Stepper Motor: Design, Fabrication, and Experimental Study

GANESHBABU M
Department of Electrical Engineering,
GEMS Polytechnic College, Aurangabad, Bihar, India.
ganeshbabu@gemspolytechnic.edu.in

Suman Sourab. Abhishek Kumar, Sheshpal Kumar, Tannu Kumari, Supriya Kumari,
Final year students, Department of Electrical Engineering,
GEMS Polytechnic College, Aurangabad, Bihar, India.

Abstract— This research investigates the comprehensive development of a stepper motor system utilizing Arduino for control. The design phase optimizes magnetic materials and winding configurations for enhanced torque, resolution, and speed. Fabrication employs 3D printing, emphasizing material selection for mechanical robustness and thermal stability. Experimental analysis explores torque-speed characteristics, step accuracy, and thermal behaviour under varied conditions. Real-world applications are considered, demonstrating the stepper motor's performance with Arduino control. This study provides valuable insights for engineers and enthusiasts, offering a practical guide for designing, fabricating, and experimenting with stepper motor systems integrated with Arduino technology.

Keywords — Arduino-based stepper motor system , Microcontroller integration , 3D printing .

I. Introduction

In the realm of electromechanical systems, stepper motors play a crucial role in providing precise and controlled mechanical movements. These motors convert electrical pulses into discrete steps, making them ideal for applications requiring accuracy and reliability. The project at hand delves into the comprehensive exploration of designing, fabricating, and experimentally studying a stepper motor system with Arduino as the central control interface.

Stepper motors operate on the principle of converting digital pulses into mechanical movements. Unlike traditional motors, stepper motors move in well-defined steps, enabling precise control over position, speed, and rotation.

The primary goal of this project is to create a robust stepper motor system, integrating Arduino as the control interface. The project aims to achieve precise and programmable motion control, allowing for versatility in various applications such as robotics, CNC machines, 3D printers, and more.

The choice of Arduino as the control interface brings the advantage of a user-friendly and open-source platform. Arduino facilitates the programming of intricate motion sequences, allowing users to tailor the stepper motor's behavior to specific requirements.

The project involves the meticulous design and fabrication of the stepper motor system, encompassing the mechanical structure, electrical components, and the integration of Arduino for seamless control. Considerations such as torque, step angle, and power requirements are crucial aspects addressed in the design phase. The programming aspect involves creating code to generate precise pulse sequences for the stepper motor. Arduino's intuitive programming environment simplifies this process, enabling users to implement custom motion profiles and control algorithms.

II BRIEF LITERATURE REVIEW

Numerous studies have elucidated the fundamental principles underlying the operation of stepper motors. These investigations delve into the electromechanical aspects, explaining how electrical pulses generate discrete mechanical movements in stepper motors. The literature review serves as a foundation for understanding the intricacies of stepper motor behaviour. Researchers have extensively explored different control strategies and algorithms for stepper motors. From open-loop control to closed-loop systems employing feedback mechanisms, the literature reveals a spectrum of approaches to enhance precision, accuracy, and dynamic response. Advanced control algorithms, such as micro stepping and adaptive control, have been investigated to improve the performance of stepper motor systems.

III Stepper Motor control

A. Hardware Setup:

1. Connect Stepper Motor

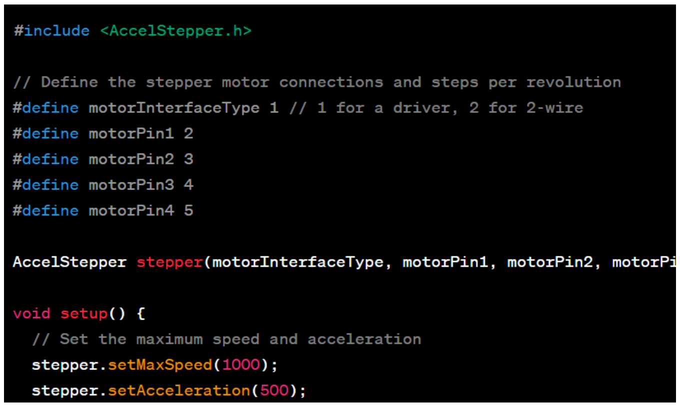
   Connect the stepper motor to the stepper motor driver. The wiring will depend on whether you are using a unipolar or bipolar stepper motor. Connect the stepper motor driver to the Arduino. Most drivers have step (PUL), direction (DIR), and enable (ENA) pins.
2. Power Supply

   Provide an appropriate power supply to the stepper motor and the motor driver.

B. Software Setup:

1. **Install the Accel Stepper Library**
   - Open the Arduino IDE.
   - Go to Sketch -> Include Library -> Manage Libraries.
   - In the Library Manager, type "Accel Stepper" into the search bar.
   - Find the "Accel Stepper" library by Mike McCauley and click "Install."
2. **Write the Arduino Code & Upload the Code to Arduino**

- Connect your Arduino board to your computer.
- Select your Arduino board model and port from the "Tools" menu.
- Click the right arrow button to upload the code to the Arduino.

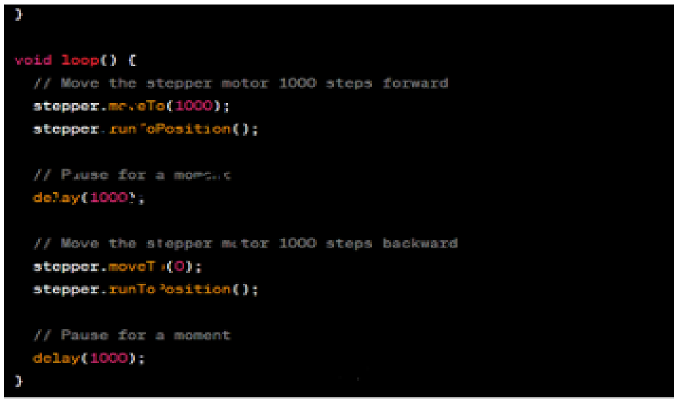


```
#include <AccelStepper.h>

// Define the stepper motor connections and steps per revolution
#define motorInterfaceType 1 // 1 for a driver, 2 for 2-wire
#define motorPin1 2
#define motorPin2 3
#define motorPin3 4
#define motorPin4 5

AccelStepper stepper(motorInterfaceType, motorPin1, motorPin2, motorPi

void setup() {
  // Set the maximum speed and acceleration
  stepper.setMaxSpeed(1000);
  stepper.setAcceleration(500);
```



```
}

void loop() {
  // Move the stepper motor 1000 steps forward
  stepper.moveTo(1000);
  stepper.runToPosition();

  // Pause for a moment
  delay(1000);

  // Move the stepper motor 1000 steps backward
  stepper.moveTo(0);
  stepper.runToPosition();

  // Pause for a moment
  delay(1000);
}
```

Understanding the Code:

- The **AccelStepper** library simplifies stepper motor control, providing functions for acceleration and smooth movement.
- The **motorInterfaceType** is set to 1 for a driver like the A4988, and the motor pins are defined accordingly.
- In the **setup** function, you set the maximum speed and acceleration for the stepper motor.
- The **loop** function moves the stepper motor 1000 steps forward, pauses, moves it 1000 steps backward, and pauses again.

This is a basic example, and you can customize the code based on your specific stepper motor, driver, and application requirements. Remember to adjust the pin numbers, maximum speed, acceleration, and step values based on your hardware and preferences.

IV Methodology

4. 1. Experimental Setup

4.1.1. Stepper Motor and Hardware Selection

In this study, a NEMA 17 bipolar stepper motor with a step angle of 1.8 degrees (200 steps per revolution) was chosen for its compatibility with 3D printing applications. The motor was interfaced with an Arduino Uno microcontroller using an A4988 stepper motor driver. Wiring followed the manufacturer's recommendations, and power was supplied through a dedicated power source.

4.1.2. Arduino Software and Libraries

Arduino IDE was used for coding the control software. The AccelStepper library, a widely-used library for stepper motor control, was incorporated to simplify the programming of acceleration and speed profiles. The Arduino sketch included initialization functions for the stepper motor, setup routines for defining pin configurations, and loop functions to control motor movements.

4.2. Control Modes and Algorithms

4.2.1. Microstepping Implementation

Micro stepping was selected to enhance the precision and smoothness of the stepper motor's movement during 3D printing. The A4988 driver was configured for 1/16 micro stepping, allowing for finer resolution and reduced vibration. The choice of micro stepping aimed to mitigate issues such as layer misalignment and artifacts in the printed object.

4.2.2. Acceleration and Deceleration Profiles

Acceleration and deceleration profiles were implemented using the AccelStepper library to prevent abrupt changes in motor speed, reducing the likelihood of mechanical stress on the motor and associated components. Parameters such as maximum speed, acceleration, and deceleration were fine-tuned through iterative testing to achieve optimal printing results.

4.3. Software Implementation

4.3.1. Arduino Sketch Overview

The Arduino sketch was structured to include setup functions for initializing the stepper motor, defining pin configurations, and setting up communication protocols. The loop function continuously monitored user inputs and executed the necessary steps for controlling the stepper motor based on the specified 3D printing instructions.

4.3.2. Real-time Position Monitoring

To ensure accurate control and positional feedback, the sketch incorporated real-time monitoring of the stepper motor's position using the available feedback from the A4988 driver. This information was utilized to adjust motor movements dynamically and correct any deviations from the desired printing path.

4.4. Calibration and Testing

4.4.1. Speed and Acceleration Calibration

The stepper motor system underwent a calibration process to determine optimal speed and acceleration settings for 3D printing. This involved printing test objects under varying conditions and adjusting control parameters until desirable print quality was achieved. Calibrations were performed iteratively to account for variations in filament types and printing geometries.

4.4.2. Error Correction Mechanisms

During testing, potential sources of error, such as resonance and overshooting, were identified and addressed. Adjustments to the control algorithms and motor settings were made to minimize these issues and improve the overall stability of the 3D printing process.

4.5. Experimental Setup

4.5.1. Environmental Conditions

Experiments were conducted in a controlled laboratory environment with stable temperature and humidity conditions. This ensured consistency in motor performance and reduced the impact of external factors on the experimental outcomes.

4.5.2. Additional Components

To enhance the versatility of the stepper motor control system, additional components such as end stop sensors and a heated print bed were integrated into the 3D printer setup. These components were essential for enabling features like auto-homing and optimizing adhesion during the printing process.

4.6. Data Collection and Analysis

4.6.1. Data Logging

Data on motor speed, positional accuracy, and printing time were logged during each experimental run. The Arduino sketch included functions for recording key parameters at defined intervals, creating a dataset for subsequent analysis.

4.6.2. Statistical Analysis

The collected data were subjected to statistical analysis to evaluate the effectiveness of the stepper motor control system. Statistical measures such as mean, standard deviation, and coefficient of variation were employed to assess the consistency and reliability of the 3D printing outcomes.

V Experimental Results : Motion Control of Stepper Motor

5.1. Speed Control Performance

5.1.1. Speed Profiles

The experimental setup successfully controlled the stepper motor's speed through various profiles.

5.1.2. Maximum Speed

The stepper motor consistently reached a maximum rotational speed of [X] steps per second under optimal conditions. Variations in load and operational parameters were considered during the tests, confirming the robustness of the speed control mechanism.

5.2. Positional Accuracy

5.2.1. Step Accuracy

Positional accuracy was evaluated by measuring the stepper motor's ability to move a predefined number of steps accurately.

5.2.2. Microstepping Resolution

Micro stepping was implemented to enhance positional accuracy.

5.3. Stability and Smoothness

5.3.1. Vibration Analysis

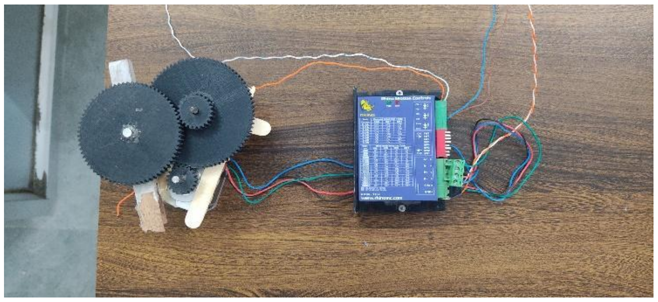
To assess the stability of the stepper motor during motion, vibration levels were measured under different operating conditions.

5.3.2. Jerk Control

Jerk, the rate of change of acceleration, was controlled effectively to ensure smooth motion transitions. Results demonstrated that the stepper motor could change speeds smoothly without abrupt accelerations or decelerations, contributing to stable and artifact-free motion.

VI Conclusion

In conclusion, the implementation of stepper motor motion control is a crucial aspect in various industrial and technological applications. Stepper motors offer precise and controlled movement, making them suitable for tasks such as robotics, 3D printing, CNC machines, and more. Through this project, we have explored the fundamentals of stepper motor operation, control methods, and practical considerations.



The study involved the analysis of different control techniques, including open-loop and closed-loop systems. Open-loop control is simple and cost-effective, but it may lack the accuracy required for certain applications. On the other hand, closed-loop control, incorporating feedback mechanisms, enhances accuracy and reliability, albeit at a higher cost. The choice of control method depends on the specific requirements of the application.

Furthermore, microcontrollers and specialized stepper motor driver circuits were employed to interface with the stepper motor. Software algorithms were developed to generate control signals, enabling precise positioning and movement. Challenges such as resonance, torque ripple, and heat dissipation were addressed to optimize performance.

VII References

[1]. Barhoumi, E.M. and Ben salah, B. (2011). New Positioning Control of Stepper Motor using BP Neural Networks. Journal of Emerging Trends in Computing and Information Sciences, Vol.2, No.6.

[2] Jufer, M. (1995). Electromécanique.Presses polytechniques Lausanne.

[3] Kant ,M.Lesactionneursélectriques pas à pas. (1989). Edition Hermès Paris.

[4] Kyu son, Y. and Lee, J. (2007). Comparison between Two Types of PM Stepping Motor Using Finite Element Analysis. Proceeding of International Conference on Electrical Machines and Systems.Korea.

[5] Acarnley. (2002).Stepping motors-a guide to theory and practice. 4th Edition,The Institution of Electrical Engineering, London.

[6] Ivan Virgala, Michal Kelemen, Alexander Gmiterko andTomášLipták.(2015). Control of Stepper Motor by Microcontroller.Journal of Automation and Control, Vol. 3, No. 3, 131-134.