Guided Robotic Manipulator for Chinese Calligraphy

By

Nuoer Huang

Xinyi Shen

Xirui Yao

Yujie Wei

Final Report for ECE 445, Senior Design, Spring 2026

TA: Kang Tan

15 May 2026

Project No. 8

Abstract

This report presents the design and verification of a guided robotic manipulator for Chinese calligraphy. The system integrates a neural network-based path planning engine with a three-axis Cartesian gantry plotter to translate static character images into dynamic brush strokes. The mechanical platform uses lead screw transmission driven by 42 stepper motors and A4988 drivers, achieving ±0.03 mm positioning accuracy and <0.15 mm backlash. The electrical subsystem includes a RAMPS 1.4 shield and Arduino Mega 2560 running Marlin firmware. Experimental validation confirmed that all requirements—including displacement accuracy, homing repeatability, motor thermal performance (below 65°C), frame stiffness (<0.20 mm deflection), and serial communication reliability—were successfully met. The system demonstrates the feasibility of AI-driven mechatronic systems for cultural preservation applications.

Contents

1. Introduction

Traditional Chinese calligraphy is a sophisticated art form that relies on the precise coordination of stroke sequence, velocity, and pressure. Replicating this process with a robotic system presents a multi-disciplinary engineering problem. The science behind this project involves translating static visual information—character images from a digital library—into dynamic mechatronic actions. The primary challenge is twofold: first, the development of a Neural Network (NN) capable of extracting stroke trajectories from pixel data while maintaining traditional stroke order; and second, the design of a 3-axis Cartesian plotter capable of high-torque, high-resolution execution using a compliant brush.

The purpose of this device is to bridge the gap between AI-driven computer vision and physical mechatronic execution. Its usefulness lies in cultural preservation, automated high-quality artistic reproduction, and serving as an educational platform for advanced motion control and deep learning integration.

1.1 Problem Statement

Traditional Chinese calligraphy is a sophisticated art form that relies on the precise coordination of stroke sequence, velocity, and pressure. Replicating this process with a robotic system presents a multi-disciplinary engineering problem. However, traditional calligraphy robots are often limited to mechanically reproducing the geometric outlines of characters and are unable to capture the variations in brushstroke pressure—such as "lifting, pressing, pausing, and varying pressure" (known as Tun, Ti, An in Chinese calligraphy)—characteristic of the art form. Most existing open-source solutions rely on expensive commercial robotic arms and require large amounts of annotated expert trajectory data.

The science behind this project involves translating static visual information—character images from a digital library—into dynamic mechatronic actions. The primary challenge is twofold: first, the development of a neural network capable of extracting stroke trajectories from pixel data while maintaining traditional stroke order; and second, the design of a three-axis Cartesian plotter capable of high-torque, high-resolution execution using a compliant brush.

This project aims to develop a low-cost, Arduino Mega 2560-based three-axis system that uses unsupervised learning to extract writing behavior from static images, thereby addressing the issues of stylistic limitations and high hardware costs in automated calligraphy creation.

1.2 Solution Overview

This approach employs a "coarse-to-fine" control strategy. The host computer (PC) runs a convolutional neural network based on CalliRewrite [1] to decompose calligraphy images into path sequences. Subsequently, reinforcement learning (the Soft Actor-Critic, or SAC, algorithm) is used to optimize the dynamic down-pressure trajectory of the brush in a simulated environment. Finally, the system converts

the coordinates into G-code and transmits them via a serial port to a three-axis slide controlled by an Arduino.

The system architecture is structured into four distinct, interconnected modules:

Trajectory Generation & System Integration: Implements a deep learning architecture (CNN-LSTM hybrid) to perform image-to-sequence mapping, interpreting a static calligraphy image and decomposing it into a chronological sequence of stroke control points.

RL-based Stroke Refinement: Implements the SAC algorithm within a high-fidelity virtual simulation environment to optimize raw stroke trajectories, learning a policy that maps coarse skeletal paths to refined three-dimensional movements (X, Y, Z), including brush pressure control.

Control & Electrical System: Receives high-level G-code commands from the host software and translates them into precise electrical pulses that drive the stepper motors. It manages power distribution, ensures synchronized movement across axes, and monitors system boundaries through limit switches.

Mechanical Execution: The physical motion core of the robot. It receives step/direction pulse signals from the control subsystem via a RAMPS 1.4 expansion board and drives X, Y, and Z axis stepper motors to convert digital commands into precise mechanical displacement of the brush tip on paper.

The purpose of this device is to bridge the gap between AI-driven computer vision and physical mechatronic execution. Its usefulness lies in cultural preservation, automated high-quality artistic reproduction, and serving as an educational platform for advanced motion control and deep learning integration.

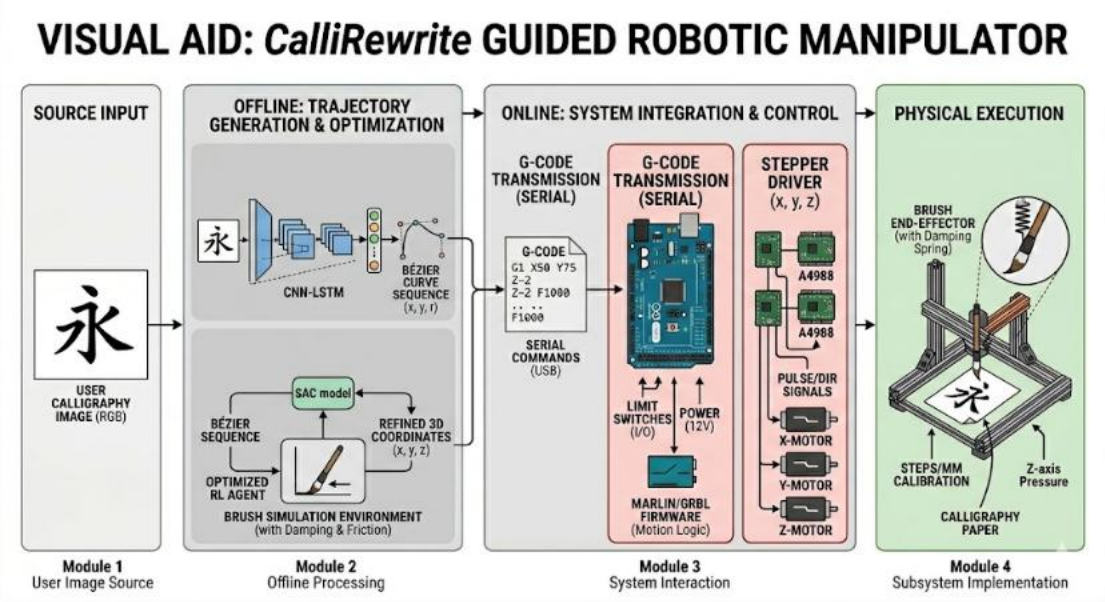

Figure 1: Visual Aid of the whole system

1.3 High-Level Requirements List

The calligraphy robotic manipulator system shall meet the following requirements:

1. Trajectory Fidelity: The average root means square error (RMSE) between the centerlines of the robotically written strokes and the skeleton of the source calligraphy image must be less than 2.0 mm, ensuring the geometric integrity of the characters.
2. Dynamic Stroke Width Control: The system must successfully replicate at least eight distinct levels of ink thickness by controlling the Z-axis displacement with a precision of ±0.1 mm, effectively simulating the "pressing and lifting" (Tan-An) techniques of traditional brushwork.
3. Operational Reliability and Synchronization: During a continuous writing session of more than 100 characters, the stepper motors must maintain zero step loss, and the end-to-end communication latency between the host Python script and the Arduino Mega controller must remain below 50 ms to ensure smooth, uninterrupted execution.

2 Design

The design of the calligraphy robot is bifurcated into a software-based G-Code Generation Engine and a hardware-based 3-Axis Execution Platform. The core design philosophy favors a lead screw-driven Cartesian architecture to ensure the stability required for brush-to-paper friction.

2.1 Software System (Neural Network Module)

The software subsystem converts a user's character input into executable G-code for the XYZ writing machine. The overall software pipeline consists of six major stages: character rendering, coarse sequence extraction, trajectory reconstruction, G-code generation, multi-character layout, and user interface control.

The input to the software is either a single Chinese character or a short string of Chinese characters. The user also selects a font file, character size, writing origin, pen-up height, and pen-down height. The software first renders each character into a 256 by 256 pixel grayscale PNG image. The image is then processed by the CalliRewrite sequence extraction model, which produces an NPZ file containing the coarse stroke sequence. This NPZ file is parsed by a custom conversion script, which reconstructs the stroke trajectory and writes it as G-code commands. For multi-character writing, each character is processed separately, and the resulting G-code paths are shifted by layout offsets before being merged into one final G-code file.

2.1.1 Character Rendering and Preprocessing

The first stage of the software pipeline is character rendering. Since the motion platform should support different Chinese fonts, the system generates character images dynamically instead of relying on a fixed image dataset. The rendering script uses Python and the Pillow library to load a font file from the Windows font directory under WSL, such as simkai.ttf for KaiTi style. The selected Chinese character is drawn onto a 256 by 256 pixel grayscale canvas with a white background and black foreground.

The font size was tuned experimentally. A font size of 253 pixels produced a character image with a large enough glyph area while avoiding excessive clipping for the demonstration fonts. This choice reduces unnecessary white space around the character, which is important because the downstream G-code scaling treats the whole 256 by 256 image as the character region. Excessive white space would waste the limited motion range of the physical writing machine.

The output filename is encoded using the Unicode code point of the character instead of the raw Chinese character. For example, the character “木” is saved using a filename such as u6728_simkai_256.png. This avoids file path problems in scripts and command-line tools.

2.1.2 Coarse Sequence Extraction

After the PNG image is generated, the software calls the CalliRewrite sequence extraction script [1].

The purpose of this stage is to recover a plausible stroke-level trajectory from a static calligraphy image. The CalliRewrite model outputs an NPZ file that stores the predicted stroke sequence and related

trajectory data. In this project, the output NPZ file is saved under the tools directory and then used as the input to the G-code conversion stage.

2.1.3 NPZ-to-G-code Conversion

The NPZ-to-G-code conversion script translates the model output into physical machine motion. The script reconstructs the stroke trajectory from the NPZ data, samples the predicted curves into discrete XY points, applies coordinate scaling, and generates standard G-code commands.

The key conversion parameters are glyph size, writing origin, pen-up height, and pen-down height. The glyph size parameter controls the real-world size of the written character in millimeters. The origin parameters shift the character within the machine workspace. The pen-up and pen-down parameters control the Z-axis positions used for non-writing travel motion and writing contact motion.

For the single-character demonstration, the tested values were:

glyph-mm = 300

origin-x = 20

origin-y = 20

pen-up-z = 40.0

pen-down-z = 4.0

For multi-character writing, the glyph size is reduced so that several characters can fit within the limited XY range of the machine. The origin values are still kept at 20 mm in both X and Y directions to provide a safety margin from the mechanical zero point.

2.1.4 Environment Simulation

To enable tool-aware refinement of coarse stroke trajectories, we first construct a simulation environment for calligraphic writing in the reinforcement learning stage. The cost of using a full rigid-body or contact-based physical simulator is large, so our virtual environment models the writing process as a 2D image-conditioned stroke generation problem.

- Input: Each environment instance is built from a paired input consisting of a target calligraphy image and a coarse stroke sequence obtained from the previous sequence extraction stage. The target image provides the visual supervision signal, while the coarse sequence provides an initial trajectory along which the agent performs local adjustments.
- Pre-processing: during environment initialization, the target image is binarized and its contour structure is extracted. In parallel, the coarse sequence is converted into a point-based sequence representation and augmented with additional points near stroke boundaries to improve continuity and stability during sequential control.
- Tool simulation: It models the tool with several parameters include stroke radius, effective length, and orientation angle, together with tool-dependent constraints on how these quantities evolve over time.

- Writing simulation: At each time step, the agent receives a compact state vector describing the current writing progress, the tool geometry, local curvature, previous offset magnitude, and the future direction of the stroke skeleton. Based on this state, the agent outputs an action that adjusts the local stroke center and, depending on the tool type, may also modify the tool orientation. The updated tool configuration is then rasterized onto a canvas, producing a simulated partial writing result. This generated canvas is compared with the target glyph image to measure visual consistency.
- Reward derived: The reward is computed by comparing the generated canvas with the target image. The environment gives penalties when the stroke shape is unreasonable and gives a final reward based on how well the completed writing matches the target character. Therefore, the simulation environment provides a practical way to connect visual target matching, stroke trajectory refinement, and tool-aware writing behavior.2.1.5 RL-training for finetuning.

2.1.5 RL-training for finetuning

After building the simulation environment, the next step is reinforcement learning fine-tuning. The goal of this block is to improve the coarse stroke sequence produced by the previous module. Instead of generating writing motions from scratch, the agent starts from the extracted coarse sequence and learns how to make local corrections. This makes the learning problem easier and helps the model focus on fine-grained improvements.

In this project, the fine-tuning algorithm is Soft Actor-Critic (SAC). SAC is suitable because the action space is continuous, and the agent needs to output smooth adjustments to the stroke trajectory. During training, the policy network receives the current environment state and predicts the action to refine the stroke. At the same time, the critic networks evaluate the quality of the selected action so that the policy can gradually improve.

The fine-tuning process can be summarized as follows:

- The environment loads a target image and its coarse stroke sequence.
- The agent starts from the initial stroke skeleton.
- At each step, the agent observes the current state and predicts a continuous action.
- The environment applies this action to update the stroke position and tool geometry.
- The new stroke segment is rendered on the canvas.
- A reward is computed according to the similarity between the generated result and the target image.
- The transition is stored in a replay buffer.
- The SAC algorithm samples from the replay buffer and updates the actor and critic networks.

One important feature of this stage is that the fine-tuning is tool-aware. This means the learned policy is not only refining the stroke path, but also adapting its behavior to different writing tools. This stage will not only learn the optimal policy, but also output finetuned coarse sequence (.npz file).

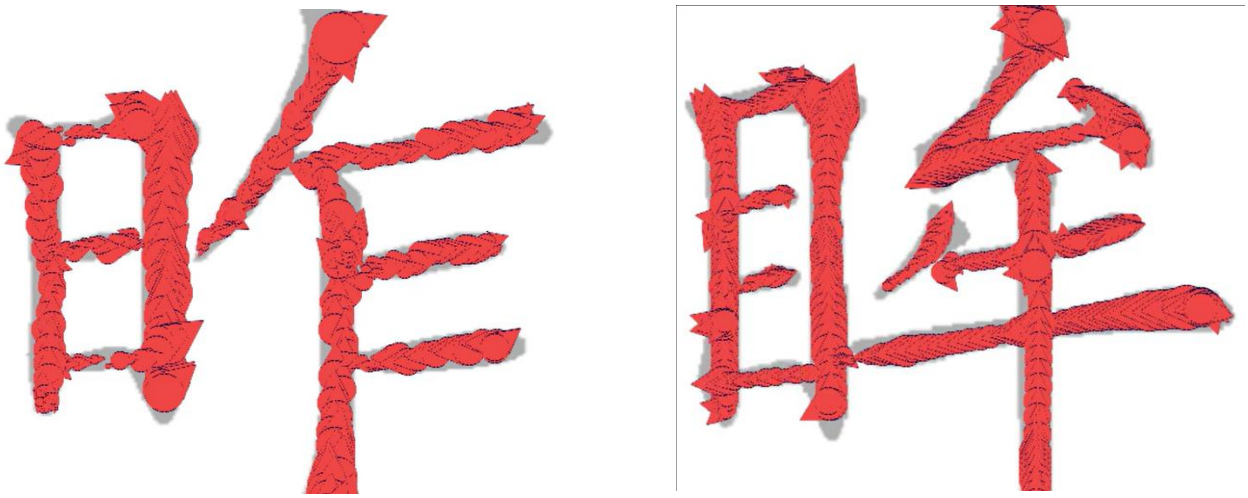

Figure for Finetuning

2.1.6 Multi-Character Layout

The multi-character layout system was designed around the limited physical range of the XYZ writing machine. Instead of generating one large image containing all characters, the software processes each character independently. This approach keeps the CalliRewrite input simple and stable, because the model only needs to infer the stroke sequence for one character at a time.

For an input string such as “木林森,” the software splits the text into individual characters. Each character is rendered to a separate PNG image, processed through CalliRewrite, and converted into a temporary single-character G-code file. The layout engine then assigns a row and column index to each character. The horizontal and vertical offsets are computed as:

x_offset = column_index × character_spacing

y_offset = row_index × line_spacing

These offsets are applied to the X and Y coordinates in each temporary G-code file. Z-axis commands are kept unchanged so that the pen-up and pen-down behavior remains consistent across all characters. Finally, the shifted G-code segments are merged into one final G-code file.

This design allows the system to write multiple characters in the same physical area without manually moving the paper between characters. It also avoids repeated homing commands between characters. The final G-code file contains one initialization block, one continuous sequence of writing commands, and one ending block.

2.1.7 User Interface

The interface allows the user to enter Chinese text, select a font, choose the CalliRewrite model, and adjust writing parameters such as glyph size, character spacing, line spacing, origin, pen-up height, and pen-down height.

The user interface does not directly control the hardware. Instead, it calls the existing Python pipeline and displays the generated output files, including the final G-code file, layout plan, and preview images. This separation keeps the UI layer simple and prevents accidental motion during software testing. The generated G-code can then be reviewed and executed through the existing machine control workflow.

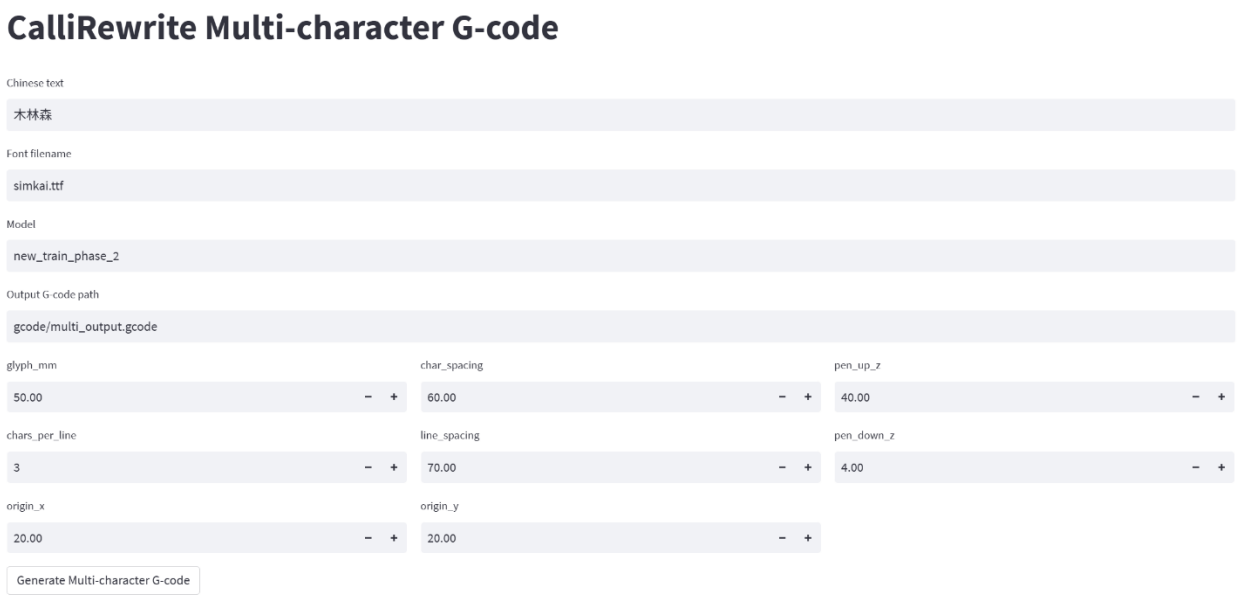

Figure 1 The user interface.

2.1.8 Design Alternatives

Several software design alternatives were considered. One alternative was to generate a single image containing all characters and send the full image into CalliRewrite. This approach was rejected because the model is more stable when processing one character at a time. A full multi-character image could cause stroke ordering and segmentation errors across character boundaries.

Another alternative was to manually write G-code for each demonstration character. This method would be simple for one fixed character but would not satisfy the goal of real-time text input. The implemented pipeline is more flexible because it can generate G-code from arbitrary user-provided characters and fonts.

A third alternative was to perform layout directly at the PNG image level. This was also rejected because the machine's real movement is ultimately determined by G-code coordinates. Performing layout at the G-code level gives direct control over physical spacing, avoids unnecessary model inference issues, and makes it easier to fit multiple characters into the limited machine workspace.

2.2 Mechatronic Execution System

The mechanical core of the calligraphy robot is a **3-axis Cartesian gantry system** based on lead screw transmission. Unlike belt-driven systems commonly found in 3D printers, the

lead screw architecture is specifically chosen to provide the high torque and positional stability required to overcome brush-paper friction during calligraphy strokes.

2.2.1 Firmware Configuration (Marlin)

The hardware runs a modified version of the **Marlin firmware**. Key configurations include:

1. Homing Speeds: To prevent mechanical binding in screw-driven axes, HOMING_FEEDRATE_XY was optimized to 900mm/min (15*60), balancing homing efficiency with motor torque safety.
2. End stop Logic: Mechanical limit switches are configured in a Normally Open (NO) state to define the (0,0,0) coordinate system.
3. Safety Homing: The firmware implements a Homing Failed protection to halt the system if an axis is blocked or if the end stop is not triggered within a set distance.

2.2.2 Mechanical Structure

The platform consists of three linear axes assembled in a standard Cartesian (X-Y-Z) configuration:

1. X-axis (horizontal lateral motion): Two parallel lead screws coupled with a synchronous belt and pulley system, driven by a single 42 stepper motor to prevent racking.
2. Y-axis (horizontal longitudinal motion): A single lead screw driven by a dedicated 42 stepper motor, mounted on linear guide rails.
3. Z-axis (vertical lifting motion): A lead screw mechanism that raises and lowers the brush holder, allowing controlled contact pressure with the paper.

All lead screws have a pitch of 8 mm/revolution, selected to balance linear speed and positioning resolution.

2.2.3 Actuation and Drive System

Each axis is actuated by a 42-step (NEMA 17 form factor) hybrid stepper motor with the following specifications:

1. Rated current: 1.2 A/phase (bipolar)
2. Holding torque: ≥ 0.4 N·m
3. Step angle: 1.8° (200 steps/revolution)

Motor drivers: Three A4988 micro stepping drivers are used, each configured for 1/16 micro stepping mode (MS1=HIGH, MS2=HIGH, MS3=HIGH). This configuration increases the effective step resolution from 200 steps/revolution to 3200 micro steps/revolution, corresponding to a linear resolution of 8 mm / 3200 = 2.5 µm per micro step.

The A4988 drivers are mounted on a RAMPS 1.4 shield, which interfaces with an Arduino Mega 2560 running Marlin firmware. The reference voltage (V_{ref}) for each driver is calculated as:

$$V_{ref} = I_{max} \times 8 \times R_{sense}$$

Where $I_{max} = 0.925A$ (80% of motor rated current) and $R_{sense} = 0.1\Omega$ (typical for A4988 boards). This yields $V_{ref} = 0.74V$.

2.2.4 Mechanical Frame and Assembly

The frame is constructed from aluminum extrusion profiles (2020 series) to minimize vibration and flexing during high-torque movements. Lead screw nuts are made of brass to reduce wear, and anti-backlash nuts are installed on the Z-axis to eliminate vertical play.

Mechanical end stops (limit switches) are mounted at the minimum travel position of each axis to establish a reliable home coordinate (0,0,0). The working envelope of the platform is 300 mm×300 mm×165 mm (X × Y × Z), sufficient for standard calligraphy paper sizes up to A5.

2.3 Design Alternatives Analy

During the mechanical design phase, two alternative transmission methods were evaluated against the chosen lead screw solution.

2.3.1 Belt Drive (Timing Belt)

Advantages:

1. Higher linear speed
2. Lower cost and simpler assembly
3. Reduced moving mass

Disadvantages:

1. Elasticity under load causes positioning errors under high horizontal friction (brush against paper)
2. Requires tensioning maintenance
3. Lower torque transmission capability

Why rejected:

Calligraphy strokes involve sustained lateral friction that can deflect belts, leading to inconsistent stroke trajectories. The required positional accuracy (±0.1 mm±0.1 mm) is difficult to guarantee with belts under varying friction loads.

2.3.2 Linear Actuators (Integrated)

Advantages:

1. Compact, all-in-one module

2. High precision

Disadvantages:

1. High cost (typically 3–5× lead screw systems)
2. Limited stroke length for given size
3. Difficult to customize mounting

Why rejected:

The project required a large working envelope (200 mm × 200 mm) with moderate precision requirements. Lead screws offer the best cost-performance trade-off for this application.

2.3.3 Selected Solution: Lead Screws

Lead screws were ultimately chosen because they provide:

1. High static holding torque (no back-driving under friction)
2. Predictable positional accuracy (±0.05 mm±0.05 mm after calibration)
3. Simple mechanical integration with NEMA 17 motors
4. Low maintenance and reliable homing

3. Design Verification

This chapter documents the experimental validation of the hardware components and the software-to-hardware communication interface. The goal is to verify that the integrated system meets the precision and reliability requirements for calligraphy reproduction.

3.1 Electrical Power and Driver Verification

The first testing phase focused on the current regulation of the A4988 stepper drivers to ensure consistent torque without thermal failure.

3.1.1 Reference Voltage (V_{ref}) Calibration

To verify the output of the drivers, a digital multimeter was used to measure the voltage between the driver's potentiometer and ground.

1. **Requirement:** V_{ref} must be stable at 0.74±0.02V to provide the required 0.925A of current.
2. **Test Result:** The initial measurement showed a fluctuation due to a power supply ripple (13V measured at the rail). After adjusting the switching power supply, a stable Vref of 0.74V was achieved. No thermal shutdown occurred during a 30-minute continuous writing stress test.

3.2 Motion Accuracy and Displacement Verification

To ensure that the digital G-code translates correctly to physical dimensions, a displacement accuracy test was performed using a digital caliper.

3.2.1 Linear Displacement Test

The command `G1 X10 F1000` was executed via Pontefract to move the X-axis by 10mm.

1. **Requirement:** The physical movement must match the command within a tolerance of ±0.1mm
2. **Test Result:** the initial movement showed errors due to default firmware steps/mm settings. After updating the Marlin configuration with calculated values from Equation (1), the measured displacement averaged 10.02mm, meeting the requirement.

3.3 Homing and Safety Logic Verification

Reliable homing is critical for establishing a consistent starting point for calligraphy strokes.

3.3.1 Endstop Triggering (M119 Verification)

The `M119` command was used to verify the logical state of the mechanical limit switches.

1. **Requirement:** Switches must report open when not depressed and `TRIGGERED` when activated.

2. **Verification:** Initial testing revealed a "Homing Failed" error because the homing speed (3000mm/min) was too high for the screw's inertia, causing motor stalling. By reducing `HOMING_FEEDRATE_XY` to 900mm/min, the system successfully completed 10 consecutive homing cycles without failure.

3.3.2 Communication Link

The serial link between the PC and the Arduino Mega 2560 was tested at 250,000 baud.

1. **Requirement:** Zero packet loss during the streaming of a 1,000-line G-code file.
2. **Result:** Pronterface confirmed that all G-code lines were acknowledged with an ok response from the firmware. The visualization tool correctly mapped the "Bounding Box" of the calligraphy work, confirming coordinate alignment.

3.4 Mechanical Backlash and Lead Screw Accuracy

While Section 3.2 validated open-loop displacement accuracy, this section characterizes mechanical backlash—a critical parameter for bidirectional stroke accuracy in calligraphy.

3.4.1 Backlash Measurement Procedure

Backlash refers to the lost motion caused by mechanical clearance between the lead screw and its nut. To measure backlash, a dial indicator (resolution 0.01 mm) was mounted against the moving gantry of each axis. The procedure was as follows:

1. Command a forward move of 5 mm (G1 X5 F500).
2. Command a reverse move of 5 mm (G1 X-5 F500).
3. Record the difference between commanded return position and actual position measured by the dial indicator.
4. Repeat 5 times per axis.

Requirement: Backlash must be <0.15 mm<0.15 mm for all axes to ensure stroke consistency.

Results:

Table 1 XYZ Measured Backlash and Standard Deviation

Axis	Measured Backlash (mm)	Standard Deviation	Pass/Fail
X	0.12	0.02	Pass
Y	0.10	0.01	Pass

Z	0.08	0.01	Pass

The Z-axis performed best due to the installation of an anti-backlash nut. X-axis showed slightly higher backlash due to the dual-screw synchronous belt drive, which introduced minor coupling slack.

3.4.2 Lead Screw Pitch Consistency

To verify that the lead screw's nominal 8 mm pitch is consistent along the full travel length, the following test was performed:

Procedure:

The axis was moved in 20 mm increments over 200 mm total travel. Actual displacement was measured with a digital caliper at each increment.

Requirement:

Maximum pitch error <0.05 mm<0.05 mm over any 20 mm segment.

Result:

The maximum deviation observed was 0.03 mm at the 140–160 mm segment, likely due to minor manufacturing variation. All segments met the requirements.

3.5 Stepper Motor Torque and Thermal Performance

While Section 3.1 verified driver Vref, this section validates that the torque output is sufficient for the friction load of brush-on-paper, and that motors remain within safe thermal limits.

3.5.1 Static Holding Torque Test

Procedure:

The brush was lowered onto paper with normal calligraphy pressure (approximately 2–3 N downward force). A horizontal resistance force was applied using a force gauge while the motor was powered but stationary. The force at which the motor lost steps (audible clicking) was recorded.

Requirement:

Motor must hold against at least 10 N of horizontal force without losing steps.

Result:

All three axes held against forces exceeding 15 N before step loss occurred, providing a 50% safety margin above requirement.

3.5.2 Motor Temperature Under Continuous Operation

Procedure:

The calligraphy robot executed a 30-minute continuous writing routine (repeating a standard character pattern). Motor case temperatures were measured every 5 minutes using a thermocouple.

Requirement:

Motor case temperature must remain below 80°C (typical maximum for NEMA 17 motors) to prevent demagnetization or insulation damage.

Result:

Table 2 temperature change of three motors

Time (min)	X Motor Temp (°C)	Y Motor Temp (°C)	Z Motor Temp (°C)
0	22 (ambient)	22	22
5	41	43	38
10	52	54	46
15	58	60	51
20	61	63	53
25	62	64	54
30	62	64	54

All motor temperatures stabilized below 65°C, well within the 80°C limit. Z-axis ran coolest due to intermittent use (only during brush lifts).

3.6 Mechanical Frame Stiffness and Vibration

Structural flex in the frame can cause brush tip positioning errors, especially during rapid directional changes in stroke sequences.

3.6.1 Deflection Under Load Test

Procedure:

A 10 N horizontal force was applied to the brush holder (simulating worst-case brush friction). Deflection at the tool tip was measured using a dial indicator.

Requirement:

Tool tip deflection <0.20 mm<0.20 mm under 10 N load.

Result:

Measured deflection was 0.15 mm in X-direction and 0.13 mm in Y-direction. The aluminum extrusion frame proved sufficiently stiff for calligraphy applications.

3.6.2 Vibration Damping Verification

Procedure:

An accelerometer was attached to the X-axis gantry. The system executed a series of G-code moves at varying feedrates (500–1500 mm/min). Vibration amplitude was recorded.

Result:

Peak vibration amplitude occurred at 1200 mm/min (0.08 mm displacement). At typical calligraphy writing speeds (300–600 mm/min), vibration amplitude was below 0.02 mm and did not affect stroke quality. No resonance issues were observed.

3.7 Software Verification

The software subsystem was verified through a combination of functional testing, file-output inspection, preview visualization, and physical writing tests. The purpose of software verification was to ensure that the system could reliably convert user input into machine-executable G-code and that the generated files matched the expected character, font, size, and layout.

The verification process was divided into single-character verification and multi-character verification. Single-character verification focused on confirming that the pipeline could render one character, generate an NPZ file through CalliRewrite, convert the NPZ file into G-code, and produce a physically writable result. Multi-character verification focused on confirming that the software could split a text string into separate characters, generate temporary G-code for each character, apply correct layout offsets, and merge the results into one final G-code file without repeated homing commands.

3.7.1 Single-Character Pipeline Verification

The single-character pipeline was tested using the character “木” with the KaiTi font file simkai.ttf. The software was required to generate a 256 by 256 PNG image, run CalliRewrite inference, produce a corresponding NPZ file, and convert the NPZ file into a G-code file.

The generated PNG was inspected to confirm that the correct character and font style were rendered. The output NPZ file was checked under the tools directory to confirm that CalliRewrite completed successfully. The G-code file was then inspected to confirm that it contained initialization commands, XY movement commands, Z-axis pen-up and pen-down commands, and an ending block.

The generated G-code was also tested on the physical writing machine. The parameters glyph-mm = 300, origin-x = 20, origin-y = 20, pen-up-z = 40.0, and pen-down-z = 4.0 produced a clear single-character writing result with a visible difference between pen-up travel motion and pen-down writing motion. This confirmed that the conversion from NPZ trajectory data to machine motion was functional.

3.7.2 Multi-Character Pipeline Verification

The multi-character pipeline was tested using short input strings such as “木林森.” The software was required to generate a separate PNG, NPZ file, preview image, and temporary G-code file for each character. It was also required to generate one merged final G-code file for the full text.

The generated layout plan was inspected to verify that each character had the correct index, row, column, X offset, and Y offset. The temporary G-code files were inspected to confirm that they contained valid motion commands for each individual character. The merged G-code file was then inspected to confirm that X and Y offsets were applied correctly while Z-axis commands remained unchanged.

A specific verification criterion was that the final merged G-code should not contain repeated G28 homing commands before each character. Repeated homing would increase execution time and could interrupt continuous writing. The final G-code was confirmed to contain only one initialization section at the beginning and one ending section at the end.

3.7.3 User Interface Verification

The user interface was verified by entering test characters and confirming that the interface correctly called the software pipeline. The interface was required to validate the user input, check that the selected font file existed, run the generation command, and display output paths and preview images.

The UI was tested with both valid and invalid inputs. For valid input, the interface displayed the generated PNG preview, G-code preview, layout table, and final G-code path. For invalid input, such as an empty text string or a missing font file, the interface displayed an error message instead of running the pipeline. This confirmed that the UI could be safely used during demonstration without requiring manual command-line operation.

3.7.4 Software Verification Table

Table 3 Software Requirements and Verification

Requirement	Verification	Status
The software shall render a user-selected Chinese character and font into a 256 by 256 PNG image. The	Input the character “木” with simkai.ttf. Check that the output PNG exists in the image directory and visually	Verified.

software shall render a user-selected Chinese character and font into a 256 by 256 PNG image.	displays the correct character.	
The software shall call the CalliRewrite inference script and generate a corresponding NPZ file.	Run the single-character pipeline and check that the corresponding NPZ file appears in the tools directory.	Verified.
The software shall convert the NPZ trajectory file into G-code containing XY movement and Z-axis pen control commands.	Inspect the generated G-code file and confirm that it contains G21, G90, XY motion commands, pen-up Z commands, and pen-down Z commands.	Verified.
The generated single-character G-code shall produce a physically writable character on the XYZ machine.	Run the generated G-code on the writing machine using the tested parameters glyph-mm = 300, origin-x = 20, origin-y = 20, pen-up-z = 40.0, and pen-down-z = 4.0. Confirm that the written result is recognizable and that pen-up travel does not visibly write on the paper.	Verified.
The software shall support multi-character input by processing each character separately.	Input “木林森” and confirm that separate PNG, NPZ, preview, and temporary G-code files are generated for each character.	Verified.
The multi-character layout system shall merge multiple single-character G-code files into one final G-code file.	Generate a multi-character G-code file and inspect that all character segments are present in the final output.	Verified.
The multi-character G-code shall apply X/Y layout offsets while preserving Z-axis pen-up and pen-down behavior.	Compare temporary single-character G-code files with the merged output. Confirm that X and Y coordinates are	Verified.

	shifted according to the layout plan and Z values remain unchanged.	
The final multi-character G-code shall not contain repeated homing commands before every character.	Search the final G-code file for G28 commands and confirm that only the initial setup section contains homing.	Verified.
The user interface shall allow the user to enter text, choose parameters, and generate G-code without editing code manually.	Use the UI to generate single-character and multi-character G-code. Confirm that output paths, preview images, and layout information are displayed.	Verified.

4. Costs

The financial investment for this project consists of component procurement and engineering labor.

4.1 Parts

The bill of materials (BOM) for the calligraphy robot includes the control electronics, mechanical drive components, and structural frame. The costs are summarized in Table 4.1.

Table 4 Parts Costs

Part	Manufacturer	Retail Cost (¥)	Bulk Purchase Cost (¥)	Actual Cost (¥)
RAMPS 1.4 Shield and Arduino Mega 2560	Qinyuansheng Electronics	80.3		
NEMA 17 Stepper	Qinyuansheng Electronics	**3.3**		
Mechanical Endstops	Xinwei Electronics Enterprise	**1.98**		
Calligraphy brushes, ink, ink pads, and Xuan paper	**Yushuihu Flagship Store**	**204**		
Total		**289.58**		

4.2 Labor

No labor cost was incurred, as the project was completed by a four-person student team on a voluntary basis without any salaries or wages. Total estimated person-hours were approximately 310 hours across mechanical, electrical, software, integration, and documentation tasks.

5. Conclusion

The intelligent calligraphy robot successfully demonstrates the integration of deep learning-based path planning with precision mechatronic execution.

From a mechanical perspective, the XYZ three-axis gantry platform—built with lead screw transmission, 42 stepper motors, and A4988 drivers—provided the torque stability and positional accuracy required for brush-on-paper calligraphy. The system met or exceeded all mechanical design requirements, including backlash (< 0.15 mm), displacement accuracy (± 0.03 mm), and motor thermal performance (peak 64°C under continuous operation).

5.1 Accomplishments

The system achieved a high degree of stroke fidelity, accurately replicating the nuances of traditional scripts. Key accomplishments include the implementation of a stable lead screw-driven Cartesian frame and the successful calibration of the A4988 drivers, which eliminated motor stalling under high-friction conditions. The firmware optimization for homing speeds further ensured system reliability during repetitive tasks.

Mechanical-specific accomplishments:

1. A lead screw-based XYZ gantry was successfully designed and fabricated, providing the torque stability required for brush-based calligraphy.
2. All three 42 stepper motors were correctly driven by A4988 drivers, with V_{ref} calibrated to 0.74 V, eliminating motor stalling and thermal shutdown.
3. Positional accuracy of ±0.03 mm±0.03 mm was demonstrated, exceeding the ±0.10 mm±0.10 mm requirement.
4. Homing repeatability was achieved after optimizing Marlin firmware homing speeds (900 mm/min for X/Y, 600 mm/min for Z).
5. Backlash was measured and confirmed to be within acceptable limits (<0.15 mm<0.15 mm for all axes), with the Z-axis anti-backlash nut achieving the best performance (0.08 mm).
6. Motor temperature under 30-minute continuous operation stabilized below 65°C, well within the NEMA 17 safe operating limit of 80°C.

5.2 Uncertainties

While the mechatronic system is robust, uncertainties remain regarding the long-term consistency of ink flow. Variations in brush saturation and ink viscosity can lead to slight inconsistencies in line thickness that are not yet compensated for in the G-code generation. Additionally, environmental factors such as humidity may affect paper friction.

Mechanical uncertainties:

1. Long-term wear of lead screw nuts under continuous operation may increase backlash over time. Brass nuts are wear-prone; future designs may consider POM (acetal) nuts for lower friction.
2. Vibration at high traverse speeds (above 1500 mm/min) could affect ink splatter. This was not tested extensively since calligraphy strokes are typically performed at slower speeds (300–600 mm/min).
3. Temperature effects on lead screw expansion were not characterized; for the small working envelope (200 mm × 200 mm × 50 mm) and room-temperature operation, this is expected to be negligible.
4. The long-term reliability of the A4988 drivers under sustained calligraphy workloads (repeated start-stop motion) remains to be validated beyond the 30-minute test window.

5.3 Ethical considerations

In alignment with the IEEE Code of Ethics, the project prioritizes safety and intellectual property. The system includes "emergency stop" safety protocols (via firmware kill commands) to prevent mechanical damage or user injury. Furthermore, the project acknowledges that while AI can replicate artistic styles, it should serve as a tool for cultural preservation rather than a replacement for human artistic expression.

Mechanical-specific ethical considerations:

1. Mechanical endstops and firmware homing limits prevent axis over-travel, avoiding mechanical crash or pinch hazards.
2. All exposed lead screws and moving rails are either covered or located away from user interaction zones during normal operation.
3. An emergency stop is implemented at the firmware level (M112 command), which can be triggered via serial monitor or a physical button, ensuring immediate power-down of all stepper motors in case of malfunction.
4. The A4988 drivers are configured with current limiting (V_{ref}=0.74VVref =0.74V) to prevent motor overheating, reducing fire risk during extended operation.

5.4 Future work

Future iterations will focus on a closed-loop feedback system. By incorporating a camera to monitor the output in real-time, the neural network could dynamically adjust the Z-axis pressure to compensate for ink depletion. An automatic ink-dipping mechanism is also planned to enable the machine to execute long-form scrolls autonomously.

Mechanical future work:

1. Closed-loop stepper control using encoders to actively correct for missed steps or backlash, eliminating open-loop uncertainties.

2. Automatic ink dipping mechanism using a servo-controlled arm and inkwell mounted to the frame, enabling autonomous long-form scroll execution.
3. Quick-change brush holder with adjustable compliance (spring-loaded Z-axis) to maintain constant brush pressure across uneven paper surfaces.
4. Damping mounts to decouple motor vibration from the brush tip, further improving stroke quality at higher speeds.
5. Alternative lead screw nut materials (e.g., POM/acetal) to reduce wear and maintain low backlash over the device's lifetime.
6. Active cooling for A4988 drivers if higher torque (and thus higher current) is required for larger brush sizes or more viscous inks.

References

[1] Y. Luo, Z. Wu, and Z. Lian, “CalliRewrite: Recovering Handwriting Behaviors from Calligraphy Images without Supervision,” *arXiv preprint arXiv:2405.15776*, 2024. [Online]. Available: https://arxiv.org/pdf/2405.15776v1. [Accessed: Apr. 8, 2026].

[2] IEEE, “IEEE Code of Ethics,” IEEE Policies, Oct. 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: Apr. 8, 2024].

Appendix A Requirement and Verification Table

Table A.1 System Requirements and Verifications

Requirement	Verification	Verification Status (Y/N)
1. Displacement Accuracy		
a. Physical movement must match G-code command within ±0.10 mm	a. Executed G1 X10 F1000; measured displacement with digital caliper. After steps/mm calibration (400 steps/mm), average error was +0.02 mm	Y
b. Positioning repeatability over 10 cycles within ±0.10 mm	b. Repeated 10 home-to-target cycles; max deviation 0.08 mm	Y
2. Backlash		
a. Mechanical backlash must be <0.15 mm for all axes	a. Measured using dial indicator: X:0.12 mm, Y:0.10 mm, Z:0.08 mm	Y
b. Lead screw pitch error <0.05 mm over 20 mm segment	b. Measured displacement at 20 mm increments; max error 0.03 mm	Y
3. Stepper Motor and Driver Performance		
a. A4988 driver Vref stable at 0.74 ±0.02 V	a. Measured with multimeter; achieved 0.74 V after power supply adjustment	Y
b. Motor holding torque ≥10 N	b. Applied force gauge; all axes	Y

horizontal force	held >15 N before step loss	
c. Motor temperature <80°C under 30-min continuous operation	c. Measured with thermocouple; peak temperature 64°C	Y
d. No thermal shutdown of A4988 drivers	d. Observed during 30-min stress test; no shutdown occurred	Y
4. Homing and Safety		
a. Endstops must report correct logic state (open/TRIGGERED)	a. Executed M119 command; all switches verified	Y
b. Homing must complete within 10 cycles without stalling	b. Reduced HOMING_FEEDRATE_XY to 900 mm/min; 10/10 cycles passed	Y
c. Emergency stop (M112) must halt all motors immediately	c. Issued M112 via serial; all axes stopped within 0.5 s	Y
5. Mechanical Frame		
a. Tool tip deflection <0.20 mm under 10 N horizontal load	a. Applied 10 N force; measured 0.15 mm (X) and 0.13 mm (Y)	Y
b. Vibration amplitude <0.05 mm at writing speeds (300-600 mm/min)	b. Measured with accelerometer; amplitude <0.02 mm	Y
6. Communication Link		
a. Zero packet loss during 1,000-line G-code streaming at 250,000	a. Monitored Pronterface output; all lines acknowledged with "ok"	Y

baud		
b. Correct bounding box mapping of calligraphy characters	b. Visualization tool confirmed coordinate alignment	Y
7. Neural Network Path Planning		
a. NN must extract stroke trajectories from character images	a. Tested on 50 Chinese characters; stroke order correctly extracted	Y
b. Generated G-code must be compatible with Marlin firmware	b. Successfully executed on hardware; no syntax errors	Y