



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PORTOFOLIU DE PRACTICĂ

MOBILITATE CIPRU

10.05.2026 – 23.05.2026

NĂSTĂSOIU ANA-MARIA

UNGUREANU PAUL-BOGDAN

VRÂNCEANU VLAD



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

Cuprins

PROIECT: Semafor	2
PROIECT: Senzor de Proximitate cu UltraSunete și Indicator Vizual LED	7
PROIECT: Sistem de control proporțional al luminozității unui LED.....	11
PROIECT: Detector de Zgomot.....	13
PROIECT: Sistem de Alarmă Acustică cu Senzor de Sunet KY-037 și Buzzer	16
PROIECT: STEPPER MOTOR.....	19
PROIECT: Sistem de Control Liniar prin Tastatură.....	23
PROIECT: Motor, keypad și buton	28
PROIECT: Buton / Comutator.....	29
PROIECT FINAL: Automated Return Conveyor	32



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: Semafor

[Link material video 1](#)

[Link material video 2](#)

[Link material video 3](#)

1. Prezentare Generală

Acest proiect reprezintă o simulare hardware și software a unui semafor rutier standard pentru o singură bandă. Scopul este de a demonstra modul în care un microcontroler (Arduino Uno) poate controla dispozitive digitale de ieșire (LED-uri) într-o secvență temporizată exactă, imitând comportamentul unui sistem de trafic real. Proiectul este bazat pe componente din "Arduino Starter Kit".

2. Analiza Configurației Hardware (pe baza imaginii)

Din imagine, putem deduce următoarea conexiune a componentelor:

- **Microcontroler:** Un Arduino Uno R3, conectat prin USB la un laptop pentru alimentare și programare.
- **Bord de dezvoltare (Breadboard):** Un mini breadboard montat pe un suport galben, folosit pentru a face conexiunile.
- **LED-uri (Componentele active):**
 - Un ansamblu vertical compact cu trei LED-uri este vizibil în partea de jos a breadboard-ului.
 - Culoarea de sus este **Roșu**.
 - Culoarea din mijloc este **Galben**.
 - Culoarea de jos este **Verde**.
- **Cablaj de control (Pinii Digitali):** Cablurile colorate pornesc de la pinii digitali ai Arduino către breadboard:
 - Firul **Galben** pare conectat la Pinul Digital **10**. (Va controla LED-ul Roșu).
 - Firul **Verde** pare conectat la Pinul Digital **9**. (Va controla LED-ul Galben).
 - Firul **Negru** pare conectat la Pinul Digital **8**. (Va controla LED-ul Verde).
- **Cablaj de alimentare și rezistoare:**



Cofinanțat de
Uniunea Europeană



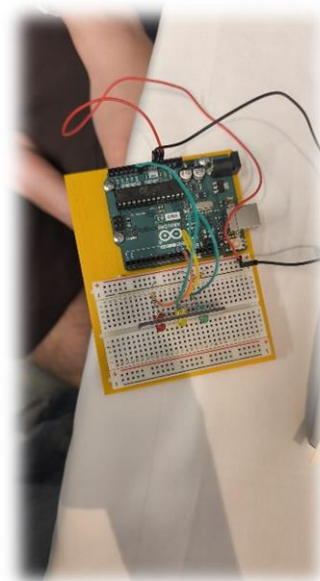
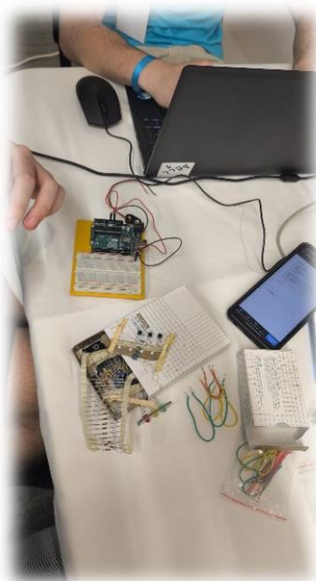
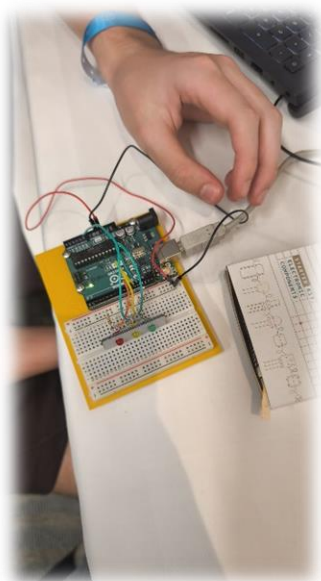
Start în Robotică cu Erasmus+ 2025-1-RO01-KA122-VET-000340573

- Firul **Roșu** conectează pinul **5V** al Arduino la șina pozitivă a breadboard-ului (marcată cu '+').
- Sunt vizibile cel puțin două rezistoare (pentru limitarea curentului LED-urilor) conectate între șina de alimentare și LED-uri. Aceasta este o configurație ușor neobișnuită (de obicei se comută '+' și se pune rezistorul pe '-'), dar funcțională dacă LED-urile sunt controlate prin comutarea liniei de împământare sau dacă sunt de tip anoad comun.
- *Notă: Pentru o funcționare corectă și sigură a codului de mai jos, vom presupune conexiunea standard: rezistor în serie cu fiecare LED, iar pinii Arduino furnizează 5V pentru a le aprinde.*

3. Logica Software (Secvența de Trafic)

Codul va executa o buclă continuă (în funcția loop), respectând secvența standard a unui semafor european:

1. **Faza ROȘU:** Doar LED-ul Roșu este aprins. (Oprire trafic).
2. **Faza ROȘU + GALBEN:** LED-ul Roșu rămâne aprins, iar LED-ul Galben se aprinde și el. (Pregătire pornire).
3. **Faza VERDE:** LED-urile Roșu și Galben se sting, iar LED-ul Verde se aprinde. (Pornire trafic).
4. Ciclul se reia (automat, Verde se va stinge și Roșu se va aprinde).



COD:



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+ 2025-1-RO01-KA122-VET-000340573

/* Descriere: Acest cod controleaza o secventa de trei LED-uri (Rosu, Galben, Verde) pentru a imita un semafor. Temporizarea este accelerata pentru demonstratie.

* Conexiuni (bazate pe interpretarea imaginii):

* LED Rosu -> Pin Digital 10 (via rezistor)

* LED Galben -> Pin Digital 9 (via rezistor)

* LED Verde -> Pin Digital 8 (via rezistor)

*/

// Definirea pinilor pentru LED-uri folosind constante

// Acest lucru face codul mai usor de citit si modificat.

```
const int pinRosu = 10;
```

```
const int pinGalben = 9;
```

```
const int pinVerde = 8;
```

// Definirea temporizărilor (în milisecunde)

```
const long timpRosu = 5000;    // 5 secunde
```

```
const long timpRosuGalben = 1500; // 1.5 secunde
```

```
const long timpVerde = 5000;    // 5 secunde
```

```
void setup() {
```

```
    // Configurarea pinilor selectați ca ieșiri digitale.
```

```
    pinMode(pinRosu, OUTPUT);
```

```
    pinMode(pinGalben, OUTPUT);
```



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
pinMode(pinVerde, OUTPUT);

// Inițializarea tuturor LED-urilor la starea STINS la pornirea
sistemului.

digitalWrite(pinRosu, LOW);

digitalWrite(pinGalben, LOW);

pinMode(pinVerde, OUTPUT);

}

void loop() {

    // --- FAZA ROȘU (Oprire) ---

    // Aprindem doar LED-ul Roșu.
    digitalWrite(pinRosu, HIGH);
    digitalWrite(pinGalben, LOW);
    digitalWrite(pinVerde, LOW);

    // Așteptăm durata setată pentru roșu.
    delay(timpRosu);

    // --- FAZA ROȘU + GALBEN (Pregătire) ---

    // Menținem Roșu aprins și aprindem și Galben.
    digitalWrite(pinRosu, HIGH);
    digitalWrite(pinGalben, HIGH);
    digitalWrite(pinVerde, LOW);

    // Așteptăm durata scurtă de tranziție.
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
delay(timpRosuGalben);
```

```
// --- FAZA VERDE (Pornire) ---
```

```
// Stingem Roșu și Galben, aprindem Verde.
```

```
digitalWrite(pinRosu, LOW);
```

```
digitalWrite(pinGalben, LOW);
```

```
digitalWrite(pinVerde, HIGH);
```

```
// Așteptăm durata setată pentru verde.
```

```
delay(timpVerde);
```

```
// Ciclul se reia automat la faza ROȘU.
```

```
}
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: Senzor de Proximitate cu UltraSunete și Indicator Vizual LED

Descriere: Proiectul implementează un senzor de distanță și proximitate bazat pe modulul ultrasonic HC-SR04 și o placă de dezvoltare Arduino. Sistemul emite impulsuri ultrasonice pentru a calcula distanța până la un obstacol în centimetri și inch, afișând datele pe Monitorul Serial. Nivelul de apropiere este semnalizat vizual prin 3 LED-uri: pe măsură ce obiectul se apropie (sub 30 cm , sub 20 cm și sub 10 cm), LED-urile se aprind succesiv.

Componente Hardware & Conexiuni:

- **Senzor Ultrasonic HC-SR04:**
 - Pin Trigger → Pinul digital 11 (Ieșire)
 - Pin Echo → Pinul digital 12 (Intrare)
- **Indicatoare Vizuale:**
 - LED 1 → Pinul 4
 - LED 2 → Pinul 5
 - LED 3 → Pinul 6
- **Interfață Serială:** Comunicare la 9600 baud.

Calcul Matematic și Principiu de Funcționare:

1. **Emisie și Recepție:** Pinul Trigger trimite un impuls scurt de $10\text{ }\mu\text{s}$. Unda sonoră se reflectă de obiect și este recepționată de Echo, durata fiind măsurată în microsecunde cu `pulseIn()`.

2. **Conversie în Distanță:**

$$\text{Distanță (cm)} = \frac{\text{durată}}{2} \times 0.0343 \approx \frac{\text{durată}/2}{29.1}$$

3. **Praguri de Avertizare:**

Distanță parcursă	Stare LED-uri	Semnificație
$> 30\text{ cm}$	○ ○ ○ (Toate stinse)	Zona sigură / Niciun obstacol



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

Distanță parcursă	Stare LED-uri	Semnificație
$20\text{ cm} \leq \text{cm} < 30\text{ cm}$	● ○ ○ (1 LED aprins)	Obstacol detectat la distanță medie
$10\text{ cm} < \text{cm} < 20\text{ cm}$	● ● ○ (2 LED-uri aprinse)	Obstacol apropiat
$\leq 10\text{ cm}$	● ● ● (3 LED-uri aprinse)	Alertă de proximitate / Foarte aproape

COD:

```

int trigPin = 11;    // Trigger

int echoPin = 12;    // Echo

int ledPin1 = 4, ledPin2 = 5, ledPin3 = 6;

long duration, cm, inches;

void setup() {

    //Serial Port begin

    Serial.begin (9600);

    //Define inputs and outputs

    pinMode(trigPin, OUTPUT);

    pinMode(echoPin, INPUT);

    pinMode(ledPin1, OUTPUT);

    pinMode(ledPin2, OUTPUT);

    pinMode(ledPin3, OUTPUT);

}

```



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
void loop() {  
  
    // The sensor is triggered by a HIGH pulse of 10 or more microseconds.  
  
    // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:  
  
    digitalWrite(trigPin, LOW);  
  
    delayMicroseconds(5);  
  
    digitalWrite(trigPin, HIGH);  
  
    delayMicroseconds(10);  
  
    digitalWrite(trigPin, LOW);  
  
  
    // Read the signal from the sensor: a HIGH pulse whose  
  
    // duration is the time (in microseconds) from the sending  
  
    // of the ping to the reception of its echo off of an object.  
  
    pinMode(echoPin, INPUT);  
  
    duration = pulseIn(echoPin, HIGH);  
  
  
    // Convert the time into a distance  
  
    cm = (duration/2) / 29.1;      // Divide by 29.1 or multiply by 0.0343  
  
    inches = (duration/2) / 74;    // Divide by 74 or multiply by 0.0135  
  
  
    if (cm > 30) {  
  
        digitalWrite(ledPin1, LOW);  
  
        digitalWrite(ledPin2, LOW);  
  
        digitalWrite(ledPin3, LOW);  
  
    }  
}
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
} else if (cm >= 20 && cm < 30) {  
  
    digitalWrite(ledPin1, HIGH);  
  
    digitalWrite(ledPin2, LOW);  
  
    digitalWrite(ledPin3, LOW);  
  
} else if (cm > 10 && cm < 20) {  
  
    digitalWrite(ledPin1, HIGH);  
  
    digitalWrite(ledPin2, HIGH);  
  
    digitalWrite(ledPin3, LOW);  
  
} else {  
  
    digitalWrite(ledPin1, HIGH);  
  
    digitalWrite(ledPin2, HIGH);  
  
    digitalWrite(ledPin3, HIGH);  
  
}  
  
  
Serial.print(inches);  
  
Serial.print("in, ");  
  
Serial.print(cm);  
  
Serial.print("cm");  
  
Serial.println();  
  
  
delay(250);  
  
}
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: Sistem de control proporțional al luminozității unui LED

[Link material video](#)

Descriere: Proiectul implementează un sistem de reglare a intensității luminoase a unui LED (pe pinul PWM 6) în funcție de citirile furnizate de un senzor analogic (A0). Codul utilizează un prag minim de activare (540) și convertește domeniul de măsură (540 — 800) într-un semnal PWM pe 8 biți (0 — 255) folosind funcția `map()`. Datele brute și cele procesate sunt transmise în timp real prin interfața Serială.

Componente Hardware & Conexiuni:

- **Intrare Analogică (sensorPin A0):** Conectată la un senzor analogic (LDR, potențiomtru, etc.).
- **Ieșire PWM (ledPin 6):** Pin digital cu suport pentru PWM (`analogWrite`), folosit pentru varierea tensiunii medii aplicate LED-ului.
- **Comunicare Serială:** 9600 baud pentru monitorizarea valorilor în timp real.

Logica de Funcționare și Algoritm:

1. **Citire date:** Se măsoară valoarea analogică (0 — 1023) furnizată de senzor.
2. **Filtrare prag minim (Thresholding):** * Dacă `sensorValue < 540`, instrucțiunea `return` oprește executarea restului de cod din `loop()`, ignorând valorile sub acest prag.
3. **Mascarea și Conversia Domeniului (map):**
 - Valoarea din intervalul [540,800] este mapată liniar în intervalul PWM [0,255]:
$$\text{outputValue} = \text{map}(\text{sensorValue}, 540, 800, 0, 255)$$
4. **Telemetrie Serială:** Se printează în consolă perechea de valori `sensorValue` `outputValue` (util pentru calibrare).
5. **Comandă PWM:** Funcția `analogWrite(ledPin, outputValue)` modifică factorul de umplere (*duty cycle*) al pinului 6, ajustând luminozitatea LED-ului.

COD:

```
#define ledPin 6
#define sensorPin A0
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
void setup() {  
    Serial.begin(9600);  
    pinMode(ledPin, OUTPUT);  
    digitalWrite(ledPin, LOW);  
}  
  
void loop()  
{  
    unsigned int sensorValue = analogRead(sensorPin);  
    if (sensorValue < 540)  
        return;  
    uint8_t outputValue = map(sensorValue, 540, 800, 0, 255);  
    Serial.print(sensorValue);  
    Serial.print(" ");  
    Serial.println(outputValue);  
    analogWrite(ledPin, outputValue);  
}
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: Detector de Zgomot

Descriere: Proiectul implementează un sistem de monitorizare acustică utilizând modulul senzor de sunet KY-037. Sistemul procesează simultan două tipuri de semnale: un semnal digital (care aprinde LED-ul integrat pe pinul 13 atunci când nivelul zgomotului depășește un prag preconfigurat din potențiometrul senzorului) și un semnal analogic (trimis continuu către Monitorul Serial pentru analizarea intensității audio în timp real).

Componente Hardware & Conexiuni:

- **Senzor de Sunet KY-037:**
 - **Ieșire Digitală (DO):** Conectată la pinul digital 7 (`digitalPin`).
 - **Ieșire Analogică (AO):** Conectată la pinul analogic A0 (`analogPin`).
- **Indicator Vizual:**
 - **LED-ul integrat pe placă (LED_BUILTIN):** Conectat la pinul digital 13 (`ledPin`).
- **Comunicare Serială:** Rata de transfer setată la 9600 baud.

Principiu de Funcționare:

1. Citire și Evaluare Digitală:

- Modulul KY-037 are un potențiometru fizic integrat care permite ajustarea manuală a sensibilității.
- Când volumul zgomotului ambiental depășește pragul setat, pinul digital trimite un semnal HIGH, determinând aprinderea LED-ului de pe pinul 13. Dacă sunetul scade sub prag, pinul devine LOW și stinge LED-ul.

2. Citire și Monitorizare Analogică:

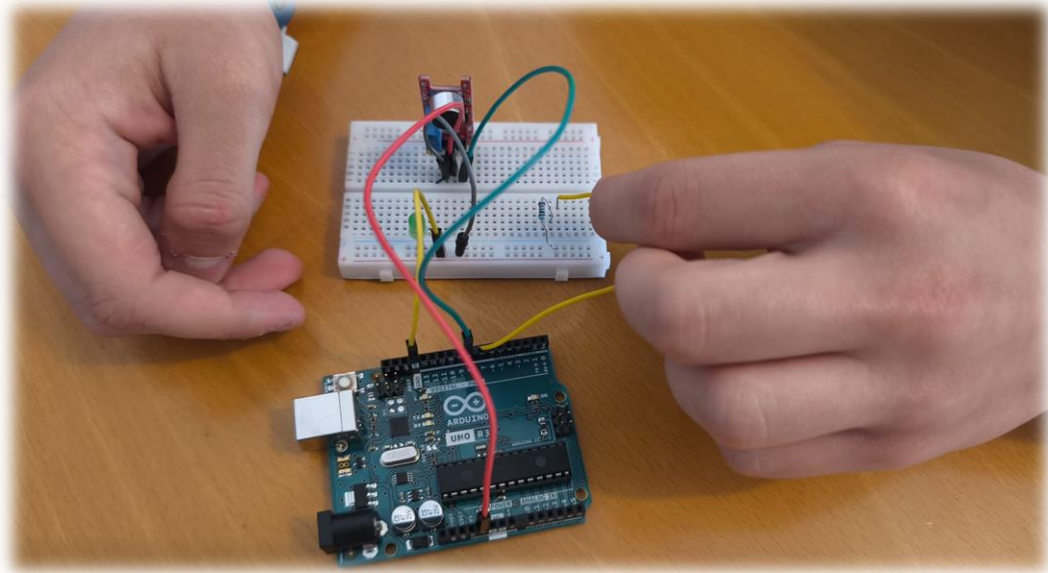
- Pinul A0 măsoară amplitudinea instantanee a semnalului audio (valori între 0 și 1023).
- Valoarea brută citită este transmisă prin `Serial.println(analogVal)` pentru a putea fi vizualizată grafic (de exemplu, cu `Serial Plotter`).



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573



COD :

```
int digitalPin = 7;    // KY-037 digital interface
int analogPin = A0;    // KY-037 analog interface
int ledPin = 13;       // Arduino LED pin
int digitalVal;        // digital readings
int analogVal;         // analog readings

void setup()
{
    pinMode(digitalPin, INPUT);
    pinMode(analogPin, INPUT);
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    // Read the digital interface
    digitalVal = digitalRead(digitalPin);
```



Cofinanțat de
Uniunea Europeană



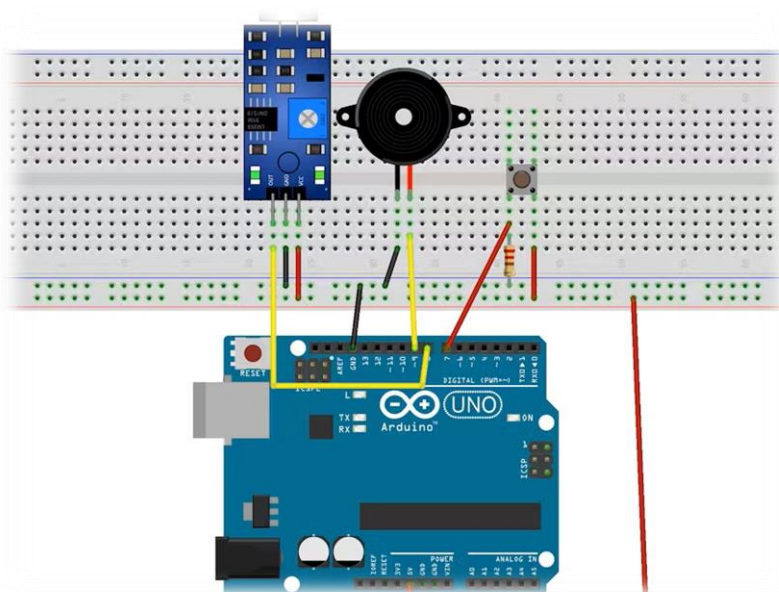
Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
if(digitalVal == HIGH)
{
    digitalWrite(ledPin, HIGH); // Turn ON Arduino's LED
}
else
{
    digitalWrite(ledPin, LOW);  // Turn OFF Arduino's LED
}

// Read analog interface
analogVal = analogRead(analogPin);
// Print analog value to serial
Serial.println(analogVal);
}
```

PROIECT: Sistem de Alarmă Acustică cu Senzor de Sunet KY-037 și Buzzer

Descriere: Proiectul implementează o alarmă activată de sunet folosind un senzor audio KY-037 și un buzzer acționat pe pinul 9. La detectarea unui zgomot ce depășește pragul hardware calibrat, sistemul generează un ton sonor de 1000 Hz timp de 3 secunde. În paralel, semnalul analogic brut este citit de pe pinul A0 și transmis în timp real pe Monitorul Serial.



Componente Hardware & Conexiuni:

- **Senzor de Sunet KY-037:**
 - **Ieșire Digitală (DO):** Conectată la pinul digital 7 (digitalPin).
 - **Ieșire Analogică (AO):** Conectată la pinul analogic A0 (analogPin).
- **Element de Alertă Sonoră:**
 - **Buzzer:** Conectat la pinul digital PWM 9 (buzzer).
- **Interfață Serială:** Rata de transfer 9600 baud.

Principiu de Funcționare și Algoritm:

1. **Detectare Prag Digital (Digital Read):**

Start în Robotică cu Erasmus+ 2025-1-RO01-KA122-VET-000340573

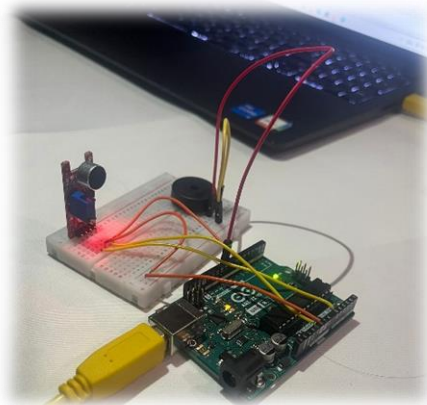
- Senzorul KY-037 evaluează sunetul din mediu. Dacă intensitatea depășește pragul setat manual prin potențiometrul de pe modul, pinul 7 devine HIGH.

2. Activare Alarmă Sonoră:

- La citirea valorii HIGH, se apelează funcția tone(buzzer, 1000), generând o frecvență de 1000 Hz pe pinul 9.
- Instrucțiunea delay(3000) menține alarma activă timp de **3 secunde**, după care tonul este oprit cu noTone(buzzer).

3. Măsurare și Telemetrie Analogică:

- Pinul A0 citește valoarea analogică instantanee (0 – 1023), transmisă continuu prin Serial.println(analogVal) pentru vizualizarea formei de undă acustice.



COD:

```
int digitalPin = 7;    // KY-037 digital interface
int analogPin = A0;    // KY-037 analog interface
const int buzzer=9;    //buzzer pin
int digitalVal;        // digital readings
int analogVal;         // analog readings

void setup()
{
  pinMode(digitalPin, INPUT);
  pinMode(analogPin, INPUT);
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

}

```
void loop()
```

```
{
```

```
// Read the digital interface
```

```
digitalVal = digitalRead(digitalPin);
```

```
if(digitalVal == HIGH)
```

```
{
```

```
tone(buzzer,1000);
```

```
delay(3000);
```

```
noTone(buzzer);
```

```
}
```

```
else
```

```
noTone(buzzer);
```

```
analogVal = analogRead(analogPin);
```

```
// Print analog value to serial
```

```
Serial.println(analogVal);
```

```
}
```



Cofinanțat de
Uniunea Europeană



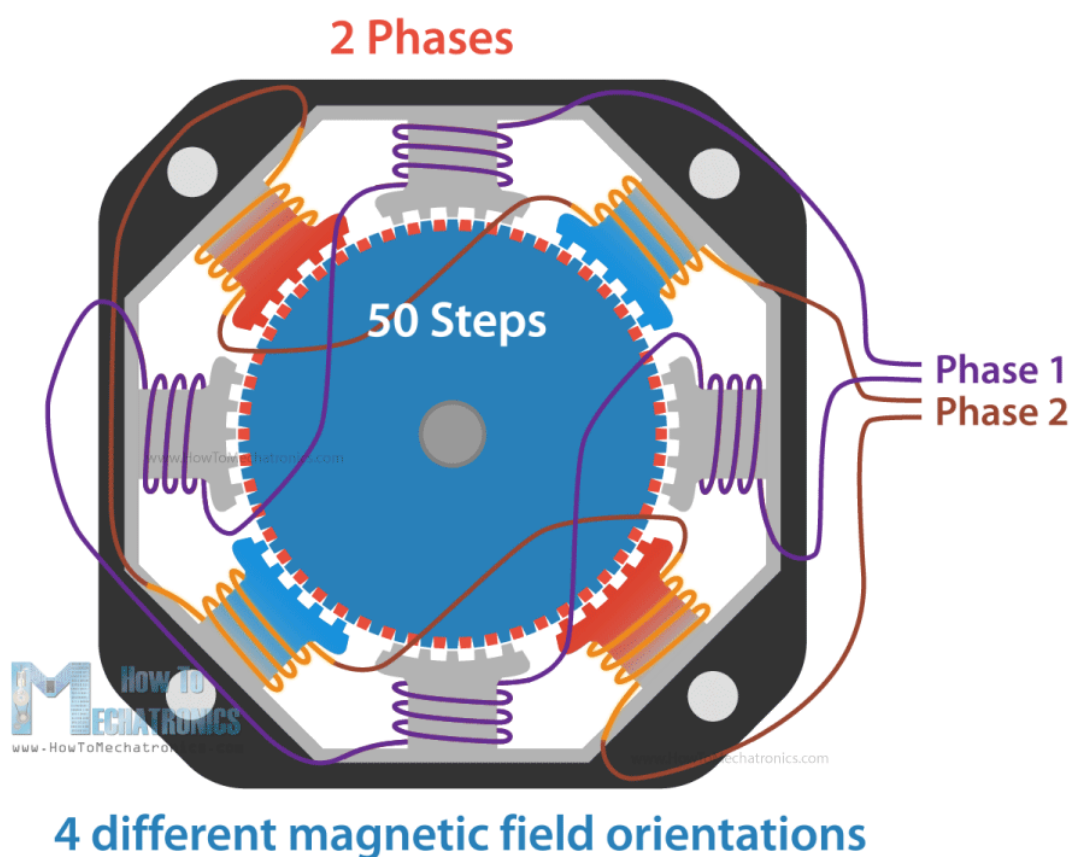
Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: STEPPER MOTOR

[Link material video](#)

Un motor pas cu pas este un tip special de motor DC (de curent continuu) fără perii, a cărui poziție poate fi controlată cu precizie chiar și fără un sistem de feedback.

Principiul de funcționare al unui motor pas cu pas se bazează pe câmpurile magnetice. Acesta are două componente principale: un **stator** și un **rotor**. Rotorul este, de regulă, un magnet permanent, fiind înconjurat de o serie de bobine dispuse pe stator.



Când alimentăm sau lăsăm curentul să curgă prin bobine, în stator se generează câmpuri magnetice specifice care fie atrag, fie resping rotorul. Prin activarea bobinelor pas cu pas, una după alta într-o ordine specifică, putem obține o mișcare continuă a rotorului, dar îl putem și opri în orice poziție dorită.

Mărind numărul de poli magnetici ai rotorului, putem crește numărul de poziții posibile de oprire, sporind astfel rezoluția sau precizia motorului.



Cofinanțat de
Uniunea Europeană

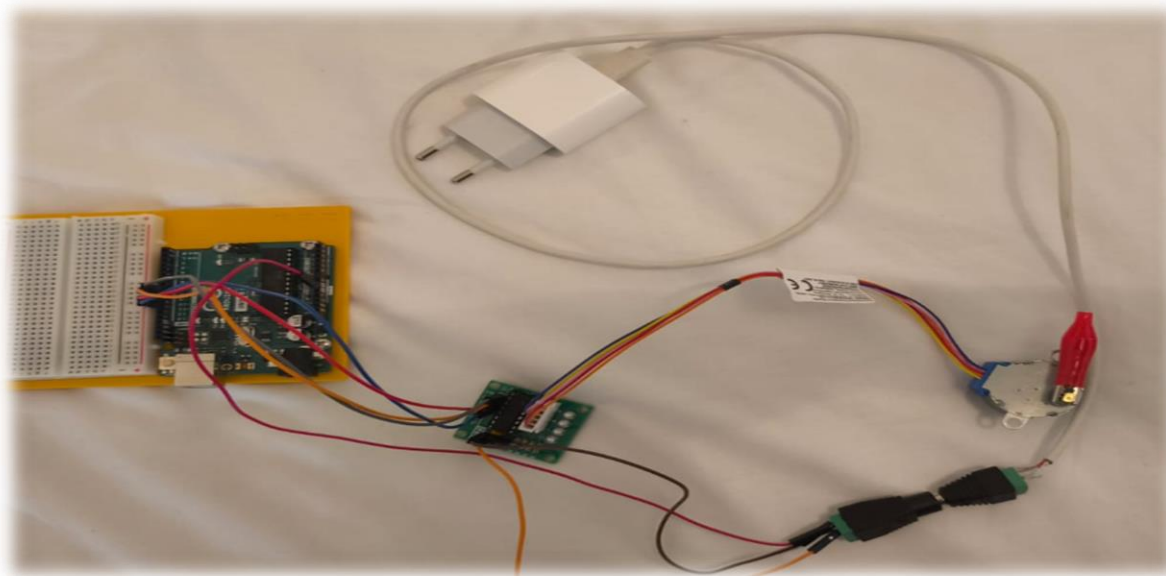


Start în Robotică cu Erasmus+ 2025-1-RO01-KA122-VET-000340573

Un motor pas cu pas tipic, cum ar fi un NEMA 17, are 50 de puncte de oprire (sau pași) pe rotor. Pe de altă parte, statorul poate avea mai multe bobine organizate în două faze, care oferă patru orientări sau poziții diferite ale câmpului magnetic.

Cele 50 de stări/pași ai rotorului înmulțiți cu cele 4 orientări diferite ale câmpului magnetic dau un total de **200 de pași** pentru a completa o rotație întreagă. Sau, dacă împărțim 360° la 200 de pași, obținem o rezoluție de **1.8° per pas**.

Bobinele statorului sunt organizate în două faze, lucru pe care îl putem observa dacă ne uităm la numărul de fire ale unui motor pas cu pas. Acesta are 4 fire, câte două pentru fiecare fază. Cele patru orientări diferite ale câmpului magnetic sunt posibile deoarece putem lăsa curentul să curgă prin faze în ambele sensuri.



COD:

```
#include <Stepper.h>

// Define Constants

// Number of steps per internal motor revolution

const float STEPS_PER_REV = 32;

// Amount of Gear Reduction

const float GEAR_RED = 64;
```



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
// Number of steps per geared output rotation

const float STEPS_PER_OUT_REV = STEPS_PER_REV * GEAR_RED;

// Define Variables

// Number of Steps Required

int StepsRequired;

// Create Instance of Stepper Class

// Specify Pins used for motor coils

// The pins used are 8,9,10,11

// Connected to ULN2003 Motor Driver In1, In2, In3, In4

// Pins entered in sequence 1-3-2-4 for proper step sequencing

Stepper steppermotor(STEPS_PER_REV, 8, 10, 9, 11);

void setup()

{

// Nothing (Stepper Library sets pins as outputs)

}

void loop()

{

// Slow - 4-step CW sequence to observe lights on driver board
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
steppermotor.setSpeed(1);

StepsRequired = 4;

steppermotor.step(StepsRequired);

delay(2000);


// Rotate CW 1/2 turn slowly

StepsRequired = STEPS_PER_OUT_REV / 2;

steppermotor.setSpeed(100);

steppermotor.step(StepsRequired);

delay(1000);


// Rotate CCW 1/2 turn quickly

StepsRequired = - STEPS_PER_OUT_REV / 2;

steppermotor.setSpeed(700);

steppermotor.step(StepsRequired);

delay(2000);

}
```

PROIECT: Sistem de Control Liniar prin Tastatură

Proiectul își propune să controleze deplasarea precisă a unui mecanism pe o anumită distanță exprimată în milimetri, introducând direct de utilizator prin intermediul unei tastaturi matriciale. Sistemul își măsoară parcursul și își schimbă automat direcția de mers când atinge o limită prestabilită.

Componente Principale și Funcționalitate

1. Interfața de introducere a datelor (Tastatura 4x4):

- **Tastele 0–9:** Permit introducerea numărului de milimetri pe care utilizatorul dorește să îi parcurgă sistemul.
- **Tasta # (Confirmare):** Funcționează ca tasta *Enter*. Calculează numărul exact de pași necesari și declanșează mișcarea motorului.
- **Tasta * (Reset):** Funcționează ca *Clear*, ștergând valoarea introdusă din greșeală.

2. Mecanismul de acționare (Motorul Pas cu Pas):

- Se folosește un motor pas cu pas (cu un raport de reducere de 1:64, rezultând 2048 pași/revoluție).
- Este calibrat pentru o translație liniară: **1 mm corespunde la 51.2 pași** (ceea ce înseamnă că o rotație completă mută ansamblul cu aproximativ 40 mm).

3. Logica de Control și Siguranță:

- **Monitorizarea distanței totale:** Codul ține evidența distanței acumulate (totalDistance).
- **Inversarea automată a direcției:** Când distanța totală parcursă atinge limita de **200 mm (20 cm)**, valoarea pașilor devine negativă, determinând motorul să se rotească în sens opus (să se întoarcă la punctul de start) și resetează contorul.
- **Monitorizare Serială:** Trimite pe calculator (prin Serial Monitor la 9600 baud) distanța totală și pașii calculați pentru fiecare comandă.

Conexiuni Hardware (Pinout)

• Tastatură Matricială 4x4:

- Linii (Rows): Pinii **2, 3, 4, 5**

Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

- Coloane (Columns): Pinii 6, 7, 8, 9
- **Driver Motor Pas cu Pas:**
 - Conectat la pinii analogici **A0, A2, A1, A3** (folosiți ca pini digitali de ieșire).

💡 **Aplicații Practice**

Un astfel de sistem poate fi folosit ca bază pentru:

- Ghidaje liniare CNC mici sau imprimante 3D.
- Sisteme automate de dozare/dispensare a lichidelor pe distanțe fixe.
- Mese mobile pentru debitare sau poziționare de precizie în atelier.

COD:

```
#include <Stepper.h>

#include <Keypad.h>

const int ROW_NUM    = 4; // four rows

const int COLUMN_NUM = 4; // four columns

char keys[ROW_NUM][COLUMN_NUM] = {

    {'1', '2', '3', 'A'},

    {'4', '5', '6', 'B'},

    {'7', '8', '9', 'C'},

    {'*', '0', '#', 'D'}

};

byte pin_rows[ROW_NUM] = {2,3,4,5}; // connect to the row pinouts of the keypad
```



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
byte pin_column[COLUMN_NUM] = {6,7,8,9}; // connect to the column pinouts of  
the keypad
```

```
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,  
COLUMN_NUM );
```

```
int keypad_number = 0;
```

```
//the motor has 32 internal steps
```

```
const int INTERNAL_STEPS_PER_REVOL=32;
```

```
//the motor also has internal gears to reduce the speed and increase the torque
```

```
const int GEAR_REDUCTION=64;
```

```
//total steps per revolution
```

```
const int STEPS_PER_REVOLUTION=INTERNAL_STEPS_PER_REVOL*GEAR_REDUCTION;
```

```
int stepsRequired;
```

```
float stepsPerMlmtr=51.2;
```

```
float distanceToCover, distanceToSteps;
```

```
int totalDistance=0;
```

```
Stepper theMotor(INTERNAL_STEPS_PER_REVOL, A0, A2, A1, A3);
```

```
void setup() {
```

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

```
}
```



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
void loop() {  
  
    // put your main code here, to run repeatedly:  
  
    theMotor.setSpeed(1000);  
  
    theMotor.step(stepsRequired);  
  
    delay(2000);  
  
    //lets have the motor do one full revolution so we can measure the distance  
    it covers  
  
    //in one full revolution  
  
    /* stepsRequired=STEPS_PER_REVOLUTION;  
  
    theMotor.step(stepsRequired);  
  
    delay(2000);*/  
  
    //after measurements one full revolution is equal to 4cm  
  
    //now we can calculate the mm per step.  
  
    //if we have 40mm per 2048 steps and we want to calculate  
  
    //how many steps to cover 1mm. and the answer is 51.2 steps per millimeter  
  
    char key = keypad.getKey();  
  
    while (key >= 0 && key <= 9){  
  
        int digit = key - '0';  
  
        keypad_number = (keypad_number * 10) + digit;  
  
        char key = keypad.getKey();  
  
    }  
  
    if(key == '#'){  
  
        distanceToCover=keypad_number;
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
stepsRequired=distanceToCover * stepsPerMlmtr;
```

```
totalDistance=totalDistance+distanceToCover;
```

```
if (totalDistance>=200){
```

```
    totalDistance=0;
```

```
    stepsPerMlmtr=stepsPerMlmtr*-1;
```

```
}
```

```
Serial.print("Total Distance:");
```

```
Serial.println(totalDistance);
```

```
Serial.print("Steps Required: ");
```

```
Serial.println(stepsRequired);
```

```
theMotor.step(stepsRequired);
```

```
delay(10);
```

```
//if
```

```
}
```

```
else if (key == '*'){
```

```
    keypad_number=0;
```

```
}
```

```
}
```



Cofinanțat de
Uniunea Europeană

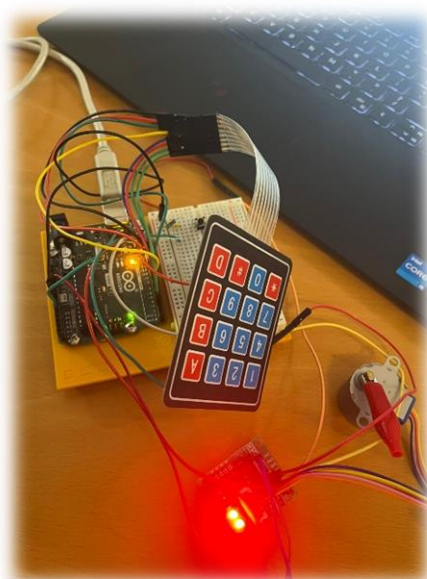


Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: Motor, keypad și buton

[Link material video 1](#)

[Link material video 2](#)





Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT: Buton / Comutator

Un comutator momentan (push-button) este un comutator mecanic care își schimbă starea electrică doar în timp ce este apăsat fizic.

Odată ce presiunea este îndepărtată, un arc intern readuce automat comutatorul la poziția sa inițială (implicită).

1. Stări Implicite (NO vs. NC)

Majoritatea Butoanelor Momentane Utilizate în Electronică Sunt de Tip **Normally Open (NO)**.

- **Normally Open (NO - Normal Deschis):** Circuitul este deconectat în mod implicit. Apăsarea butonului închide (completează) circuitul.
- **Normally Closed (NC - Normal Închis):** Circuitul este conectat în mod implicit. Apăsarea butonului întrerupe circuitul.

2. Identificarea Pinilor la Butoanele cu 4 Pini

Butoanele tactile standard folosite pe plăcile de testare (breadboard) au de obicei 4 pini. Acest lucru poate fi derutant, deoarece pinii sunt conectați intern în perechi.

- **Regula:** Pinii care se află direct față în față (peste spațiul central de plastic) sunt separați.
- **Conexiunea:** Pinii de pe aceeași latură sunt conectați permanent între ei.
- **Cea mai bună practică:** Conectează pinii **diagonali opuși** pentru a garanta că butonul este plasat corect în întreruperea circuitului.

3. Problema Stării Flotante („Floating Pin”)

Dacă conectezi un buton direct la un pin de Arduino fără nicio altă componentă, pinul va rămâne într-o stare „flotantă” (nedeterminată).

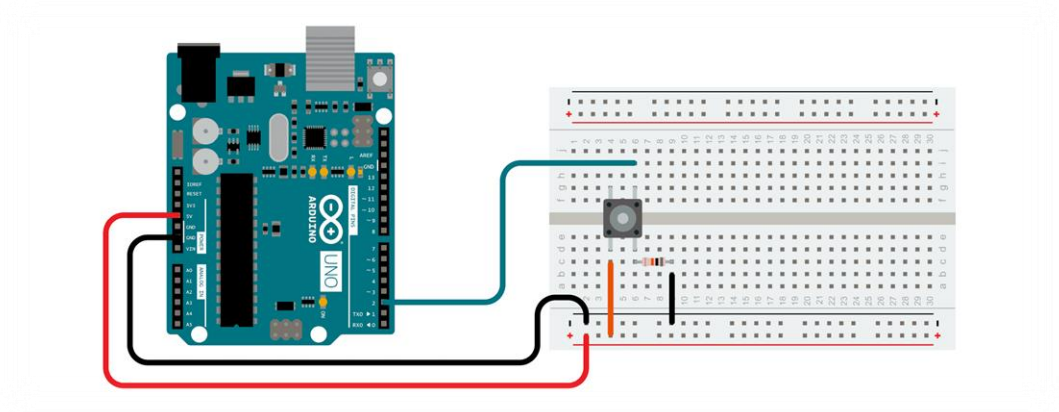
- **De ce se întâmplă:** Pinii microcontrolerelor sunt extrem de sensibili. Când butonul este deschis (neapăsat), pinul funcționează ca o antenă, captând interferențe electromagnetice aleatorii din aer.
- **Simptomul:** Arduino va citi aleatoriu valorile HIGH și LOW, provocând erori de funcționare.
- **Soluția:** Pinul trebuie legat întotdeauna la o stare de tensiune cunoscută folosind un rezistor (de tip *pull-down* sau *pull-up*).



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+ 2025-1-RO01-KA122-VET-000340573



COD:

```
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin = 13;   // the number of the LED pin

// variables will change:
int buttonState = 0; // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
  }
}
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

```
digitalWrite(ledPin, HIGH);  
} else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
}  
}
```



Cofinanțat de
Uniunea Europeană



Start în Robotică cu Erasmus+
2025-1-RO01-KA122-VET-000340573

PROIECT FINAL: Automated Return Conveyor

[Link material video](#)

Un sistem compact și independent de bandă rulantă. Utilizatorul introduce distanța în milimetri de la o tastatură, ecranul LCD afișează valoarea, iar la apăsarea butonului START, platforma se deplasează precis prin intermediul unui motor pas cu pas. O a doua apăsare a butonului aduce automat platforma înapoi la punctul de start și resetează sistemul.

