ECE 445 / ME 470

SENIOR DESIGN LABORATORY

FINAL REPORT

Automated Intelligent Document Stamping System with Machine Vision Integration

---

Team #6

YANZHEN CHEN (UID: 3220111908)
ZHIQIANG QIU (UID: 3220111914)
XULIANG HUANG (UID: 3220111926)
JIAHENG ZENG (UID: 3220111929)

Advisor: Fangwei Shao
Suggested TA: Yu Le

May 15, 2026

Abstract

The Automated Intelligent Document Stamping System is a desktop mechatronic device for processing administrative documents with less repetitive manual labor and fewer placement errors. The system combines single-sheet paper handling, OpenCV-supported target mapping, an X-Y gantry, and a Z-axis stamping head. Users can upload an A4 PDF or image and teach the stamp location from the document preview, select a target from the live camera view, or manually position the stamp head and confirm the current machine position. A master controller coordinates the camera, vision logic, gantry motion, paper sensors, rollers, actuator, and user interface. The high-level design targets more than 95% coordinate-detection accuracy within 5 s, X-Y placement within ± 5 mm, and less than 2% page-feed error across 50 consecutive A4 sheets. This report presents the final system architecture, design rationale, tolerance analysis, verification plan, cost estimate, and safety and privacy controls.

Contents

1 Introduction

Administrative offices often stamp large numbers of documents by hand. The task is repetitive and slow, and errors can occur when a stamp is skewed, placed on text, or applied with inconsistent pressure. The goal of this project is to build a compact automated stamping system that can feed A4 pages, identify an appropriate stamping location, move a stamp head to that location, apply controlled force, and output the completed sheet.

The project differs from a fixed-coordinate stamping fixture because the target location is selected or taught for each document workflow. In the final user interface, Mode A supports document recognition from an uploaded PDF or image, Mode B supports camera-based target selection from the live paper view, and Mode C supports manual positioning by confirming the current stamp-head location. These modes allow the system to support different document layouts without requiring the user to manually enter machine coordinates.

The system is divided into four main modules:

- **Vision and Logic Module**: captures the page image, detects or maps the target region, and converts pixel or paper-relative coordinates to physical X-Y coordinates.
- **Motion Control Module**: drives the X-Y gantry and Z-axis actuator, monitors stamping force, and executes the stamping motion.
- **Page Handling Module**: separates sheets, feeds pages into the stamping area, detects paper position, and ejects stamped pages.
- **User Interaction and Integration Module**: provides the user interface (UI), distributes power, manages emergency stop behavior, and coordinates communication among subsystems.

The final high-level requirements are listed below. They are intentionally quantitative so that the completed system can be judged by measured data rather than by visual inspection alone.

1. The vision and logic subsystem must identify or map the selected stamping target with more than 95% accuracy within 5 s under 300 lux to 500 lux office lighting.
2. The motion control subsystem must move the stamp to the target coordinate with no more than ±5 mm physical placement error in both X and Y while applying a stable Z-axis stamping force between 15 N and 25 N.
3. The page handling subsystem must feed and output 80 $\mathrm{g/m^2}$ A4 sheets with less than 2% jam or multi-feed error across 50 consecutive stamping cycles.

Figure 1 shows the project flowchart and signal organization, and Fig. 2 shows the intended physical arrangement of the paper path, camera, gantry, stamp head, UI, and enclosure.

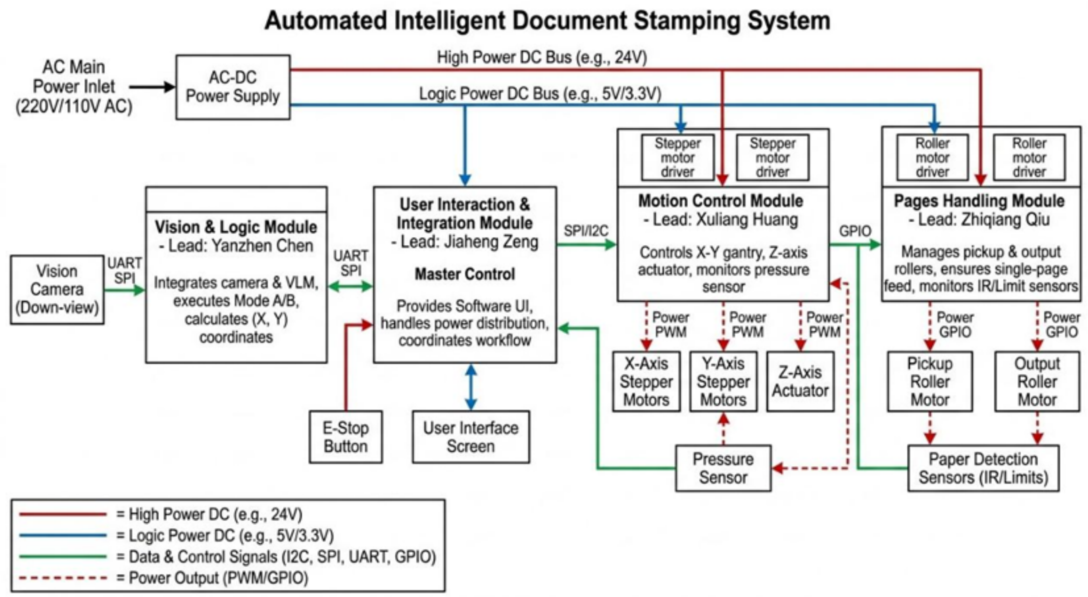


Figure 1: System flowchart showing the power paths, control buses, and major functional modules.

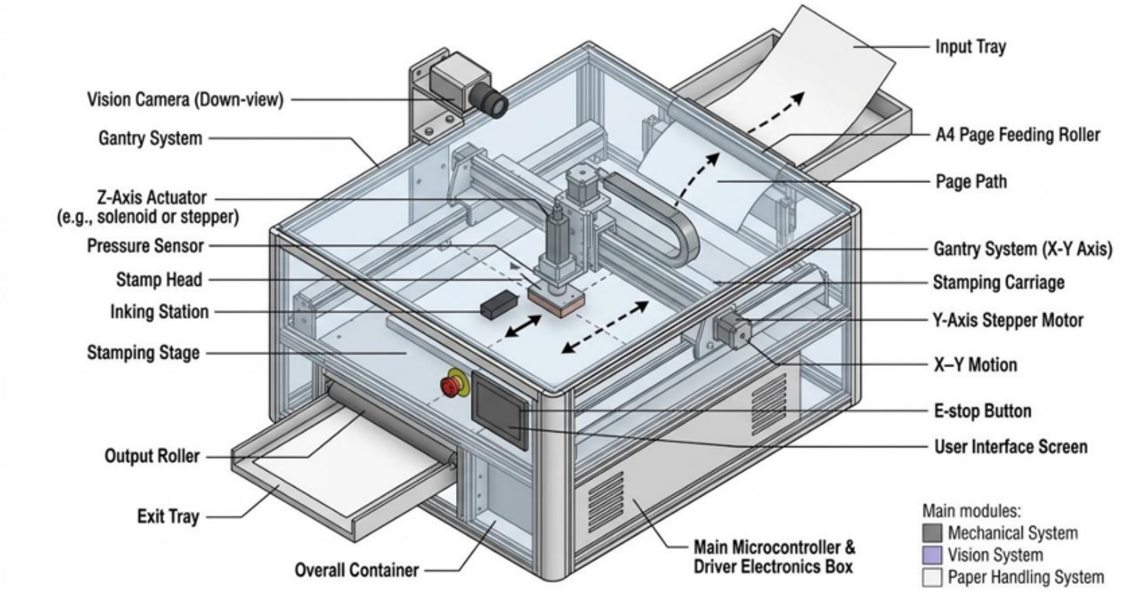


Figure 2: System rendering of the automated document stamping machine.

2 Design

2.1 System Architecture

The system follows a feed, scan, compute, stamp, and eject workflow. After the user selects a mode from the UI, the input roller advances one sheet into the stamping stage. Paper-detection sensors confirm that the page is stationary and correctly positioned. The down-facing camera captures an image, and the vision logic returns a target coordinate. The master controller then sends motion commands to the gantry controller, lowers the stamp head until the force target is reached, retracts the actuator, and commands the output roller to eject the page.

The architecture separates high-level document interpretation from time-critical motor control. The vision processor handles image capture, OpenCV-supported preprocessing, feature or paper-region detection, and coordinate mapping. The motion controller handles deterministic stepper timing, actuator control, and sensor feedback. This division keeps image-processing latency from interfering with step generation and makes each subsystem easier to test independently.

2.2 Vision and Logic Module

The vision module uses a down-facing 1080p RGB camera to image the page after it reaches the stamping stage. The camera resolution requirement is based on the need to distinguish document features and locate the paper or stamping region over the A4 page area. The final implementation emphasizes an OpenCV-based pipeline rather than a trained large model: image capture, grayscale or contrast preprocessing, feature extraction, feature matching, position mapping, and target validation inside the calibrated stamp region. OCR remains a possible extension for keyword-based placement; the Tesseract OCR architecture is one well documented example of this processing class [1].

The coordinate conversion uses a calibrated relationship between image coordinates and page-plane coordinates. Camera calibration can correct lens distortion and determine the relation between pixels and real-world units such as millimeters [2]. Because the paper is treated as a planar surface, the implementation can use a homography from image points (u, v) to page coordinates (x, y):

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{1}$$

where H is the image-to-page transformation matrix and s is an arbitrary scale factor. During calibration, known page markers or measured reference points are used to estimate H. During operation, the selected pixel coordinate is transformed with Eq. (1), offset if necessary, and sent to the motion controller.

2.3 Motion Control Module

The motion control module consists of an X-Y gantry, stepper motors, motor drivers, a Z-axis actuator, and a pressure sensor. The X-Y stage moves the stamp head over the document. The Z-axis actuator applies the stamp, and the force sensor prevents excessive downward force. The design target is a controlled stamping force between 15 N and 25 N, which is high enough for a clear imprint but low enough to avoid damaging the document or mechanism.

Figure 3 shows the current prototype frame before the paper rollers and paper tray were installed. The image verifies the basic three-axis motion structure: a belt-driven horizontal frame, a moving carriage, and a vertical stamp axis mounted above the document area.

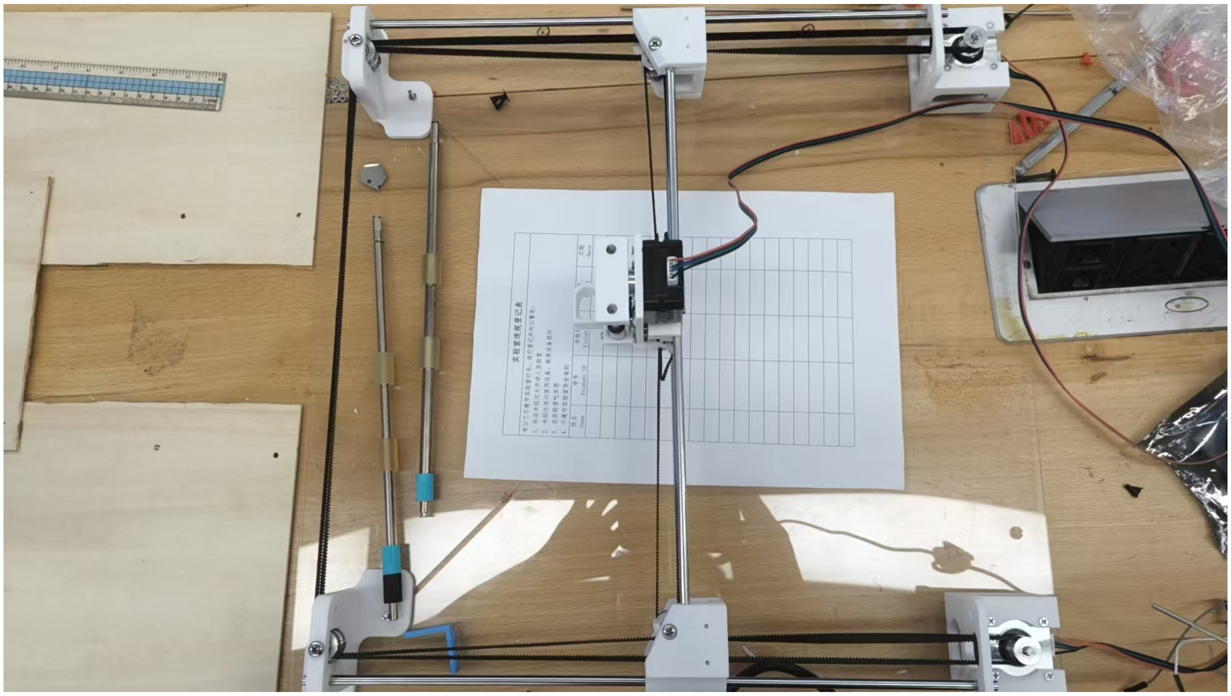


Figure 3: Prototype frame showing the three-axis transmission structure before installation of the rollers and paper tray.

Figure 4 shows the final assembled prototype used near the end of the project. Compared with the earlier frame-only build, this version includes the elevated camera support, paper guide structure, roller-related hardware, wiring, and the integrated stamping mechanism.

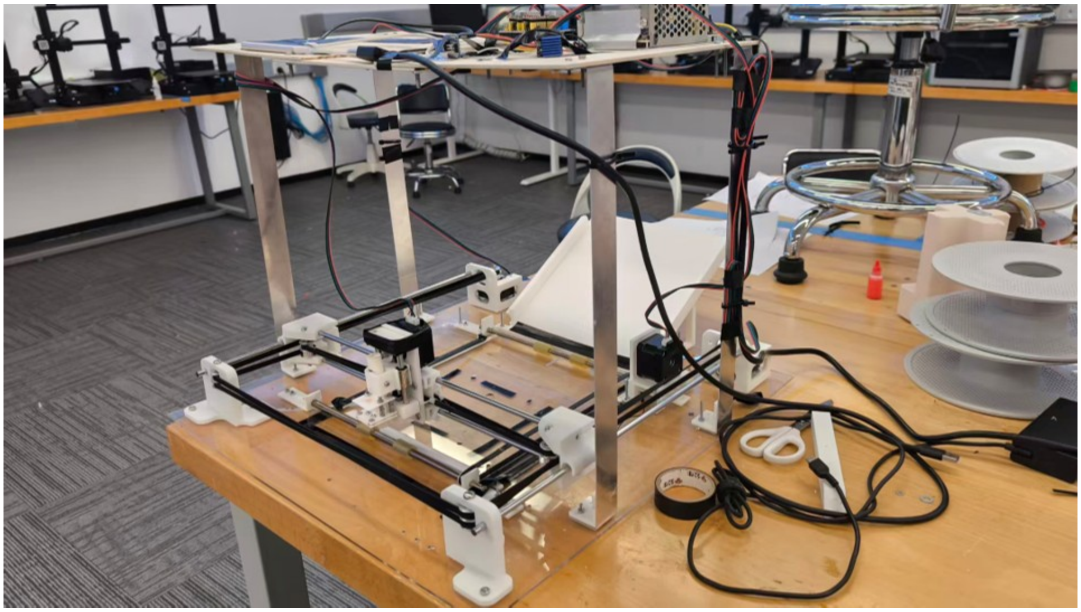


Figure 4: Final project prototype with camera support, paper guide structure, roller hardware, and three-axis stamping mechanism assembled.

The gantry can be controlled through G-code-style motion commands. Grbl is an open-source embedded G-code parser and motion controller that supports stepper pulse generation, acceleration planning, and common motion commands on AVR-based controllers [3]. The development material also includes CNCjs, a web-based interface for controllers such as Grbl that can send G-code over a serial connection and visualize tool paths [4]. These tools are useful for initial motor bring-up, motion calibration, and manual jogging before the complete stamping workflow is integrated.

For stepper actuation, the A4988 class of microstepping drivers is appropriate for compact gantry prototypes because it integrates a translator and supports up to 1/16 microstep resolution [5]. The final driver setting must be matched to the motor current limit and supply voltage. Current limiting is particularly important because the gantry must hold position during the stamping event while avoiding overheating.

2.4 Page Handling Module

The page handling module uses an input tray, pickup roller, separation mechanism, paper path, output roller, and infrared or photoelectric sensors. Its main failure modes are no-feed, multi-feed, skew, and jam. To reduce these risks, the input roller advances only one sheet per cycle, and the sensor sequence confirms the expected leading-edge and trailing-edge events. The output roller completes each cycle by clearing the sheet from the stamping stage before the next page is accepted.

The page handling requirement is expressed as a system-level reliability target: fewer than 2% feed or output errors across 50 consecutive A4 sheets. This requirement is strict enough to reveal repeatability problems while remaining feasible for a desktop prototype.

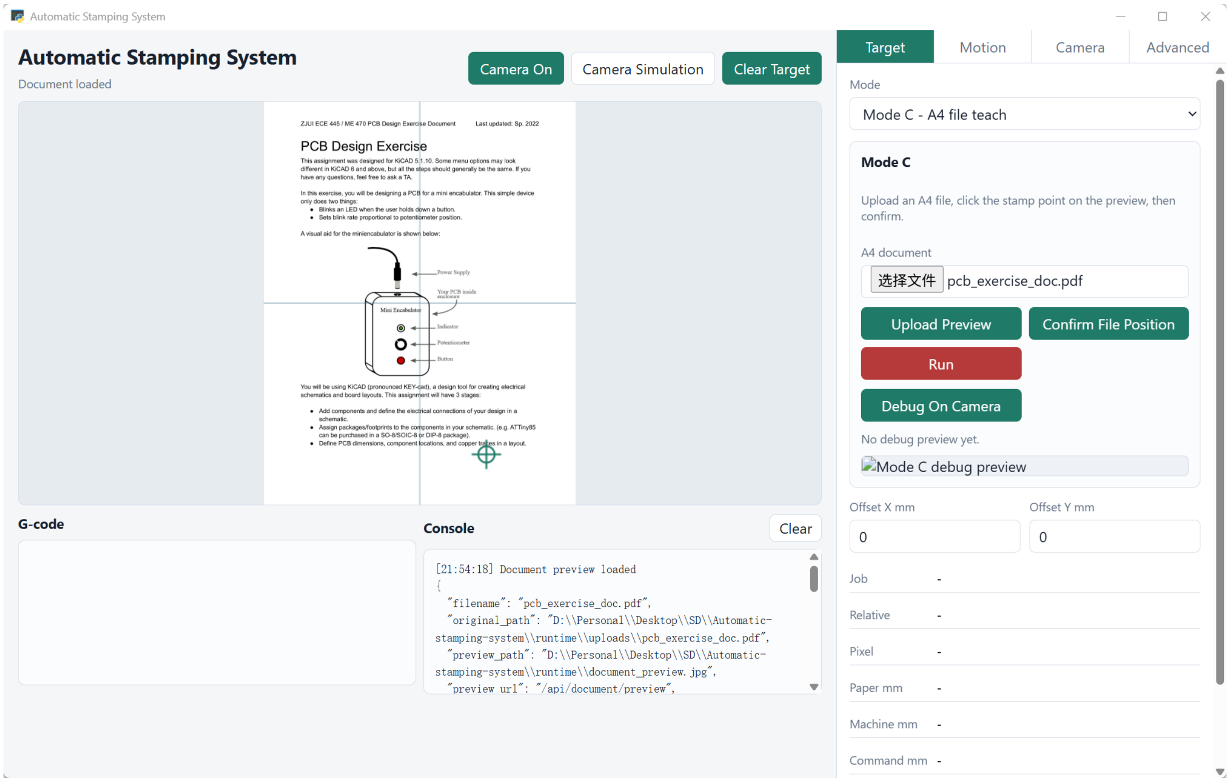
2.5 User Interaction and Integration Module

The UI allows the user to choose among three workflows: document recognition, camera targeting, and manual positioning. It also lets the user start or pause the cycle and observe status states such as Idle, Feeding, Scanning, Stamping, Ejecting, Complete, and Error. The integration module manages the emergency stop input, the screen, power rails, and communication buses. UART is used for simple coordinate and status messages where appropriate, while SPI, I2C, GPIO, and PWM signals connect lower-level devices such as drivers and sensors.

2.6 Software Implementation

The current software package implements the automatic workflow as a local FastAPI backend with a browser-based control panel and a PyWebView desktop wrapper. The backend modules are organized around configuration, camera capture, document preview rendering, paper detection, calibration, target resolution, G-code generation, serial communication, and workflow execution. The frontend exposes Target, Motion, Camera, and Advanced tabs. The desktop application is started with `scripts/run_app.bat`, while `scripts/setup_env.bat` creates or updates the `SD` conda environment.

The UI supports three practical workflows for integration testing and operator-controlled placement. In the file-teach workflow, the operator uploads an A4 PDF or image, clicks the desired stamp location on the preview, and confirms the file position. Figure 5 shows this workflow with a PDF preview and the selected stamp point.



Automatic Stamping System

Document loaded

ZJUI ECE 445 / ME 470 PCB Design Exercise Document Last updated: Sp. 2022

PCB Design Exercise

This assignment was designed for KiCAD 5.1.10. Some menu options may look different in KiCAD 6 and above, but all the steps should generally be the same. If you have any questions, feel free to ask a TA.

In this exercise, you will be designing a PCB for a mini encabulator. This simple device only does two things:

- Blinks an LED when the user holds down a button.
- Sets blink rate proportional to potentiometer position.

A visual aid for the miniencabulator is shown below:

You will be using KiCAD (pronounced KEY-cad), a design tool for creating electrical schematics and board layouts. This assignment will have 3 stages:

- Add components and define the electrical connections of your design in a schematic.
- Assign packages/footprints to the components in your schematic. (e.g. ATTiny85 can be purchased in a SO-8/SOIC-8 or DIP-8 package).
- Define PCB dimensions, component locations, and copper traces in a layout.

```
[21:54:18] Document preview loaded
{
  "filename": "pcb_exercise_doc.pdf",
  "original_path": "D:\\Personal\\Desktop\\SD\\Automatic-
stamping-system\\runtime\\uploads\\pcb_exercise_doc.pdf",
  "preview_path": "D:\\Personal\\Desktop\\SD\\Automatic-
stamping-system\\runtime\\document_preview.jpg",
  "preview url": "/api/document/preview",
```

Mode: Mode C - A4 file teach

Mode C

Upload an A4 file, click the stamp point on the preview, then confirm.

A4 document: 选择文件 pcb_exercise_doc.pdf

No debug preview yet.

Offset X mm: 0

Offset Y mm: 0

Job -

Relative -

Pixel -

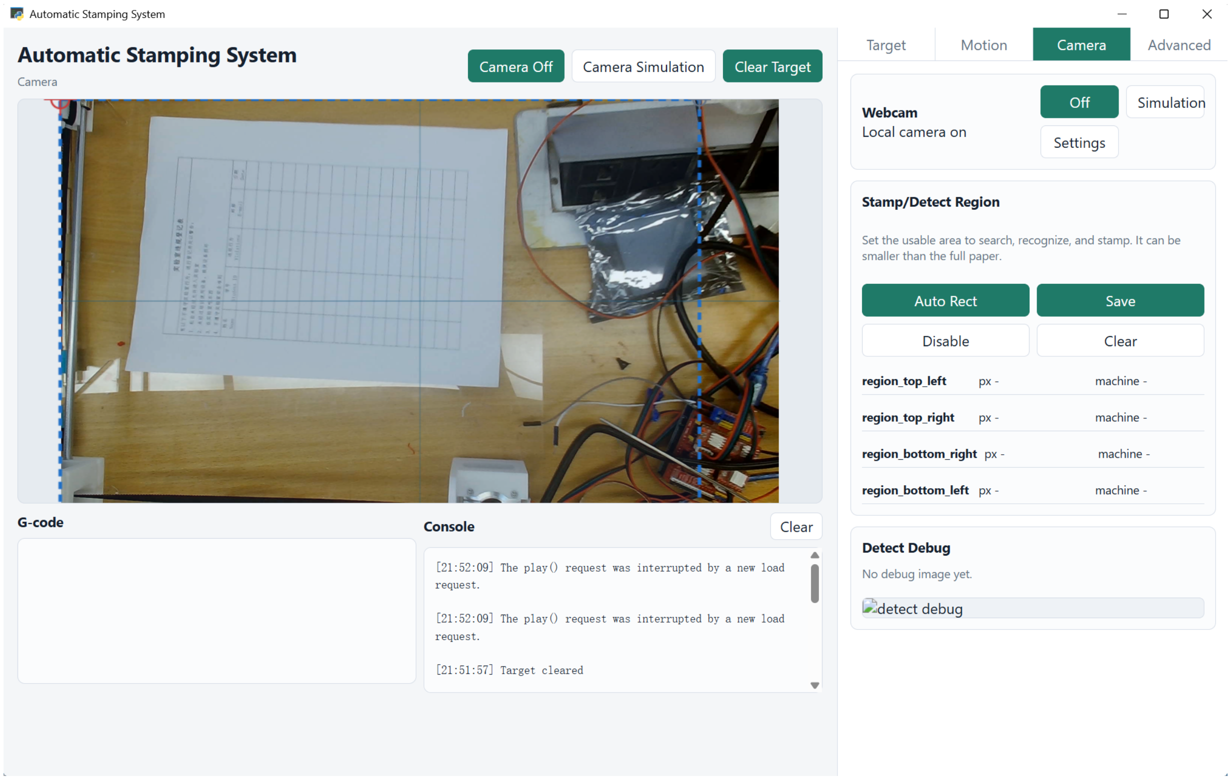
Paper mm -

Machine mm -

Command mm -

Figure 5: UI file-teach mode: the operator uploads an A4 document preview and clicks the desired stamp point.

In the camera-teach workflow, the operator uses the live camera image to define the usable stamp/detect region or to select the target on the physical sheet. Figure 6 shows the camera view, paper region overlay, and camera controls. The manual-positioning workflow lets the operator jog the stamp head and confirm the current machine position as the target. These workflows store the result as either a paper-relative coordinate (r_x, r_y) in the range $[0, 1]$ or a real machine coordinate, so later commands can map the same logical stamp point to the current physical page location.



Set the usable area to search, recognize, and stamp. It can be smaller than the full paper.

region_top_left px - machine -

region_top_right px - machine -

region_bottom_right px - machine -

region_bottom_left px - machine -

No debug image yet.

```
[21:52:09] The play() request was interrupted by a new load
request.

[21:52:09] The play() request was interrupted by a new load
request.

[21:51:57] Target cleared
```

Figure 6: UI camera mode: the live camera view is used to set the usable stamp/detect region and support camera-based target selection.

The backend resolves every target into three coordinate descriptions: relative paper coordinates, real machine coordinates in millimeters, and corrected commanded coordinates. The correction layer accounts for axis scale, offset, inversion, and configured motion limits. The G-code generator then emits a repeatable stamp sequence: set millimeter units, use absolute positioning, lift or hold safe Z, move X-Y, press to stamp Z, dwell, retract, and optionally trigger the paper-feed command.

3 Verification

3.1 Verification Strategy

The verification strategy follows the high-level requirements and then decomposes them into module-level tests. The most important principle is that each test produces quantitative evidence: detection accuracy, processing time, coordinate error, force, feed success rate, sensor response time, packet error rate, or voltage tolerance. Table 1 summarizes the system-level verification plan.

Table 1: High-level requirements and verification plan

Requirement	**Verification method**	**Acceptance criterion**
Vision target detection	Test sample documents under 300 lux to 500 lux lighting. Log target result and processing time for each page.	More than 95% correct target selection and coordinate output within 5 s.
X-Y positioning and stamping force	Command known target points, mark or measure stamp location, and record force with a force gauge or load cell.	Placement error no greater than ± 5 mm in X and Y; force between 15 N and 25 N.
Page handling reliability	Load 50 sheets of 80 g/m^2 A4 paper and run continuous feed, stamp, and eject cycles.	Jam, no-feed, or multi-feed rate less than 2%.

3.2 Vision Verification

The camera test verifies that the captured frame is 1920 by 1080 pixels and that standard office lighting is sufficient for OpenCV-based paper and feature detection. For the file-teach mode, the selected document-preview point is compared with the mapped physical target on the page. For the camera-targeting mode, the selected live-camera point is checked against the calibrated stamp region and measured machine position. For manual positioning, the confirmed work position is compared with the expected physical location.

Processing latency is measured from image capture to coordinate output. The original target is 5 s at the system level. In final presentation testing, the OpenCV-supported detection path completed within 300 ms in the tested cases, leaving margin for communication and workflow overhead.

3.3 Motion Verification

The X-Y gantry is tested by commanding repeated moves to a set of reference points and measuring the resulting position. The repeatability target from the design document is ± 2.0 mm, while the high-level requirement allows ± 5.0 mm final stamping error after vision and mechanics are combined. The Z-axis actuator is tested with a force gauge placed at the stamping location. Later prototype testing estimated more than 40 N of center pressing force, but the effective stamp quality still depended on how evenly that force was distributed across the stamp face.

3.4 Page Handling Verification

The page handling subsystem is tested over repeated A4 cycles. Each trial records whether zero, one, or multiple sheets were fed; whether the front edge stopped at the reference location; and whether the sheet exited fully after stamping. Sensor timing is measured separately by blocking each sensor and recording response on a logic analyzer or oscilloscope.

3.5 Integration Verification

Integration tests check whether the UI state matches the physical state within 1 s, whether UART packets reach the receiving controller correctly, and whether the power rails stay within tolerance during peak mechanical load. The final report should be updated with measured values after the completed prototype is tested. The detailed requirement and verification matrix is included in Appendix A.

3.6 Software Verification

The software package includes unit tests for homography calibration, relative coordinate conversion, G-code generation, axis scaling, configuration round trip, Mode A zeroing, and paper-feed command generation. These tests provide regression coverage for the coordinate and command-generation layers before live machine testing. During report preparation, the tests were reviewed and run in a local Python environment; one preset-preview test did not pass because the current Y-axis machine limits in `machine.toml` do not yet cover the A4 preset target. The test should be rerun after final machine travel limits are measured and the production configuration is updated.

4 Cost and Schedule

4.1 Parts Cost

Table 2 lists the updated prototype cost estimate used for the final presentation. The total recorded parts cost is approximately 630 CNY, helped by reused components and custom 3D-printed parts such as frame mounts and timing pulleys. This estimate does not include labor, spare parts, shipping, or final enclosure revisions.

Table 2: Prototype parts cost estimate

Part	**Cost (CNY)**
Camera	180
Pillars and base	100
Electronic components	300
Other components and reused/printed hardware	50
Total	**630**

4.2 Labor Cost

The ECE 445 final report guideline recommends estimating labor cost as hourly rate multiplied by actual hours and then multiplied by 2.5. The final submission should replace the placeholder hours in Table 3 with each member's actual reported contribution.

Table 3: Labor cost worksheet

Team member	**Primary responsibility**	**Hours**	**Labor cost formula**
Yanzhen Chen	Vision and logic	Not yet reported	hourly rate × hours $\times 2.5$
Zhiqiang Qiu	Page handling	Not yet reported	hourly rate × hours $\times 2.5$
Xuliang Huang	Motion control	Not yet reported	hourly rate × hours $\times 2.5$
Jiaheng Zeng	UI and integration	Not yet reported	hourly rate × hours $\times 2.5$

4.3 Project Schedule

The schedule allocated mechanical, vision, motion, and integration tasks in parallel. Early work focused on CAD, model setup, component sourcing, and roller prototypes. Midproject work focused on OpenCV testing, motor calibration, sensor integration, and power design. Final work focused on assembly, debugging, verification, presentation preparation, and report preparation.

Table 4: Summary schedule

Week	Zhiqiang Qiu	Yanzhen Chen	Xuliang Huang	Jiaheng Zeng
5	Paper path CAD	Vision setup	Gantry assembly	Component sourcing
6	Roller prototype	OpenCV testing	Motor calibration	UI design
7	IR sensors	Pixel-to-mm mapping	Z-axis actuator test	Power PCB layout
8	Mechanism refinement	Logic integration	Accuracy tuning	UI-MCU communication
9	Final assembly	Debugging	Testing	Final report

5 Ethics and Safety

The project handles documents that may contain personal or administrative information. The system design therefore follows a local-processing privacy policy: document images are used only to compute stamping coordinates, are processed locally, and are not intentionally saved to non-volatile storage or uploaded to cloud services. This approach aligns with the IEEE Code of Ethics requirement to protect privacy and make engineering decisions consistent with safety and public welfare [6].

The main mechanical risks are pinch points at rollers, moving belts, and the X-Y carriage. These risks are reduced by using an enclosure around the moving gantry and paper path, placing the UI outside the motion envelope, and installing a front-panel emergency stop button. During testing, operators should keep hands clear of the paper path and should use manual jog controls only when the stamp head is raised.

The main electrical risks are AC mains input, motor supply current, and voltage transients from inductive loads. The AC-DC power supply must be enclosed, grounded, fused, and strain-relieved. Low-voltage logic and motor wiring should be separated where practical. The power distribution verification test checks that the 24 V rail remains within ± 0.5 V under load and that logic rails remain within their specified operating range.

The broader impact of the system is mainly economic and societal. Automating repetitive document stamping can reduce administrative workload and make batch processing more consistent. At the same time, the system should be deployed as an assistive tool rather than as a substitute for human judgment on official document approval. The machine can place a stamp, but the organization using it remains responsible for deciding when a stamp is authorized.

6 Conclusion

The automated document stamping system integrates machine vision, page transport, motion control, and user interaction into a single desktop workflow. The final prototype supports three operating modes for different user needs: document recognition from an uploaded file, camera targeting from the live paper view, and manual positioning from the current stamp-head location.

The most critical engineering risk is coordinate mapping from camera pixels to physical gantry motion. The design document's tolerance analysis estimates 0.4 mm vision error and 2.0 mm mechanical backlash, giving a worst-case accumulated error of 2.4 mm. This leaves 2.6 mm of margin relative to the ± 5.0 mm placement requirement. The current prototype and software already demonstrate the main structure of the final approach: a three-axis transmission frame, file-teach UI, camera-teach UI, homography-based coordinate mapping, and G-code generation for motion and stamping.

Several uncertainties remain after the final demo. First, the paper-feed roller belt loosened during operation, which caused tooth skipping and increased the time required to feed paper. The team remade a tighter belt that better fits the roller structure, and the demo video after modification showed improved behavior. Second, the camera holder was not rigid enough; pillar oscillation could change calibration and cause incorrect stop-region judgment, including stopping the paper before it reached the output roller. Third, the stamp head did not always press down evenly. Although the estimated center pressing force exceeded 40 N, uneven force distribution caused inconsistent stamp quality.

Future work should focus on mechanical optimization. The camera support should be redesigned with a more rigid fixture, such as triangular pillars or added beams, although this may require redesigning the base board. The Z-axis stamping mechanism should also be redesigned to reduce side rotation and distribute force more uniformly; one suggested approach is to guide the stamp head with two vertical poles, and another possible improvement is a mechanism that rolls or rotates the stamp head during pressing. Within the project timeline and our current mechanical-design experience, we adjusted the mechanism as much as possible, but these remaining issues would benefit from more specialized mechanical redesign.

References

[1] R. Smith, “An overview of the tesseract ocr engine,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991. [Online]. Available: https://tesseract-ocr.github.io/docs/tesseracticdar2007.pdf.

[2] OpenCV. “Camera Calibration With OpenCV,” Accessed: May 15, 2026. [Online]. Available: https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html.

[3] Grbl Project. “Grbl: An Open Source, Embedded, High Performance G-Code Parser and CNC Milling Controller,” Accessed: May 15, 2026. [Online]. Available: https://github.com/grbl/grbl.

[4] CNCjs Project. “CNCjs: A Web-Based Interface for CNC Controllers,” Accessed: May 15, 2026. [Online]. Available: https://cnc.js.org/docs/.

[5] Allegro MicroSystems. “A4988 DMOS Microstepping Driver with Translator,” Accessed: May 15, 2026. [Online]. Available: https://www.allegromicro.com/en/products/motor-drivers/brush-dc-motor-drivers/a4988.

[6] IEEE. “IEEE Code of Ethics,” Accessed: May 15, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A Requirement and Verification Matrix

Table 5 records the detailed block-level requirements and verification procedures derived from the proposal and design document.

Table 5: Detailed requirement and verification matrix

Module	**Requirement**	**Verification**
Vision camera	Capture images at 1920 by 1080 resolution under 300 lux office lighting.	Capture a standard 12 pt text document and verify image resolution and OCR readability on a PC.
Vision algorithm	Output selected target coordinates for file-teach, camera-targeting, or manual-positioning workflows within 5 s.	Run sample documents or camera frames with known targets and log capture-to-coordinate latency.
Coordinate mapping	Convert pixel points to physical X-Y coordinates with no more than ± 0.5 mm mapping error.	Input known calibration points and compare converted coordinates with caliper measurements.
X-Y gantry	Reach target coordinates with ± 2.0 mm repeatability.	Command five points repeatedly and measure return-to-origin and target-point deviations.
Z-axis actuator	Apply 20 N ± 2 N at the stamp contact point in block-level tests.	Place a force gauge at the stamping stage and record peak force over repeated cycles.
Pickup roller	Feed exactly one A4 sheet per cycle with no more than 2% failure.	Load 50 A4 sheets and record no-feed, multi-feed, and jam events.
Paper sensors	Detect paper presence within 100 ms.	Block each sensor and measure signal response with an oscilloscope or logic analyzer.
UI and workflow	Display file-teach, camera-targeting, and manual-positioning controls, and update workflow state within 1 s of physical state changes.	Run a full cycle and compare UI state transitions against observed machine state.
Communication	Maintain UART communication at 115200 bps with no more than 1% corrupted or dropped packets.	Send 1000 test strings and count missing or corrupted messages.
Power distribution	Supply 24 V DC within ± 0.5 V under a 2 A load.	Connect an electronic load and monitor output voltage with a multimeter or oscilloscope.

Appendix B Provided Project Materials Inventory

The workspace was reviewed before writing this report. The supplied materials fall into the following categories.

Table 6: Source material inventory and role in this report

Material category	Role
ECE final report guidelines PDF	Defines formatting, report organization, figure/table treatment, citation expectations, cost discussion, and verification discussion.
LaTeX template archive	Provides the required project structure, title page style, abstract, table of contents, bibliography setup, and appendix setup used for this report.
Project proposal PDF	Primary source for project title, team information, high-level requirements, module division, tolerance analysis, ethics, and safety goals.
Design document DOCX	Primary source for problem statement, final module descriptions, block-level requirement tables, cost estimate, and schedule.
System flowchart image	Inserted as Fig. 1; used to describe power, control, and module interactions.
System rendering image	Inserted as Fig. 2; used to describe the physical layout and user-facing structure.
Motion firmware source package	Contains Grbl C source, configuration files, upload sketch, and documentation for G-code-based motion control.
CNCjs and firmware flashing guides	Used to understand bring-up workflow, serial connection, baud rate, G-code testing, and motion-parameter adjustment.
G-code generation tools and sample files	Provide context for vector-to-motion command generation and testing with sample G-code.
3D model, CAD, and STL files	Provide mechanical reference for gantry, frame, rollers, brackets, holder parts, and custom printable components.
Power, wiring, and driver images	Provide supporting reference for motor-driver current adjustment, adapter wiring, and cable layout.
Driver packages and executable tools	Treated as setup assets only; they were inventoried but not executed while preparing this report.
Automatic stamping software package	Contains the Python backend, Web UI, PyWebView desktop entry point, Arduino protocol example, machine configuration, firmware images, run scripts, and unit tests.
Prototype and UI screenshots	Added as figures for the three-axis frame, file-teach UI, and camera-region UI.
Final presentation deck	Used to update the final operating modes, OpenCV-based vision description, updated cost estimate, demo issues, uncertainties, and future work.
Final prototype photo	Added as Fig. 4 to document the assembled machine near the final presentation stage.

Appendix C Software Structure and Code Excerpts

The current code package is organized as a local desktop application. The main user-facing scripts are `scripts/setup_env.bat`, which creates or updates the `SD` conda environment, and `scripts/run_app.bat`, which starts the PyWebView desktop application. The core backend is in `src/stamping_system/`, the browser UI is in `web/`, and an optional Arduino protocol example is in `arduino/stamping_controller.ino`.

The following excerpt shows the central preview, move, and stamp pipeline. Each target is resolved to machine coordinates before the backend builds G-code and sends it through the serial transport.

Listing 1: Backend target preview and stamp pipeline excerpt

```
def preview_target(target, config=None, include_paper_feed=True):
    cfg = config or load_config()
    resolved = resolve_target(target, cfg)
    plan = build_stamp_gcode(
        resolved.real_xy_mm,
        resolved.commanded_xy_mm,
        cfg,
        include_paper_feed=include_paper_feed,
    )
    return PreviewResult(target=resolved, gcode=plan.lines)

def stamp_target(target, dry_run=None, config=None):
    cfg = config or load_config()
    preview = preview_target(target, cfg)
    result = SerialTransport(cfg,
   force_dry_run=dry_run).execute(preview.gcode)
    return preview, result
```

The target resolver supports direct pixels, pixels on a detected paper quadrilateral, paper-relative coordinates, presets, OCR keywords, and real machine coordinates. Both file-teach and camera-teach workflows reduce to the relative-paper representation shown in Listing 2.

Listing 2: Relative paper target resolution excerpt

```
def _resolve_relative(rx, ry, target, config, note):
    pixel = None
    if config.paper.use_detected_quad_for_relative_targets and
   target.paper_quad:
        pixel = relative_to_pixel_on_quad(rx, ry, target.paper_quad)
        real = pixel_to_real(pixel, config)
        note += " Detected paper quad was used."
    else:
        real = relative_paper_to_real(rx, ry, config)
        note += " Static paper origin was used."

    real = (real[0] + target.offset_mm[0], real[1] + target.offset_mm[1])
    paper = (rx * config.paper.width_mm, ry * config.paper.height_mm)
    commanded = real_to_commanded(real, config)
```

```
    return TargetResult(
        source=target.source,
        input_point=(rx, ry),
        pixel_xy=pixel,
        paper_xy_mm=paper,
        relative_xy=(rx, ry),
        real_xy_mm=real,
        commanded_xy_mm=commanded,
        note=note,
    )
```

The G-code generator produces a concise sequence for each stamping cycle. It validates the X, Y, and Z commanded values against configured machine limits before returning the command list.

Listing 3: Stamp G-code generation excerpt

```
def build_stamp_gcode(real_xy_mm, commanded_xy_mm, config,
   include_paper_feed=True):
    plan = build_move_gcode(real_xy_mm, commanded_xy_mm, config)
    safe_z = z_real_to_commanded(config.machine.safe_z_mm, config)
    stamp_z = z_real_to_commanded(config.machine.stamp_z_mm, config)
    lines = [
        *plan.lines,
        f"G1 Z{format_float(stamp_z)}
   F{format_float(config.machine.z_feed_mm_min)}",
        f"G4 P{format_float(config.machine.stamp_dwell_s)}",
        f"G1 Z{format_float(safe_z)}
   F{format_float(config.machine.z_feed_mm_min)}",
    ]
    if include_paper_feed and config.paper_feed.enabled and
   config.paper_feed.command:
        lines.append(config.paper_feed.command)
        if config.paper_feed.settle_s > 0:
            lines.append(f"G4 P{format_float(config.paper_feed.settle_s)}")
    return GcodePlan(real_xy_mm=real_xy_mm, commanded_xy_mm=commanded_xy_mm,
   lines=lines)
```