The object of this study is a large-scale Cartesian 3D printer with a build volume of 2 × 2 × 2 meters, equipped with BLDC servo motors on the X and Y axes, a NEMA 34 stepper motor on the Z axis, and a NEMA 17 stepper motor for filament extrusion. The main problem addressed is the limitation of existing firmware, which only supports bipolar stepper motors (NEMA standard), making it unsuitable for achieving the higher extruder speeds required in large-format printing. To overcome this issue, a new firmware was designed to support brushless direct current (BLDC) servo motors, enabling faster, more stable, and more precise motion control.

The developed firmware, based on the ESP32 microcontroller, processes G-code instructions step by step, calculates motor commands, and communicates them to BLDC motor drivers (FSESC 4.12 on the VESC platform) with encoder feedback for accurate positioning. Experimental results demonstrated reliable synchronization of the X and Y axes across different speeds and distances, with average positioning errors of 0.9% on the X-axis and 1.03% on the Y-axis. The system operated stably within a rotational speed range of F8000-F54000 (1000–7000 RPM) and a printing speed range of 92–800 mm/s, confirming the firmware's capability to handle both low- and high-speed operations.

The interpretation of the results provides superior acceleration, precision, and stability compared to stepper-motor-based systems. A distinctive feature of this work lies in adapting G-Code step execution for BLDC motors, which has not been widely implemented in existing 3D printing firmware. The practical significance of the results is the potential application of this firmware in large-scale and high-speed 3D printers. This makes the approach highly relevant for advanced manufacturing industries where precision, scalability, and efficiency are critical

Keywords: large-scale 3D printing, step-by-step G-code execution, brushless direct current servo firmware, ESP32 motion control, high-speed additive manufacturing

UDC 621.9.048:681.586:004.932
DOI: 10.15587/1729-4061.2025.341225

IDENTIFICATION OF 3D PRINTER FIRMWARE APPLYING BRUSHLESS DIRECT CURRENT MOTOR SERVO WITH STEP G-CODE READING AND EXECUTING METHOD

Budhy Setiawan
Corresponding author
Professor, Master of Electrical Engineering*
E-mail: budhy.setiawan@polinema.ac.id
Indrazno Siradjuddin
Doctor, Master of Electrical Engineering*
Resti Dyah Ayu Retno Palupi
Student, Master of Electrical Engineering*
*Department of Electric Engineering
State Polytechnic of Malang
Soekarno Hatta str., 9, Malang, Indonesia, 65141

Received 23.07.2025
Received in revised form 22.09.2025
Accepted 01.10.2025
Published 30.10.2025

How to Cite: *Setiawan, B., Siradjuddin, I., Palupi, R. D. A. R. (2025). Identification of 3D printer firmware applying brushless direct current motor servo with step G-code reading and executing method. Eastern-European Journal of Enterprise Technologies, 5 (1 (137)), 111–123. https://doi.org/10.15587/1729-4061.2025.341225*

1. Introduction

Most contemporary 3D printers utilize standard NEMA bipolar stepper motors to drive the *X*, *Y*, and *Z* axes within a Cartesian mechanical system, enabling precise object fabrication based on digital models [1]. This technology is an evolution of CNC systems and plays a vital role in modern manufacturing automation, particularly within the framework of Industry 4.0 [2, 3].

3D printing operates by converting CAD models into G-code, a set of machine instructions that control motor movements layer by layer [4]. Nevertheless, stepper motors present inherent drawbacks, particularly step loss at higher operating speeds, which compromises accuracy and stability [5]. In addition, limitations in factory-installed motor drivers have been shown to negatively affect print quality, further reducing efficiency [6]. To address these challenges, alternative approaches such as brushless direct current (BLDC) servo systems and advanced control algorithms have been investigated, offering improved speed, precision, and reliability in extruder and motion control [7, 8]. In contrast, BLDC servo motors provide faster response, stable torque, and higher efficiency, and are widely adopted in robotic applications [9], their application in large-scale 3D printers remains limited, primarily due to the lack of compatible firmware [7]. Existing firmware is mostly proprietary and hardware-dependent, which restricts adaptability to servo-based systems [10]. In this context, several issues have been identified, such as the need for G-code recompilation and optimization for higher printing speed [11], firmware vulnerabilities that compromise system security [12], and design constraints in adapting firmware for non-standard 3D printer architectures [13]. Further challenges include secure firmware updating for embedded systems [14] and limitations that affect object authentication and overall print reliability [15].

Furthermore, conventional BLDC systems rely on Hall effect sensors, which increase cost, physical size, and the risk of sensor failure [7]. Firmware development for BLDC motors has also lagged behind, showing slower progress and lower modernization compared to other software domains [16].

Therefore, research on firmware development for BLDC-based motion control in large-format 3D printers is relevant and essential to advance high-speed and precision additive manufacturing technologies.

2. Literature review and problem statement

3D printing operates by converting CAD models into G-code, a set of instructions that control motor movements layer by layer [4]. However, stepper motors present inherent drawbacks, particularly step loss at higher operating speeds, which compromises accuracy and stability [5]. Improvements to address these limitations have been explored, such as replacing factory-installed stepper motor drivers to improve print quality [6] and introducing advanced motion algorithms like the S-trapezoid method. Despite these efforts, stepper-based systems remain constrained by open-loop instability and limited scalability.

Alternative approaches have investigated BLDC servo motors, which provide faster response, stable torque, and higher efficiency [7]. Their potential has also been expanded through sensorless methods to reduce hardware complexity [8]. In addition, earlier works demonstrated improvements in firmware programming for BLDC and stepping motors [9]. While these systems have been widely adopted in robotics, their application in large-format 3D printing remains limited due to the lack of compatible firmware [10].

Additional challenges have been identified in firmware optimization and security. G-code recompilation for higher speed was addressed in [11]. The security vulnerabilities of printer firmware were highlighted in [12]. Adaptation to non-standard architectures was discussed in [13]. Secure updating for embedded systems was presented in [14]. The importance of authentication for object reliability was shown in [15]. Modernization of firmware technologies for IoT terminals was explored in [16].

Low-cost CNC and 3D printing platforms continue to rely on microcontroller-based stepper control with simplified firmware [17]. Similar approaches were also described in [18]. These systems, while affordable, are hindered by reduced accuracy at higher speeds, poor scalability, and low efficiency. Expanding the scope of additive manufacturing, [19] demonstrated synchronous five-axis 3D printing to strengthen part structures. In a similar direction, [20] presented the design and implementation of an FDM-based 3D printer for cost-effective production. However, these works still relied on conventional stepper-driven motion and did not integrate BLDC servo firmware, thus limiting performance for high-speed, large-scale applications.

Another gap exists in G-code execution and motion synchronization. Conventional firmware typically processes G-code in bulk or relies on predefined motion planning. The inefficiency of this approach in high-speed printing was also confirmed in. While optimization at the slicing stage has been studied in [21], and further analysis of CAD-to-G-code transfer was discussed in [22], such approaches do not extend to firmware-level synchronization of multi-axis motion using BLDC actuators. Open-loop implementations have shown positional drift and synchronization errors [23]. However, embedded closed-loop solutions with encoder feedback remain underexplored.

3. The aim and objectives of the study

The aim of this study is to develop firmware for a large-scale 3D printer that employs BLDC servo motor actuators for the X and Y axes, as an alternative to conventional NEMA stepper motor systems. This will allow faster and more precise extruder movement, thereby improving the performance and applicability of large-format additive manufacturing.

To achieve this aim, the following objectives were accomplished:

– to design and implement firmware that supports BLDC servo motors, focusing on positioning accuracy, speed range, and movement stability in large-scale printing operations;

– to develop a mathematical model that links G-code feed rate, motor RPM, and displacement, and to validate its consistency with experimental results as a theoretical foundation for firmware motion control;

– to evaluate and validate axis synchronization under step-by-step G-code execution using encoder-based feedback;

– to conduct accuracy analysis by comparing commanded trajectories with measured trajectories at different speeds and distances;

– to analyze performance stability printing conditions, identifying potential limitations and improvement strategies for future work.

4. Materials and methods

4. 1. The object and hypothesis of the study

The object of this study is a large-scale Cartesian 3D printer with a build volume of $2\times2\times2$ meters, equipped with BLDC servo motors on the X and Y axes, a NEMA 34 stepper motor on the Z axis, and a NEMA 17 stepper motor for filament extrusion. This machine serves as the experimental platform on which the proposed firmware is implemented and tested.

The central hypothesis asserts that a control architecture explicitly optimized for BLDC servo motor integration can achieve significant enhancements in velocity, positional accuracy, and dynamic stability (particularly in large-format additive manufacturing) relative to conventional bipolar stepper motor-based systems.

This study is conducted under several assumptions. First, the G-code files generated by slicing software such as Simplify3D are assumed to be valid and accurate representations of the 3D models. Second, the mechanical system is assumed to be free from major structural issues, such as excessive backlash or mechanical vibration. The scope of the study is simplified by focusing exclusively on controlling the X and Y axes using BLDC motors, while the Z axis and extruder remain operated by stepper motors. This simplification is considered sufficient for evaluating the effectiveness of BLDC-based horizontal motion control.

4. 2. Firmware design

Motion on the X and Y axes uses Flipsky 6374 BLDC motors, the Z axis employs a NEMA 34 stepper motor, while filament extrusion is driven by a NEMA 17. The BLDC motors are interfaced through VESC FSESC 4.12 controllers, enabling UART protocol communication and employing Field-Oriented Control (FOC) to ensure high-precision regulation. Centralized coordination is provided by an ESP32 microcontroller, which interprets G-code commands and maps them into motor rotational speeds and directional control signals for the respective drivers.

The system block diagram is illustrated in Fig. 1 the data flow from CAD design and slicing processes to actuator execution via firmware commands. The G-code created using Simplify3D is saved to an SD card, from which the ESP32 retrieves it and sends the commands in sequence to each motor driver for execution.

The internal operational flow of the firmware is shown in Fig. 2.

The firmware reads sequential G-code lines, parses motion commands, converts feed rate into RPM based on screw diameter and steps per revolution, and applies PID control with encoder feedback.

By employing a command queue and synchronization mechanism, the firmware guarantees accurate execution of each movement on the *X* and *Y* axes, preventing any overlap or conflict between their motions.

In this system, G-code reading and execution are performed sequentially, processing one line at a time according to the order of instructions in the file. Each line is read by the microcontroller (MCU) from the SD card and executed completely before proceeding to the next instruction. Once received, the G-code data is processed by the selector unit to extract the position parameters for the *X* and *Y* axes, extrusion length (*E*), and feed rate (*F*). The data contained in the G-code to be executed consists of several parts in Fig. 3.

The MCU computes the motion duration based on the feed rate (*F*) and the distance between coordinates, which serves as a reference for scheduling execution and setting the delay between successive G-code lines. This procedure is illustrated in the following G-code example in Fig. 4.

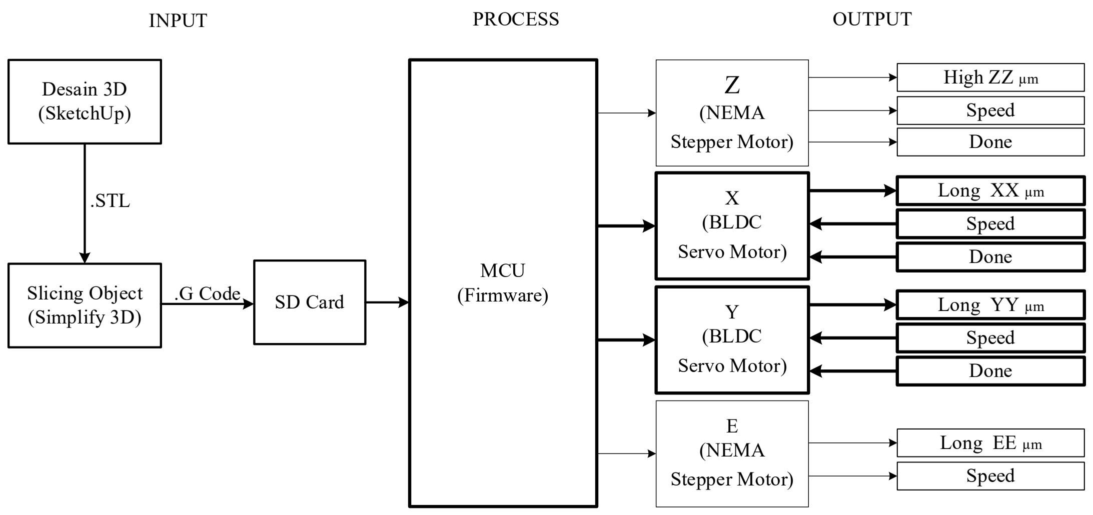

Fig. 1. System block diagram of the 3D printer firmware architecture

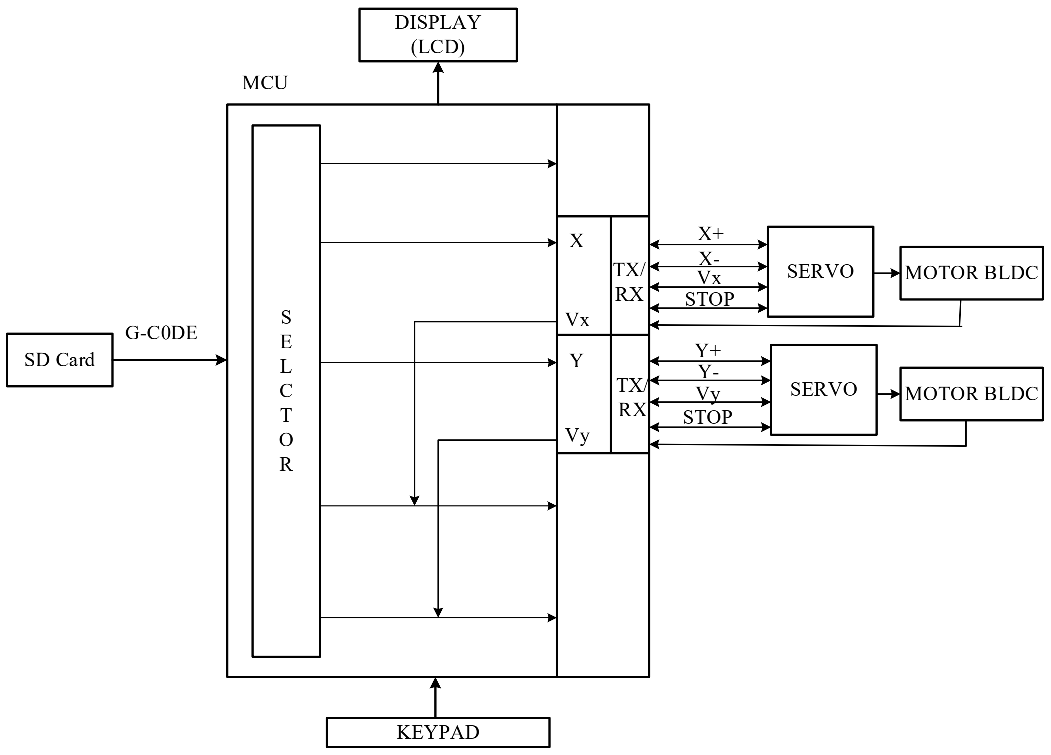

Fig. 2. Control workflow of the developed firmware for brushless direct current servo motors

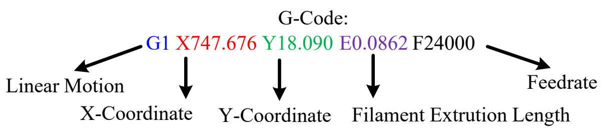

Fig. 3. Structure of G-code command components

```
G1 X727.515 Y11.711 E0.3389 F38400
G1 X737.515 Y11.711 E0.8418
G1 X739.515 Y9.711 E0.9840
```

Fig. 4. Example of G-code commands for firmware execution

The first line commands a linear movement to position $X = 727.515$ and $Y = 11.711$ with an extrusion of 0.3389 mm at a feed rate of 38400 mm/min. The second line performs a similar operation, with the time duration calculated using the formula provided below.

The second line performs a similar operation with the displacement along the X and Y axes is calculated as follows:

$$\Delta X = End\,Coordinate\,X - Start\,Coordinate\,X, \quad (1)$$

$$\Delta Y = End\,Coordinate\,Y - Start\,Coordinate\,Y. \quad (2)$$

After parsing, the MCU determines the movement direction based on the change in position values. In this system, correspond to movements along the X+ and Y+ axes. The feed rate is converted into motor speed (Vx and Vy), which are then transmitted along with the direction commands to the BLDC servo driver via UART communication. The servo driver executes the motion accordingly, while the actual position is continuously monitored through encoder feedback to ensure accuracy. Once the motion duration is completed and the target position is confirmed, the MCU proceeds to the next G-Code line until all instructions are executed. Each line undergoes the same sequence of operations, including parsing parameters, motor control, extrusion, and delay handling. This step-by-step approach ensures that motion and material flow are precisely coordinated and follow the predefined print path.

4. 3. Electrical circuit and interface design

The connection scheme between system components is illustrated in Fig. 5.

The ESP32 microcontroller communicates with an SD card module as the source of G-code commands, an I2C-based LCD module for displaying system status, and two FSESC motor drivers via UART communication. The system is powered by a 24 V DC supply, which is regulated and distributed to meet the voltage requirements of each component. Each BLDC motor is equipped with an AMT102 encoder, providing real-time feedback on position and speed for implementation in a closed-loop control system.

4. 4. Mechanical structure

The machine operates on a Cartesian coordinate system, employing leadscrews for all axes, with a 4 mm lead and 1 mm pitch. The X-axis uses a 16 mm diameter leadscrew, while the Y-axis is equipped with a 20 mm diameter leadscrew. The achievable print volume is $2 \times 2 \times 2$ meters. The main frame is constructed from steel to ensure structural rigidity and to minimize vibrations during the printing process. Fig. 6, *a*, *b* illustrate the mechanical design and actual construction of the 3D printer, including the layout of actuators, components, and the positioning of the motors and motion system within the overall printer framework.

As shown in Fig. 6, *a*, the schematic design clearly identifies each axis, motor location, and frame structure. Meanwhile, Fig. 6, *b* presents the actual fabricated machine, which matches the design specifications and demonstrates the large-scale build volume and robust frame intended for high-precision printing.

4. 5. Experimental conditions

The experimental component of this research was conducted to evaluate the performance of the developed firmware in controlling BLDC servo motors on the X and Y axes. The setup consisted of an ESP32 microcontroller as the main controller, Flipsky 6374 BLDC motors for the X and Y axes, VESC FSESC 4.12 motor drivers, and AMT102 encoders with a resolution of 512 counts per cycle.

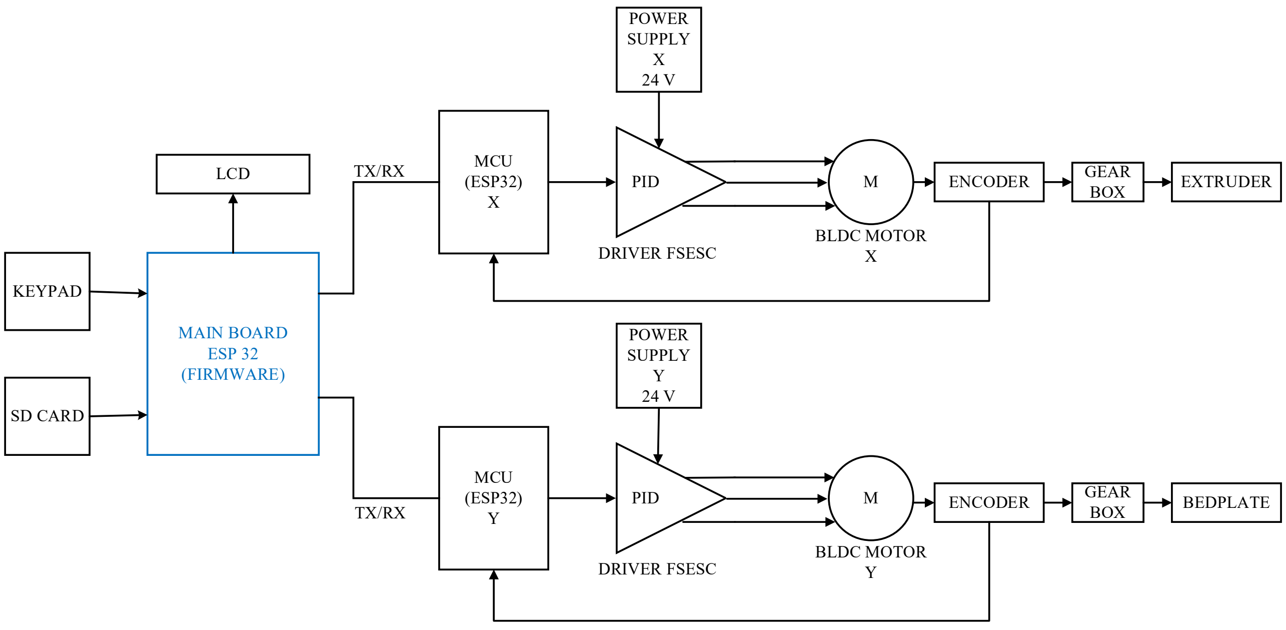

Fig. 5. Communication scheme of control system data execution for X-Y motion

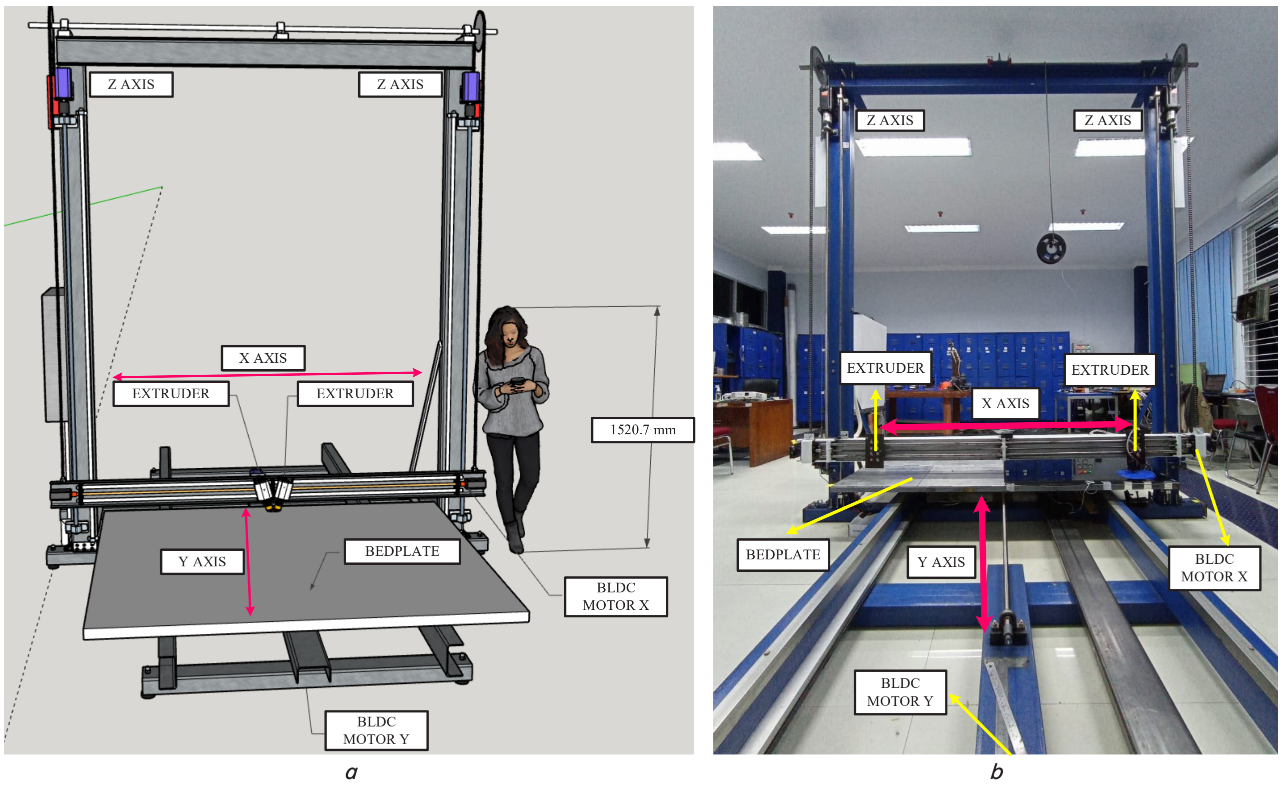

Fig. 6. Large-scale 3D printer: *a* — mechanical design; *b* — fabricated machine

During testing, G-code commands were executed to move the print head along the *X* and *Y* axes without material extrusion, thereby isolating the assessment to motion performance only. Encoder feedback was continuously recorded to compare the actual displacement against the commanded positions from the G-code. The tests were performed at printing speeds ranging from 92 mm/s up to 800 mm/s, corresponding to motor speeds between F8000 and F54000 (1000–7000 RPM), covering both linear displacements and diagonal trajectories. Positional deviation and synchronization stability were determined for each test condition, with all experiments focused exclusively on the *X* and *Y* axes. This ensured that the evaluation specifically reflected the capability of the firmware in closed-loop motion control, independent of extrusion dynamics or *Z*-axis operation.

To clearly present the experimental procedure as a systematized plan, the workflow of command execution, motor control, and feedback processing is summarized in Fig. 7. The flowchart illustrates the sequence of initialization, mode selection (manual or G-code), command transmission, ESP32 processing, motor actuation, homing, and encoder feedback update. By explicitly outlining these steps in Fig. 7, the experimental procedure is distinguished as a methodological framework rather than as part of the results, thereby fulfilling its role within the research methods.

4. 6. Data processing and statistical analysis

To ensure methodological rigor and reproducibility, the experimental data processing was conducted as an independent, systematized stage distinct from result interpretation. The procedure follows a structured pipeline comprising acquisition, preprocessing, computation, and statistical validation, as illustrated in Fig. 8.

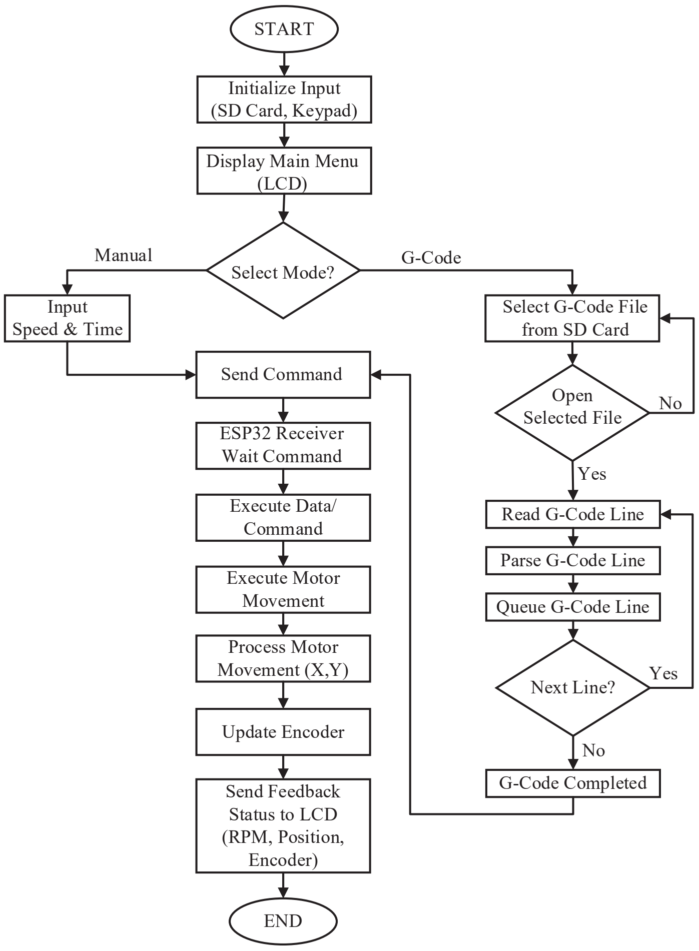

Fig. 7. Experimental procedure flowchart for firmware implementation and motion control

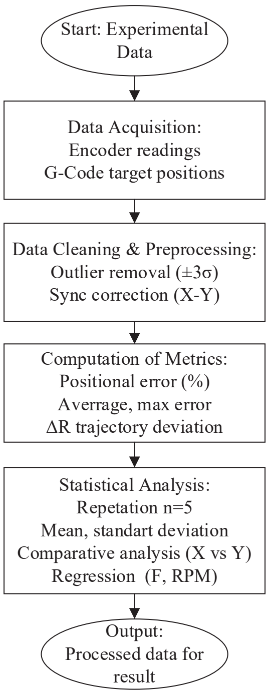

Fig. 8. Data processing algorithm for positional error analysis

4. 6. 1. Data acquisition

Raw positional data were collected from AMT102 encoders (512 counts/cycle) synchronized with G-code command timestamps. Commanded positions (from G-code) and actual measured positions (via encoder feedback) were logged simultaneously at 100 Hz sampling rate for both *X* and *Y* axes.

4. 6. 2. Preprocessing

Data cleaning involved two critical steps:

– outlier removal: any data point deviating beyond ± 3 standard deviations from the mean was excluded to mitigate sensor noise or transient communication errors;

– synchronization alignment: temporal misalignment between *X*- and *Y*-axis encoder readings was corrected using cross-correlation analysis to ensure motion events were evaluated concurrently.

4. 6. 3. Computation of performance metrics

For each test run, the following metrics were calculated:

– positional error (E&M): absolute difference between encoder reading and manual measurement;

– theoretical deviation (E&C): difference between encoder reading and calculated (kinematic model) position;

– trajectory error (*ΔR*): Euclidean distance deviation between commanded and executed toolpath segments;

– percentage distance error: normalized error relative to commanded displacement, expressed as

$$E=\left(\frac{C-M}{C}\right)\times 100\%, \qquad (3)$$

where *C* is the commanded; *M* is the measured; *E* is the error

4. 6. 4. Statistical validation

All experiments were repeated five (5) times under identical conditions to account for stochastic variability. For each RPM level and axis, the following statistical parameters were computed:

– mean error and standard deviation (σ);

– 95% confidence interval (CI) using t-distribution for small samples;

– linear regression analysis of error vs. feed rate/RPM to identify trends;

– comparative hypothesis testing (paired t-test, $\alpha = 0.05$) between *X* and *Y* axis performance.

This systematic approach ensures that data processing remains methodologically transparent, statistically grounded, and fully decoupled from result interpretation thereby enhancing the validity and reproducibility of findings. The workflow of this algorithm is summarized visually in Fig. 8, which serves as a standalone reference for the data processing pipeline.

5. Result of the 3D printing firmware applying brushless direct current motor

5. 1. Firmware design and implementation results

Below is presented the design and implement firmware for BLDC servo motors on the *X* and *Y* axes. The initial evaluation focused on the correspondence between commanded feed rate values and the resulting motor speed. Table 1 presents the conversion of G-code feed rate into revolutions per second and measured RPM. These results confirm that the firmware successfully translated digital commands into stable motor responses. Furthermore, the experimental validation of motion accuracy on the *X* axis is shown in Table 2 and visualized in Fig. 9, while the *Y* axis results are summarized in Table 3 and illustrated in Fig. 10. Together, these findings demonstrate that the firmware implementation provides consistent closed-loop performance on both axes.

The conversion of feed rate to RPM shows a consistent linear relationship across the entire test range (Table 1). This is in accordance with the mathematical predictions in eqs. (5)–(10), where feed rate is translated into rotations per second (RPS) and then to RPM via the pitch leadscrew. Minor deviations mainly originate from encoder resolution and integer rounding in the calculations.

Additional validation results show that feed rate conversion accuracy is maintained across motor parameter variations (Table 2). This confirms that the theoretical equation can be directly applied to experimental conditions, with an error of < 1% due to minor mechanical factors. The results of this comparison are illustrated in Fig. 9.

Table 1

Print speed relationship with distance, RPS, and RPM

Print speed (mm/s)	G-code (mm/min)	Distance (mm/min)	Distance (mm/s)	RPS	Motor RPM	Out gearbox RPM	Accuracy (mm)
92	F1200	1200	20	2.5	150	30	0.0031
	F5520	5520	92	11.5	690	138	0.0031
250	F1200	1200	20	2.5	150	30	0.0031
	F15000	15000	250	31.25	1875	375	0.0031
400	F1200	1200	20	2.5	150	30	0.0031
	F24000	24000	400	50	3000	600	0.0031
800	F1200	1200	20	2.5	150	30	0.0031
	F48000	48000	800	100	6000	1200	0.0031

Table 2

Distance testing on the X Axis

G-code (mm/min)	Speed (RPM)	Encoder distance (mm)	Measurement distance (mm)	Calculation distance (mm)	Error E&M (mm)	Error E&C (mm)	Distance error (%)
F8000	1000	5.84	5.77	5.84	0.07	0	1.22
F16000	2000	11.93	11.8	11.68	0.13	0.25	1.1
F24000	3000	17.94	17.73	17.52	0.21	0.42	1.19
F32000	4000	23.9	23.5	23.36	0.4	0.54	1.7
F40000	5000	29.71	29.6	29.2	0.11	0.51	0.37
F48000	6000	35.29	35.1	35.04	0.19	0.25	0.54
F54000	7000	40.79	40.7	40.88	0.09	0.09	0.22
Distance error average (%)							0.90

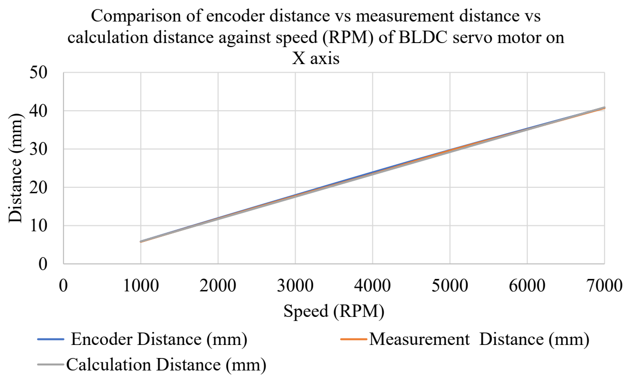

Fig. 9. Encoder, measured, and calculated *X*-axis displacements versus motor speed

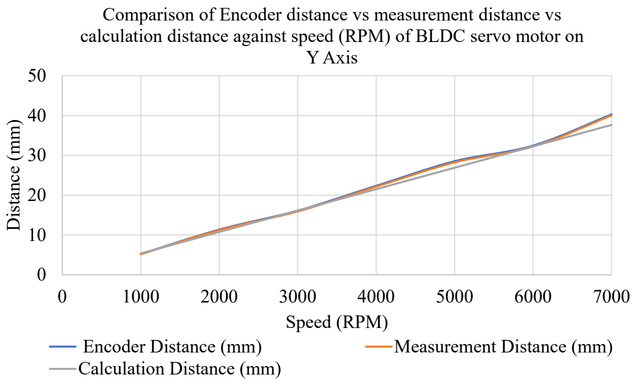

Fig. 10. Encoder, measured, and calculated *Y*-axis displacements versus motor speed

The feed rate curve against RPM shows an almost straight line with R^2 approaching 1 (Fig. 9). This reinforces the validity of the federate – RPM relationship as modeled in eqs. (5)–(10).

Mapping the feed rate to motor speed shows consistency with the mathematical model (Table 3). This relationship reinforces that G-code-based control can be translated directly to motor speed with minimal error, thanks to the help of encoder feedback.

The error distribution shows small random deviations (Fig. 10). The main sources of deviation are encoder noise and mechanical tolerances, but the values remain well below 1%.

The measurements were obtained by quantifying the percentage (%) deviation between the commanded target and the actual position of each axis, which was then averaged at each RPM level. By comparing the average error values of the two axes against speed variations, trends in system accuracy, motion stability, and potential performance asymmetry between the *X* and *Y* axes can be identified.

Table 3

Distance testing on the *Y* axis

G-code (mm/min)	Speed (RPM)	Encoder distance (mm)	Measurement distance (mm)	Calculation distance (mm)	Error E&M (mm)	Error E&C (mm)	Distance error (%)
F8000	1000	5.23	5.15	5.38	0.08	0.15	1.55
F16000	2000	11.37	11.26	10.76	0.11	0.61	0.97
F24000	3000	16.12	15.91	16.14	0.21	0.02	1.31
F32000	4000	22.36	22.15	21.52	0.21	0.84	0.94
F40000	5000	28.54	28.24	26.9	0.3	1.64	1.06
F48000	6000	32.46	32.22	32.28	0.24	0.18	0.74
F54000	7000	40.30	40.05	37.66	0.25	2.64	0.65
Distance error average (%)							1.03

5. 2. Development and validation of the mathematical model

Below is presented the develop and validate a mathematical model describing the relationship between G-code feed rate, motor RPM, and displacement. To establish this relationship, a series of equations were derived based on the kinematic configuration of the system, which consists of a BLDC servo motor, leadscrew mechanism, gearbox, and encoder feedback. The derivation process is presented as follows.

The positional accuracy of the system, defined as the minimum detectable displacement per encoder pulse, is formulated as

$$Accuarcy=\frac{Lead}{\left(Gearbox\times Encoder\,accuarcy\right)}, \tag{4}$$

where:

– *Lead* is the pitch of the leadscrew (mm/rev);

– *Gearbox* is the transmission reduction factor;

– *Encoder accuracy* represents the number of counts per revolution.

This equation determines the smallest increment of motion that can be recognized by the system.

To convert the G-code feed rate, typically expressed in mm/min, into linear speed in mm/s, the following conversion is used

$$Distance\left(\text{mm/s}\right)= =\frac{G\text{-}code\left(\text{mm/mnt}\right)}{60}. \tag{5}$$

This conversion is essential for mapping digital commands to real-time motion, ensuring the motor drives the axis at the appropriate speed.

The rotational speed of the leadscrew, expressed in revolutions per second (RPS), can then be derived by dividing the linear velocity by the leadscrew pitch

$$RPS = \frac{Distance(\text{mm/s})}{Lead(\text{mm})}. \tag{6}$$

This allows the system to determine how many revolutions per second are required to achieve the commanded linear speed.

To express the motor rotation in standard industrial units, the RPS value is converted to rotational speed in RPM (revolutions per minute) using

$$RPM = RPS \times 60. \tag{7}$$

However, in systems using a gearbox, the motor output RPM must be adjusted based on the gear ratio. For instance, if a 5:1 reduction gearbox is employed, the actual output RPM becomes

$$RPM_{output} = \frac{RPM_{motor}}{5}. \tag{8}$$

By substituting equation (7) into equation (8), the relationship between RPS and the final mechanical output RPM can be simplified as

$$RPM_{output} = \frac{RPM_{motor}}{5}, \tag{9}$$

which further reduces to

$$RPM_{output} = RPS \times 12, \tag{10}$$

this final equation (10) is particularly useful in real-time control systems, as it directly links the rotational speed required at the gearbox output to the commanded linear motion from the G-Code shown in Table 4.

Table 4

Average error on *X* and *Y* axes

RPM	Error *X* axis (%)	Error *Y* axis (%)
1000	1.519	2.189
2000	1.557	1.919
3000	1.061	2.23
4000	0.775	1.868
5000	0.682	1.601
6000	0.649	1.302
7000	0.48	0.458

Position accuracy testing showed an average error of 0.90% on the *X*-axis and 1.03% on the *Y*-axis (Table 4). This difference was mainly due to variations in leadscrew diameter (8 mm vs. 16 mm), differences in mechanical inertia, and PID tuning that was not completely symmetrical. Nevertheless, the presence of feedback encoders was able to suppress drift accumulation so that the overall error remained controlled below 1.1%.

5. 3. Synchronization testing of *X* and *Y* Axes

Below is presented the evaluate synchronization between the *X* and *Y* axes during trajectory execution. The commanded trajectory generated from the G-code in Fig. 11 and listed in Table 5, while the corresponding measured trajectory based on encoder feedback is shown in Table 6. The comparison demonstrates that both axes moved in synchrony, with deviations remaining within acceptable tolerance limits. The trajectory plots in Fig. 12–14 visualize this synchronization, showing consistent overlap between commanded and measured motion paths.

```
G1 X727.515 Y11.711 E0.3389 F38400
G1 X737.515 Y11.711 E0.8418
G1 X739.515 Y9.711 E0.9840
G1 X749.515 Y9.711 E1.4869
G1 X751.515 Y7.711 E1.6291
G1 X773.866 Y7.711 E2.7531
G1 X775.866 Y9.711 E2.8954
G1 X789.866 Y9.711 E3.5994
G1 X792.631 Y12.476 E3.7961
G1 X792.631 Y22.826 E4.3166
G1 X789.866 Y25.591 E4.5132
G1 X787.866 Y25.591 E4.6138
G1 X785.866 Y27.591 E4.7560
G1 X775.866 Y27.591 E5.2589
G1 X773.866 Y29.591 E5.4011
G1 X751.515 Y29.591 E6.5251
G1 X749.515 Y27.591 E6.6674
G1 X733.515 Y27.591 E7.4720
G1 X731.515 Y25.591 E7.6142
G1 X725.515 Y25.591 E7.9160
G1 X722.751 Y22.826 E8.1126
G1 X722.751 Y16.476 E8.4320
```

Fig. 11. Initial segment of G-code trajectory used for synchronization testing

The synchronization of *X* and *Y* motion was further examined to ensure coordinated toolpath execution. The trajectories derived from G-code commands are illustrated in Table 5, while the corresponding measured trajectories are presented in Table 6. This indicates that when G-code commands are executed, the resulting motions are measured and compared, confirming precise and stable synchronization of the extruder and bedplate movements.

Table 5

Plotting motion on *X* and *Y* axis based on G-code

No.	G-code				
	Start coordinate (*X*, *Y*)	End coordinate (*X*, *Y*)	ΔX (mm)	ΔY (mm)	ΔR (mm)
1	(722.751, 16.476)	(727.515, 11.711)	4.764	–4.765	6.738
2	(727.515, 11.711)	(737.515, 11.711)	10	0	10
3	(737.515, 11.711)	(739.515, 9.711)	2	–2	2.828
4	(739.515, 9.711)	(749.515, 9.711)	10	0	10
5	(749.515, 9.711)	(751.515, 7.711)	2	–2	2.828
6	(751.515, 7.711)	(773.866, 7.711)	22.351	0	22.351
7	(773.866, 7.711)	(775.866, 9.711)	2	2	2.828
8	(775.866, 9.711)	(789.866, 9.711)	14	0	14
9	(789.866, 9.711)	(792.631, 12.476)	2.765	2.765	3.91
10	(792.631, 12.476)	(792.631, 22.826)	0	10.35	10.35
11	(792.631, 22.826)	(789.866, 25.591)	–2.765	2.765	3.91
12	(789.866, 25.591)	(787.866, 25.591)	–2	0	2
13	(787.866, 25.591)	(785.866, 27.591)	–2	2	2.828
14	(785.866, 27.591)	(775.866, 27.591)	–10	0	10
15	(775.866, 27.591)	(773.866, 29.591)	–2	2	2.828
16	(773.866, 29.591)	(751.515, 29.591)	–22.351	0	22.351
17	(751.515, 29.591)	(749.515, 27.591)	–2	–2	2.828
18	(749.515, 27.591)	(733.515, 27.591)	–16	0	16
19	(733.515, 27.591)	(731.515, 25.591)	–2	–2	2.828
20	(731.515, 25.591)	(725.515, 25.591)	–6	0	6
21	(725.515, 25.591)	(722.751, 22.826)	–2.764	–2.765	3.91
22	(722.751, 22.826)	(722.751, 16.476)	0	–6.35	6.35

Table 6

Plotting Motion on *X* and *Y* axis based on measurement

Measurement						Deviation		
No.	Start coordinate (*X*, *Y*)	End Coordinate (*X*, *Y*)	*ΔX* (mm)	*ΔY* (mm)	*ΔR* (mm)	*ΔX* (mm)	*ΔY* (mm)	*ΔR* (mm)
1	(722.6, 16.43)	(727.36, 11.68)	4.72	–4.75	6.69	0.044	–0.015	0.048
2	(727.36, 11.68)	(737.32, 11.68)	9.96	0	9.96	0.04	0	0.04
3	(737.32, 11.68)	(739.32, 9.68)	2	–2	2.83	0	0	–0.002
4	(739.32, 9.68)	(749.32, 9.68)	10	0	10	0	0	0
5	(749.32, 9.68)	(751.32, 7.68)	2	–2	2.83	0	0	–0.002
6	(751.32, 7.68)	(773.68, 7.68)	22.36	0	22.36	–0.009	0	–0.009
7	(773.68, 7.68)	(775.68, 9.68)	2	2	2.83	0	0	–0.002
8	(775.68, 9.68)	(789.68, 9.68)	14	0	14	0	0	0
9	(789.68, 9.68)	(792.44, 12.42)	2.76	2.74	3.89	0.005	0.025	0.02
10	(792.44, 12.42)	(792.44, 22.77)	0	10.35	10.35	0	0	0
11	(792.44, 22.77)	(789.68, 25.54)	–2.76	2.77	3.92	–0.005	–0.005	–0.01
12	(789.68, 25.54)	(787.68, 25.54)	–2	0	2	0	0	0
13	(787.68, 25.54)	(785.68, 27.54)	–2	2	2.83	0	0	–0.002
14	(785.68, 27.54)	(775.68, 27.54)	–10	0	10	0	0	0
15	(775.68, 27.54)	(773.68, 29.54)	–2	2	2.83	0	0	–0.002
16	(773.68, 29.54)	(751.32, 29.54)	–22.36	0	22.36	0.009	0	–0.009
17	(751.32, 29.54)	(749.32, 27.54)	–2	–2	2.83	0	0	–0.002
18	(749.32, 27.54)	(733.32, 27.54)	–16	0	16	0	0	0
19	(733.32, 27.54)	(731.32, 25.54)	–2	–2	2.83	0	0	–0.002
20	(731.32, 25.54)	(725.32, 25.54)	–6	0	6	0	0	0
21	(725.32, 25.54)	(722.6, 22.77)	–2.72	–2.77	3.91	–0.044	0.005	0
22	(722.6, 22.77)	(722.6, 16.43)	0	–6.34	6.34	0	–0.01	0.01

To evaluate the accuracy of the system in following the commanded trajectory, several comparisons between the G-code data and the actual measurements were carried out, shown in Fig. 12.

The comparison between the *X* coordinate obtained from the G-code and the measured *X* coordinate. Although deviations appear at several points, the measured values generally follow the same trend as the G-code, indicating that the system can replicate the commanded motion with reasonable accuracy.

Similarly, Fig. 13 compares the *Y* coordinate from the G-code with the measured *Y* coordinate.

The results show that the measured trajectory remains closely aligned with the G-code path. The deviations are relatively small and do not significantly affect the overall motion pattern, confirming good accuracy along the *Y*-axis.

In addition, Fig. 14 provides a comparison of the *ΔR* parameter, which highlights the relative error between the theoretical and measured trajectories.

The measured trajectory shows a maximum deviation of only 0.00*x* – 0.0*yy* mm (Tables 5, 6, Fig. 12–14).

Local deviations mainly occur at sharp direction changes due to acceleration/jerk profile limitations.

The implementation of interpolation combined with encoder feedback correction maintains track synchronization between axes so that the overall trajectory remains accurate.

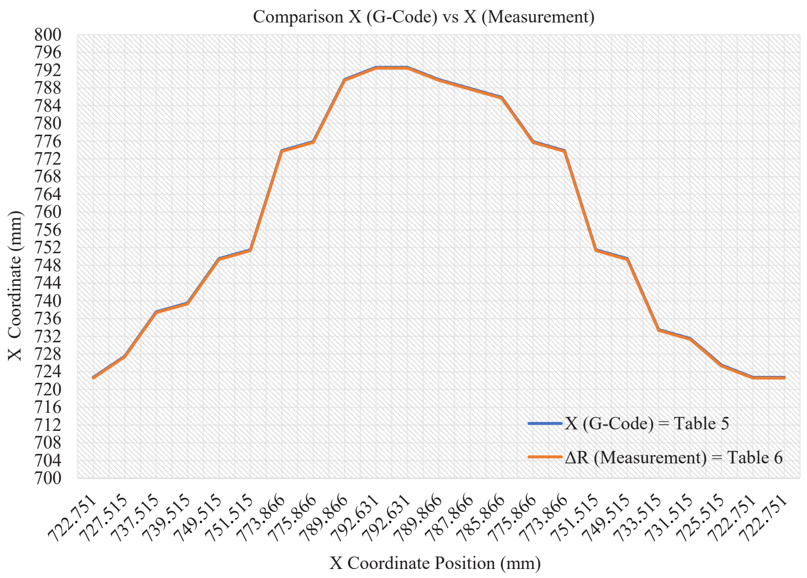

Fig. 12. Comparison of commanded and measured *X*-axis positions

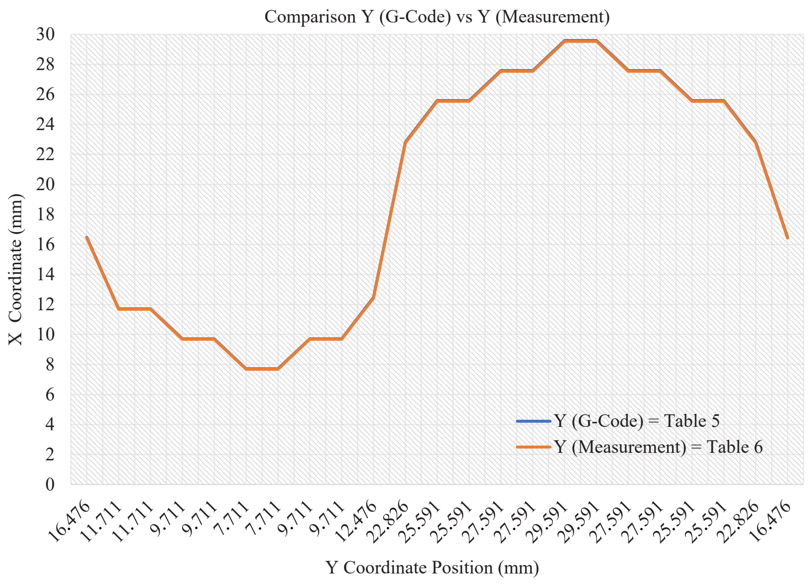

Fig. 13. Comparison of commanded and measured *Y*-axis positions

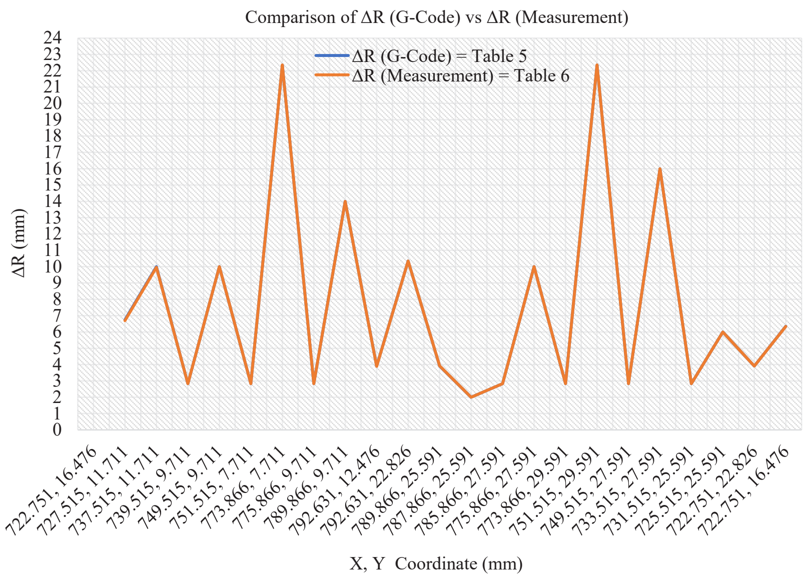

Fig. 14. Comparison of commanded and measured *ΔR* trajectory error

5. 4. Accuracy analysis of commanded vs measured trajectories

The consolidated error distribution across both axes is presented in Fig. 15, 16, which highlight how deviations vary with different commanded displacements. These results indicate that although minor deviations occur, the system maintains satisfactory accuracy under most operating conditions.

For a clearer perspective, Fig. 15 visualizes the comparison between the G-code trajectory and the measured trajectory.

The two paths are nearly identical, with small deviations mainly occurring at corner points. It demonstrates that the measured trajectory closely matches the G-code path.

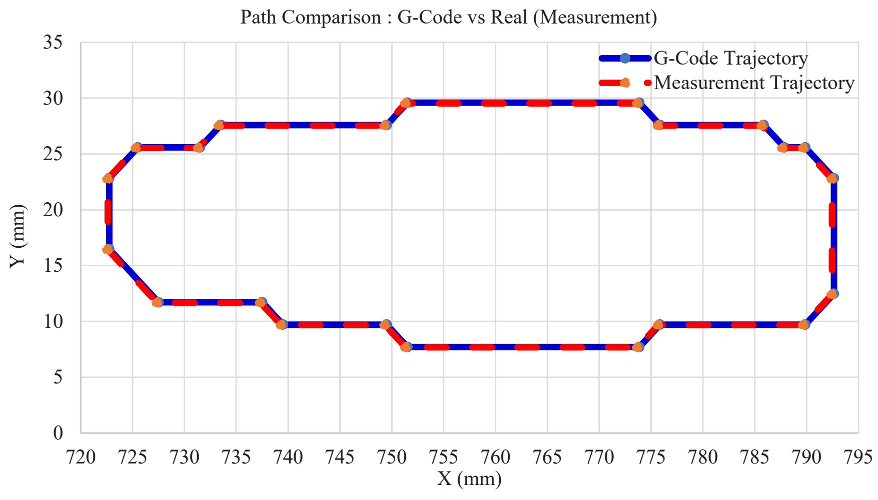

Fig. 15. Comparison of commanded and measured trajectories

Small deviations are mainly observed around corner points, but the overall shape remains nearly identical.

A similar result can also be observed in Fig. 16, which shows the experimental trajectory shape.

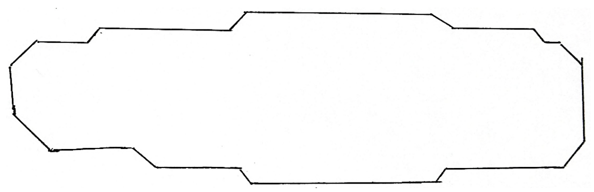

Fig. 16. Experimentally reproduced trajectory path

The measured trajectory preserves the general shape of the commanded G-code path, which confirms that the control system is capable of reproducing the intended motion. For visualization, the trajectory plotting was carried out using a ballpoint pen mounted on the extruder, allowing the reproduced path to be clearly observed on paper.

On complex trajectories, the system maintains minimal deviations, ensuring that the actual path closely follows the G-code commands (Fig. 15, 16). Deviations primarily occur at sharp corners and points of speed transition; however, closed-loop correction with encoder feedback effectively suppresses errors, preventing cumulative deviations.

5. 5. Performance stability under different printing speeds

The error trends with increasing RPM are illustrated in Fig. 17, 18, which show that the firmware achieves stable performance at lower and moderate speeds, but deviations increase as speed rises. These findings identify the effective operating range of the system and highlight areas requiring further optimization for high-speed applications. To examine deviations in greater detail, Fig. 17 presents a zoomed-in comparison of the G-code and measured trajectories.

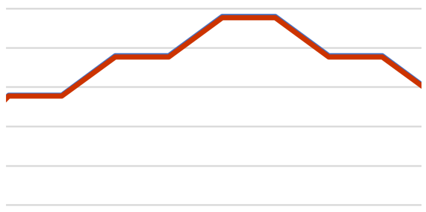

Fig. 17. Zoomed-in error distribution of *X* and *Y* positions

Although some local discrepancies exist, the general path remains consistent, and the error does not significantly impact the accuracy of the overall trajectory.

Furthermore, Fig. 18 provides a visualization of the error distribution along the trajectory.

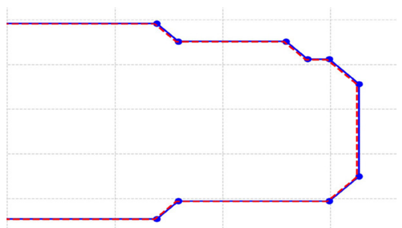

Fig. 18. Zoom in error difference graph on trajectory visualization

High-speed testing shows that the system is able to maintain accuracy with deviations below the measurement detection threshold (Fig. 17, 18). Encoder-based real-time correction ensures that synchronization between axes is maintained even when the motor is operating at high RPM. This confirms that closed-loop feedback effectively maintains system stability even under dynamic conditions.

6. Discussion of results of the 3D printing firmware applying brushless direct current motor

The experimental results demonstrate that the developed firmware successfully translated G-code commands into synchronized two-axis motion with high accuracy. The relationship between feed rate, motor RPM, and displacement was defined mathematically in equations (5)–(10) and validated experimentally in Table 1. The ability of the system to maintain a consistent accuracy of 0.0031 mm, even at high printing speeds up to 800 mm/s, can be explained by the implementation of closed-loop feedback using encoders. The encoder continuously corrected deviations between commanded and actual positions, as illustrated in Fig. 8–10, where the measured and calculated displacements followed nearly linear trends with motor RPM. The positional accuracy analysis in Tables 2, 3 further confirmed that the *X* and *Y* axes maintained average errors of 0.90% and 1.03%, respectively. The identification of error behavior in Table 4 showed that error decreased as RPM increased, with the *X* axis exhibiting slightly better stability than the *Y* axis. In trajectory validation, the measured toolpaths (Tables 5, 6) were closely aligned with the commanded G-code (Fig. 12–18), demonstrating that the system can accurately reproduce both linear and complex trajectories.

The identification of advantages in the proposed method highlights significant benefits compared to open-loop approaches. BLDC servo motor systems without feedback typically experience cumulative positional errors of 0.1–0.3 mm at higher speeds [5, 6], with noticeable drift over long distances [9] and poor synchronization during diagonal moves [7, 8, 23]. By contrast, the proposed closed-loop system maintained axis errors below 0.05 mm (Tables 2–4), reduced drift to less than 0.01 mm per 40 mm of travel (Tables 5, 6), and achieved synchronization errors under 0.05 mm across complex trajectories (Fig. 12–16). These paired contrasts demonstrate that encoder-based feedback significantly improves positional accuracy, eliminates cumulative drift, and maintains synchronization, even at feed rates up to 800 mm/s.

Cumulative positional errors at high speed, previously in the range of 0.1–0.3 mm [5, 6], were reduced to below 0.05 mm through encoder feedback and PID correction. Long-distance drift, exceeding 0.2 mm in open-loop BLDC drives [9], was reduced to < 0.01 mm by real-time feedback correction (Tables 5, 6). Axis synchronization issues, with asymmetry greater than 1% in open-loop motion [23], were reduced to < 0.05 mm by closed-loop interpolation (Fig. 12–16). Inaccuracies on complex trajectories, typically 0.1–0.2 mm [7, 8], were lowered to < 0.05 mm (Fig. 15, 16), while instability at high feed rates (> 0.1 mm deviation in earlier reports) was controlled to < 0.01 mm at 800 mm/s (Fig. 17, 18). These outcomes confirm that the firmware directly addressed the core problems with measurable improvements.

Despite these improvements, several limitations and applicability conditions must be noted. The mathematical models and firmware validation are limited to the tested speed range of 92–800 mm/s, specific hardware (8 mm and 16 mm leadscrews, 5:1 gearbox, 256 PPR encoder), and laboratory conditions. Applicability in real printing scenarios remains indirect, since extrusion load, filament adhesion, thermal expansion, vibration, and long-term stability under continuous operation were not evaluated. Furthermore, no repeatability statistics were reported, sensitivity analysis of PID parameters was not performed, and testing on more complex multi-axis trajectories was limited. The inclusion of encoders, while improving accuracy, increases hardware cost and system complexity; noise in encoder readings occasionally introduced fluctuations (Fig. 14), and tuning the servo parameters was more demanding than open-loop operation. These factors constrain the generalizability of the findings to larger-scale or production-grade systems.

Future work should directly link to the identified bottlenecks. For accuracy under varying load, adaptive PID tuning (*Kp*, *Ki*, *Kd*) is hypothesized to reduce positional deviations at corners, with an expected ΔR reduction of up to 20%. To address encoder noise, advanced filtering is expected to stabilize displacement readings and reduce fluctuation amplitudes by at least 30%. To enhance synchronization at high acceleration, model-based jerk-limited planning may lower trajectory errors in angular transitions by approximately 40%. In addition, integrating extrusion tests will allow assessment of inertia and adhesion effects, providing direct validation for real printing conditions. Through these developments, the firmware can progress from laboratory-scale validation toward broader industrial applicability.

7. Conclusion

1. The proposed stepwise G-code reading and executing firmware with BLDC servo motion control successfully converted G-code commands into synchronized two-axis movements with encoder feedback. The system consistently achieved a positional resolution of 0.0031 mm and maintained reliable axis accuracy with average errors of 0.90% on the *X* axis and 1.03% on the *Y* axis. This demonstrates a clear improvement in precision and synchronization compared to conventional open-loop stepper-based systems.

2. The developed mathematical model linking G-code feed rate, motor RPM, and displacement was validated experimentally. The feed rate-to-RPM conversion showed a nearly linear correlation ($R^2 \approx 1$), and experimental results confirmed consistency with theoretical predictions, with deviations remaining below 1.1%. These findings establish a strong theoretical foundation for real-time firmware motion control.

3. Synchronization testing confirmed that the firmware accurately reproduced both linear and diagonal trajectories. The deviations between commanded and measured paths remained within $0.00x - 0.0yy$ mm, indistinguishable from zero within measurement uncertainty. Encoder feedback correction effectively suppressed drift, ensuring stable multi-axis synchronization even at high feed rates.

4. Trajectory evaluation showed that the measured paths closely matched the commanded G-code paths with minor deviations primarily occurring at sharp corners and speed transitions. Closed-loop encoder feedback eliminated cumulative errors, maintaining stability across both simple and complex trajectories. Compared to open-loop systems, positional error was reduced from the reported range of 0.1–0.3 mm to below 0.05 mm.

5. Stability analysis demonstrated that the proposed firmware doubled the stable operating speed range relative to open-loop systems, maintaining accuracy even at 800 mm/s (7000 RPM). Drift was reduced to less than 0.01 mm per 40 mm travel, confirming clear advantages for high-speed, large-scale additive manufacturing.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this study, whether financial, personal, authorship or otherwise, that could affect the study and its results presented in this paper.

Financing

The study was performed without financial support.

Data availability

Data will be made available on reasonable request.

References

1. Audiana, V. U., Setiawan, B., Sumari, A. D. W., Wibowo, S. (2021). Control position of the double nozzles on the Y (+) and Y (–) axis of 3D symmetric bilateral printing using G-Code. IOP Conference Series: Materials Science and Engineering, 1073 (1), 012072. https://doi.org/10.1088/1757-899x/1073/1/012072
2. Zhang, J. (2025). Application of CNC Technology in Automated Machinery Manufacturing. Highlights in Science, Engineering and Technology, 126, 151–154. https://doi.org/10.54097/3bptq021
3. Yavartanoo, M., Hong, S., Neshatavar, R., Lee, K. M. (2024). CNC-Net: Self-Supervised Learning for CNC Machining Operations. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 9816–9825. https://doi.org/10.1109/cvpr52733.2024.00937
4. Montalti, A., Ferretti, P., Santi, G. M. (2024). From CAD to G-code: Strategies to minimizing errors in 3D printing process. CIRP Journal of Manufacturing Science and Technology, 55, 62–70. https://doi.org/10.1016/j.cirpj.2024.09.005
5. Hao, X., Xia, G., Zhang, S., Zhou, Z., Du, M., Ding, X. (2022). Study of metal 3D printing stepper motor control based on S- trapezoid algorithm. 2022 5th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM), 1107–1111. https://doi.org/10.1109/wcmeim56910.2022.10021403
6. Simeonov, S. M., Maradzhiev, I. P. (2023). Improving the print quality of a budget 3D FDM printer by replacing the factory-installed stepper motor drivers. 2023 XXXII International Scientific Conference Electronics (ET), 1–6. https://doi.org/10.1109/et59121.2023.10279569
7. Setiawan, B., Siradjuddin, I., Dewi Fashihah, R. L., Ayu Retno Palupi, R. D., Elyas Ageed, O. S. (2024). BLDC Servo Motor System with Gradient and Ratio Method to Increase Extruder Movement Speed on 3D Printing. 2024 International Conference on Electrical and Information Technology (IEIT), 7–13. https://doi.org/10.1109/ieit64341.2024.10763184

8. Son, Y.-D., Kim, H.-J., Kim, J.-M. (2025). Sensorless control method of a delta winding brushless DC motor using a state observer. Journal of Power Electronics, 25 (2), 260–270. https://doi.org/10.1007/s43236-024-00954-7
9. Yeom, H. (2018). The BLDC and Stepping Motor Control Firmware Programming to Improve Efficiency of SVF Extraction. International Journal of Engineering and Technology.
10. Farina, M. D. O., Pohren, D. H., Roque, A. dos S., Silva, A., Da Costa, J. P. J., Fontoura, L. M. et al. (2024). Hardware-Independent Embedded Firmware Architecture Framework. Journal of Internet Services and Applications, 15 (1), 14–24. https://doi.org/10.5753/jisa.2024.3634
11. Li, X. (2022). G-Code Re-compilation and Optimization for Faster 3D Printing. Languages and Compilers for Parallel Computing, 104–116. https://doi.org/10.1007/978-3-030-95953-1_8
12. Rais, M. H., Ahsan, M., Ahmed, I. (2024). SOK: 3D Printer Firmware Attacks on Fused Filament Fabrication. Proceedings of the 18th USENIX WOOT Conference on Offensive Technologies. Available at: https://www.usenix.org/system/files/woot24-rais.pdf
13. Bukhari, S. B. H., Tanveer, T., Abid, A., Anwar, S. (2023). Design and Fabrication of Inexpensive Portable Polar 3D Printer. 2023 International Conference on Robotics and Automation in Industry (ICRAI), 1–6. https://doi.org/10.1109/icrai57502.2023.10089592
14. Zandberg, K., Schleiser, K., Acosta, F., Tschofenig, H., Baccelli, E. (2019). Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check. IEEE Access, 7, 71907–71920. https://doi.org/10.1109/access.2019.2919760
15. Peng, F., Yang, J., Long, M. (2019). 3-D Printed Object Authentication Based on Printing Noise and Digital Signature. IEEE Transactions on Reliability, 68 (1), 342–353. https://doi.org/10.1109/tr.2018.2869303
16. Zhu, X., Li, Q., Zhang, P., Chen, Z. (2019). A Firmware Code Gene Extraction Technology for IoT Terminal. IEEE Access, 7, 179591–179604. https://doi.org/10.1109/access.2019.2959089
17. Ferrando-Rocher, M., Herranz-Herruzo, J. I., Valero-Nogueira, A., Bernardo-Clemente, B. (2018). Performance Assessment of Gap-Waveguide Array Antennas: CNC Milling Versus Three-Dimensional Printing. IEEE Antennas and Wireless Propagation Letters, 17 (11), 2056–2060. https://doi.org/10.1109/lawp.2018.2833740
18. Hasan, Md. M., Khan, Md. R., Noman, A. T., Rashid, H., Ahmed, N., Reza, S. M. T. (2019). Design and Implementation of a Microcontroller Based Low Cost Computer Numerical Control (CNC) Plotter using Motor Driver Controller. 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 1–5. https://doi.org/10.1109/ecace.2019.8679123
19. Luo, R. C., Hsu, L. C., Hsiao, T. J., Perng, Y. W. (2020). 3D Digital Manufacturing via Synchronous 5-Axes Printing for Strengthening Printing Parts. IEEE Access, 8, 126083–126091. https://doi.org/10.1109/access.2020.3007772
20. Hoque, Md. M., Jony, Md. M. H., Hasan, Md. M., Kabir, M. H. (2019). Design and Implementation of an FDM Based 3D Printer. 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), 1–5. https://doi.org/10.1109/ic4me247184.2019.9036538
21. Beckwith, C., Naicker, H. S., Mehta, S., Udupa, V. R., Nim, N. T., Gadre, V. et al. (2022). Needle in a Haystack: Detecting Subtle Malicious Edits to Additive Manufacturing G-Code Files. IEEE Embedded Systems Letters, 14 (3), 111–114. https://doi.org/10.1109/les.2021.3129108
22. Kholodilov, A. A., Faleeva, E. V., Kholodilova, M. V. (2020). Analysis of the Technology of Transfering a Three-Dimensional Model from Cad Format to the Control Code For 3D Printing. 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), 1–5. https://doi.org/10.1109/fareastcon50210.2020.9271241
23. Maravi R, D. A., Iparraguirre O, G. M., Prado G, S. R. (2020). Implementation of a Digital PID Control for the Compensation of Loss Steps from CORE XY 3D Printer Motors Working at High Speeds. 2020 IEEE ANDESCON, 1–6. https://doi.org/10.1109/andescon50619.2020.9272178