



Cofinanțat de
Uniunea Europeană



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

PORTOFOLIU DE PRACTICĂ

MOBILITATE CIPRU

10.05.2026 – 23.05.2026

DRAGOMIR THEODOR

LUPU ȘTEFAN

MANEA-IVAN ȘTEFAN

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.”



Cofinanțat de
Uniunea Europeană



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

Cuprins

PROIECT: Controlul unui motor pas cu pas cu clemă alligator.....	2
PROIECT: Conveyor Belt Positioning and Distance Control System	5

PROIECT: Controlul unui motor pas cu pas cu clemă alligator

Obiectivul proiectului:

Crearea unui sistem care controlează mișcarea precisă a unui motor pas cu pas (stepper) ce are atașat o clemă alligator. Sistemul va demonstra controlul vitezei și al direcției de rotație oferind și un feedback vizual prin intermediul LED-urilor de pe placa de driver.

Lista de piese necesare:

- 1 x Placă Arduino Uno
- 1 x Motor pas cu pas (28BYJ-48)
- 1 x Placă de driver pentru motor (ULN2003)
- 1 x Clemă alligator (atașată de axul motorului)
- Fire de conexiune (jumper wires) și conectori

Schema de conectare:

Componentă / Pin Driver	Pin Arduino	Rol / Descriere
IN1	Pin 8 Digital	Semnal fază 1 motor
IN2	Pin 10 Digital	Semnal fază 3 motor (pentru secvențiere corectă)
IN3	Pin 9 Digital	Semnal fază 2 motor
IN4	Pin 11 Digital	Semnal fază 4 motor
GND	GND	Masă comună
VCC / 5V	5V	Alimentare driver logică/motor
Mufă Motor	Conector Driver	Conectarea directă a motorului 28BYJ-48 în placa ULN2003



Cofinanțat de
Uniunea Europeană



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

Codul proiectului:

```
// 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
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);
}
```



Cofinanțat de
Uniunea Europeană



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

```
// Rotate CCW 1/2 turn quickly  
StepsRequired = - STEPS_PER_OUT_REV / 2;  
steppermotor.setSpeed(700);  
steppermotor.step(StepsRequired);  
delay(2000);  
}
```

Exerciții și Provocări:

1. **Modificarea unghiului de rotație:** Modifică variabilele din cod astfel încât clema alligator să facă o rotație completă (o tură completă de 360 de grade) în loc de o jumătate de tură.
2. **Experimentarea cu viteza:** Modifică valoarea din `steppermotor.setSpeed()` în cadrul secvenței rapide. Ce se întâmplă dacă pui o viteză prea mare (de exemplu peste 1000)? Mai are motorul forță să se învârtă sau doar scoate un bâzâit?
3. **Observare vizuală:** Urmărește LED-urile de pe placa driverului ULN2003 în timpul primei secvențe (când viteza este setată la 1). Ce observi la ordinea în care se aprind acestea?

!!! Sfat: Ai grijă la ordinea pinilor în cod! Pentru ca motorul pas cu pas 28BYJ-48 să funcționeze corect cu librăria `Stepper.h`, pinii trebuie declarați în secvența 8, 10, 9, 11 (inversând pinii din mijloc față de ordinea fizică de pe driver). Dacă motorul doar vibrează și nu se învârtă, verifică această ordine!

PROIECT: Conveyor Belt Positioning and Distance Control System

Project Objective:

The creation of a system that controls the precise positioning and displacement of a carrier plate along a conveyor belt using a stepper motor. The system will allow users to either manually jog the plate left/right or input a specific travel distance in millimeters via a 4x4 Keypad, with real-time system feedback displayed on an LCD screen.

Required Components List:

- 1 x Arduino Uno Board
- 1 x Stepper Motor (28BYJ-48 or similar internal 32-step motor)
- 1 x Stepper Motor Driver (ULN2003)
- 1 x 16x2 Character LCD Display (with HD44780 driver or compatible)
- 1 x 4x4 Matrix Keypad
- 1 x Breadboard and Jumper Wires
- 1 x Conveyor belt mechanics with a carrier plate attached to the stepper shaft

Connection Diagram:

Component / Pin	Arduino Pin	Description
LCD RS	Pin A1	Register Select control pin
LCD Enable (E)	Pin A0	Read/Write enable control pin
LCD D4, D5, D6, D7	Pins 13, 12, 11, 10	4-bit data bus pins for screen updates
Stepper Driver IN1	Pin A5	Phase 1 coil signal
Stepper Driver IN2	Pin A3	Phase 3 coil signal (sequenced for library compatibility)
Stepper Driver IN3	Pin A4	Phase 2 coil signal
Stepper Driver IN4	Pin A2	Phase 4 coil signal



Cofinanțat de
Uniunea Europeană



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

Component / Pin	Arduino Pin	Description
Keypad Rows (1, 2, 3, 4)	Pins 9, 8, 7, 6	Keypad row scan outputs
Keypad Columns (1, 2, 3, 4)	Pins 5, 4, 3, 2	Keypad column sense inputs
VCC / GND	5V / GND	Shared power rails for the LCD, Keypad, and Driver

Potential Applications of This Circuit:

- **Automated Dispensing Systems:** Can be used in pharmaceutical or food processing applications to move a container exactly under a nozzle, fill it based on a pre-defined input, and then advance it along the line.
- **Camera Slider / Linear Motion Rig:** Perfect for time-lapse photography or cinematography, allowing filmmakers to input a precise distance and speed for smooth, repeatable automated tracking shots.
- **3D Printers and CNC Components:** Demonstrates the core foundation of Cartesian mechanics, where entering coordinates or numerical steps moves an axis (X, Y, or Z) to a precise geometric location.
- **Automated Sorting Gates:** By expanding the keypad options, this system can route items to different bin stations along a conveyor line based on numeric sorting rules or product SKU codes input by an operator.

Advantages and Disadvantages of This System:

Advantages (Pros)	Disadvantages (Cons)
High Positioning Precision: Stepper motors provide open-loop, accurate incremental steps, eliminating the immediate need for complex position encoders.	Risk of Lost Steps: If the conveyor belt experiences a mechanical jam or heavy load, the motor can stall or skip steps, causing position drifting without a feedback sensor.
User-Friendly Interface: The physical 4x4 keypad combined with the 16x2 LCD offers an intuitive, standalone HMI (Human-Machine Interface) without needing a PC.	Complex Wiring & Pin Consumption: Utilizing a parallel LCD, matrix keypad, and stepper driver consumes nearly all available I/O pins on a standard Arduino Uno.



Cofinanțat de
Uniunea Europeană



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

Advantages (Pros)	Disadvantages (Cons)
Dual-Control Modes: Provides versatility through manual positioning (Jog/Position Mode) for calibration and automatic travel (Distance Mode) for workflow.	Blocking and Low Speed: Stepper motors generally operate slower than DC or servo motors, and the standard Stepper library can introduce blocking delays during execution.

Project Code:

```
#include <Keypad.h>
#include <Stepper.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(A1, A0, 13, 12, 11, 10);

const int INTERNAL_STEPS_PER_REVOL = 32;
float stepsPerMlmtr = 51.2;
Stepper theMotor(INTERNAL_STEPS_PER_REVOL, A5, A3, A4, A2);

const int ROW_NUM = 4;
const int COLUMN_NUM = 4;

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] = {9, 8, 7, 6};
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2};

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

enum Mode { CHOOSE_MODE, POSITION_MODE, DISTANCE_MODE,
EXECUTING_DISTANCE };
Mode currentMode = CHOOSE_MODE;

String inputDistance = "";
String lastState = "";

long stepsRemaining = 0;
```



Cofinanțat de
Uniunea Europeană



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

```
int stepDirection = 1;

void setup(){
  Serial.begin(9600);
  theMotor.setSpeed(1000);

  lcd.begin(16, 2);
  showMainMenu();
}

void loop(){
  char key = keypad.getKey();
  KeyState state = keypad.getState();
  char activeKey = keypad.key[0].kchar;

  if (key == 'C' || (state == PRESSED && activeKey == 'C')) {
    currentMode = CHOOSE_MODE;
    inputDistance = "";
    stepsRemaining = 0;
    lastState = "";
    showMainMenu();
    return;
  }

  if (key == 'A' && currentMode != EXECUTING_DISTANCE) {
    currentMode = POSITION_MODE;
    lastState = "";
    lcd.clear();
    return;
  }

  if (key == 'B' && currentMode != EXECUTING_DISTANCE) {
    currentMode = DISTANCE_MODE;
    inputDistance = "";
    lastState = "";
    lcd.clear();
    return;
  }

  if (currentMode == POSITION_MODE) {
    if (state == PRESSED || state == HOLD) {
      if (activeKey == '4') {
        if (lastState != "LEFT") {
          lcd.clear();
          lcd.print("Mode: Position");
          lcd.setCursor(0, 1);
          lcd.print("<- LEFT");
          lastState = "LEFT";
        }
        theMotor.step(-5);
      }
    }
  }
}
```



Cofinanțat de
Uniunea Europeană



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

```
}
else if (activeKey == '6') {
    if (lastState != "RIGHT") {
        lcd.clear();
        lcd.print("Mode: Position");
        lcd.setCursor(0, 1);
        lcd.print("RIGHT ->");
        lastState = "RIGHT";
    }
    theMotor.step(5);
}
}
else {
    if (lastState != "IDLE") {
        lcd.clear();
        lcd.print("Mode: Position");
        lcd.setCursor(0, 1);
        lcd.print("4<-|6-> [C]Exit");
        lastState = "IDLE";
    }
}
}

else if (currentMode == DISTANCE_MODE) {
    if (lastState != "INPUT_WAIT") {
        lcd.setCursor(0, 0);
        lcd.print("Dist(mm) [C]Cancl");
        lcd.setCursor(0, 1);
        lcd.print("-> " + inputDistance + "    ");
        lastState = "INPUT_WAIT";
    }

    if (key >= '0' && key <= '9') {
        if (inputDistance.length() < 4) {
            inputDistance += key;
            lastState = "";
        }
    }
    else if (key == '*') {
        inputDistance = "";
        lastState = "";
    }
    else if (key == '#') {
        if (inputDistance.length() > 0) {
            int mmToMove = inputDistance.toInt();

            lcd.clear();
            lcd.print("Moving...");
            lcd.setCursor(0, 1);
        }
    }
}
```



Cofinanțat de
Uniunea Europeană



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

```

    lcd.print(inputDistance + " mm [C]Stop");

    stepsRemaining = (long)mmToMove * stepsPerMlmtr;
    stepDirection = 1;

    currentMode = EXECUTING_DISTANCE;
    lastState = "";
}
}
}

else if (currentMode == EXECUTING_DISTANCE) {
    if (stepsRemaining > 0) {
        int pasiCurenti = (stepsRemaining > 5) ? 5 :
stepsRemaining;

        theMotor.step(pasiCurenti * stepDirection);
        stepsRemaining -= pasiCurenti;
    }
    else {
        lcd.clear();
        lcd.print("Movement Done!");
        delay(1500);

        currentMode = DISTANCE_MODE;
        inputDistance = "";
        lastState = "";
    }
}
}

void showMainMenu() {
    lcd.clear();
    lcd.print("System Ready");
    lcd.setCursor(0, 1);
    lcd.print("Press A or B");
}

```

Exercises and Challenges:

1. **Adjusting the Calibration:** Look at the global variable stepsPerMlmtr. Test the conveyor belt by executing a 100 mm run. Measure the actual physical distance traveled with a ruler. If it did not travel exactly 100 mm, adjust the stepsPerMlmtr variable value to calibrate your unique belt setup.



Cofinanțat de
Uniunea Europeană



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

2. **Direction Inversion:** In `DISTANCE_MODE`, the system currently defaults to a positive step direction (`stepDirection = 1`). Can you modify the system logic so that if a user hits the D key before pressing #, the conveyor runs the calculated distance in the opposite direction (`stepDirection = -1`)?
3. **Emergency Stop Optimization:** Pressing C resets the state machine back to the main menu. Open your Serial Monitor (`Ctrl+Shift+M`) and debug what happens if you press C while the machine is actively in `EXECUTING_DISTANCE`. Does the motor halt instantly? Explain how splitting steps into chunks of 5 helps maintain keypad responsiveness.

!!! Hint: Stepper motors draw significantly higher current when moving under load. Avoid drawing power for the stepper motor directly from the Arduino's 5V rail if your carrier plate is heavy, as it could cause the Arduino to brown out or reset. Always ensure your driver board grounds are tied together with your Arduino ground.