



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

PORTOFOLIU DE PRACTICĂ

MOBILITATE CIPRU

10.05.2026 – 23.05.2026

LUPU IRIS

TURTURICĂ SARA-MARIA

PÎRJOL IONUȚ



Cofinanțat de
Uniunea Europeană



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

Cuprins

PROIECT: Traffic Light	2
<i>PROIECT</i> : Sistem de monitorizare a umidității solului	4
<i>PROIECT</i> : Controlul unui Motor Pas cu Pas	6
<i>PROIECT</i> : Actuator Liniar cu Motor Pas cu Pas și Calibrare în Milimetri	8
PROIECT: Actuator Liniar Interactiv cu Motor Pas cu Pas, Buton Start/Stop și Autorevers	11
PROIECT: Actuator Liniar Programabil cu Tastatură Matricială 4x4 și Arduino	14
PROIECT FINAL: Sistem Automat de Detecție a Pachetelor pe Conveior.....	17

PROIECT: Traffic Light

Descriere: Proiectul reprezintă un sistem de control al traficului ce simulează funcționarea unui semafor rutier cu 3 LED-uri (Roșu, Galben, Verde). Codul execută o secvență temporizată standard de tranziție a stărilor: fază de oprire (Roșu - 3s), fază de pregătire (Roșu + Galben - 1s) și fază de permisiune a trecerii (Verde - 3s), repetând ciclul în mod automat.

Arhitectură și Configurație Hardware:

- **Ieșiri digitale:** Pinii 4 (LED Roșu), 5 (LED Galben) și 6 (LED Verde).
- **Stare inițială:** Toate LED-urile sunt setate pe LOW (stînse) în secțiunea setup().

Secvența de Funcționare (Ciclul loop):

1. **Faza ROȘU (Oprire):** Doar LED-ul Roșu este aprins timp de **3 secunde** (3000 ms).
2. **Faza ROȘU + GALBEN (Tranziție/Pregătire):** LED-ul Roșu rămâne aprins și se alătură LED-ul Galben timp de **1 secundă** (1000 ms), semnalizând pregătirea pentru pornire.
3. **Faza VERDE (Trecere):** LED-urile Roșu și Galben se sting, iar LED-ul Verde se aprinde timp de **3 secunde** (3000 ms).
4. **Ciclul se reia automat de la faza ROȘU.**

```
// Pin definitions
```

```
const int RED_PIN = 4;
```

```
const int YELLOW_PIN = 5;
```

```
const int GREEN_PIN = 6;
```

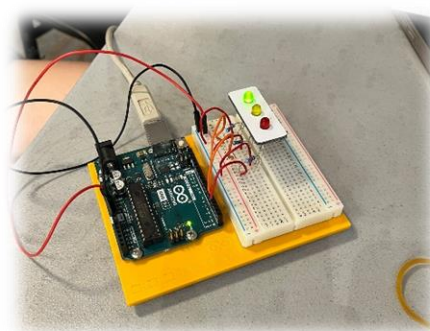
```
void setup() {
```

```
    // Configure LED pins as outputs
```

```
    pinMode(RED_PIN, OUTPUT);
```

```
    pinMode(YELLOW_PIN, OUTPUT);
```

```
    pinMode(GREEN_PIN, OUTPUT);
```





Cofinanțat de
Uniunea Europeană



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

```
// Initialize all LEDs to OFF

digitalWrite(RED_PIN, LOW);

digitalWrite(YELLOW_PIN, LOW);

digitalWrite(GREEN_PIN, LOW);

}

void loop() {

    // RED phase: 3 seconds

    digitalWrite(RED_PIN, HIGH);

    digitalWrite(YELLOW_PIN, LOW);

    digitalWrite(GREEN_PIN, LOW);

    delay(3000);

    // RED + YELLOW phase: 1 second (transition)

    digitalWrite(RED_PIN, HIGH);

    digitalWrite(YELLOW_PIN, HIGH);

    digitalWrite(GREEN_PIN, LOW);

    delay(1000);

    // GREEN phase: 3 seconds

    digitalWrite(RED_PIN, LOW);

    digitalWrite(YELLOW_PIN, LOW);

    digitalWrite(GREEN_PIN, HIGH);

    delay(3000);

    // Cycle repeats automatically

}
```



Cofinanțat de
Uniunea Europeană



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

PROIECT: Sistem de monitorizare a umidității solului

Descriere: Proiectul reprezintă un sistem automatizat de monitorizare a nivelului de apă din sol folosind un senzor analogic și un microcontroler Arduino. Informațiile sunt citite în timp real și afișate prin intermediul a două LED-uri, oferind un feedback vizual rapid pe trei niveluri: **Uscat** (fără LED-uri aprinse), **Nivel mediu de apă** (un LED aprins) și **Nivel optim/Multă apă** (ambele LED-uri aprinse).

Principiu de Funcționare & Logică:

- **Senzor analogic (A0):** Măsoară rezistența/conductivitatea solului. Valoarea citită este **invers proporțională** cu cantitatea de apă (valoare mare = uscat, valoare mică = umed).
- **Interfață Serială:** Trimite datele în timp real la 9600 baud prin `Serial.println(sensorValue)` pentru monitorizare și calibrare.

Praguri de Comutare:

1. **Sol Uscat ($\text{sensorValue} > 340$):** LED 1 = LOW, LED 2 = LOW (Solul necesită udare).
2. **Umiditate Moderată ($290 < \text{sensorValue} \leq 340$):** LED 1 = HIGH, LED 2 = LOW (Nivel intermediar).
3. **Umiditate Ridicată ($\text{sensorValue} \leq 290$):** LED 1 = HIGH, LED 2 = HIGH (Sol suficient udat).

COD:

```
#define led1 6
#define led2 7
#define sensorPin A0

void setup() {

  Serial.begin(9600);

  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);

  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
}

void loop() {
```



Cofinanțat de
Uniunea Europeană



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

```
int sensorValue = analogRead(sensorPin);

Serial.println(sensorValue);

// ● USCAT
if (sensorValue > 340) {

    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
}

// ○ PUȚINĂ APĂ
else if (sensorValue > 290) {

    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
}

// ● MULTĂ APĂ
else {

    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
}

delay(100);
}
```



Cofinanțat de
Uniunea Europeană



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

PROIECT: Controlul unui Motor Pas cu Pas

Descriere: > Proiectul exemplifică controlul preciziei de rotație a unui motor pas cu pas cu reductor folosind biblioteca standard Stepper.h. Codul calculează numărul exact de pași necesari pentru o rotație completă raportat la raportul de reducere al angrenajului (64: 1) și execută o rutină demonstrativă ce include rotații în sens orar (CW) și antiorar (CCW) la viteze variabile.

2. Descriere Tehnică Detaliată (Raport de Laborator)

Hardware & Calcul Matematic:

- **Motor & Driver:** Motor pas cu pas 28BYJ-48 acționat prin placa driver ULN2003 conectată la pinii digitali 8, 10, 9, 11 (secvență 1-3-2-4).
- **Calcul pași per rotație:**

$$\text{Pași ieșire} = \text{Pași interni (32)} \times \text{Raport reducere (64)} = 2048 \text{ pași/rotație}$$

Rutina de Funcționare (Ciclul loop):

1. **Pas fin (Observare vizuală):** Execută 4 pași la viteză foarte mică (setSpeed(1)) pentru a observa aprinderea secvențială a LED-urilor pe placa driver.
2. **Semirotăție lentă (CW):** Rotește axul la 180° în sens orar la 100 RPM.
3. **Semirotăție rapidă (CCW):** Rotește axul înapoi la 180° în sens antiorar la 700 RPM.

```
//Include the Arduino Stepper Library

#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;

// Number of steps per geared output rotation

const float STEPS_PER_OUT_REV = STEPS_PER_REV * GEAR_RED;

// Define Variables

// Number of Steps Required
```



Cofinanțat de
Uniunea Europeană



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

```
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
    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);
}
```



Cofinanțat de
Uniunea Europeană



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

PROIECT: Actuator Liniar cu Motor Pas cu Pas și Calibrare în Milimetri

Descriere: > Proiectul implementează un sistem de poziționare liniară bazat pe un motor pas cu pas 28BYJ-48. Prin calibrare experimentală (1 rotație = 40 mm), codul convertește mișcarea de rotație în distanță liniară (51.2 pași/mm). Sistemul execută deplasări de câte 50 mm, monitorizează distanța totală și își inversează automat direcția la atingerea bornei de 200 mm.

2. Descriere Tehnică Detaliată (Raport de Laborator)

Calcul Matematic & Calibrare:

- **Rezoluție motor:** 32 pași × 64 (reducere) = 2048 pași/rotație.
- **Conversie Liniară:** O rotație completă mută ansamblul 40 mm (4 cm).
- **Raport de conversie:**

$$\text{Factor Pași} = \frac{2048 \text{ pași}}{40 \text{ mm}} = 51.2 \text{ pași/mm}$$

Funcționare (Ciclul loop):

1. **Pas de deplasare:** Proiectul comanda deplasarea cu câte 50 mm per ciclu ($50 \times 51.2 = 2560$ pași).
2. **Monitorizare & Monitor Serial:** Calculează și afișează distanța acumulat stocată în totalDistance și numărul de pași necesari prin interfața Serială (9600 baud).
3. **Schimbare de sens:** Când totalDistance atinge sau depășește 200 mm, valoarea stepsPerMlmtr este multiplicată cu -1, inversând direcția de rotație a motorului (cursa dus-întors).

COD:

```
#include <Stepper.h>

//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
```



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

```
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,8,10,9,11);


void setup() {

    // put your setup code here, to run once:

    Serial.begin(9600);

}

//one full revolution is equal to 4cm
//the motor is operating at 8.5 volts
//at this voltage is working much better


void loop() {

    // put your main code here, to run repeatedly:

    theMotor.setSpeed(1000);

    //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);*/
```



Cofinanțat de
Uniunea Europeană



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

```
//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

distanceToCover=50; //lets move 50mm

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);

}
```



Cofinanțat de
Uniunea Europeană



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

PROIECT: Actuator Liniar Interactiv cu Motor Pas cu Pas, Buton Start/Stop și Autorevers

Descriere: > Proiectul implementează un sistem de poziționare liniară automatizat, controlat interactiv printr-un buton fizic (comutare Start/Stop cu debouncing). În timpul funcționării, motorul pas cu pas acționează ansamblul pe o cursă maximă de 800 mm (40.960 pași), calculând poziția pas cu pas. La atingerea limitelor electronice de cursă (originea 0 sau valoarea maximă de 800 mm), direcția de deplasare se inversează automat.

2. Descriere Tehnică Detaliată (Raport de Laborator)

Arhitectură Hardware & Logica de Control:

- **Interfață Buton:** Conectat la pinul digital 7 utilizând rezistența internă de pull-up (INPUT_PULLUP). Detectează apăsarea pe front coborâtor și comută starea sistemului (`motorRunning = !motorRunning`) folosind un *delay* de 200 ms pentru debouncing mecanic.
- **Control pas cu pas non-blocant:** Motorul execută doar câte 1 pas per ciclu loop, ceea ce permite citirea continuă a butonului în timp ce motorul este în mișcare.

Limitare Electronică de Cursă (Soft Endstops):

- **Rezoluție liniară:** 51.2 pași/mm.
- **Cursă maximă:** $800 \text{ mm} \times 51.2 = 40.960 \text{ pași}$.
- **Autorevers:** * Când `currentDistance >= maxDistance * 51.2`, direcția devine -1 (înapoi).
 - Când `currentDistance <= 0`, direcția devine 1 (înainte).

COD:

```
#include <Stepper.h>

const int STEPS_PER_REVOLUTION = 2048;

Stepper theMotor(STEPS_PER_REVOLUTION, 8, 10, 9, 11);

// buton

const int buttonPin = 7;

bool motorRunning = false;

bool lastButtonState = HIGH;
```



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

```
int direction = 1;

int currentDistance = 0;

const int maxDistance = 800; // mm

void setup() {

    Serial.begin(9600);

    theMotor.setSpeed(5);

    pinMode(buttonPin, INPUT_PULLUP);

}

void loop() {

    // read button

    bool buttonState = digitalRead(buttonPin);

    // toggle ON/OFF

    if (buttonState == LOW && lastButtonState == HIGH) {

        motorRunning = !motorRunning;

        delay(200);

    }

    lastButtonState = buttonState;

    // run motor

    if (motorRunning) {

        // move only 1 step

        theMotor.step(direction);

        // distance tracking

        currentDistance += direction;

    }

}
```



Cofinanțat de
Uniunea Europeană



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

```
// reverse direction

if (currentDistance >= maxDistance * 51.2) {

    direction = -1;

}

if (currentDistance <= 0) {

    direction = 1;

}

}

}
```



Cofinanțat de
Uniunea Europeană



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

PROIECT: Actuator Liniar Programabil cu Tastatură Matricială 4x4 și Arduino

Descriere: > Proiectul implementează un sistem inteligent de poziționare liniară în care distanța de deplasare (în milimetri) este introdusă direct de utilizator prin intermediul unei tastaturi matriciale 4×4 . Sistemul procesează cifrele introduse în timp real, le afișează pe monitorul serial și pregătește comanda de mișcare pentru motorul pas cu pas (28BYJ-48) la confirmarea utilizatorului.

2. Descriere Tehnică Detaliată (Raport de Laborator)

Componente Hardware & Conexiuni:

- **Tastatură Matricială 4x4:** Liniile pe pinii 2, 3, 4, 5 și Coloanele pe pinii 6, 7, 12, 13.
- **Motor Pas cu Pas (28BYJ-48):** Pinii 8, 10, 9, 11 acționați prin driver ULN2003.
- **Intrare Buton Auxiliar:** Pinul analogic A2 configurat ca intrare digitală (buttonPin).
- **Comunicare Serială:** 9600 baud pentru ghidaj și afișarea valorii tastate.

Logica de Programare:

1. **Preluare taste (customKeypad.getKey()):** Analizează fiecare apăsare de tastă.
2. **Formare număr multi-cifră:** Dacă tasta apăsată este o cifră ('0' – '9'), valoarea anterioară este multiplicată cu 10 și se adaugă noua cifră convertită din ASCII în întreg (key - '0').
3. **Feedback în timp real:** Valoarea parțială introdusă este afișată pe monitorul serial sub forma Typing (mm): X.

COD:

```
#include <Keypad.h>

#include <Stepper.h>

const int button = A2;

int buttonstate=0;

bool ok=0;

const float STEPS_PER_REV = 32;

const float GEAR_RED = 64;
```



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

```
const float STEPS_PER_OUT_REV = STEPS_PER_REV * GEAR_RED;

long StepsRequired;

long inputValue = 0;

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


const byte ROWS = 4;

const byte COLS = 4;

char hexaKeys[ROWS][COLS] = {

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

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

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

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

};

byte rowPins[ROWS] = {2, 3, 4, 5};

byte colPins[COLS] = {6, 7, 12, 13};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);


void setup() {

    Serial.begin(9600);

    Serial.println("Enter a number and press # to confirm.");

    pinMode(button, INPUT);

}


void loop() {

    char key = customKeypad.getKey();
```



Cofinanțat de
Uniunea Europeană



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

```
if (key) {  
    if (key >= '0' && key <= '9') {  
        inputValue = (inputValue * 10) + (key - '0');  
        Serial.print("Typing (mm): ");  
        Serial.println(inputValue);  
    }  
}
```



Cofinanțat de
Uniunea Europeană



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

PROIECT FINAL: Sistem Automat de Detecție a Pachetelor pe Conveior

[Link material video](#)

Acest proiect reprezintă un mini-sistem automatizat de transport și detecție a pachetelor, realizat cu ajutorul plăcii **Arduino**. Sistemul simulează modul în care coletele sunt transportate și detectate pe benzile rulante din fabrici sau depozite logistice.

O bandă transportoare aflată în mișcare poartă pachete de mici dimensiuni, în timp ce un **senzor ultrasonic de distanță** detectează momentul în care un pachet trece prin dreptul său. La detectare, un **buzzer** emite un semnal sonor de avertizare. Microcontrolerul Arduino coordonează întregul sistem și procesează datele primite de la senzor.



Principiu de Funcționare

- **Mișcare continuă:** Banda transportoare se deplasează continuu, fiind acționată de un motor controlat de Arduino.
- **Transport automat:** Pachetele sunt plasate pe bandă și transportate automat către zona de detecție.
- **Monitorizare distanță:** Senzorul ultrasonic este poziționat deasupra/lângă bandă și măsoară în permanență distanța până la obiectele care trec prin dreptul lui.
- **Detecție obiect:** Când un pachet ajunge în dreptul senzorului, valoarea distanței măsurate scade brusc.
- **Semnalizare sonoră:** Placa Arduino procesează modificarea de distanță și activează imediat buzzer-ul.
- **Oprire alarmă:** După ce pachetul trece de senzor, buzzer-ul se oprește automat, iar sistemul revine la starea de monitorizare.
- **Ciclu continuu:** Procesul se repetă pentru fiecare pachet, simulând o linie industrială reală.



Avantaje și Dezavantaje



Cofinanțat de
Uniunea Europeană



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

Avantaje	Dezavantaje
Cost redus & Simplu: Ușor de asamblat și ieftin.	Senzitiv la poziționare: Senzorul necesită aliniere precisă.
Consum mic: Eficiență energetică ridicată.	False citiri: Poate detecta accidental obiecte din apropiere.
Didactic: Excelent pentru învățarea electronicii și programării.	Sensibil la alimentare: Necesită o sursă stabilă de energie.
Extensibil: Se pot adăuga afișaje (LCD), contoare sau LED-uri.	Viteză limitată: Banda nu poate rula la viteze foarte mari.



Aplicații Practice

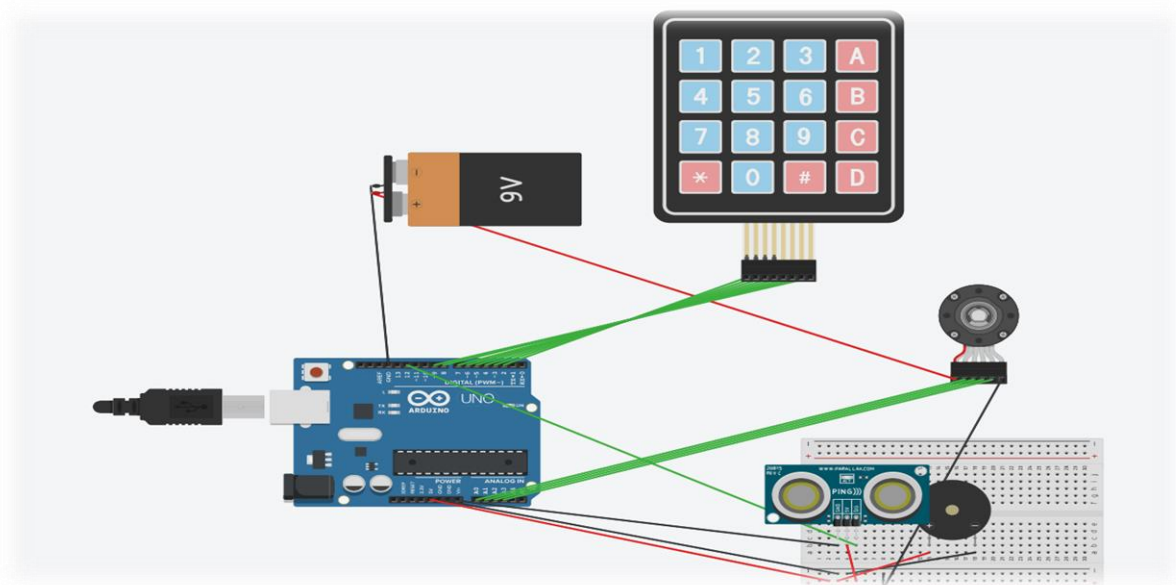
- Automatizări în depozite logistice și curierat.
- Contorizarea mecanică/electronică a pachetelor pe linie.
- Sisteme de transport al bagajelor în aeroporturi.
- Linii de producție și sortare în fabrici (Smart Factory).
- Benzile rulante de la casele de marcat din supermarketuri.



Cofinanțat de
Uniunea Europeană



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



COD:

```
#include <Stepper.h>

#include <Keypad.h>

// ----- STEPPER -----

const int STEPS_PER_REVOLUTION = 2048;

Stepper motor(STEPS_PER_REVOLUTION, A0, A1, A2, A3);

// ----- KEYPAD -----

const byte ROWS = 4;

const byte COLS = 4;

char keys[ROWS][COLS] = {

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



Cofinanțat de
Uniunea Europeană



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

```
{'4','5','6','B'},  
  
{'7','8','9','C'},  
  
{' ','0','#','D'}  
  
};  
  
byte rowPins[ROWS] = {8, 7, 6, 5};  
  
byte colPins[COLS] = {4, 3, 9, 2};  
  
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);  
  
// ----- MOTOR -----  
  
bool moving = false;  
  
long targetSteps = 0;  
  
long currentSteps = 0;  
  
int direction = 1;  
  
float stepsPerMM = 10.0;  
  
String inputNumber = "";  
  
// ----- ULTRASONIC -----  
  
const int trigPin = 12;  
  
const int echoPin = 13;  
  
const int buzzerPin = 11;  
  
float distance = -1;  
  
unsigned long lastSensorRead = 0;  
  
const int sensorInterval = 80;  
  
// ----- FUNCTION -----  
  
float readDistance() {
```



Cofinanțat de
Uniunea Europeană



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

```
digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH, 30000);

if (duration == 0) return -1;

return duration * 0.034 / 2;

}

// ----- SETUP -----

void setup() {

  Serial.begin(9600);

  motor.setSpeed(20);

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  pinMode(buzzerPin, OUTPUT);

  Serial.println("System ready");

}

// ----- LOOP -----

void loop() {

  // =====

  // KEYPAD

  // =====
```



Cofinanțat de
Uniunea Europeană



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

```
char key = keypad.getKey();

if (key) {

    if (key >= '0' && key <= '9') {

        inputNumber += key;

        Serial.println(inputNumber);

    }

    if (key == '*') {

        inputNumber = "";

        Serial.println("Cleared");

    }

    if (key == '#') {

        if (inputNumber.length() > 0) {

            float mm = inputNumber.toFloat();

            targetSteps = mm * stepsPerMM;

            currentSteps = 0;

            moving = true;

            Serial.print("Move mm: ");

            Serial.println(mm);

            inputNumber = "";

        }

    }

}

// =====
```



Cofinanțat de
Uniunea Europeană



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

```
// MOTOR (IMPORTANT: mic delay, nu blocăm senzorul)

// =====

if (moving) {

    motor.step(direction);

currentSteps++;

    if (currentSteps >= targetSteps) {

        moving = false;

        Serial.println("Done!");

    }

}

// =====

// ULTRASONIC (NON-BLOCKING TIMING)

// =====

if (millis() - lastSensorRead >= sensorInterval) {

    lastSensorRead = millis();

    distance = readDistance();

    Serial.print("Distance: ");

    Serial.println(distance);

    if (distance > 0 && distance <= 30) {

        tone(buzzerPin, 300);

    } else {

        noTone(buzzerPin);

    }

}
```



Cofinanțat de
Uniunea Europeană



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

