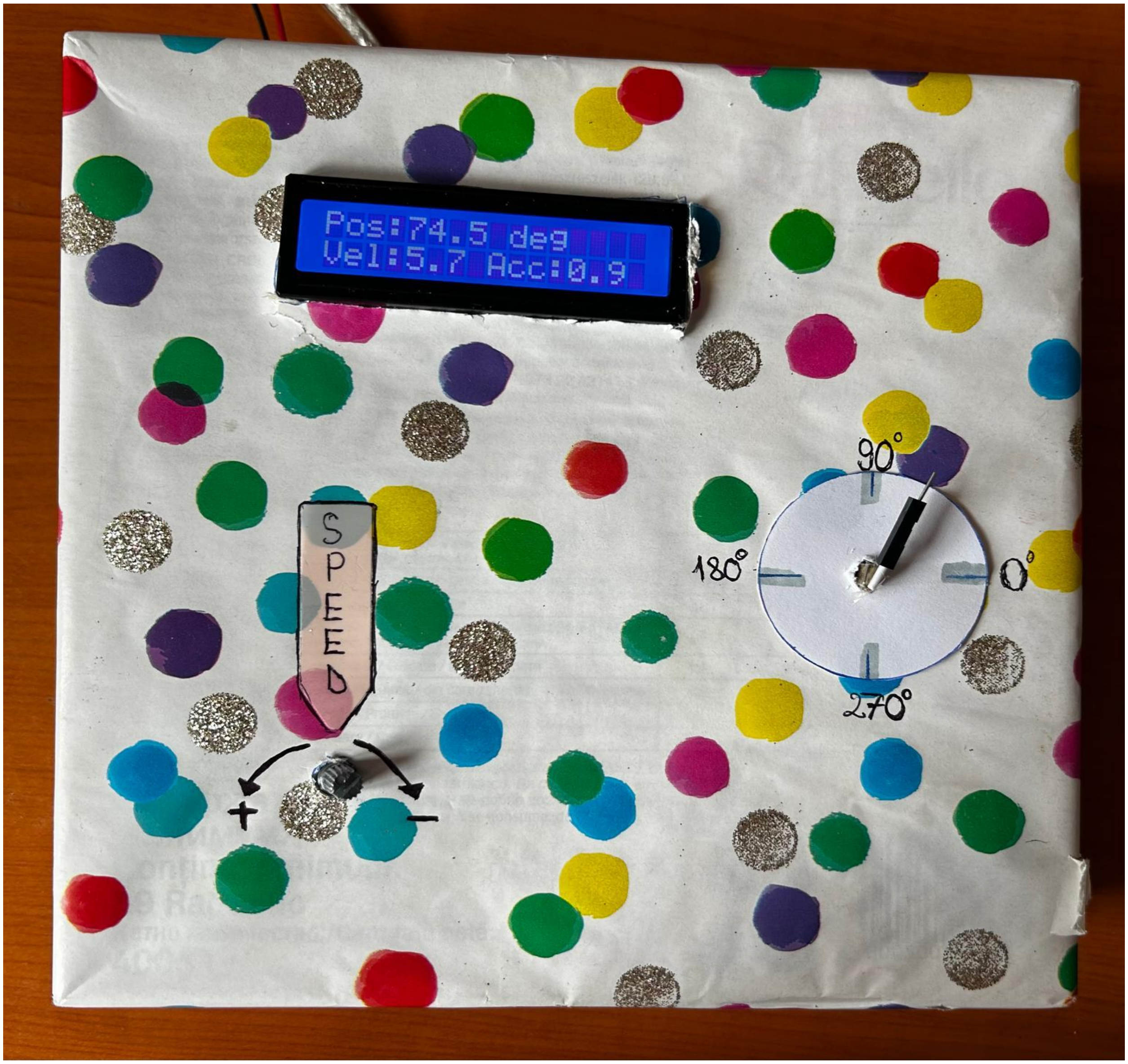
Stepper Motor Control with Real-Time Data Display



Student: Olar Paula-Maria

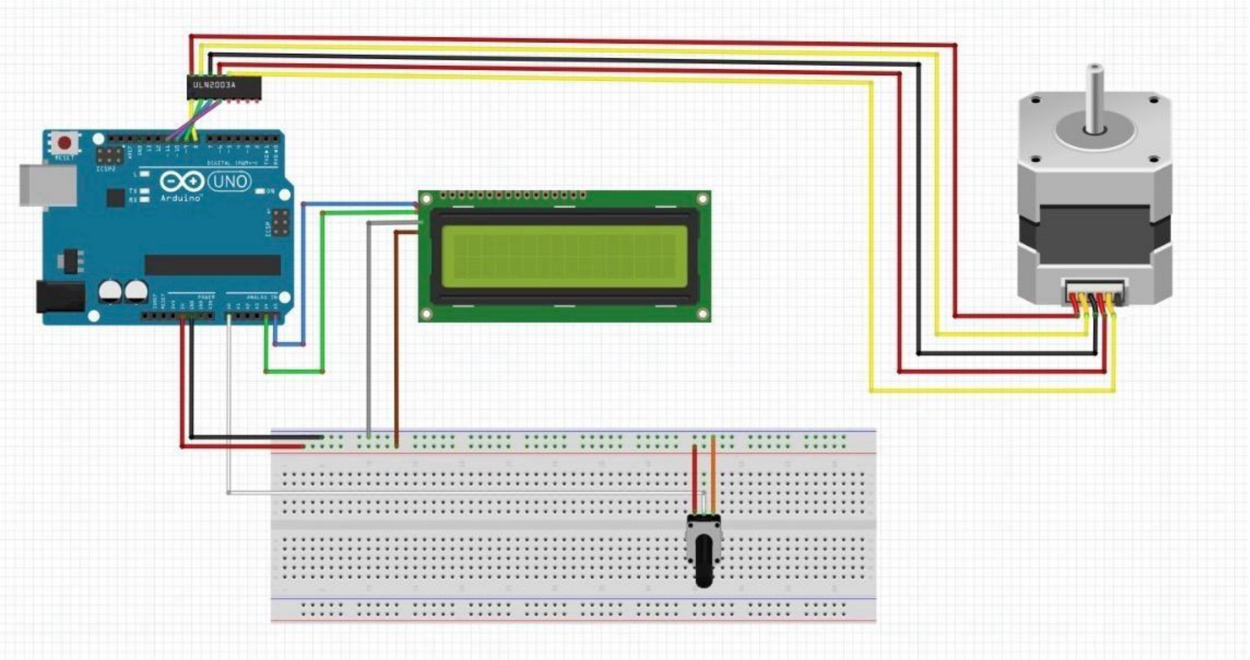
General Description

This project involves a **microcontroller-based system** that controls a stepper motor to perform a defined number of rotations and then return to its initial position. The system includes a **user interface with an LCD display** that shows **real-time angular position, rotation speed, and acceleration**. Additionally, the motor speed can be adjusted using a potentiometer.

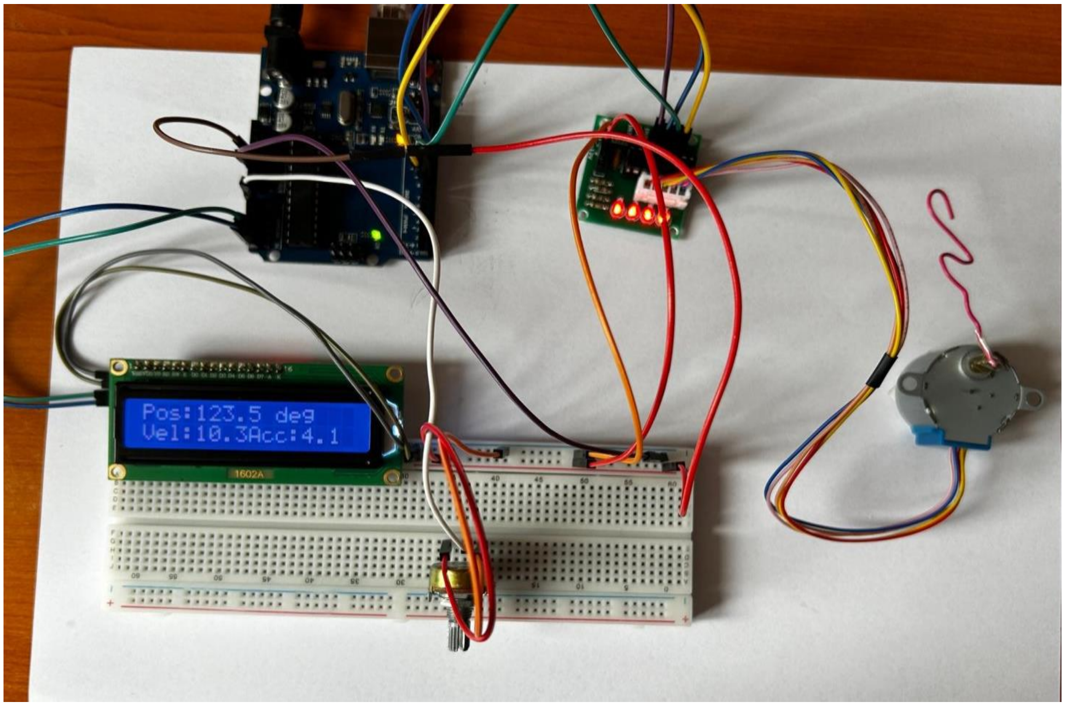
Components Used

1. **Microcontroller (e.g., Arduino)** – controls the motor and user interface.
2. **Stepper Motor 28BYJ-48 + ULN2003 Driver** – provides precise movement control.
3. **16x2 I2C LCD Display** – shows real-time motor parameters.
4. **Potentiometer** – allows the user to adjust the motor speed.
5. **Power Supply** – 5V for the motor and microcontroller.

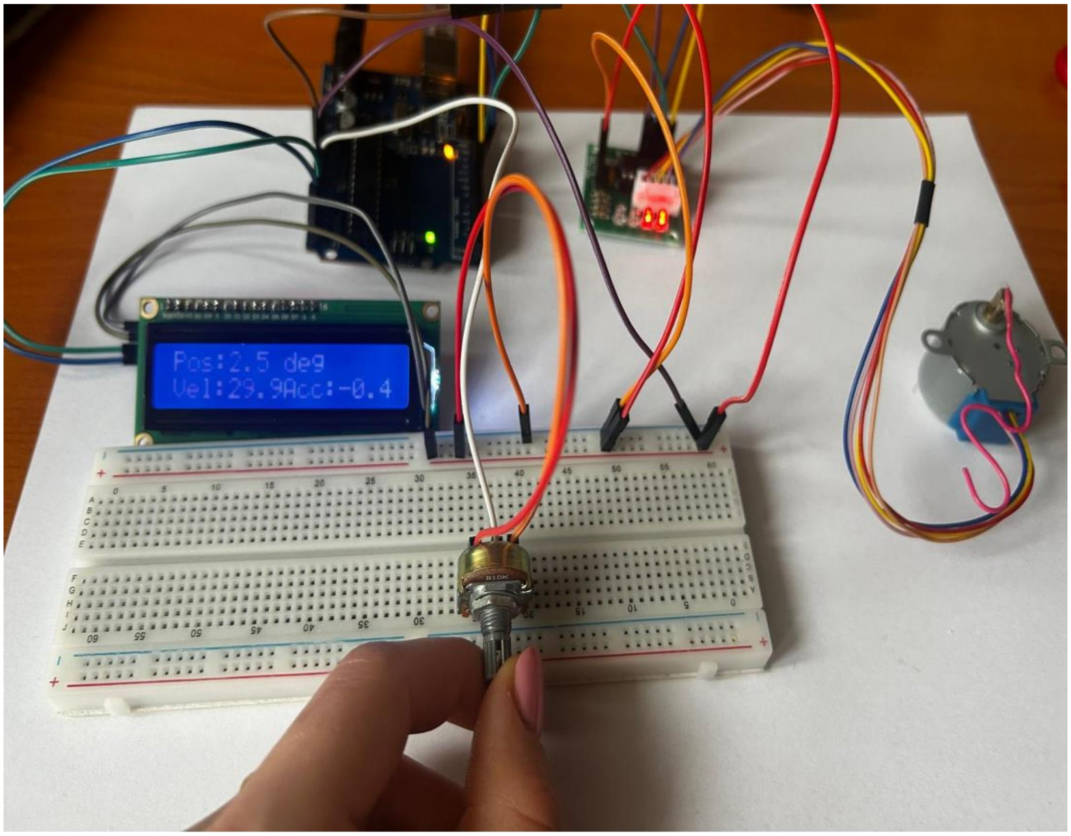
The fritzing



The system assembley



```
Pos:123.5 deg
Vel:10.3Acc:4.1
```



```
Pos:2.5 deg
Vel:29.9Acc:-0.4
```

Operation Flow

1. The system initializes the **LCD display and motor settings** upon startup.
2. The stepper motor performs **a defined number of turns forward** followed by **one reverse rotation** back to its home position.
3. During rotation, the system continuously calculates and displays:
   - **Current angular position** in degrees.
   - **Velocity** in degrees per second.
   - **Acceleration** to track motion dynamics.
4. The user can adjust the **motor speed in real-time** using the potentiometer.
5. The data is displayed both on the **LCD** and **Serial Monitor** for real-time monitoring.

The source code:

```
#include <Stepper.h>
#include <LiquidCrystal_I2C.h>
#define POT_PIN A0
int stepsPerRevolution = 2048;
int minRPM = 1;
int maxRPM = 30;
int rpm = 10;
int numberOfRotations = 2;

float angularPosition = 0;
float angularVelocity = 0;
float angularAcceleration = 0;
float previousAngularVelocity = 0;
float alphaFilter = 0.2;
```

```
unsigned long lastStepTime = 0;
unsigned long currentTime = 0;
unsigned long lastReadTime = 0;
const int readInterval = 500;

LiquidCrystal_I2C lcd(0x27, 16, 2);
Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  myStepper.setSpeed(rpm);
  executeRotations();
}

void loop() {
}

void executeRotations() {
  for (int i = 0; i < numberOfRotations; i++) {
    executeOneRotation(1);
  }
  delay(1000);
  executeOneRotation(-1);
  Serial.println("Start position reached");
}

void executeOneRotation(int direction) {
  for (int stepCount = 0; stepCount < stepsPerRevolution; stepCount++) {
    myStepper.step(direction);
    updatePositionAndSpeed(direction);
    if (millis() - lastReadTime >= readInterval) {
      readPotentiometer();
      lastReadTime = millis();
    }
  }
  delay(500);
}

void readPotentiometer() {
  int potValue = analogRead(POT_PIN);
  rpm = map(potValue, 0, 1023, minRPM, maxRPM);
  myStepper.setSpeed(rpm);

  displayData();
}

void updatePositionAndSpeed(int direction) {
  currentTime = micros();
  unsigned long deltaTime = currentTime - lastStepTime;

  if (deltaTime > 0) {
    previousAngularVelocity = angularVelocity;
    angularVelocity = (360.0 / stepsPerRevolution) / (deltaTime / 1000000.0);

    float rawAcceleration = (angularVelocity - previousAngularVelocity) / (deltaTime /
1000000.0);
```

```
        angularAcceleration = angularAcceleration * (1 - alphaFilter) + rawAcceleration * alphaFilter;
    }

    angularPosition += direction * (360.0 / stepsPerRevolution);

    if (angularPosition >= 360.0) {
        angularPosition -= 360.0;
    } else if (angularPosition < 0.0) {
        angularPosition += 360.0;
    }

    lastStepTime = currentTime;
}

void displayData() {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Pos:");
    lcd.print(angularPosition, 1);
    lcd.print(" deg");

    lcd.setCursor(0, 1);
    lcd.print("Vel:");
    lcd.print(angularVelocity, 2);
    lcd.print(" d/s");

    lcd.setCursor(8, 1);
    lcd.print("Acc:");
```

```
  lcd.print(angularAcceleration, 2);

  Serial.print("Position: ");
  Serial.print(angularPosition, 1);
  Serial.print("° | Velocity: ");
  Serial.print(angularVelocity, 2);
  Serial.print("°/s | Acceleration: ");
  Serial.print(angularAcceleration, 2);
  Serial.print("°/s² | RPM: ");
  Serial.println(rpm);
}
```