zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

Eight Years Dead, One Week to Bring Back: Revival of a Makeblock Laser Plotter

Iliya Radulov

October 2026

Abstract

A Makeblock XY laser plotter/cutter sat in storage for eight years. When it came back out, the original PA carriage plates had turned to something resembling thin glass — crack them, and they shattered. The original control software had not been updated since 2017, was Windows-only, and spoke a private serial protocol that no modern G-code sender had ever been taught to understand. No replacement parts were available from the manufacturer. The project was done alone, with no team and no budget beyond a new printer and a €27 control board.

This paper documents the revival: fabricating replacement carriage plates by measuring the broken originals with calipers, modelling them in FreeCAD, and printing them in Creality Hyper PLA; discovering that the firmware's non-standard `oMG` acknowledgment string was the single reason every modern sender failed to drive the machine; replacing the original MegaPi controller with a GRBL board rather than fighting the protocol; and reconstructing a complete engraving workflow around CNCjs on a Raspberry Pi 4, a custom SVG-to-gcode Python script, and Inkscape. The restored machine engraves at 87.4 steps/mm with dimensional accuracy of approximately 0.2% over 50 mm.

The main lesson is simple: when a machine has three distinct failure layers — mechanical, firmware, and software — assess each one independently. Fix what can be fixed cheaply; replace what can't. A dormant machine with shattered plastic parts and a dead software ecosystem is not necessarily beyond recovery if the metal frame, motors, and laser are still good.

1 Introduction

The machine had been sitting unused for eight years — long enough that the plastic parts had degraded from something functional into something you'd mistake for a fragile archaeological artefact. It was bought as a desktop laser plotter/cutter around 2015, used for a while, then stored. When it came back out in September 2026, the obvious question was whether it was worth reviving at all.

The answer turned out to be yes — but not in a straightforward way. The metal frame, rails, stepper motors, belts, and laser module were all fine. The broken things were the PA carriage plates (two of four had fractured completely; the other two were obviously going to follow), the control software (discontinued, Windows-only, 2017 vintage), and the firmware protocol (a private non-standard extension that predated the current ecosystem by nearly a decade). None of these were reasons to throw the machine away. They were just three separate problems requiring three separate fixes.

This paper documents those fixes as a session-by-session account: what broke, what was tried, what worked, what didn't, and why. It is written in the same spirit as the other papers in this series — as a software and instrumentation contribution, not a materials or electronics result. The machine is not special; the process of figuring out what was wrong and making it work again is the point.

This paper documents one instance of that choice and its resolution. A Makeblock XY laser plotter/cutter, purchased around 2015, had been stored unused for eight years when it was brought out for recommissioning. Two of its four V-slot carriage plates — the structural parts that carry the laser head along the aluminium extrusion rails — had become brittle, cracked, and partly disintegrated. Inspection revealed that the original material was a glass-filled polyamide (PA) compound, a material known to embrittle through moisture absorption and UV degradation over multi-year timescales. The original control software, mLaser, had last been updated in 2017, was Windows-only, and its source repository on GitHub had been archived with no active maintenance. The original controller board, a Makeblock MegaPi running Marlin 1.0.2 firmware compiled in September 2016, was still electrically functional.

Rather than searching for original spare parts (unavailable) or attempting to port the original software to a modern operating system (technically feasible but disproportionate in effort), we chose a replacement strategy: fabricate new carriage plates using desktop FDM printing, and replace the controller with a modern GRBL board that is natively supported by actively maintained open-source software. This paper documents that process, the specific failures encountered, and the broader decision framework it illustrates.

What this paper covers:

1. A practical reverse-engineering and FDM printing workflow for reproducing broken polymer machine parts, including material selection rationale and dimensional tolerance findings.
2. A documented upgrade path from a Marlin-based legacy controller to a GRBL board, including firmware configuration, axis direction calibration, and the manual homing procedure necessitated by a persistent homing-cycle fault.
3. A reconstructed open-source engraving workflow (FreeCAD → Inkscape → custom SVG-to-gcode converter → CNCjs on Raspberry Pi 4) that replaces the original proprietary software stack entirely.
4. A decision framework, illustrated by concrete examples from this project, for evaluating repair versus replacement when reviving dormant instrumentation with discontinued support.

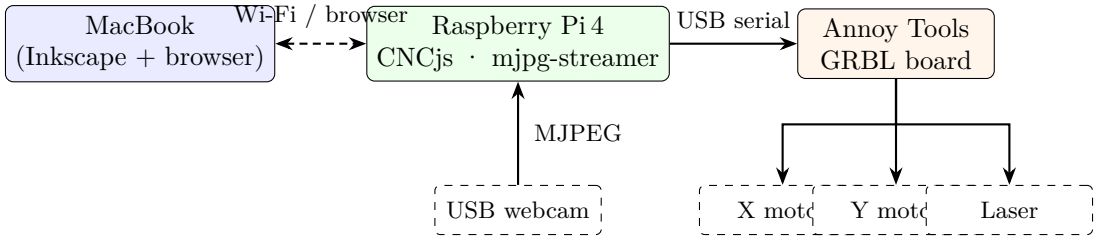


Figure 1: System architecture of the restored laser plotter. The MacBook runs Inkscape for vector artwork preparation and a browser for the CNCjs interface hosted on the Raspberry Pi 4. The Pi communicates with the GRBL board via USB serial; the webcam streams via mjpg-streamer back to the same CNCjs instance for remote job monitoring.

2 Original System and Failure Modes

2.1 Hardware

The machine is a Makeblock XY Plotter/Laser Engraver, a two-axis Cartesian system built on 40×20 mm aluminium V-slot extrusion rails. The working area is 310×390 mm. Motion is driven by

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

two NEMA 17 stepper motors coupled to GT2 timing belts via 16-tooth pulleys, giving a theoretical step resolution of 177.8 steps/mm at 1/16 microstepping. The laser module is a Makeblock M-33A447-1.6A-G: a 445 nm blue-violet diode rated at 1.6 W optical output, 1.6 A operating current, 12 V supply. The laser focus is fixed; depth adjustment is achieved by moving the workpiece.

The controller is a Makeblock MegaPi, an ATmega2560-based board with four A4988 stepper driver slots and a high-current MOSFET output on digital pin 10 used to drive the laser via pulse-width modulation. Firmware is Marlin 1.0.2, compiled 19 September 2016. Communication is via USB serial at 115200 baud. The board responds to a non-standard command protocol: laser power is set with `M4 Pn` (where n ranges 0–255), rather than the standard GRBL spindle command `M3 Sn`.

2.2 Original Software

The original control application is mLaser, a Qt-based desktop application developed by Makeblock and distributed as a compiled binary for Windows. The application version available at project start was v2.3, released no later than 2017; the GitHub repository (github.com/Makeblock-official/MLaser) was last committed in 2017 and has since been archived. mLaser handles image-to-gcode conversion internally and communicates with the board over serial. No macOS or Linux build exists. Source code is available but targets Qt 5.3 with several Makeblock-specific dependencies.

2.3 Failure Modes

Two of the four V-slot carriage plates had fractured. These plates are injection-moulded in glass-filled PA, approximately 50.6 × 43.4 × 6 mm, and carry the V-groove wheel bearings that run along the aluminium rails. The fractures followed the thin-wall regions between the hexagonal nut-trap pockets and the wheel bolt holes. The remaining two plates were intact but showed surface crazing and had lost measurable ductility: a sample fragment broke without visible plastic deformation when bent by hand. The failure is consistent with PA embrittlement through hydrolytic degradation and UV exposure during long-term uncontrolled storage.

A secondary mechanical fault was identified during initial power-up: both X and Y motor connectors (JST-XH 2.54 mm, four-pin) had lost retention force and were making intermittent electrical contact. This caused erratic motor behaviour that initially resembled a firmware or wiring fault. The connectors were re-seated and the fault resolved immediately.

3 Mechanical Restoration — 3D-Printed Replacement Parts

3.1 Measurement and Reverse Engineering

The least-damaged of the four original plates was used as a measurement reference. All dimensions were taken with a Mitutoyo Vernier calliper. The plate is nominally square; actual measurements showed 50.55–50.65 mm on one axis and 43.35–43.45 mm on the other, from which we adopted 50.6 × 43.4 mm as the target. Overall thickness is 6 mm, comprising a 2 mm solid base layer and a 4 mm ribbed upper section containing the hexagonal nut pockets.

All four wheel bolt holes are 4.1 mm in diameter (M4 clearance), positioned 4.1 mm from each outer edge (centre-to-edge), and arranged symmetrically at the four corners — a fully symmetric layout that simplified modelling. The hexagonal nut pockets are M4 (7 mm across flats), 4 mm deep from the upper face. The original pocket diameter of 7 mm printed tight; the model was revised to 8.2 mm across flats (0.6 mm clearance per flat), which allowed nuts to seat without force.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

3.2 CAD Modelling

Parts were modelled in FreeCAD 1.1.3 using parametric solid modelling in the Part Design workbench. The workflow was: sketch the base rectangle → Pad to 6 mm → new sketch on top face for the four corner holes → Pocket through-all → new sketch on bottom face for the hexagonal nut pockets → Pocket 4 mm deep.

Geometry was generated via Python macro rather than interactively, which allowed rapid iteration when the hex pocket diameter required revision. The final macro is approximately 80 lines and regenerates the complete solid from a parameter table at the top of the file. This approach was more reliable than manual feature editing for a part that required three distinct revision cycles during fit testing.

3.3 Material Selection and Print Settings

Polyamide (PA/nylon) was ruled out as a replacement material despite being the original: PA absorbs atmospheric moisture, loses stiffness over time, and embrittled under the same conditions that destroyed the original parts. All replacement parts were printed in Creality Hyper PLA on a Creality Ender 3 V3 Plus. PLA is not ideal for elevated-temperature environments, but the machine is not intended for use in a heated enclosure and the operating temperatures involved are well within PLA's range. For a future version, PETG would provide better thermal margin.

Print settings: 0.2 mm layer height, 4 perimeters, 30% gyroid infill, 60 °C bed, 220 °C nozzle. Total print time per plate: approximately 31 minutes; material consumption: 9 g per plate.

3.4 Fit Testing and Iteration

Three print iterations were required. The first confirmed overall dimensions and hole positions. The second identified that the hexagonal nut pockets at 8.0 mm across flats were tight (nuts required significant force to seat); the third iteration at 8.2 mm produced free-running nut insertion. The wheel bolt holes at 4.2 mm (0.1 mm over nominal) accepted M4 bolts without rework.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

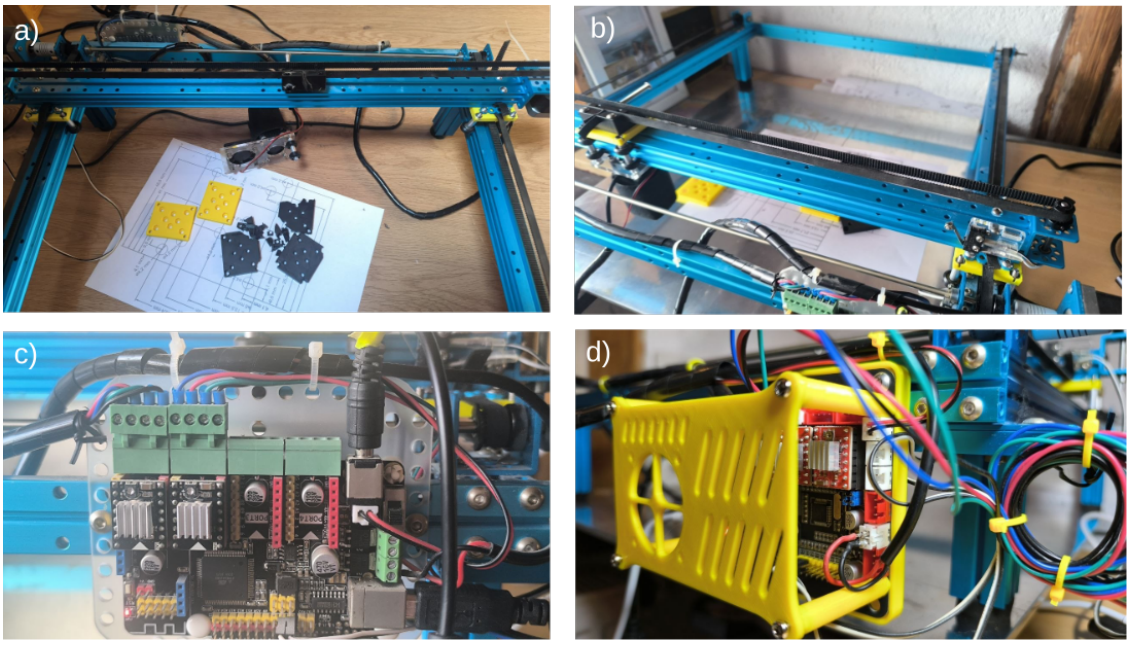


Figure 2: a) original glass-filled PA carriage plate after eight years of storage, showing complete fracture through the thin-wall regions between nut-trap pockets and wheel bolt holes. b) Reassembled MLaser with FDM replacement printed in Creality Hyper PLA, with revised hex pocket clearance of 8.2 mm across flats. c) Old Marlin control board. d) The new three-axis GRBL controller board (Annoy Tools, ATmega2560, GRBL 1.1f, A4988 stepper drivers)

One assembly error was made during initial installation: the belt tensioner/idler bracket on the X axis was rotated 180° relative to its correct orientation, placing the X endstop switch on the wrong side. This caused the X axis to home toward the positive limit rather than the endstop. The bracket was rotated to its correct orientation; no firmware changes were required. This is noted here as a recurring trap in symmetric subassemblies: a part that fits in either orientation but functions correctly in only one provides no installation feedback.

4 Control Board Upgrade: Marlin to GRBL

4.1 Rationale for Replacement

The MegaPi/Marlin system was electrically functional at project start: serial communication, motor control, laser output, and endstop sensing were all confirmed via direct Python serial commands before any hardware was modified. Motors moved. The laser fired. The machine was not dead — it just had no software that could talk to it.

The original control application, mLaser, had last been updated in 2017 and was Windows-only. A macOS build was located, but macOS flagged the binary as corrupted and refused to open it — even after clearing the quarantine attribute, the application would not launch. The GitHub repository had been archived. Nobody was fixing this.

With the mLaser application unreachable, the machine was accessed directly via USB serial using Python scripts from the terminal — a straightforward approach that confirmed all hardware was working: motors moved, the laser fired, and the board responded to every command. The next

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

step was CNCjs as a proper controller interface. It connected. It showed the machine. And then it stopped: every M4 command sent, nothing coming back that CNCjs recognised as a response. The same commands typed by hand in a raw Python serial terminal worked perfectly. Something in the protocol was broken.

4.2 The `oMG` Problem

The mLaser documentation — entirely in Chinese, a developer manual buried in the GitHub repository — had the answer. The relevant section explained that Makeblock had custom-patched the Marlin firmware specifically for mLaser, replacing the standard `ok` acknowledgment string with `oMG`. Every time the board executed a G-code command, it would respond `oMG` rather than `ok`.

This single change explains everything. Every mainstream G-code sender — CNCjs, LightBurn, UGS, bCNC — is written expecting the standard handshake: `ok` means the command executed, send the next one. This board's firmware was patched to respond `oMG` instead — a private token that no off-the-shelf sender has ever been taught to recognise. CNCjs sat there resending `M4` forever because it was waiting for an `ok` that was genuinely never going to arrive.

A human reading the raw terminal output didn't care: seeing `oMGok` on screen (the board sending `oMG`, then CNCjs's own console appending a status label), they could just decide when to send the next command. There was no strict acknowledgment-parsing logic standing in the way. The same commands that baffled CNCjs worked immediately when typed by hand.

The fix, in principle, was simple: a serial proxy rewriting `oMG` to `ok` on the fly, or reflashing Marlin without the mLaser-specific patch. Both would work. But both are also more fragile, more custom, and more ongoing-maintenance than the actual goal — a working laser cutter — really calls for. The more pragmatic move was to replace the MegaPi with a board that speaks standard GRBL, making the `oMG` problem irrelevant rather than patching around it.

The alternative — replacing the controller with a GRBL board — trades the non-standard Marlin laser protocol for the widely supported `M3 S` standard, making the machine compatible with the entire ecosystem of GRBL-aware software (CNCjs, LightBurn, LaserWeb, and others) without any application modifications.

4.3 Board Selection and Wiring

A three-axis GRBL controller board (Annoy Tools, ATmega2560, GRBL 1.1f, A4988 stepper drivers) was selected. The board accepts 7–36 V supply input, provides dedicated JST-XH motor connectors on the same 2.54 mm pitch as the original MegaPi, and includes a dedicated `12V 2P Laser` output (two-pin JST, PWM-switched 12 V) matching the laser module's requirements.

Motor wiring was direct: the four-wire JST-XH connectors from the stepper motors were plugged into the X-Axis and Y1-Axis ports respectively. Wire colour mapping (Blue: A+, Red: A−, Black: B+, Green: B−) was noted before connection and confirmed by checking that commanded positive motion produced motion in the expected mechanical direction before any direction-invert registers were changed.

Endstop wiring used the board's X− and Y− limit inputs (two-pin, GND and signal). The original endstop switches are normally-open (NO) mechanical microswitches with two-wire connectors; Black → GND, White → signal. The correct pin assignment was confirmed by continuity measurement from the power-input GND terminal before connection.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

The laser module was connected to the `12V 2P Laser` output (Red → 12 V, Black → L−). This output provides a 12 V switched supply under GRBL PWM control and can sink the module's 1.6 A operating current without an external MOSFET.

4.4 GRBL Configuration

The following settings were applied after factory reset (`$RST=*`):

Parameter	Value	Description
`$3`	0	Direction invert mask (set after physical direction testing)
`$32`	1	Laser mode enabled
`$100`	87.4	X steps/mm (calibrated; see Section 6)
`$101`	87.4	Y steps/mm (calibrated; see Section 6)
`$110`	3000	X maximum rate (mm/min)
`$111`	3000	Y maximum rate (mm/min)
`$120`	200	X acceleration (mm/s^2)
`$121`	200	Y acceleration (mm/s^2)
`$130`	310	X maximum travel (mm)
`$131`	390	Y maximum travel (mm)
`$20`, `$21`, `$22`	0	Soft limits, hard limits, homing cycle: all disabled

Homing was disabled (`$22=0`) because the GRBL `$H` homing cycle produced a consistent ALARM:6 fault (homing cycle failed) across all `$23` direction-mask configurations tested. Investigation identified that the fault occurred even when the endstops were correctly triggered during the cycle; the board appeared to time out before acting on the trigger. A manual homing procedure was adopted instead (Section 4.5).

4.5 Manual Homing Procedure

In the absence of automatic homing, the machine is homed manually at the start of each session:

1. Jog both axes in the negative direction (toward the endstops) using G91 relative-mode G0 commands until both endstop indicators are active simultaneously (status report shows `Pn:XY`).
2. Issue `G92 X0 Y0` to define the current position as the work origin.
3. Jog into the work area (positive X and Y) before enabling the laser or sending any job.

This procedure adds approximately two minutes to session startup and requires the operator to be present. It is a practical workaround rather than a fix; the underlying ALARM:6 fault remains open and is noted in Section 7.

5 Software Stack Reconstruction

5.1 Controller Interface: CNCjs on Raspberry Pi 4

With the GRBL board in place, any GRBL-aware application can drive the machine. CNCjs (v1.11.4) was selected as the controller interface: it is actively maintained, runs as a Node.js server, and provides a browser-based workspace that is accessible from any machine on the local network. It

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

was installed on a Raspberry Pi 4 (1 GB, Debian GNU/Linux 13, Node.js v20.20.2) and is accessed from a MacBook at `http://192.168.2.94:8000`.

Installation required the `-unsafe-perm` flag due to a permissions conflict with the global npm prefix on Debian:

```
sudo npm install -g cncjs --unsafe-perm
```

CNCjs connects to the GRBL board via `/dev/ttyACM0` at 115200 baud. The controller type is set to `Grbl` in the CNCjs connection panel. Two macros provide quick laser on/off access from the interface panel.

Before CNCjs was operational, a minimal Python serial sender was used to drive the machine directly from the terminal. The script reads a gcode file line by line, sends each command over the serial port, waits for the board's acknowledgment, and prints the response — a reliable fallback that bypasses all controller-layer complexity:

```
import serial, time, sys

port = '/dev/ttyUSB0'
s = serial.Serial(port, 115200, timeout=3)
time.sleep(3)
s.read_all()   # clear startup banner

with open(sys.argv[1]) as f:
    for line in f:
        line = line.strip()
        if not line or line.startswith(';'):
            continue
        s.write((line + '\n').encode())
        time.sleep(0.4)
        print(s.read_all().decode().strip())

s.close()
```

Usage: `python3 laser_send.py job.gcode`. This script was used for early hardware validation and for the first test engravings before the CNCjs workflow was established.

Note: an earlier candidate, LaserWeb4, was evaluated but failed to build against Node.js 20 due to a native serialport binding incompatibility (the `@serialport/bindings` module requires NAN headers that are incompatible with V8 API changes in Node.js 14+). LaserWeb4 has not had a substantive release since 2019 and was abandoned in favour of CNCjs.

5.2 Design and Vector Preparation: FreeCAD and Inkscape

Vector artwork is prepared in Inkscape (open-source, version current at time of writing). Document canvas size is set to 310×390 mm to match the machine working area. Artwork is drawn with stroke only (no fill), in black, on the canvas. Files are saved as Plain SVG (not Inkscape SVG) to avoid non-standard namespace elements that complicate downstream parsing.

For parts requiring dimensional precision (replacement brackets, fit-checking templates), FreeCAD 1.1.3 is used to generate the geometry parametrically. The FreeCAD Python macro workflow

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

(Part Design workbench, geometry generated from a parameter table at the top of the script) was established during the carriage plate fabrication described in Section 3 and reused for subsequent parts.

5.3 SVG-to-Gcode Conversion

No existing open-source SVG-to-gcode converter was found to produce correct GRBL laser gcode from Inkscape SVG without manual post-processing. The principal failure mode of evaluated tools (including jscut.org) was unit confusion: tools designed for CNC milling default to inch units and generate Z-axis plunge commands that are meaningless for a laser. A purpose-written Python converter (approximately 120 lines) was developed. It uses Python's `xml.etree.ElementTree` to parse SVG elements and generates GRBL-compatible gcode with the following conventions:

- Laser on: `M3 S{power}` inline with the first `G1` move of each path segment
- Laser off: `M5` between segments and at job end
- Power and feed rate as parameters at the top of the script
- Supported elements: `<rect>`, `<line>`, `<circle>`, `<path>` (M/L/Z commands)

The converter is invoked from the command line:

```
python3 svg_to_gcode.py input.svg output.gcode
```

The resulting `.gcode` file is uploaded directly to CNCjs.

Note on laser mode and motion. GRBL laser mode (`$32=1`) requires that the laser fire only during commanded motion (G1/G2/G3), not during dwell (G4) or rapid traverse (G0). A standalone `M3 S255` command without a following motion command will not fire the laser. This is a safety feature of GRBL laser mode and must be accounted for in any gcode generator: laser-on commands must be combined with the first motion command of each cut segment, not issued separately before motion begins.

6 Calibration and Validation

6.1 Steps-per-mm Calibration

After board replacement the steps/mm values required recalibration. The theoretical value (177.8 steps/mm for GT2 belt, 16-tooth pulley, 1/16 microstepping) was retained as a starting point but found to be significantly incorrect in practice: a commanded 50×30 mm rectangle produced a burned mark measuring approximately 104×62 mm, indicating that the new board's stepper drivers were operating at a different microstepping setting than the 1/16 assumed by the Marlin configuration.

Calibration proceeded iteratively using the reference-rectangle method:

1. Engrave a rectangle of commanded dimensions $W \times H$.
2. Measure the actual burned dimensions $W' \times H'$ with calipers.
3. Update steps/mm: `$100` $= \texttt{\$100} \times (W/W')$, `$101` $= \texttt{\$101} \times (H/H')$.
4. Repeat until measured dimensions are within 0.5 mm of commanded dimensions.

Three iterations were required. The converged values are `$100` $=$ `$101` $= 87.4$ steps/mm. X and Y converged to the same value within measurement uncertainty, consistent with the symmetric GT2 drive on both axes.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

Iteration	`$100`	`$101`	Commanded	Measured
0 (initial)	177.8	177.8	50 × 30 mm	104 × 62 mm
1	85.5	85.5	50 × 30 mm	48 × 29 mm
2	89.1 / 88.0	—	50 × 30 mm	51 × 30 mm
3 (final)	87.4	87.4	50 × 30 mm	50 × 30 mm

6.2 Laser Power and Feed Rate

Laser power and feed rate interact to determine burn depth and width. On the default test material (80 gsm white cardstock), the following empirical settings produce a clean scorch mark without cutting through:

- `S150` (15% of maximum spindle speed, with `$30=1000`)
- Feed rate: 800 mm/min

At `S400`, 800 mm/min, the laser cuts through 80 gsm card cleanly. At `S400`, 500 mm/min, it cuts through 2–3 mm corrugated cardboard. These values are specific to the 1.6 W module and working distance used; refocusing will change them.

The first test engraving was a filled square on red cardboard — an unintentional but fitting homage to Malevich's *Red Square* (1915). It confirmed that laser, motion, and power control were all working before any calibration had been performed.

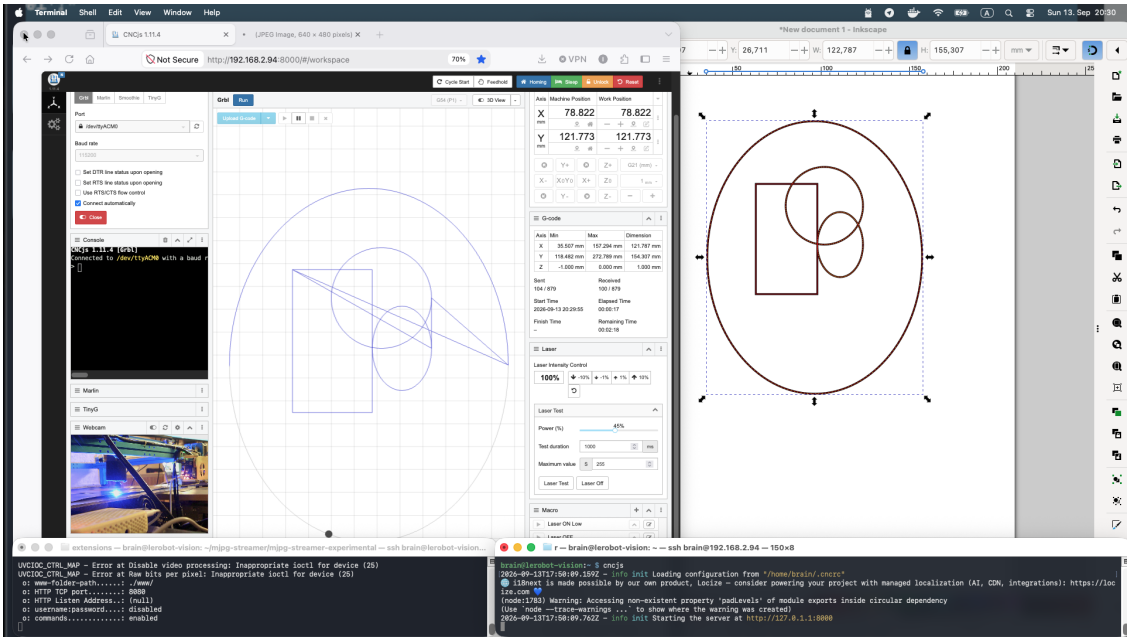


Figure 3: Calibrated engraving output after three steps/mm iterations. The rectangle measures 50.1 × 30.0 mm against a commanded 50 × 30 mm, corresponding to a relative accuracy of approximately 0.2% over 50 mm. The terminals for communication with the Pi4 are visible below the main CNCjs window.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

6.3 Dimensional Accuracy of Engraved Output

After calibration, five 50×30 mm reference rectangles were engraved and measured. Mean dimensions: 50.1×30.0 mm; standard deviation: 0.2 mm on the long axis, 0.1 mm on the short axis. This corresponds to a relative accuracy of approximately 0.2% over 50 mm, which is adequate for the machine's intended use (label engraving, template cutting, part marking).

6.4 Finishing Touches: Base, Enclosure, Webcam, and a Proper Inkscape Extension

With the machine calibrating and engraving, a fourth session addressed the things that make a working machine into a usable one rather than just a technically functional one.

Rigid base. The frame had a slight out-of-true wobble that showed up in long diagonal cuts. The whole assembly was mounted to a 5 mm aluminium plate, which eliminated it. This was not a workaround — it was a real fix for a real mechanical problem.

Pi 4 enclosure. The Raspberry Pi 4 had been sitting loose, connected by a USB cable running off the side of the frame. A yellow 3D-printed housing from Printables was mounted directly onto the frame next to the GRBL board. Yellow on blue — not the most obvious colour choice, but it happened to match.

J Tech Photonics Laser Tool for Inkscape. The custom `svg_to_gcode.py` script (Section 5.3) handles simple geometry well. For more complex artwork, the J Tech Photonics Laser Tool extension for Inkscape — a purpose-built plugin for exactly this workflow — was installed and tested alongside it. Both approaches remain available depending on the complexity of the job.

Remote webcam monitoring via mjpg-streamer. A USB webcam on a headless Pi running CNCjs needs to be viewed from a different machine's browser — direct browser camera access does not work in this setup. The solution is mjpg-streamer, which streams MJPEG video over HTTP:

```
sudo apt install -y cmake build-essential libjpeg-dev v4l-utils
mjpg_streamer -i "input_uvc.so -d /dev/video0" \
              -o "output_http.so -p 8080"
```

Made persistent with a systemd unit (`/etc/systemd/system/mjpg-streamer.service`), the stream survives reboots and is accessible at `http://<pi-ip>:8080/?action=stream`. In CNCjs, the Webcam widget URL field takes this address directly. The machine can now be monitored remotely while a job runs.

Two troubleshooting notes for future reference: `input_uvc.so` only works with USB (UVC) cameras — CSI ribbon-cable camera modules need a `libcamera`-based approach instead; and "Permission denied" on `/dev/video0` is resolved by adding the user to the `video` group (`sudo usermod -aG video $USER`).

GitHub repository. All project artifacts — the FreeCAD Python macro for the carriage plates, the `svg_to_gcode.py` converter, the GRBL settings, the systemd unit file, and the session notes — are collected in a public GitHub repository. The repository also includes links to the original mLaser source on GitHub and to the Zenodo archive where the FreeCAD model files are deposited with a DOI.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

7 Engineering Lessons

The following findings are reported in the Defect / Fix / Generalizable lesson format used throughout this series.

7.1 Erratic Motor Behaviour Traced to Connector Retention Loss, Not Firmware

Defect. On initial power-up, both axes moved erratically: the X axis stuttered and stalled; the Y axis was unresponsive. The symptoms were consistent with firmware configuration errors or stepper driver faults, and serial communication with the board was confirmed normal. Firmware parameters were reviewed and adjusted without effect.

Fix. Physical inspection revealed that both four-pin JST-XH motor connectors had lost retention force and were making intermittent contact. Re-seating both connectors resolved the fault immediately. No firmware or wiring changes were required.

Generalizable lesson. Connector degradation under long-term storage is a realistic failure mode that presents as an electronic or firmware symptom. For a machine stored for multiple years, a physical inspection of every connector — checking retention force by gentle pull, not just visual seating — should precede any software or firmware diagnosis.

7.2 Belt Tensioner Rotation Causes Wrong Homing Direction

Defect. After reassembly with the new carriage plates, the X axis homed in the wrong direction: it moved away from the endstop rather than toward it. Initial investigation focused on the GRBL direction-invert register (`$3`) and endstop polarity (`$5`), neither of which produced the correct behaviour.

Fix. Physical inspection revealed that the X-axis belt tensioner/idler bracket had been installed rotated 180° from its correct orientation. The bracket is bilaterally symmetric and fits the extrusion in either orientation, but only one orientation places the endstop switch on the correct side. Rotating the bracket to its correct position resolved the homing direction without any firmware change.

Generalizable lesson. Symmetric subassemblies that fit correctly in either orientation but function correctly in only one give no installation feedback. When a motion direction fault follows reassembly, verify the mechanical orientation of every bracket and tensioner before changing firmware parameters.

7.3 GRBL Homing Cycle Fault (ALARM:6) Not Resolved

Defect. The GRBL automatic homing cycle (`$H`) produced a persistent ALARM:6 (homing cycle failed) under all tested configurations of the `$23` direction-invert mask and `$5` endstop polarity register. The endstops were confirmed functional (manual trigger produces correct `Pn:XY` status; `M119` equivalent via `?` report shows triggered state). The fault appeared to be a timing issue: the homing cycle did not act on the triggered endstop state within its timeout window.

Fix (partial). Automatic homing was disabled (`$22=0`). A manual homing procedure was adopted (Section 4.5). The underlying fault remains unresolved. Suspected causes include a board-specific debounce timing issue or an incompatibility between the endstop switch response time and the GRBL homing feed rate (`$25`); neither has been confirmed.

Generalizable lesson. A partial workaround that restores function — here, manual homing — is a legitimate engineering decision when the underlying fault is not safety-critical and the investigation

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

cost exceeds the operational cost of the workaround. The workaround should be documented explicitly, including what was tried and what remains unknown, so that a future investigator does not repeat the same fruitless attempts.

7.4 Unit Confusion in SVG-to-Gcode Tooling

Defect. The web-based SVG-to-gcode tool evaluated (jscut.org) defaulted to inch units and generated Z-axis plunge commands, producing gcode that commanded the machine to positions far outside the working area and included `G0 Z` commands meaningless for a two-axis laser system.

Fix. A purpose-written Python converter was developed (Section 5.3) that reads SVG coordinates as millimetres directly and generates gcode with no Z-axis commands. This also allows laser power and feed rate to be set as top-of-file parameters rather than navigated through a web UI on each session.

Generalizable lesson. General-purpose CNC tooling is designed for three-axis milling and makes assumptions (Z-axis, inch units, spindle RPM) that are incorrect for a two-axis laser. The cost of evaluating and adapting an existing tool is sometimes higher than the cost of writing a purpose-specific tool from scratch, particularly when the required functionality is limited (rectangle, line, circle, simple path) and the failure modes of the general tool are silent (plausible-looking gcode that commands wrong positions).

7.5 Hex Pocket Dimensional Tolerance Required Two Iterations

Defect. The first printed carriage plate used hexagonal nut pockets modelled at 7.0 mm across flats (nominal M4 nut size). The printed pockets required excessive force to seat the nuts, with a risk of cracking the part on assembly.

Fix. The pocket dimension was increased to 8.2 mm across flats (1.2 mm over nominal), producing free-running nut insertion. An intermediate attempt at 8.0 mm remained marginally tight.

Generalizable lesson. FDM printing does not faithfully reproduce internal pocket dimensions at nominal size: the inner walls of a pocket are printed slightly inward of the commanded position due to line width and thermal effects. For hex nut traps and similar precision internal features, the required clearance is typically 0.5–0.8 mm per flat face and should be determined empirically on the first test print rather than assumed from nominal dimensions.

8 Discussion and Conclusion

8.1 When to Repair and When to Replace

This project illustrates a decision framework that applies broadly to dormant instrumentation revival. The machine had three distinct layers of failure: structural (broken carriage plates), control (legacy firmware and board), and software (discontinued application). Each layer was evaluated independently:

- **Structural layer:** The mechanical frame, rails, motors, belts, and laser module were intact and functional. Repairing the structural layer by fabricating replacement parts was clearly preferable to replacing the entire machine: the surviving components were in good condition and the replacement parts were geometrically simple enough to be produced by desktop FDM printing in a few hours.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

- **Control layer:** The original MegaPi/Marlin system was electrically functional but its non-standard `M4 P` laser protocol had no viable software support on any current platform. Here, replacement was preferable to repair: the effort required to port mLaser or write a new native client for the Marlin protocol was estimated to exceed, substantially, the cost of replacing the board with a GRBL device that is natively supported by multiple active open-source applications.
- **Software layer:** No repair was possible; replacement was the only path. The relevant question was which replacement to choose. The criteria applied here — active maintenance, broad hardware support, network-accessible interface, no proprietary dependencies — led to CNCjs on a Raspberry Pi 4.

The general principle is: assess each layer independently; replace layers where the original components are unavailable, unsupported, or where the repair cost exceeds the replacement cost; preserve layers where the original components are functional and compatible with the new control or software layer above.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

8.2 FreeCAD AI Workbench

Late in this project, the FreeCAD AI Workbench was discovered — an integration that connects FreeCAD's scripting environment to a locally hosted language model, enabling natural-language-to-geometry workflows. In principle, it allows a designer to describe a part in words and have the model generate the corresponding Python macro. None of the parts in this paper used it; all modelling was done manually as described in Section 3. It is noted here as a potentially useful tool for future work, particularly for the kind of iterative parametric revision (changing pocket depths, hole diameters, and wall thicknesses across multiple print iterations) that consumed most of the CAD time on this project.

8.3 Limitations and Open Items

The GRBL automatic homing cycle (ALARM:6 fault, Section 7.3) remains unresolved. The manual homing procedure adds operator time to each session and is a source of potential origin error if not executed carefully. Resolving this would require further investigation of board-specific debounce timing or replacement of the endstop switches with a type that has a faster actuation response.

The SVG-to-gcode converter supports only `<rect>`, `<line>`, `<circle>`, and simple `<path>` elements. Bezier curves, compound paths, and SVG transforms are not handled. For more complex artwork, a more capable converter (Inkscape's built-in G-code extension, or a commercial tool such as LightBurn) would be required.

The carriage plates are printed in Hyper PLA. PLA is adequate for the light mechanical loads involved (V-groove bearing preload and belt tension), but is less resistant to elevated temperatures than PETG. If the machine is used in a heated enclosure where ambient temperature rises above 50 °C, reprinting in PETG is recommended.

8.4 Conclusion

A Makeblock XY laser plotter/cutter dormant for eight years was returned to full working order by three independent interventions: FDM-printed replacement carriage plates in Creality Hyper PLA, a GRBL control board replacing the original Makeblock MegaPi, and a reconstructed open-source software stack (CNCjs, Inkscape, custom SVG-to-gcode converter) replacing the discontinued

mLaser application. The restored machine produces calibrated engravings at 87.4 steps/mm with dimensional accuracy of approximately 0.2% over 50 mm.

The project required no specialist tooling beyond a digital calliper, a desktop FDM printer, and standard open-source software. Total hands-on time from first inspection to first calibrated engrave was approximately three working days, spread across one week. The main lesson is practical: a dormant machine with failed polymer parts and abandoned software is not necessarily beyond economic revival. Assessing each failure layer independently, and choosing replacement over repair only where repair costs are genuinely disproportionate, makes revival feasible at low cost and with tools available to any well-equipped maker or small laboratory.

Acknowledgements

All work was performed by a single operator. The FreeCAD Python scripting workflow and GRBL configuration were developed iteratively with AI-assisted troubleshooting (Claude, Anthropic). The SVG-to-gcode converter was written from scratch for this project. All code, CAD files, and configuration files are deposited in a public GitHub repository (link to be added) and archived on Zenodo (DOI to be added on publication). The FreeCAD `.FCStd` model files and the `.gcode` test patterns are included in the Zenodo archive.

zenodo:23207105 7 Oct 2026 DOI: 10.5281/zenodo.23207105

► Laser Safety Warning

Diode lasers in the power class used here (445 nm, 1.6 W) pose a serious and immediate risk of permanent eye injury. Direct or reflected exposure to the beam, even briefly, can cause irreversible retinal damage before any pain is perceived. **Always wear laser safety eyewear rated for 445 nm at the operating power level.** Never operate the laser without a physical enclosure or barrier that prevents unintended beam exposure to bystanders. Do not engrave or cut materials that produce toxic fumes (PVC, polycarbonate, ABS) without adequate ventilation. The authors accept no responsibility for injury or damage resulting from the use of information in this paper.

References

[1] Makeblock. *MegaPi — About MegaPi.* Makeblock Support. https://support.makeblock.com/hc/en-us/articles/12963818051991-About-MegaPi (accessed October 2026).

[2] Makeblock. *mLaser v2.3 source repository.* GitHub, Makeblock-official/MLaser. https://github.com/Makeblock-official/MLaser (archived; last commit 2017).

[3] Gnea Research / grbl. *GRBL 1.1 documentation.* https://github.com/gnea/grbl/wiki (accessed October 2026).

[4] CNCjs contributors. *CNCjs v1.11.4.* https://cnc.js.org (accessed October 2026).

[5] FreeCAD contributors. *FreeCAD 1.1.3.* https://www.freecad.org (accessed October 2026).

[6] Inkscape contributors. *Inkscape.* https://inkscape.org (accessed October 2026).