalRead(pin)</code>	Citește starea unui pin digital (<code>HIGH</code> / <code>LOW</code>).
<code>analogRead(pin)</code>	Citește un pin analogic → valoare 0-1023.
<code>analogWrite(pin, val)</code>	Trimite PWM (0-255) pe un pin <code>~</code> .
<code>delay(ms)</code>	Așteaptă un număr de milisecunde.
<code>delayMicroseconds(us)</code>	Așteaptă un număr de microsecunde.
<code>map(x, in1, in2, out1, out2)</code>	Transformă <code>x</code> dintr-un interval în altul.
<code>tone(pin, freqv[, durata])</code>	Emite un sunet pe buzzer la frecvența dată.
<code>noTone(pin)</code>	Oprește sunetul.
<code>pulseIn(pin, stare)</code>	Măsoară durata unui impuls (ex. senzor ultrasonic).
<code>Serial.begin(9600)</code>	Pornește comunicarea cu calculatorul.
<code>Serial.print</code> / <code>println()</code>	Afișează text/valori pe Monitorul Serial.

I A2. Codul culorilor pentru rezistențe

Inel	Culoare	Cifră	Mult.	Exemplu
	Negru	0	×1	
	Maro	1	×10	maro-negru-portocaliu = 10 kΩ
	Roșu	2	×100	roșu-roșu-marou = 220 Ω
	Portocaliu	3	×1 000	
	Galben	4	×10 000	
	Verde	5	×100 000	
	Albastru	6	×1 000 000	
	Violet	7	×10 000 000	
	Gri	8		
	Alb	9		

Toleranță: auriu = $\pm 5\%$, argintiu = $\pm 10\%$.



Calculator online pentru rezistențe

Selectezi culorile inelelor și primești valoarea în ohmi (sau invers).

<https://www.calculator.net/resistor-calculator.html>

I A3. Pinout Arduino UNO



Figura 15: Harta pinilor plăcii Arduino UNO (pinout)

I A4. Pinii folosiți în curs

Pin	Rol tipic în proiecte
5V / GND	Alimentarea oricărui montaj.
0-13	Pini digitali (LED-uri, butoane, buzzer, senzori digitali).
~3, ~5, ~6, ~9, ~10, ~11	Pini PWM (analogWrite): intensitate, culori, viteză.
A0-A5	Pini analogici (potențiometru, LDR) sau pini digitali suplimentari (ecranul LCD din Proiectul 9).