PORTABLE FILM DIGITIZER

By

Yuhong Chen

Guyan Wang

Arnav Gaddam

Final Report for ECE 445, Senior Design, Spring 2026

TA: Gerasimos Gerogiannis

06 May 2026

Project No. 23

Abstract

This is a report detailing the design of a device built to convert analog film to a digital format using raster scanning and a computer vision processing pipeline.

The film digitizer system is a standalone device that uses an onboard Raspberry Pi 4 computer and Sony IMX477 camera to move row-by-row and capture several high-fidelity images of negative and positive film. Following the scanning step, computer vision algorithms are used to stitch macro frames together and apply further post-processing steps such as denoising and negative inversion. Camera position is controlled by a precision X-Y stage driven by NEMA 17 stepper motors and an STM32 microcontroller. The device exposes a user interface through a web application hosted on the Raspberry Pi, which allows for a completely automated scanning process and digital receipt of final outputs.

Contents

1. Introduction

1.1 Problem

In an age where most modern media is captured on devices like cell phones and digital cameras, analog film is harder than ever to distribute. Enthusiasts can use professional drum scanners, flatbed scanners, or manual DSLR scanning to convert their analog captures to digital formats, but these solutions are complex, non-portable, and require expert knowledge.

1.2 Solution

We aim to address this with a portable film digitizer, which will allow users to convert their film to a digital format without specialized knowledge or equipment using a raster scanning process. The digitizer will be a fully self-contained device that can perform scans without external power or manual intervention. A high-quality camera is positioned at specific positions over a piece of film, and several captures are stitched together before being processed into an end result.

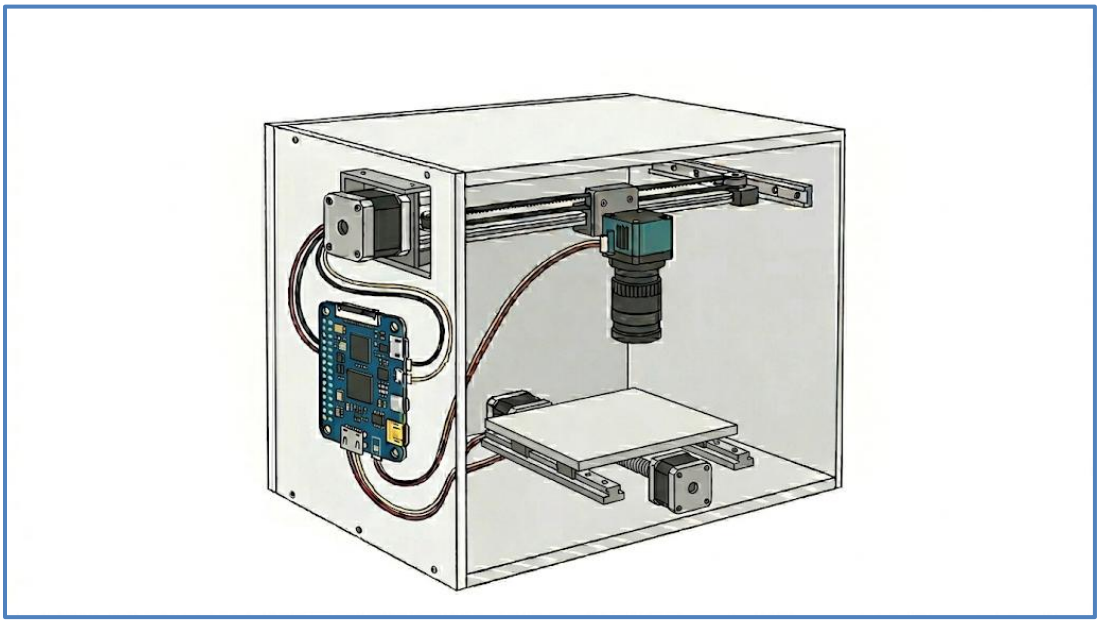

Figure 1: Prototype Design

1.3 High Level Requirements

To quantify the success of this project, we propose the following requirements:

1.3.1 Resolution and Stitching

The system must produce a final image with a minimum resolution of 3840x2160 pixels by autonomously moving the camera to capture and stitch at least 4 overlapping macro frames with an alignment error of less than 5 pixels.

1.3.2 Portability

The complete device, including power, optics, and mechanical parts, must fit within a physical volume of 15cm (L) X 15cm (W) X 20cm (H) and weigh less than 1.5kg to meet our portability requirements. There should be no reliance on external power, network connection, or specialized equipment for hardware tuning.

1.3.3 Automation Speed

The system must complete the full cycle of automatic exposure, X-Y scanning, image capture, and software stitching for a single frame in under 30 seconds without human intervention.

2 Design Procedure

The design process for this project involved making different decisions and tradeoffs along the way with respect to physical constraints, engineering challenges, and impact on user experience.

The prototype combines five major subsystems: an imaging subsystem, an illumination subsystem, a motion subsystem, a Raspberry Pi and STM32 scanner/control subsystem, and a cloud processing subsystem. The Raspberry Pi controls camera capture, motion, and lighting. The cloud server owns user-facing job state, API routing, RAW processing, and final artifact storage.

2.1 Block Diagram

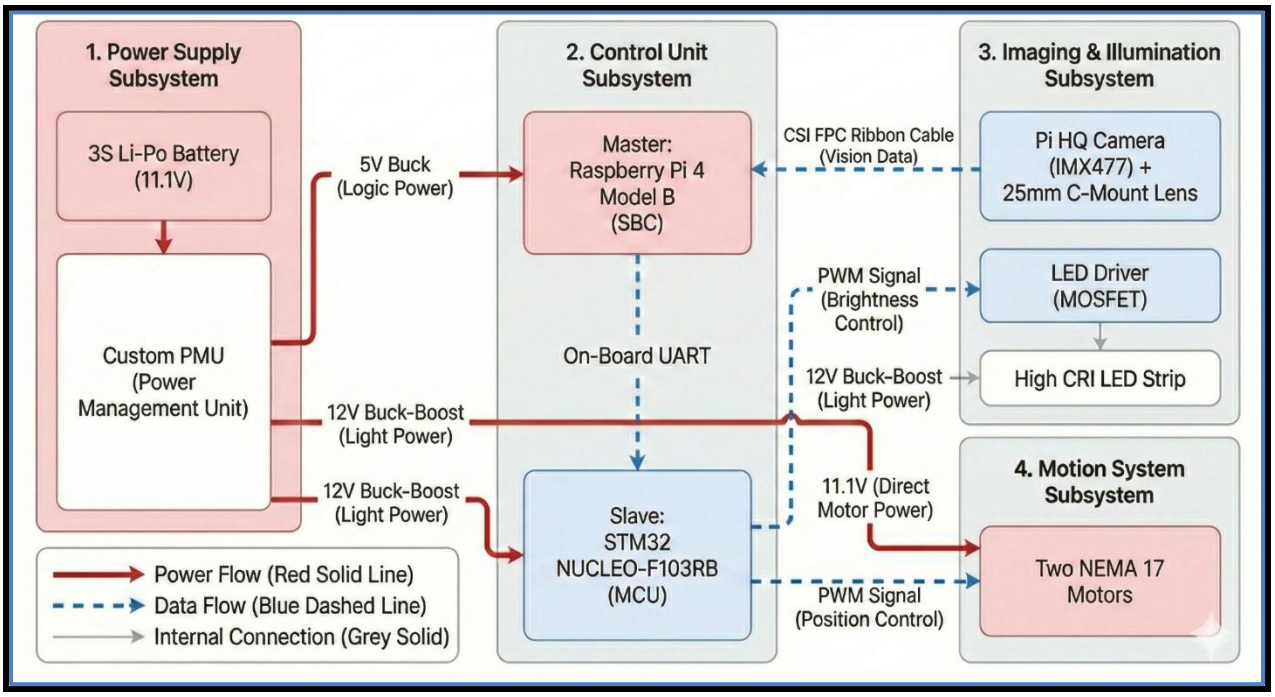

Figure 2: System Block Diagram

2.2 Power Supply and Management

The power subsystem is responsible for allowing the digitizer device to operate without the aid of an external power source or wall outlet in accordance with the portability requirements. This system is comprised of a rechargeable Li-Po battery, charging circuit, and voltage regulators to step down source voltage to the required logic levels.

Motor Driver Current Limiting (A4988)

To ensure our A4988 stepper motor drivers do not draw excessive current and overheat the power system, the maximum current limit ($I_{TripMax}$) must be set using the on-board potentiometer (V_{REF}) and the current sense resistor (R_S).

$$I_{TripMAX} = \frac{V_{REF}}{8 \times R_S} \quad (1)$$

Micro stepping Current: During micro stepping, the internal DAC further reduces this maximum current in precise percentage steps based on the step sequence:

$$I_{trip} = \left(\frac{\%I_{TripMAX}}{100} \right) \times I_{TripMAX} \quad (2)$$

Fixed Off-Time (t_{off}): Utilizing a resistor on the ROSC pin to ground to set a custom fixed off-time for PWM current control, the duration in microseconds is calculated as:

$$t_{OFF} \approx \frac{R_{OSC}}{825} \quad (3)$$

Logic Level Trigger Thresholds (V_{IH})

This equation mathematically proves why our initial testing failed when sending 3.3V logic signals from the Raspberry Pi to the 5V-powered A4988 and WS2812B LEDs. Both components calculate their minimum "High" logic input voltage (V_{IH}) based on their main supply voltage (V_{DD}).

$$V_{IH} = 0.7 \times V_{DD} \quad (4)$$

If V_{DD} is 5V, the minimum trigger voltage is 0.7 * 5 V = 3.5 V. Because the Raspberry Pi outputs exactly 3.3V, it falls short of the 3.5V threshold, necessitating the 3.3V logic supply for the A4988 and the 3.3V-to-5V level shifter for the LEDs.

Voltage Regulator Power Dissipation & Thermal Limits

Our voltage regulation system utilizes linear regulators (AMS1117) to drop voltages for sub-systems; the power dissipation equations are used to ensure the components do not trigger thermal shutdown.

Power Dissipation

$$P_D = (V_{IN} - V_{OUT}) \times I_{OUT} \quad (5)$$

Junction Temperature

$$T_J = T_{A(MAX)} + P_D \times \theta_{JA} \quad (6)$$

Variables: $T_{A(MAX)}$ is the maximum ambient temperature, and ϑ_{JA} is the package's thermal resistance from junction to ambient. The maximum junction temperature (T_J) must never exceed 125°C.

I/O Dynamic Current Consumption

When our Raspberry Pi switches GPIO pins (such as sending the 800Kbps data stream to the WS2812B LEDs), it consumes dynamic current to charge and discharge the capacitive load of the line.

$$I_{SW} = V_{DD} \times f_{SW} \times C \quad (7)$$

Variables: I_{SW} is the sunk current, f_{SW} is the I/O switching frequency, and C is the total capacitance of the pin and external trace ($C_{INT} + C_{EXT}$).

2.3 Motion System

The motion subsystem is responsible for accurately positioning the camera module and film stage in two-dimensional space. Due to physical constraints of our mechanical design, we independently control the position of the camera and film. Our camera mounted to the top of the assembly moves along the X axis, while the film stage at the bottom moves along the Y axis. This allows us to physically achieve a full range of motion in the X-Y plane and take captures with the camera centered at any coordinate within the surface of the film.

The main components are two precision translation stages with attached NEMA-17 stepper motors. Rather than controlling the motors directly, our system relies on two A4988 stepper motor drivers to convert digital motor control signals to analog current. GPIO pins on microcontrollers such as the STM32 output 3.3V and are limited to 20mA of current. The NEMA-17 motors we need require at least 12V and over 1A per phase [6]. A microcontroller cannot physically support this, which is why we decided to use motor drivers [3].

Another advantage of using motor drivers is the capability to perform micro stepping. Directly controlling a motor only allows full or half steps of rotation. However, motor drivers like the A4988 have internal circuitry to allocate current between motor coils, which allows each physical motor step to be split into 16 parts. This offers much greater control and smoothness, which is crucial for our project since even small perturbations in camera displacement can have drastic impact on macro frame captures.

D_{step} – Linear displacement per microstep
P – Lead screw pitch (mm/rev) = 2 mm
S_{rev} – Full steps per motor revolution = 200 steps
M – Microstepping factor = 16

V – Linear velocity (mm/s)
f – Step frequency (Hz)

N_{pulses} – Total number of pulses issued
ΔL – Target displacement (mm)

$$D_{step} = \frac{P}{S_{rev} \times M} = \frac{2}{32}\ mm/step \tag{8}$$

$$V = f \times D_{step} \tag{9}$$

$$N_{pulses} = \frac{\Delta L}{D_{step}} \tag{10}$$

Observing the equations, it can be seen that linear displacement ΔL is a function of the number of pulses issued to the motor driver. The speed at which this displacement is achieved is a linear function of step frequency. By doubling the frequency, which is the same as halving the interval between steps, the speed of motion is doubled. By changing the duration and frequency of a square wave, a control signal can manipulate the displacement and speed achieved by each motor.

2.4 Imaging and Illumination

The imaging system consists of a high-quality camera and 25mm lens used to capture clear images at a distance of 8.5 cm from the film [2]. An LED backlight array will be used to provide illumination for negative film, with brightness being controlled with pulse-width modulation from the control system to support multiple film densities. A separate frontlight LED array will be fixed to the upper side of the device to illuminate positive and Instax film. One major change made to this system after initial design was to use an off-the-shelf LED light for backlight capabilities.

2.5 Processing Pipeline

In addition to onboard computer vision processing on the Raspberry Pi used in the design, we will also employ cloud resources where appropriate to accelerate compute-intensive post processing tasks. This mainly consists of negative inversion, denoising, sharpening, and any additional steps such as LLM-augmented generation. The algorithms will be executed as Python scripts running in a serverless environment on a cloud provider such as Amazon Web Services to keep costs low and have minimal off-device dependencies [4].

We designed our pipeline to be modular and able to be partitioned between the Raspberry Pi and cloud environment. This is a tradeoff that users can make depending on what level of cloud dependency they prefer. For example, the pipeline can be configured to run entirely on device, but at the cost of a few minutes of real-world time due to the weaker hardware. Meanwhile, the opposite case would be performing only the stitching algorithm locally and then offloading the more intensive steps to the cloud

[7]. Depending on user preference and network conditions, the pipeline can run for a varying duration with various capabilities.

Table 1: API Endpoints and Capabilities

Interface	Endpoint or route	Verified role
Browser to Cloud	POST /api/jobs	Creates a user-facing scan or processing job.
Browser to Cloud	GET /api/jobs/{job_id}	Returns state, progress, artifacts, and errors.
Browser to Cloud	GET /api/jobs/{job_id}/artifacts/raw or final	Serves browser-safe raw preview and final result artifacts.
Browser to Cloud (dev)	/api/dev/*	Provides development capture, calibration, and status tools.
Cloud to Pi	POST /scan/start	Starts Raspberry Pi scan sequence.
Cloud to Pi	GET /scan/status	Reports scan state, tile progress, and errors.
Pi to Cloud	/internal/jobs/{job_id}/receive_tile	Uploads per-tile DNG/JPG captures for processing.

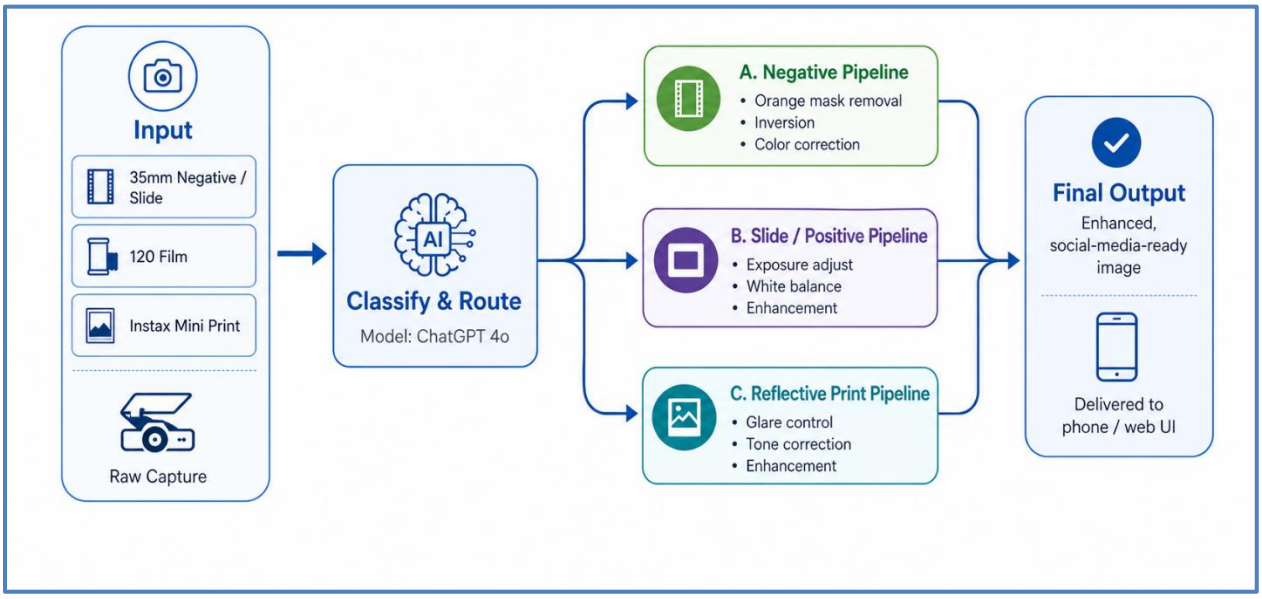

Figure 3: Post-Processing Overview

2.6 Control Unit

The control unit is responsible for integrating the other subsystems together and serving as the interface between device and user. The main components of this subsystem are a Raspberry Pi computer, STM32 microcontroller, and the software applications running to manage user interfaces and hardware interaction.

When designing this system, there were multiple approaches to component selection and defining integration points. The simplest way would be to only use a microcontroller without the presence of a Raspberry Pi computer. This would still allow for coordination of motor movement with camera captures but would need an additional persistent storage medium like an SD card. One frame capture would take about 18.5 MB, which is an entire order of magnitude greater than the 512 KB of on-chip flash memory of a standard microcontroller such as the STM32f446re [5]. One workaround for this would have been to use directory memory access (DMA) to directly transfer captures to an SD card, which would not involve the microcontroller CPU at all.

Resolution: 4056 x 3040 pixels
Bit Depth: 12-bit RAW

$$\frac{Total\ Pixels\ \times\ Bits/Pixel}{8} = 18.5\ MB \tag{11}$$

However, another issue with this approach is that all capture post-processing would need to be offloaded to cloud compute resources, as standard microcontrollers do not have the processing power or memory capacity to efficiently handle several large frames. In addition to adding a dependence on network availability, this would also greatly impact latency as each capture would need to be uploaded at full resolution.

S_{total} – data transfer size
S_{raw} – size of single capture
S_{proc} – size of processed output
N – number of frames
BW – network speed in Mbps

$$S_{total} = (N \times S_{Raw}) + S_{Proc}\ = 320.9\ MB \tag{12}$$

$$T_{network}\ = \frac{S_{total} \times 8}{BW} = 256.72\ seconds \tag{13}$$

Network transfer alone would take around 4 minutes, which would be the fastest possible completion time for the processing step, which excludes the physical scanning process. A better approach was to incorporate a Raspberry Pi into the project, as this provided a complete solution for capture storage, coordination, and processing. Having a full computer available allows for complete local processing, without a need for network connectivity or complex peripheral interactions with the camera module.

Another reason we used the Raspberry Pi was for the ability to expose a control interface over a local wireless network without the need for an internet connection. This allows users to use any client device such as a laptop or mobile phone to access the scanner's control interface in the browser and start scanning.

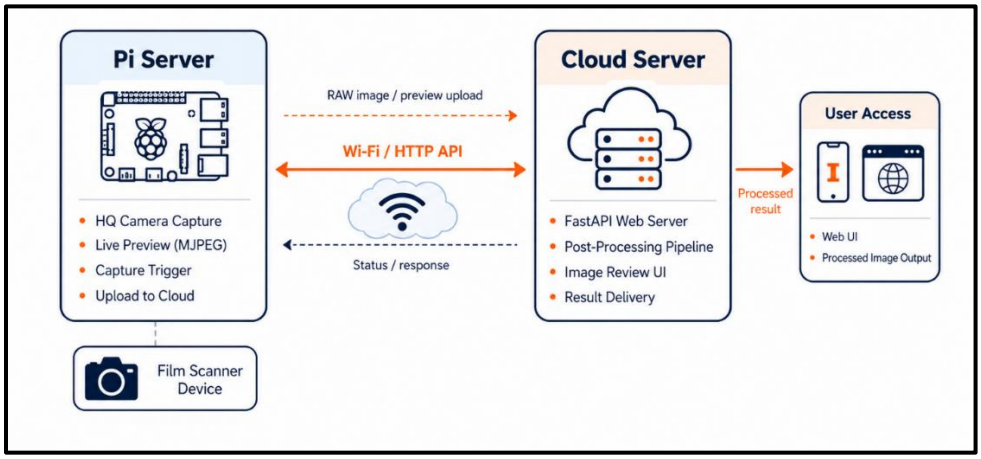

Figure 4: Software Architecture

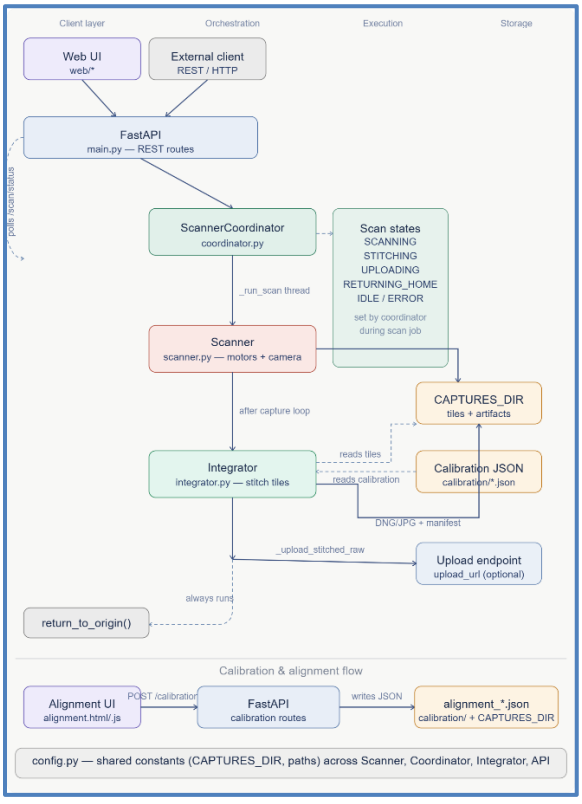

Figure 5: Control Flowchart

3 Design Verification

3.1 Power Supply and Management

To verify the power supply system we independently tested components in isolation. The circuit takes as input 11.1 V from a lithium-polymer battery and steps it down to 5V and 3.3V. The first step in verification was to charge the battery, then connect it to the power rail of a breadboard and verify with a multimeter that the reading was within the expected range. Then, we checked the output of the 5V regulator module and AMS1117 3.3V regulator the same way. An oscilloscope reading was also taken to ensure that voltage spikes from the battery and capacitor system did not have an impact on the interpretation of logic signals.

Figure 6: Battery Voltage Reading

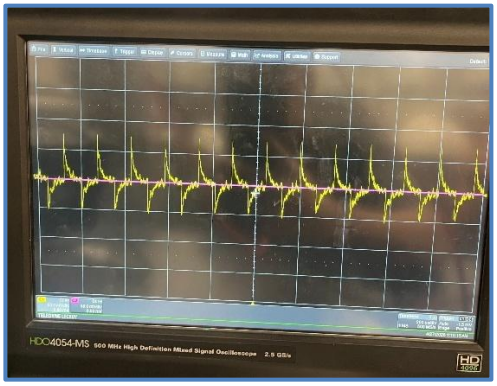

Figure 7: Logic Signal Voltage Ripple Reading

3.2 Motion, Illumination, and Imaging

To verify the motion and imaging subsystem, a bench power supply was used in place of the battery and voltage regulators. The setup for this used two output voltages: one at 11V and another at 5V. An A4988 motor driver was installed on a breadboard according to the circuit schematic, with the appropriate control pins wired HIGH or LOW. Two GPIO pins of an STM32 development board were connected to STEP and DIR pins on the driver, and the four output pins were connected to stepper motor coils.

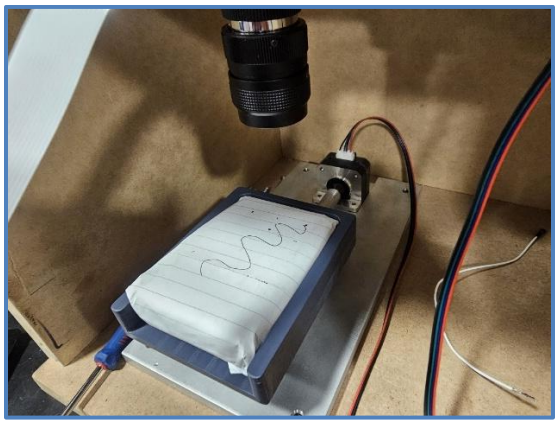

Figure 8: Calibration Test

3.3 Processing Pipeline

The final output requirement was at least 3840 x 2160 pixels using overlapping macro frames. The final system met this requirement. The scanner captured a calibrated multi-frame sequence, and the integrated output was suitable for 4K display. The stitching requirement also specified no visible discontinuities and alignment error within 5 ±1 pixels. Test scans using textured/grid-like targets showed continuous features across tile boundaries after calibration and overlap refinement.

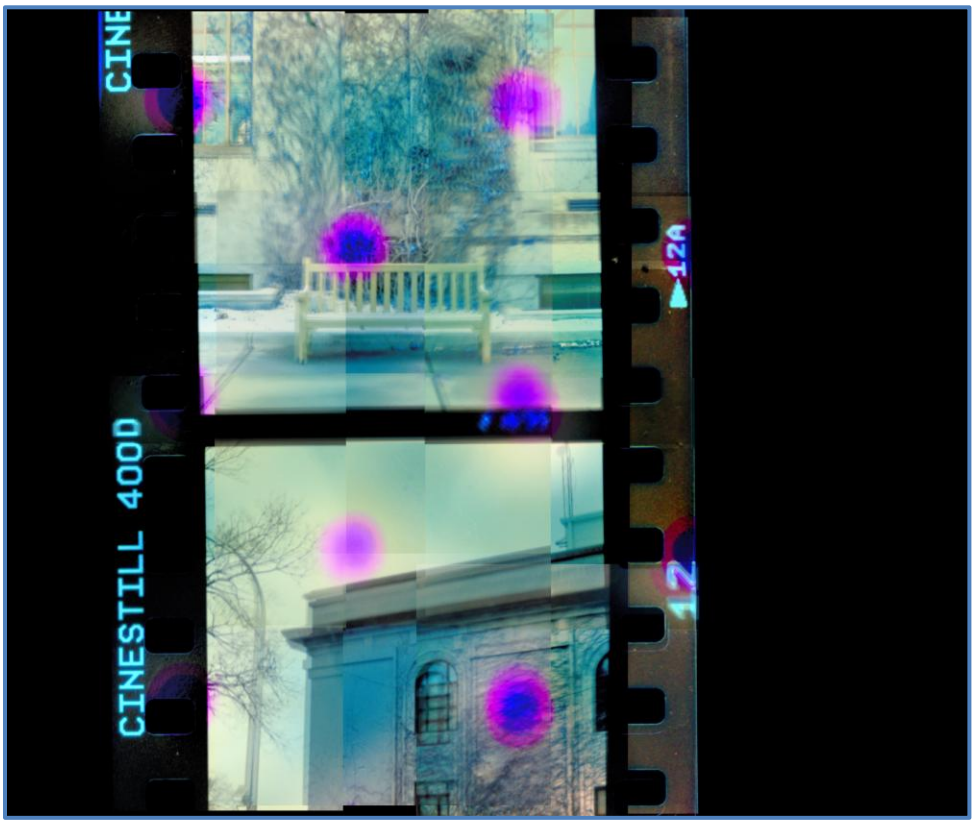

Figure 9: Processed Negative Film

The negative processing sub requirement was met. The fastest preview path runs in less than 1 second, the RAW fast preview path takes 1 to 3 seconds per tile, and the full high-quality negative RAW pipeline meets the less than 30 seconds per-image target. The complete mechanical scan and process cycle still exceeds 30 seconds, so the original full-cycle speed target is marked partially met rather than fully met.

3.4 Control Unit

Table 2: Processing Latency

Mode	Input/output	Typical latency	Use
Simple inversion/direct preview	JPEG/PNG preview to preview image	< 1 s	Fast UI/debug path; low quality.

RAW fast preview	Single DNG tile to RGB preview	about 1 to 3s per tile	Classification and base-region detection.
Full negative RAW pipeline	Single DNG tile to corrected output	< 30s per tile	High-quality negative processing path.

The API requirement was verified by sending scan commands to the Raspberry Pi controller process and receiving uploaded image artifacts and processed result URLs on the server [10]. The current implementation supports per-tile DNG/JPG upload and retains a legacy stitched-RAW route for compatibility, but the preferred final path is per-tile upload.

4. Costs

4.1 Parts

Table 3: Part Costs

Item	Vendor	Quantity	Unit Cost ($)	Total Cost ($)
Power & Control Subsystem				
12V 2400mAh AA NI-MH Battery	ECE Supply Center	1	0.17	0.17
SPST 20A On-Off Toggle Switch	ECE Supply Center	1	4.27	4.27
PCB Mount Fuse Holder 5x20mm	ECE Supply Center	1	0.80	0.80
IRF520 MOSFET N-Channel	DigiKey	1	3.00	3.00
10K Ohm 10 Watt Resistor	ECE Supply Center	1	0.75	0.75
DFR0379 Voltage Regulator 20W	DigiKey	1	4.90	4.90
AMS1117 Linear Voltage Regulator	DigiKey	1	0.12	0.12
SWD 0.05" 10-Pin SMT Box Header	DigiKey	1	1.50	1.50
Capacitors (1µF, 4.7µF, 3.3pF)	Online Purchase	1 ea.	Var.	0.35

Motion & Automation				
NEMA 17 Stepper Motor	StepperOnline	2	4.43	8.86
Allegro A4988 Stepper Driver	DigiKey	2	6.95	13.90
Illumination System				
High-CRI LED Strip	Amazon	1	14.99	14.99
High-CRI LED Plate	Amazon	1	12.99	12.99
Optical & Vision System				
Raspberry Pi HQ Camera (12MP)	Adafruit	1	55.00	55.00
Fujian 25mm f/1.4 C-Mount Lens	Amazon	1	19.54	19.54
25mm 5mm Extension Tube	Amazon	2	4.99	9.98
15CM FFC Cable	Amazon	1	6.99	6.99
			Total Estimated Cost	**$158.11**

4.2 Labor

Table 4: Labor Costs

Member	Primary contribution area	Weeks Worked	Hours / Week	Total Cost
Guyan Wang	Hardware, power, mechanical, verification support	14	10	$8,400
Yuhong Chen	PC/Pi software integration, processing pipeline, web/API, documentation	14	10	$8,400
Arnav Gaddam	Raspberry Pi scanner control, motor/camera server, scan calibration	14	10	$8,400
Machine Shop	Mechanical design and assembly	2	5	$700
Student Hourly Rate: $60/hr Machine Shop Hourly rate: $70/hr			Total Cost: $25,900	

Table 5: Labor Schedule

Week Of	Task	Group Members
2/9	Finalize component selection	All
	Draft STM32 pinout	Guyan
	Define UART serial protocol	Arnav
	Setup AWS cloud environment	Arnav
2/16	Complete schematic and PCB layout	Guyan
	Setup STM32 IDE project and test GPIO	Arnav
	Setup Raspberry Pi OS and remote access	Yuhong
2/23	Submit 1st round PCB order after finishing layout	Guyan
	Write PWM logic for LED control and STEP/DIR for motors	Arnav
	Create a self-hosted control plane UI on Raspberry Pi	Yuhong, Arnav
3/2	Order mechanical parts and lab equipment	Yuhong
	Experiment with software tradeoffs before design review	Yuhong, Arnav
	Develop image inversion and cropping algorithm	Yuhong
3/9	Prepare a circuit for breadboard demo	Arnav, Guyan
	Demonstrate motor movement through STM32 MCU	Arnav, Guyan
3/16	Assemble components on circuit	Arnav, Guyan
	Optimize processing pipeline end to end latency	Yuhong
3/23	Hardware testing and debugging for power system	Guyan
	Integrate Raspberry Pi camera with control software	Arnav
3/30	Test motors and LEDs under load	Guyan
	Continue soldering components, testing incrementally	All
4/6	Demonstrate stable power delivery and voltage regulation	Guyan
	Demonstrate image capture and negative inversion step	All

4/13	Optimize motor acceleration profiles to avoid skipped steps	Arnav
	Optimize image resolution and color correction	Yuhong
4/20	Verify physical power limits and motor current	Guyan
	Work on presentation slides	All
	Continue to debug physical motion and system integration	All
4/27	Debug integration before final demo	All
	Finalize requirements and verification results	All
	Make software robust for demo	Yuhong, Arnav
5/4	Work on final presentation	All
	Lab equipment check out	All

5. Conclusion

5.1 Accomplishments

The project produced a working macro-stitching film digitizer prototype. The final system captures overlapping film tiles with a Raspberry Pi camera and motorized stage, uploads per-tile RAW artifacts to a PC server, performs RAW-based correction and film processing, and integrates processed tiles into a final image. The system met key final-system requirements: 4K-class output, more than the minimum number of overlapping frames, seam alignment within 5 ±1 pixels, automated API upload/result flow, motion control, return-to-home behavior, regulated power, and GPIO-controlled illumination.

5.2 Uncertainties

The main unsatisfactory results were physical portability and full-cycle latency. The final prototype exceeded the original size/mass target, and the complete scan plus processing cycle exceeded the original 30 s target. The positive-film color pipeline also needs more tuning; the negative pipeline is more mature because it received more debugging time with real film data.

Aside from functionality, we also could not get our PCB to work. We suspect that this is because of the physical trace constraints of our board, where under load one of our motor drivers received lower current, which meant that the corresponding motor in turn did not have enough current to spin. As for the design of the PCB, many of the parts we included also had very small or irregular footprints that were hard to solder and find replacements for. Furthermore, we did not order a stencil, which made it even more difficult to solder on said components. One way to improve on the design for another iteration would be to choose more generic and well-document parts, so that replacements can be easily substituted in.

5.3 Ethical considerations

The project involves battery-powered electromechanical motion, bright illumination, and image data handling. Safety risks were mitigated by using regulated power rails, avoiding motor commands outside configured bounds, and keeping debug controls explicit. Privacy risk is limited because captured images are locally generated film scans, but future cloud deployment should avoid uploading user images without clear consent and should protect API keys and stored artifacts. These practices align with the IEEE Code of Ethics emphasis on public safety, honest reporting, and protection of privacy [1].

In accordance with the IEEE Code of Ethics [1], our team has identified and mitigated several ethical and safety concerns associated with the Portable Film Digitizer.

Public Safety and Health: We have prioritized safety regarding the 3S Li-Po battery design by integrating a Battery Management System to prevent overcharging and fire risks. Additionally, all mechanical pinch points created by the NEMA 17 motors are enclosed within the device housing to prevent user injury.

Disclosure and Honesty: We acknowledge the use of open-source libraries, specifically OpenCV [7] and libcamera, and do not claim their algorithms or code as our own work.

Intellectual Property and Misuse: We recognize that high-speed digitization could assist the unauthorized copying of copyrighted photographs. While the hardware cannot prevent this by itself, our user documentation and instructions will explicitly warn users only to digitize content that they own or have permission to use.

Environmental Responsibility: By repurposing precision X-Y linear stages and utilizing LED illumination, we aim to minimize electronic waste and energy consumption.

5.4 Future work

Future work should improve the positive-film color pipeline, reduce mechanical size and mass, add formal ISO 12233 chart measurements, and optimize scan latency through faster motion profiles and parallel capture/upload/processing. One major physical improvement is to switch to a closed-loop design for motor control. The current design has no feedback mechanism to detect missed steps due to inertia or mechanical resistance, causing discrepancies in measured vs expected displacement. An immediate improvement in scan quality could come from integrating rotary encoders into the motion system, which would transmit real-time data back to the controller. The motor control logic on the STM32 microcontroller would shift from pulse generation to a PID loop, allowing for self-correction during the scan. This would reduce the need for frequent manual calibration as well as avoid disruptions due to mechanical issues such as a stuck linear stage. A cleaner production build should also remove transitional debug routes and package the deployment path for continuous integration with protection against functional regressions and major bugs.

References

[1] IEEE Code of Ethics, IEEE, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[2] Raspberry Pi Ltd, "Raspberry Pi High Quality Camera Product Brief," RP-008344-DS, May 2025. [Online]. Available: https://datasheets.raspberrypi.com/camera/high-quality-camera-product-brief.pdf

[3] Allegro MicroSystems, "A4988 DMOS Microstepping Driver with Translator and Overcurrent Protection," Datasheet, 2020

[4] Amazon Web Services (AWS), "Operating Lambda: Performance optimization – Part 1," AWS Architecture Blog. [Online]. Available: https://aws.amazon.com/blogs/compute/operating-lambda-performance-optimization-part-1/

[5] STMicroelectronics, "STM32F446xC/E Datasheet," DS10693 Rev 10, 2024. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f446re.pdf

[6] NEMA 17 Stepper Motor Specification Standard, "Standard 1.8 Degree Stepper Motor Performance Data."

[7] Open Source Computer Vision Library (OpenCV), "Image Stitching Algorithms," Version 4.x, 2026. [Online]. Available: https://docs.opencv.org/

[8] G. Wang, Y. Chen, and A. Gaddam, "Portable Automated Macro-Stitching Film Digitizer," ECE 445 Design Document, University of Illinois Urbana-Champaign, Spring 2026.

[9] G. Wang, Y. Chen, and A. Gaddam, "Team 23 - Portable Film Digitization System Requirements and Verifications," University of Illinois Urbana-Champaign, Spring 2026.

[10] S. Ramirez, "FastAPI framework documentation." Available: https://fastapi.tiangolo.com/

[11] International Organization for Standardization, "ISO 12233: Photography - Electronic still picture imaging - Resolution and spatial frequency responses," ISO standard.

Appendix A: Requirement and Verification Table

Table 6: Power Subsystem Requirements and Verification [9]

Requirements	Verification	Status
Power		
Regulate 11.1 battery input to 5V± 0.25V for logic components and 12V± 1V for motor drivers under a load of 100mA to 2000mA	Connect battery to input and measure output rails using a multimeter under no load and full-load	X
Isolate motor noise from logic signals using star topology. Voltage ripple should not exceed 50mV peak-to-peek while motors are running.	Use an oscilloscope to measure voltage ripple on logic rail when motors are under peak current draw.	X
Motion		
Achieve a minimum translation resolution of 1 mm += 0.5mm per step using Nema-17 motors.	Issue a command to move the motor for a set number of steps (such as 50 or 100). Measure the physical displacement of the linear stage using a caliper and verify that it matches the desired theoretical displacement.	
Execute a row-by-row scanning pattern across a film-sized area and verify that the stage returns to the origin position with a maximum variance of 0.25mm.	Confirm that the stage can run through a scan and return to the starting position for a full reset.	X
Imaging and Illumination		

Maintain a consistent focal distance of 8.5cm ± 1mm between the lens element and the film surface to ensure high output image quality.	Mount the camera and lens, capturing a test image at the specified distance and visually inspect image clarity.	X
Adjust backlight brightness using PWM and avoid artifacts during image capture.	Capture images at different PWM duty cycles (such as 10%, 50%, 90%) and inspect frames for artifacts or image deformities	X
Post Processing		
The serverless compute environment must be able to execute a Python image processing pipeline to perform negative inversion and denoising with sub 30-second latency per image, with a 5-second margin of error.	Make requests to cloud endpoints with raw images and inspect logs to verify that end-to-end latency is below the target threshold. Use a timing library to record end-to-end latency.	X
The server needs to expose an API interface that the Raspberry Pi can use to upload raw camera data and receive the end result.	Perform HTTP Post requests from the Raspberry Pi controller process with sample images. Verify that the service returns a success message upon receipt and a valid URL to the processed output image.	X
Control System		

The Raspberry Pi needs to synchronize camera captures through libcamera with motion commands to the microcontroller. It needs to ensure that the stage is stationary during a capture to prevent motion blur. The capture will need to be taken 500ms ± 50ms after the last motor pulse is issued.	Use a logic analyzer to monitor the control signals from the Pi and the pulses issued from the microcontroller to the motors. Verify that the delay between the last pulse and the capture signal is large enough to avoid motion blur.	X
The STM32 needs to interpret serial commands from the Raspberry Pi and generate motor and LED output signals with low latency. The STM32 must issue a motor pulse within 10ms ± 2ms of receiving a completed control command.	Manually issue movement commands through a serial console. Use an oscilloscope to measure the end-to-end transmission time and evaluate against the threshold.	X
The image stitching algorithm on the Raspberry Pi needs to successfully combine multiple overlapping frames into a single 4K (3840x2160) image with no visible discontinuities or alignment errors (> 5 pixels ± 1 pixel).	Run the capture sequence on a test image (perhaps a printed grid). Inspect the stitched output to visually verify that there are no alignment issues. Use a calibrated ISO 12233 resolution chart and measure pixel offset at stitch boundaries using an image editing tool [11].	X

Appendix B: PCB Design

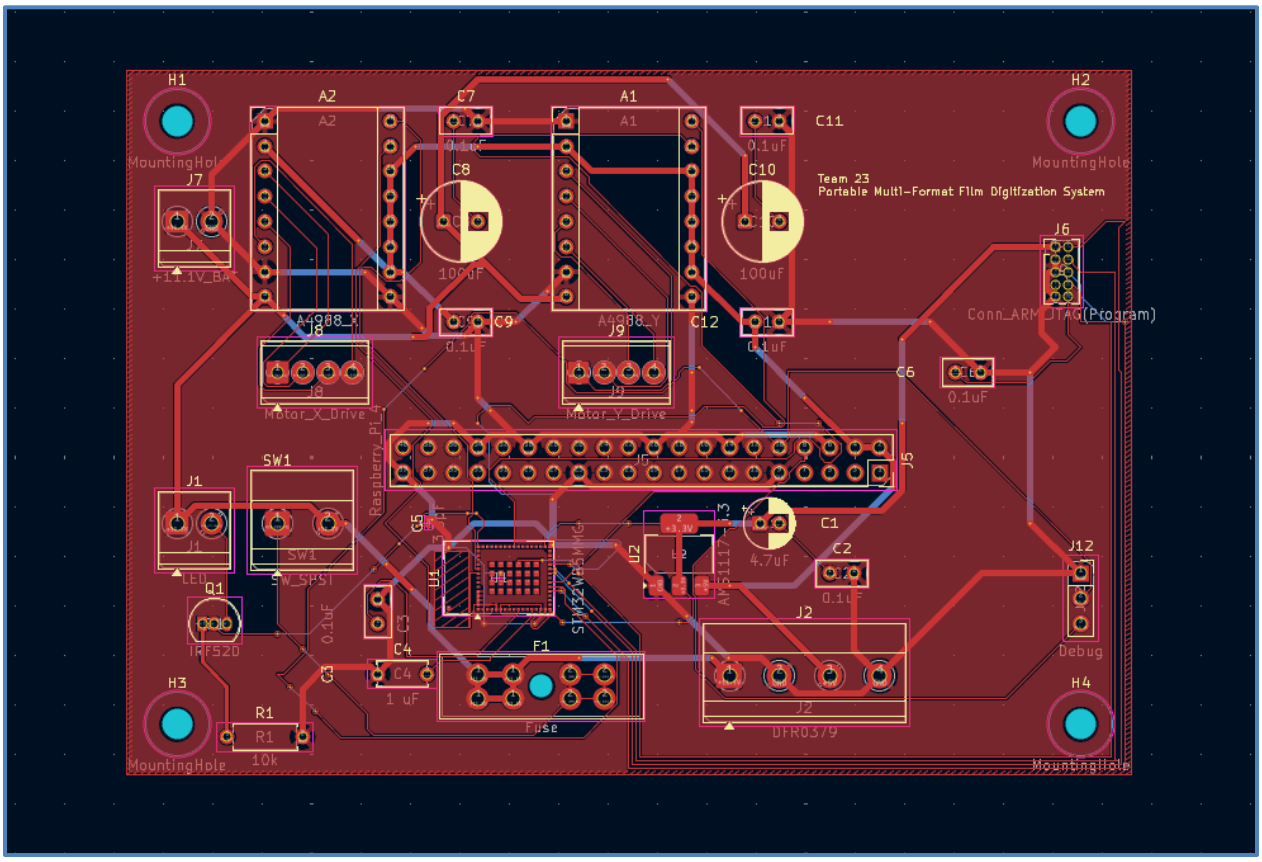

Figure 10: KiCAD PCB Design

Appendix C: Circuit Schematic

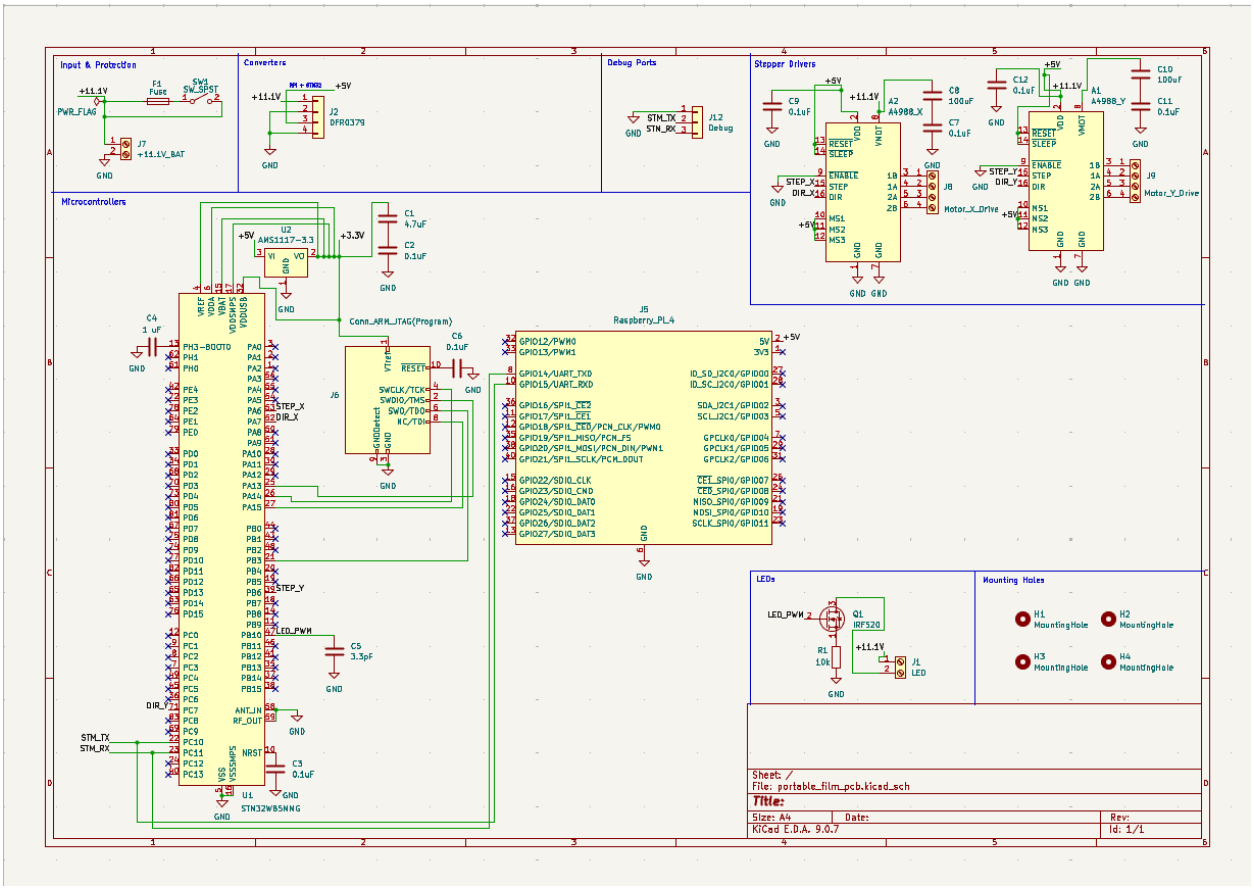

Figure 11: KiCAD Circuit Schematic

Appendix D: API and Processing Details

The final deployment uses explicit server roles. The control server normally runs on port 8000 and exposes browser-facing /api/* routes. The Raspberry Pi scanner server normally runs on port 5000 and exposes scan, motion, camera, and calibration routes. IP addresses should be configured explicitly because hotspot/SSH network addresses change between deployments.

The processing path used for the final test is per-tile DNG processing followed by RGB integration. Classification and base-reference bootstrapping use early scan tiles, while later tiles reuse the global classification and base reference. This avoids repeated inconsistent reference estimation and reduces memory pressure compared with processing one synthetic stitched RAW DNG.

Table 7: Processing Parameters

Item	Current setting or evidence needed
Input tile format	Per-tile DNG + JPG/PNG preview from Raspberry Pi camera. Each tile is 4056 x 3040, 12-bit RAW Bayer DNG, about 18.5 MB; final scan uses a 4 x 4 tile grid.
Backlight reference	Matching per-tile backlight set stored at software/web/calibration/backlight/tiles; latest calibration captured 2026-04-27 13:05, mode per_tile_raw_set, expected grid 4 x 4. Flat-map manifest reports 16/16 source tiles and 16/16 precomputed illumination maps complete.
Flat-field correction	Applied before classification and processing using precomputed per-tile Bayer-domain illumination maps. Final config: strength = 1.0, sigma_frac = 0.0005, max_side = 4096; CFA pattern BGGR; flat-field maps are stored in software/web/calibration/backlight/flat_maps.
Classification bootstrap	First three scan tiles are converted to flat-field-corrected preview RGB and classified by VLM vote. Current config uses OpenAI-compatible gpt-4o; observed classifier confidence range across saved jobs: 0.85-0.95; vote confidence range: 2/3 to 3/3. Saved results include both negative_35mm -> negative_film and positive_35mm -> positive_film.
Base reference	For negative film, the current final path uses a manual global base reference. Current base RGB, normalized linear scale: [0.0661, 0.1272, 0.0636], sample count 164,762, source tile row_0_col_0, raw bbox [84, 152, 710, 481]. Across recent manual-reference jobs, base RGB range was approximately R 0.0661-0.0673, G 0.1272-0.1348, B 0.0621-0.0636.

Final integration	Processed RGB tiles are integrated on the PC using the Pi manual alignment profile and conservative overlap refinement. Latest alignment mode: manual_alignment_profile, 4 x 4 grid, nominal tile size 4056 x 3040, refinement radius 8 px, step 2 px, minimum overlap 80 px, minimum improvement threshold 0.08