International Journal of Computer Aided Manufacturing

Vol: 12

Issue: 1

Year: 2026

E-ISSN:2456-642X

Type of article: Review

Received Date: 26 February, 2026

Accepted Date: 27 February, 2026

Published Date: 13 March, 2026

Planar-to-Rotary G-code Transformation Via Post-Processing for Discrete 4-Axis Machining

Pedro Portugal [1*], Damian Venghaus [2] and Diego Lopez [3,]

[1] Tecnologico de Monterrey, Queretaro, 76130, Mx. Independent Researcher., Independent Researcher, Querétaro, 76130, Mexico. (Formerly: School of Engineering and Sciences,
[2] Grafisch Lyceum Rotterdam, Rotterdam, 3013 AK, Nl. Independent Researcher.
[3] Tecnologico de Monterrey, Queretaro, 76130, Mx. Independent Researcher.
* Correspondence: pedroportugalsil@gmail.com

Abstract

Conventional 4-axis CNC machining remains inaccessible to many due to the high costs of industrial hardware and the complexity of firmware modifications for entry-level controllers. This paper presents a software-defined framework that enables Planar-to-Rotary G-code Transformation, allowing standard 3-axis CNC systems to perform discrete 4-axis machining without hardware retrofits or firmware changes. The core of the proposed method is a custom Python-based post-processor that maps Cartesian XZ toolpaths onto a cylindrical coordinate system by injecting indexed Y-axis rotations.

The framework incorporates an automated calculation module that determines optimal angular displacement based on stock diameter and tool geometry, utilizing an 80% overlap factor to ensure surface continuity. To facilitate user adoption, the system was implemented as both a desktop GUI with 3D visualization capabilities and a platform-independent web interface. Experimental validation using hardwood and copper specimens demonstrated high dimensional fidelity, with average deviations within ±0.25 mm. By shifting the complexity from hardware to a post-processing software layer, this approach provides a cost-effective solution for indexed rotary fabrication, expanding the capabilities of desktop CNCs in educational, prototyping, and makerspace environments.

Keywords: *Planar-to-Rotary Transformation; G-code Post-Processing; Discrete 4-Axis Machining; Indexed Rotary CNC; Software-Defined Fabrication.*

1. Introduction

Historically, advances in manufacturing technologies—from the mechanization of the Industrial Revolution to contemporary FabLabs and desktop CNC machines [1-17]—have progressively lowered the barriers to designing and producing complex artifacts (Srai, Kumar, Graham & Phillips 2020) [18].

Commercial 4-axis desktop machines typically cost between USD $5,000–$10,000, compared to <$1,000 for GRBL-based systems, making them financially inaccessible for most educational and maker contexts (Lena-Acebo & García-Ruiz, 2019). For instance, the Carvera Air desktop CNC machine, which includes a 4th axis module, is priced at approximately USD $5,799 (Makera, 2025) [16].

GRBL, an open-source firmware for Arduino platforms, provides reliable planar motion but lacks native rotary-axis support, restricting fabrication of shafts, gears, or cylindrical models (García-Ruiz & Lena-Acebo, 2022) [6]. To address this limitation, we propose a post-processing framework that injects indexed rotary commands into conventional G-code, enabling low-cost machines to approximate 4-axis functionality.

Our method emulates a rotary axis on GRBL-based CNC machines using only post-processing and inexpensive off-the-shelf mechanics, without requiring specialized controllers, unlike most existing solutions [19].

1.1 Contribution and Significance

We present a post-processing framework that extends GRBL-based CNC systems with indexed rotary capability. Unlike prior firmware hacks or external controllers, our method achieves virtual A-axis functionality without modifying GRBL, relying only on commodity mechanics already used in low-cost CNCs. It converts standard linear G-code trajectories into segmented toolpaths interleaved with indexed rotations. The required hardware—stepper motor, pulleys, and chuck—is readily available and fully compatible with existing GRBL machines.

1.2 Broader Impact

By unlocking rotary-axis capability on low-cost CNC platforms, this method expands access to multi-axis fabrication for education, hobbyists, and small-scale manufacturers. It also aligns with open-source maker ecosystems and decentralized production networks, supporting community-based fabrication of components such as teaching aids, prototype parts, and low-cost functional mechanisms (Jiang & Li, 2019 [8]; Srai et al., 2020 [18]). The approach contributes to democratizing CNC technology and offers opportunities for hands-on learning in resource-constrained environments, enhancing both creativity and technical skill development. For example, students can fabricate cylindrical teaching aids, makers can prototype gears, and local workshops can produce shafts without access to industrial equipment.

1.3 Overview of the Paper

Section 2 situates our work within related attempts at low-cost multi-axis CNC control. Section 3 describes our post-processing methodology, while Section 4 details implementation and validation. Section 5 discusses implications and limitations, and Section 6 concludes with opportunities for future work

2. Background and Related Work

GRBL and Low-Cost CNC Systems

GRBL is an open-source CNC firmware for Arduino-based platforms that provides precise stepper-motor control, including acceleration planning for three-axis systems (Sarguroh & Rane, 2018) [17]. Its simplicity and low hardware requirements have made it popular among hobbyists and small-scale manufacturers, particularly in educational and maker settings.

Extending GRBL to a fourth axis requires significant modifications to its motion planner and interrupt routines, effectively redesigning the firmware. Alternatives such as grblHAL, Mega-5X, or TinyG support multi-axis machining but require higher-end hardware (e.g., ARM-based boards with 32-bit processors) and present steeper learning curves, limiting accessibility for typical GRBL users. GRBL's low-

cost and open architecture make it a suitable target for software-based approaches to emulate additional axes without hardware upgrades.

2.1 Existing Solutions for Rotary Machining

Several approaches have been explored to achieve rotary machining on low-cost CNC platforms. One common method involves using external rotary tables or dedicated four-axis CNC machines enabling indexed rotary commands and improved interoperability with CAM systems. These solutions provide full rotational control but are often expensive and require additional drivers, firmware integration, and advanced CAM support. For instance, commercial software such as DeskProto Multi-Axis Edition provides full 4th-axis machining, but its licensing costs place it beyond the reach of many hobbyist or small-scale users.

Manual indexing or macro-based workarounds rotate parts in discrete steps, requiring operator intervention and lacking continuous motion. These solutions increase cost, complexity, or operator workload, highlighting a gap in accessible low-cost multi-axis machining (Araujo & Lins, 2020) [2].

2.2 Software-Driven Rotary Axis Simulation

Software-driven methods emulate rotary motion at the G-code level using inexpensive, off-the-shelf components, broadening access to advanced manufacturing and supporting decentralized production systems.

For example, Xu, Anwer & Lavernhe (2014) [19] demonstrated that converting G-code programs into STEP-NC format facilitates reuse of legacy programs and interoperability between CAD, CAM, and CNC systems, illustrating the potential for software-driven rotary emulation.

2.3 Broader Implications and Applications

The adoption of software-driven rotary-axis simulation has significant implications across multiple sectors.

In education, virtual CNC environments and emulators offer hands-on experience in multi-axis machining without requiring expensive hardware—lowering barriers and enhancing learning opportunities (Lo Valvo, Licari, & Adornetto, 2012) [14].

In small-scale manufacturing and regional innovation contexts, leveraging accessible controllers and digital toolpath simulation enables the cost-effective production of complex parts. This supports practical applications in resource-

constrained environments (Moroşanu et al., 2025) [16]. Open-source platforms such as EMC2 further reflect the ethos of decentralized production, contributing to the democratization of manufacturing and advancing community-level fabrication systems (Lo Valvo et al., 2012) [14].

These methods allow users to experiment with multi-axis designs and manufacturing techniques, supporting both education and prototyping. As the technology continues to evolve, it holds the potential to transform the landscape of digital fabrication, making advanced manufacturing capabilities more accessible and inclusive.

However, to date, no lightweight, software-only method provides indexed rotary functionality on GRBL while preserving its low-cost, low-complexity advantages. This paper addresses this gap with a post-processing framework that emulates rotary motion without firmware changes.

3. Methodology

3.1 Toolpath Design

Standard computer-aided manufacturing (CAM) tools can be used to generate the initial toolpaths for the proposed method. Commercially available software packages such as CAMWorks, Mastercam, or Fusion 360 are fully compatible with the developed post-processor. In this work, Fusion 360 is used to illustrate the generation and structure of the toolpaths. Solids of revolution were selected as a test case due to their prevalence in mechanical components and suitability for illustrating the discrete rotary indexing method.

The workflow begins by importing a CAD model into the CAM environment. After import, the geometry must be pre-processed to ensure compatibility with the post-processor. The part is oriented such that the longitudinal axis aligns with X and radial axis with Z, and is reduced to a 2D XZ cross-sectional profile (Figure 1). The profile thickness must remain below the cutting tool diameter to keep all tool movement within the XZ plane.

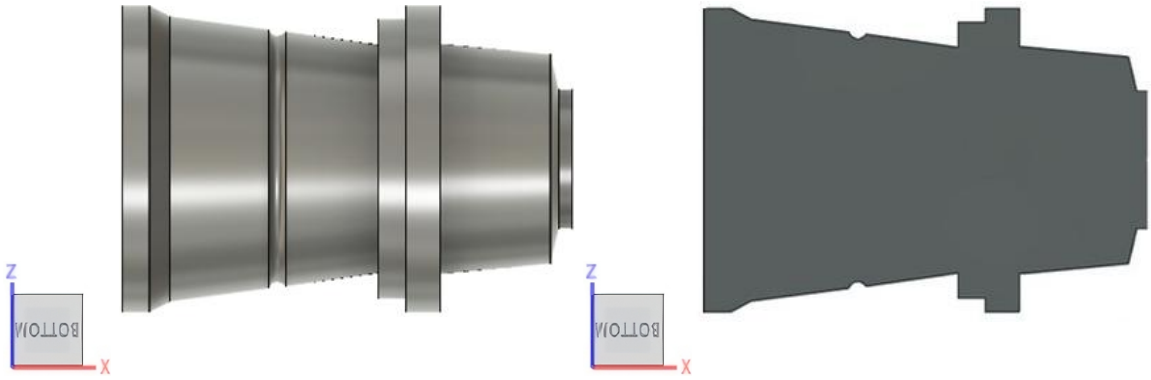

Figure 1. CAD model of the part to be manufactured. The original revolved geometry (left) and its cross-sectional representation (right), both oriented along the XZ plane.

Following pre-processing, the toolpath is generated using a 3D parallel machining strategy, which is commonly employed due to its consistent engagement and predictable cut loading (Fahmi, Abidin & Nafis, 2019) [5]. While specific cutting parameters such as feed rate, spindle speed, and step-over are left to the discretion of the user, it is recommended that they be reduced to approximately 75% of conventional milling values to compensate for the continuous indexing of the stock material and to minimize tool loading. This value is based on preliminary trials to reduce tool load during discrete indexing.

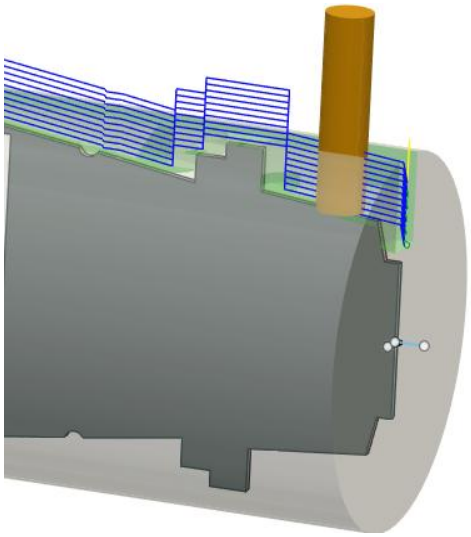

Figure 2. CAM simulation showing a 3D parallel toolpath along the XZ cross-section contour.

The 3D parallel strategy was chosen over alternatives such as Z-level or radial machining. Its simplicity and planar compatibility make it well suited to our discrete indexing approach. While radial strategies typically assume simultaneous angular control and Z motion, and Z-level strategies rely on vertical layering, the 3D parallel toolpath maintains consistent tool engagement along the X and Z axes, simplifying toolpath predictability and minimizing abrupt directional changes during Y-axis indexing.

Once generated, the toolpath should conform to the outline of the XZ cross-section (see Figure 2). The post-processor subsequently interprets this planar motion and injects synchronized Y-axis commands to emulate rotation, resulting in the final revolution geometry when executed on the machine.

3.2 Post-Processor Design

The conversion of conventional planar toolpaths into rotary motion is facilitated by a custom developed post-processor implemented in Python. While post-processing for multi-axis CNC toolpaths has been explored in contexts such as five-axis machining with rotary axes (Jung, Hwang, Park & Jung, 2011; Lee, Lin & Kuo, 2010) [9], our tool uniquely applies discrete indexing logic to emulate a fourth axis on standard three-axis hardware. Functioning both as a command-line utility and a GUI built with Tkinter, it supports both batch processing and interactive use.

The input to the system consists of standard G-code files that encode tool motion exclusively in the XZ plane, as described in Section 3.1. These toolpaths represent the cross-sectional contour of a solid of revolution. The post-processor operates by iteratively parsing each line of the input G-code and filtering out any pre-existing Y axis instructions to avoid interference with the rotary indexing process.

Unlike continuous synchronization approaches, the implemented logic relies on **discrete angular indexing**. Given the stock diameter and the effective diameter of the toolpath (derived from the tool diameter and total radial depth), the system calculates the number of indexed passes required to cover a full 360° revolution of the part. This is achieved using an overlap factor to ensure adequate surface coverage and finish quality. This approach is analogous to intermittent indexing methods used in high-end rotary systems where rotational motion is divided into discrete steps, such as those described in recent studies on bi-rotary milling heads (Dai, Li, Li & Wen, 2022) [4].

When the tool plunges vertically (Z-direction), as shown in Figure 2, and assuming an overlap factor α (typically 0.8 for 80% overlap), the width of each cut becomes:

$$w = \alpha D_{tool} \quad (1)$$

The number of indexed passes N needed to cover the full circumference of the stock is then:

$$N = \left[\frac{\pi D_{Stock}}{w}\right] \quad (2)$$

From this, the angular displacement per pass θ is determined as:

$$\theta = \frac{360^\circ}{N} \quad (3)$$

Since the purpose of this post processor is to democratize the use of this technology, θ is used in deg for the ease of understanding of the final user.

Alternatively, if the goal is to control the radial faceting error introduced by discrete indexing (e.g., for improved fidelity in surface finish), users may directly compute N from a specified maximum error tolerance e. The maximum radial deviation between the circular surface and its faceted approximation is:

$$e = R_{Stock}(1 - cos(\frac{\theta\pi}{180^{\circ}}) \quad (4)$$

For large N (small angular steps), using the second-order Taylor expansion of the cosine function leads to:

$$e \approx \frac{R_{Stock}\,\pi^2}{2N^2},\ N \approx \pi\sqrt{\frac{R_{Stock}}{2e}} \quad (5)$$

This allows intentional control of surface quality, especially in applications where visual or dimensional fidelity is critical. This equation (5) may be used instead of (2) when quality, not just overlap, drives the indexing density.

Prior to each repetition, a G-code command indexes the rotary axis (Y) by θ. This yields a sequence of G-code blocks, each beginning with a new Y-axis position and followed by the original XZ toolpath. The result is a **solid of revolution** generated by discrete angular steps, avoiding the complexity of time-synchronized rotation.

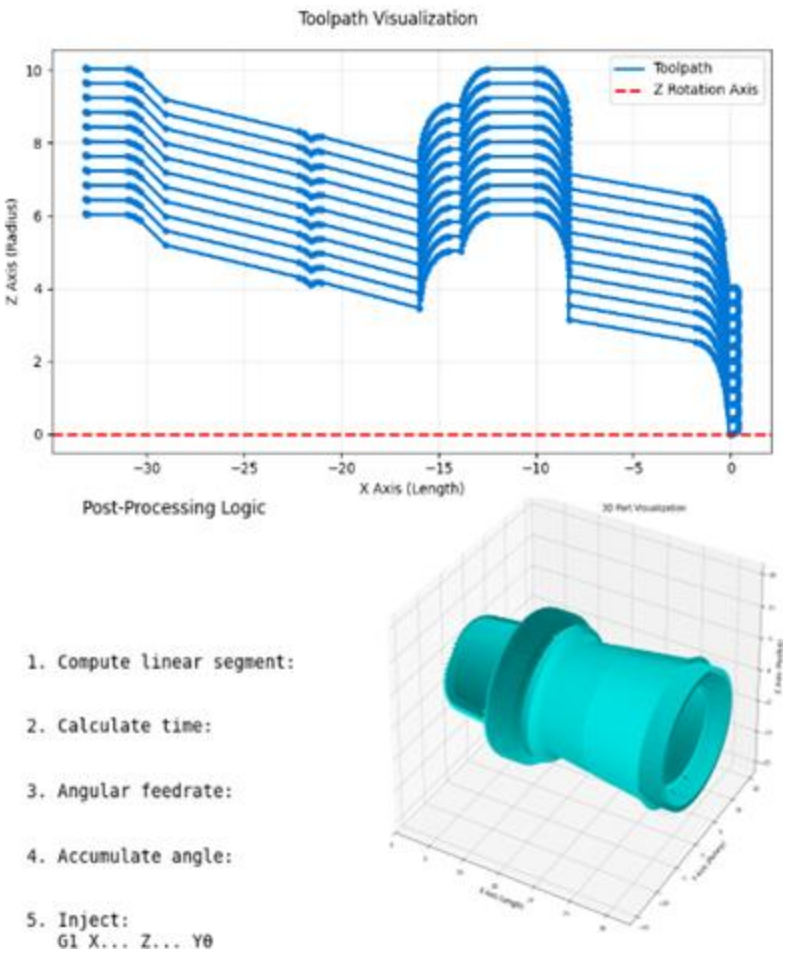

Figure 3. Post-Processor Workflow From Planar Toolpath to Rotary Motion

All rotary motion is introduced via G0 Yθ commands inserted at the start of each pass. These displacements are only applied to indexing steps, and no angular values are injected into G1 commands.

This strategy ensures that tool motion remains planar within each pass, while the workpiece is reoriented incrementally between passes. Indexing is performed with the spindle stopped and Z-axis retracted to avoid collisions.

Safety protocols—spindle stop, Z-axis retraction, and controlled restarts—are automatically inserted during indexing, ensuring safe operation with Y-axis repurposing and calibrated step-per-unit settings.

To support ease of use, the post-processor includes automatic detection routines to extract key machining parameters, such as tool diameter and initial feedrate, from structured comments within the G-code. These values, when available, are parsed using regular expression matching and used to pre-populate processing fields.

The post-processor further incorporates a visualization module based on Matplotlib, which renders both the 2D toolpath and the 3D reconstruction of the final part geometry. This allows users to validate the resulting rotary toolpath prior to execution, reducing the likelihood of runtime errors or machining artifacts.

A visual overview of the post-processing workflow is presented in Figure 3. The first panel illustrates a sample toolpath in the XZ plane. The second panel outlines the computational pipeline from parameter extraction and angular pass

calculation to G-code injection. The last panel visualizes the resulting part as a 3D geometry generated by discrete rotation of the planar profile.

3.3 Hardware Configuration

Y-axis motion is translated into synchronized rotary movement via a mechanical assembly on the CNC platform used in this study. The assembly consists of a 2-phase NEMA 23 stepper motor, a belt and pulley transmission, and a 63 mm 3-jaw chuck mounted on a rigid aluminum fixture. All components employed are commercially available and were selected to ensure compatibility with low-cost CNC machines operated via GRBL firmware.

The stepper motor is connected to the chuck through a timing belt and pulley system, providing mechanical decoupling between the motor and machining forces. This configuration not only reduces the axial load experienced by the motor shaft but also permits easy adjustment of gear ratios to suit specific applications. The chuck is mounted on a supported shaft equipped with precision bearings, ensuring smooth and concentric rotation of the stock material.

While the setup described above illustrates one practical implementation, it should be understood as an example rather than a prescriptive design. Different users will inevitably require alternative solutions depending on their machine dimensions, available components, and specific application constraints. If the stepper motor can be interfaced with the control board and the rotary assembly securely mounted within the machine's work envelope, the post-processor remains agnostic to the hardware details. The emphasis of this work lies in the post-processing methodology, which is designed to accommodate diverse mechanical realizations rather than enforce a single hardware configuration.

In the present implementation, the pulley transmission was configured with a 1:1 ratio to maintain direct correspondence between motor steps and rotary axis increments calculated by the post-processor. This eliminates the need for additional scaling factors in the G-code. If users adopt a non-1:1 pulley ratio—for example, to increase torque at the expense of angular resolution, the post-processor must be modified accordingly to account for the transmission factor. While direct drive is possible, the belt transmission provides decoupling and allows easier gear ratio adaptation.

Additionally, the described assembly is inherently modular. It can be detached and reinstalled without altering the base machine configuration, allowing users to switch between standard 3-axis operation and rotary emulation as needed. This flexibility supports rapid reconfiguration in shared lab environments, educational settings, or multi-purpose maker spaces where equipment reuse, and adaptability are essential.

It is important to note that the described implementation is specific to the physical constraints and performance characteristics of the CNC machine under study. Users applying this methodology to other systems must adapt the rotary mechanism accordingly based on the machine's structural design, motor capacity, and intended workpiece dimensions. Depending on user needs and resources, the rotary assembly can be either custom-manufactured using the machine's existing milling capabilities or obtained as a commercially available module.

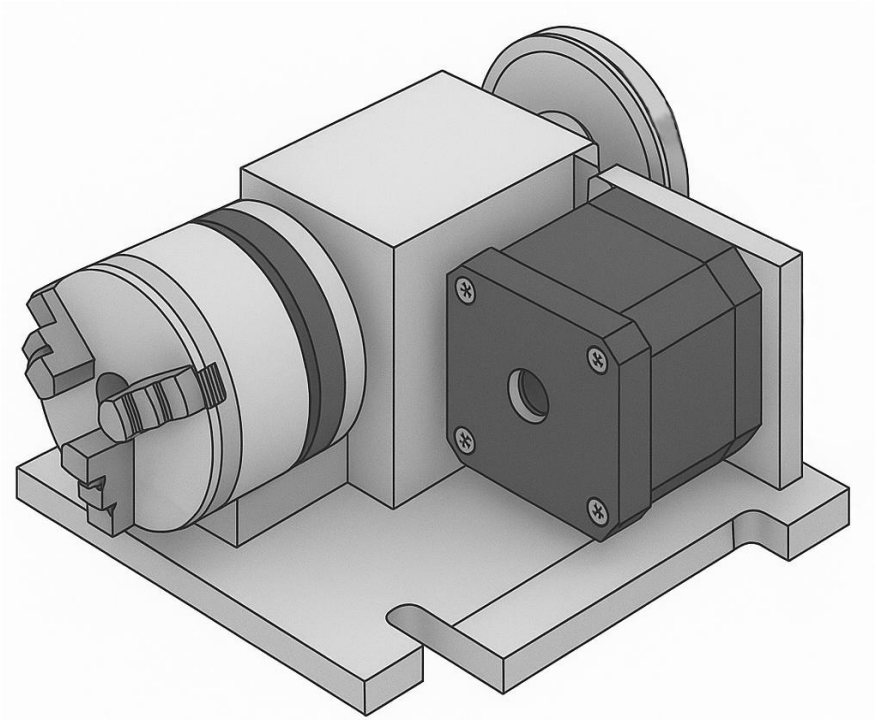

Figure 4. Isometric view of the rotary axis assembly used on the CNC machine, showing the 2-phase NEMA 23 stepper motor, timing belt, pulley system, and mounted 3-jaw chuck.

4. Implementation and Validation

4.1 Post-Processor Implementation

To implement the indexing framework described in Section 3.2, a standalone post-processing tool was developed in Python. The codebase follows a modular architecture organized into three main subsystems: (a) G-code parsing and modification, (b) graphical user interface (GUI), and (c) visualization and simulation module. Python's simplicity, strong text-processing, and extensive scientific libraries make it particularly well suited for CNC post-processing

applications, as demonstrated in recent G-code generation systems for lathes and milling machines (Abolarin, 2024).

The system remains compatible with standard GRBL-style G-code and requires no modifications to the CNC firmware. However, it assumes the presence of a rotary chuck and stepper-driven axis, as described in Section 3.3, but no controller-side changes are needed. This modular design aligns with previous approaches to CNC post-processing, including STEP-NC conversion tools built on G-code parsing frameworks that improve workflow flexibility without modifying machine firmware (Xú, Anwer, & Lavernhe, 2014).

The core logic resides in a function that parses the input G-code line by line. The parser uses regular expressions to extract coordinate information, specifically the X and Z positions. All pre-existing Y-axis commands are stripped to avoid conflicts with the added rotary indexing logic. The tool processes the original 2D toolpath and prepares it for repeated execution at discrete angular intervals. A similar filtering method has been used in academic systems retrofitted for multi-axis motion, which disable unintended commands to avoid interference with added axes (Breaz, Racz, Girjob, & Tera, 2020).

The post-processor then constructs a modified G-code program by repeating the same 2D XZ toolpath N times. Prior to each repetition, a G0 Yθ command is injected to index the part by the corresponding angular increment. These angular updates are applied only at the beginning of each pass and not during tool motion commands, simplifying the simulation of rotary operations while maintaining geometric accuracy.

Each G-code block added by the post-processor consists of:

1. A header specifying the current angular index (Y position),
2. A rapid move (G0 Yθ) to the appropriate rotary position,
3. The complete original toolpath in XZ coordinates.

Rapid traversals (G0) are preserved unmodified, ensuring rotary motion occurs only at the start of each indexed pass. To ensure accessibility and ease of use, the post-processor includes a graphical user interface built using Tkinter and a visualization module powered by Matplotlib. The interface allows users to input custom parameters, preview the original toolpaths, and visualize a 3D reconstruction of the revolved part geometry prior to execution. This interactive approach aligns with modern CNC interfaces designed for rapid parameter tuning

and on-screen feedback, such as those supported in STEP-NC simulations (Krantz & Niggemann, 2023).

Basic metadata (e.g., tool diameter, feedrate) is automatically extracted from structured G-code comments using regular expressions. Fallback values that match the most common configurations for GRBL machines have been added to the code. This reduces manual configuration and improves usability, particularly with standard CAM-generated files.

The resulting output is a fully GRBL compatible G-code file containing planar motion commands (X, Z) combined with indexed rotary emulation via the Y-axis. This approach enables rotary operations using standard linear CNC controllers, bridging the gap between conventional CAM outputs and rotary manufacturing on low-cost platforms.

A selected excerpt of the implementation, including the core parsing and indexing logic, is provided in Appendix A. The complete source code and documentation are also available in a public repository for reproducibility and further development.

In practice, the user workflow proceeds as follows: an XZ-plane G-code file generated by any CAM package is loaded into the interface. The program automatically extracts metadata such as tool diameter and feedrate, which can be edited if required. The user specifies the stock diameter and overlap factor, after which the software computes the required number of indexed passes and angular step. A 3D preview of the revolved geometry is then displayed, enabling verification prior to machining. Once confirmed, the modified G-code is exported and ready for direct execution on a GRBL-based controller.

4.2 Web Interface

The web-based component of the rotary G-code converter provides an accessible, installation-free interface for users to perform planar-to-rotary G-code conversion directly from any modern browser. This design prioritizes simplicity and portability, making it suitable for maker spaces, educational settings, or individual workshops without imposing platform-specific requirements. Users can upload standard XZ plane G-code files generated by any CAM package and configure rotary machining parameters through an intuitive web interface.

The interface is built on a Next.js framework, which supports fast server-side rendering of React pages. Tailwind CSS is employed to achieve a clean, responsive layout, while Zod performs real-time schema validation on user inputs, ensuring numerical integrity and preventing configuration errors. The platform is designed

to handle common G-code formats including .gcode, .nc, and .txt, and accepts both drag-and-drop and conventional file selection workflows.

Upon file upload, users specify the stock radius, rotary resolution in steps per revolution, and an optional feedrate to override values in the original file. The interface automatically validates these inputs, identifying missing values, invalid file types, or physically implausible parameters. Immediate feedback is provided to guide users in correcting any errors before initiating the conversion process.

After processing, the web interface generates a modified G-code file that incorporates indexed Y-axis motion corresponding to the specified rotary parameters. A 3D preview of the resulting revolved geometry is presented within the browser alongside the original 2D XZ toolpath, allowing users to verify the correctness of the toolpath and the final part geometry prior to machining. This visualization mirrors the functionality available in the desktop GUI, ensuring consistent verification across platforms (Figure 5).

The resulting workflow enables rapid, reliable conversion of planar G-code into rotary-compatible commands without requiring local software installation. Users can upload an XZ G-code file, input the required rotary parameters, preview the 3D geometry, and download a fully compatible G-code file for execution on a GRBL-controlled CNC machine. By combining portability, validation, and visualization, the web interface extends the accessibility of the proposed indexing methodology to educational, prototyping, and maker environments.

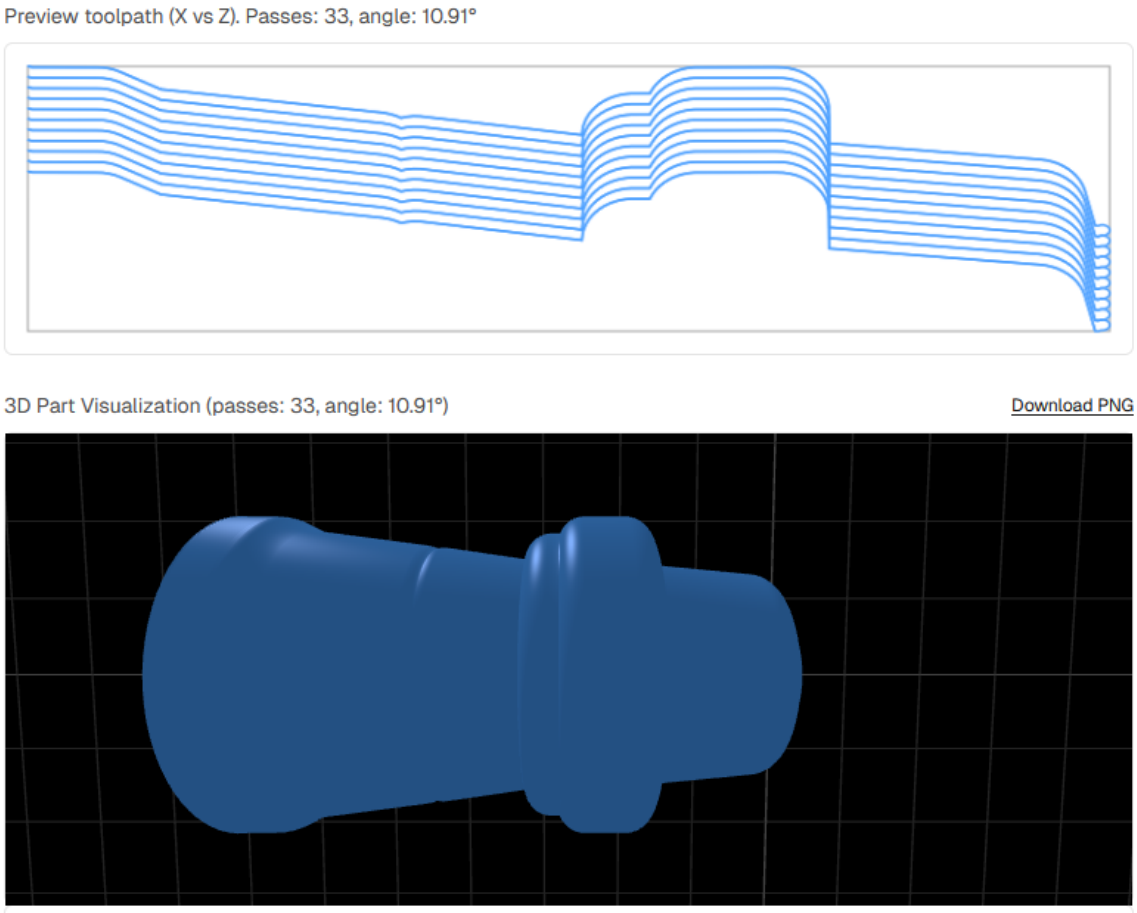

Figure 5. Web interface of the rotary G-code converter showing the original 2D XZ toolpath (left) and the corresponding 3D reconstruction of the revolved solid (right), enabling users to verify toolpaths prior to G-code export.

The web interface is hosted on Vercel and executes all file parsing and conversion server-side in ephemeral containers. Uploaded files are processed in memory and automatically discarded after conversion, and no machine control commands are executed. The service enforces a default file-size limit (≈5 MB) consistent with Vercel's runtime constraints. This ensures safe operation and protects user data while allowing browser-based access to the post-processor functionality. See Appendix B for direct access to the web interface.

4.3 Validation and Test Results

To evaluate the proposed approach, a series of test parts were machined using both hardwood and copper as stock materials. The chosen geometry corresponds to the solid of revolution shown throughout this work (see Figures 1–3), originally designed to highlight varying radial depths and curvature transitions. The same 2D toolpath, generated in Fusion 360, was post-processed using our GUI interface, with cutting parameters adjusted per material. This demonstrates the method's consistency across different setups.

Figures 6–9 summarize the results: wood output (Fig. 6), copper counterpart (Fig. 7), direct comparison (Fig. 8), and a close-up of copper toolmarks (Fig. 9) illustrating discrete angular indexing behavior.

Machining was performed on a generic 3020 CNC router equipped with a 500W air-cooled spindle. A 3.175 mm HSS two-flute flat end mill was used for all operations. The wooden part was cut dry at 1,300 mm/min feedrate, while the copper workpiece required coolant and a reduced feedrate of 500 mm/min to ensure proper chip evacuation and thermal control. Spindle RPM was increased to 12,000 for copper to reduce cutting forces and

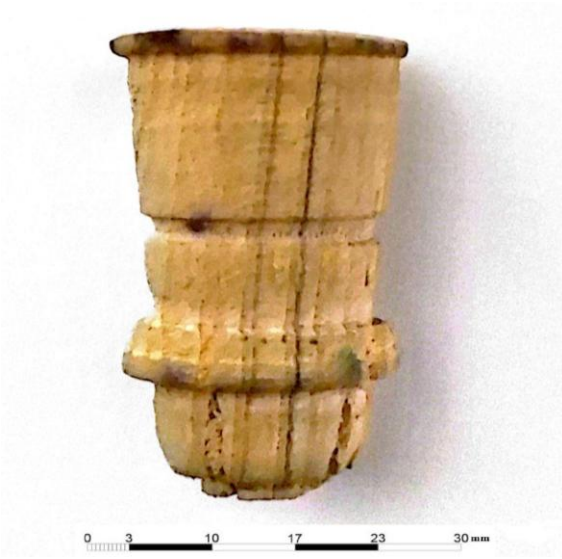

Figure 6. Machined test part in hardwood using the proposed indexing method.

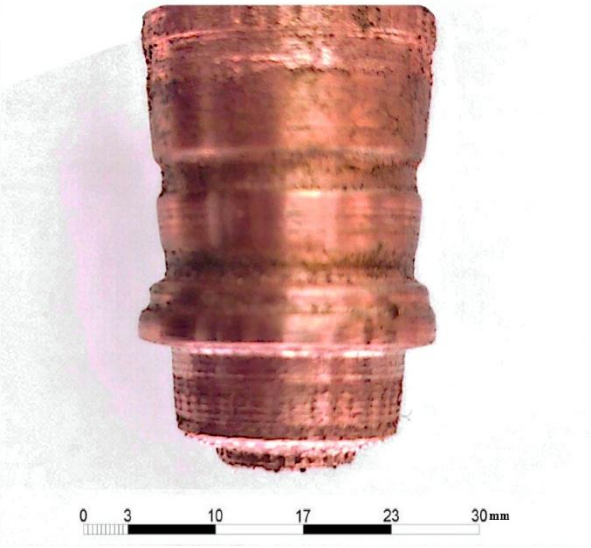

Figure 7. Copper version of the sample geometry, machined with coolant and adjusted parameters.

Figure 8. Side by side comparison of wood and copper parts, showing material related differences.

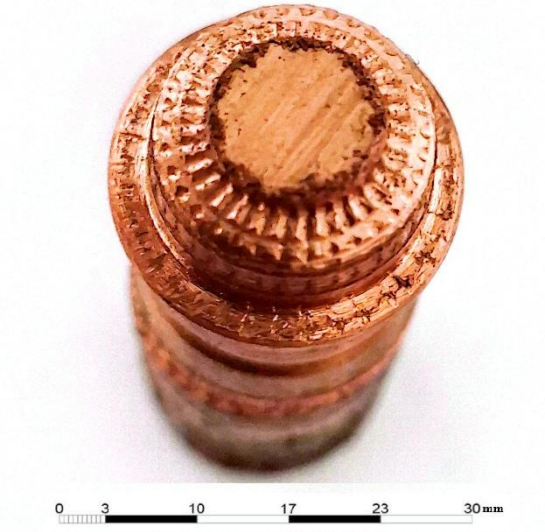

Figure 9. Close-up of copper base showing concentric toolmarks from discrete angular indexing.

avoid chatter, while wood was machined at 8,000 RPM. Depth of cut remained fixed at 0.5 mm per pass.

The post-processor was configured with a tool diameter of 3.175 mm. For a stock diameter of 22 mm, the resulting number of indexed passes was computed as N = 80, with an angular step size of approximately 4.5°. These parameters were automatically extracted from G-code comments by the interface, minimizing the risk of user error.

Dimensional validation was performed using digital calipers. The intended maximum diameter of the part was 20.50 mm. The wooden part measured 20.30 mm (−0.20 mm error), likely from tool deflection or overcut. The copper version measured 20.75 mm (+0.25 mm error), likely from burrs or elastic recovery.

Two users independently reproduced the wooden and copper parts using the same rotary hardware, relying exclusively on the GUI. Users noted the utility of the 3D preview, particularly when validating the 2D toolpath and overall 3D geometry of each indexed model prior to execution. No post-processing beyond part removal and light sanding was required.

These results confirm the system can reliably produce rotational parts with minimal intervention, validating both the indexing logic and real-world applicability

Figure 10 contrasts two parts machined from the same model but with different indexing resolutions. The right specimen was produced with a coarse angular step (low number of indexed passes), resulting in visible artifacts between successive orientations on cylindrical regions and more pronounced scallops at curvature transitions. The left specimen used a finer angular step (higher number of indexed passes), yielding a noticeably smoother surface; facets are largely imperceptible except under grazing light. The comparison makes the trade-off explicit: increasing the number of indexed passes improves surface fidelity in line with the faceting model of §3.2 [Eqs. (2) and (5)] but increases machining time and G-code size.

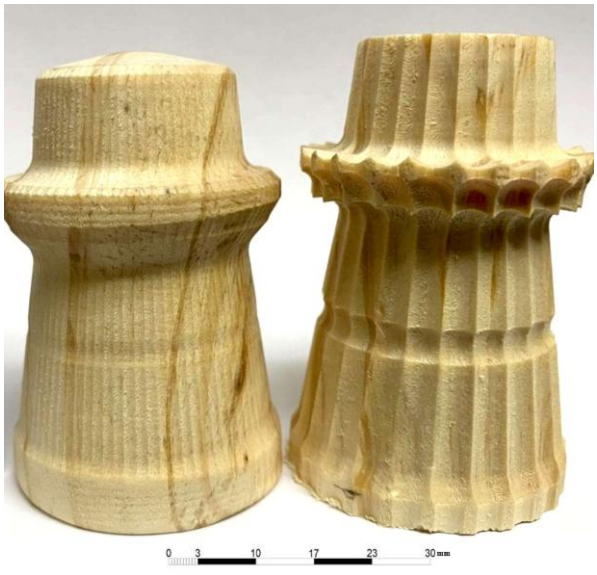

Figure 10. Side-by-side comparison of coarse N=22 (right) and fine N=80 (left) indexing. Coarse indexing exhibits visible indexing artifacts and scalloping; finer indexing improves surface continuity at the cost of longer cycle time and larger programs.

The machined parts illustrate the practical application of the post-processor; reported deviations (wood −0.20 mm, copper +0.25 mm) are indicative only, as physical results depend on CNC machine, spindle, tooling, and material. The primary validation is the correct injection of indexed rotary commands, accurate angular step calculations, and generation of GRBL-compatible G-code.

5. Discussion

The ability to emulate a fourth axis on low-cost CNC machines represents a significant step toward broadening access to advanced manufacturing techniques. By leveraging a purely software-driven approach, this method circumvents traditional hardware limitations and invites a reconsideration of what is achievable with open-source CNC platforms. Specifically, the post-processor developed here translates planar G-code into indexed rotary commands without requiring hardware modifications.

Here, we critically examine the practical benefits, inherent trade-offs, and broader implications of this approach, contextualizing it within existing manufacturing paradigms and educational frameworks.

5.1 Accessibility and Democratization of Rotary Machining

One of the most significant outcomes of this research is the improvement in accessibility to rotary machining. Traditional fourth axis integration typically requires either a specialized machine or at least an external rotary table with dedicated driver electronics and compatible CAM post processors. These systems often assume both financial and technical resources, barriers that are particularly severe in educational, hobbyist, and low-income environments.

Our method eliminates the need for any hardware modification or special firmware. Since the GRBL firmware does not natively support a fourth axis, rather than attempting to override or extend GRBL's limited motion planning, we opted to work entirely at the G-code level. The result is a strategy that allows standard 3-axis machines to *simulate* 4-axis motion by translating rotational commands into synchronized linear motion along an unused axis (in our case, the Y-axis). The post-processor transparently converts any rotary G-code into discretely indexed toolpaths compatible with standard machines.

This innovation dramatically lowers the threshold for exploring multiaxis manufacturing. By removing the dependency on specialized hardware, our tool empowers users previously excluded from advanced processes. It aligns with ongoing trends in the democratization of digital fabrication and distributed design, where the focus shifts from maximizing industrial throughput to maximizing access and creative freedom. In short, the method offers users capabilities that would otherwise require thousands of dollars in upgrades or hardware swaps.

5.2 Technical Trade Offs and Limitations

Naturally, a purely software-based approach entails trade-offs. The most significant limitation is the precision of simulated rotation. Emulated rotary motion introduces discretization artifacts, visible as longitudinal tool marks and small flat facets on curved surfaces. The severity of this artifact is directly related to the angular resolution chosen during G-code generation. While tighter step resolution improves fidelity, it also increases file size and machining time.

Furthermore, true rotary systems retain mechanical advantages—especially torque consistency—that software emulation cannot replicate. Dedicated rotary axes deliver more consistent torque than emulated rotation using coordinated X, Z, and Y motion, which can impact tool life and surface finish. This matters most when cutting materials with high hardness or when using large diameter tools, where mechanical stress can degrade both tool life and surface finish. Measured deviations of ±0.25 mm in illustrative test parts highlight practical accuracy limits for this software-only method under typical low-cost CNC settings.

The software-only approach lacks real-time feedback. Unlike closed loop systems that can adjust feedrates dynamically in response to tool load or deflection, the interpolated method assumes constant machine conditions. In highly dynamic machining environments, this could lead to overcuts or tool chatter. While these are not critical issues for prototyping or light duty engraving, they limit the method's use in precision tooling applications.

Finally, the approach presupposes accurate machine calibration. Since motion is now distributed across all three axes, any backlash, missed steps, or inconsistencies in step/mm configuration can compound over time, leading to cumulative alignment errors. While these can be minimized with careful tuning and mechanical maintenance, they remain a factor in any emulated system.

Measured deviations highlight practical accuracy achievable with the software-only method; careful tuning and maintenance can minimize cumulative errors, though some remain compared to true fourth-axis systems. Discretization artifacts, such as flat facets along curved surfaces, are inherent to the indexed approach and scale with angular step size.

5.3 Comparative Performance with True 4 Axis Systems

Our method is not intended to replace industrial fourth axes but should be evaluated for its cost-performance ratio in low-cost, open-source, and educational contexts. For instance, traditional index only rotary systems require users to pause machining, rotate the part manually or via macro controlled movement, and resume operation.

While the motion is implemented as discrete indexed passes, the workflow simulates quasi-continuous rotary machining by minimizing idle time between rotations.This class of functionality is particularly relevant in artistic or customized manufacturing, such as cylindrical engraving, totem carving, or pipe threading. The experimental results (Section 4.3) show that despite minor dimensional deviations of −0.20 mm (wood) and +0.25 mm (copper), the method reliably produces parts with acceptable accuracy for prototyping, educational, or decorative applications.

When benchmarked against low-end fourth axis attachments, our system trades some fidelity for radical gains in simplicity and price. This trade-off is particularly relevant for decorative, prototyping, or educational applications where extreme precision is less critical. No extra drivers, firmware hacks, or circuit modifications—just software. For many users, particularly those making decorative or educational parts, this tradeoff is not just acceptable, but preferred.

5.4 Educational and Decentralized Manufacturing Implications

From a pedagogical perspective, this method enables safe, low-cost exploration of rotary motion, complementing existing industrial curricula. In traditional curricula, students rarely interact with rotary systems unless institutions have access to expensive CNC setups. Even when such machines are available, student access tends to be limited due to risk, complexity, or maintenance costs.

Our system, however, can be used on almost any GRBL-based machine, some of which cost as little as $200. This democratizes access to advanced machining concepts. By allowing students to explore rotary paths, tool engagement at changing angles, and rotational motion mathematics in a safe, low-cost environment, the tool provides hands-on experience previously inaccessible due to cost or hardware constraints. This aligns with the broader goal of democratizing advanced CNC capabilities discussed in 5.1.

The implications extend beyond education. In decentralized or local manufacturing contexts, particularly in underserved regions, cost constraints remain the dominant limiting factor for CNC capability. By offering a method that simulates fourth-axis behavior using only software, we extend the functional capacity of existing hardware. This contributes to a vision of low-cost, distributed fabrication cells that can locally produce parts previously reserved for centralized factories.

6. Conclusion

Our results show that a simple Python post-processor can extend GRBL-based CNCs with practical rotary functionality, achieving reproducible deviations within ±0.25 mm on hardwood and copper, which is sufficient for prototyping, educational use, and light-duty manufacturing.

The main contributions of this study are threefold:

1. **Accessibility and Cost Reduction:** Eliminates the need for specialized rotary hardware or firmware modifications, lowering barriers to multi-axis CNC use, particularly for hobbyists, educational institutions, and small-scale manufacturers.

2. **Software-Driven Rotary Emulation:** Discrete angular indexing and automated G-code modification simulate rotational motion on standard GRBL controllers. GUI and visualization modules reduce errors and enable rapid toolpath validation.

3. **Broader Educational and Decentralized Manufacturing Impact:** Provides inexpensive exposure to multi-axis machining, enhancing CNC learning and supporting low-cost, decentralized production.

Despite these advantages, the method carries inherent trade-offs. Discretized indexing introduces minor surface artifacts and limits precision (observed deviations ±0.25 mm in Section 4.3) compared to true four-axis systems. The approach assumes stable, well-calibrated hardware and does not provide real-time feedback or closed-loop error correction, which may restrict its use in high-precision or heavy-duty applications.

Future work could explore continuous interpolation between discrete steps to reduce surface faceting, integration with higher-resolution motion controllers, and adaptive feedrate strategies to improve cutting performance in diverse materials. Additionally, extending compatibility to other low-cost CNC platforms and further refining the web-based interface could enhance accessibility and adoption in educational and maker communities worldwide.

In conclusion, this software-driven methodology represents a practical, low-cost approach to expand the capabilities of three-axis CNC machines, bridging the gap between accessibility and advanced multi-axis machining. It demonstrates that functional multi-axis enhancements can be achieved purely through software, advancing the democratization of digital fabrication.

7. Acknowledgments

Author affiliations reflect the authors' academic backgrounds; the research itself was conducted independently, without institutional support or funding. The authors gratefully acknowledge Jim Fong for independent testing of the proposed method (results shown in Figure 10), representing valuable insights from the CNC community through engagement on related online forums.

8. References

1. Abolarin, V. O. (2024). Generating G and M code for lathe machine using Python. ResearchGate. https://www.researchgate.net/publication/383870792

2. Araujo, P. R. M., & Lins, R. G. (2020). Computer vision system for workpiece referencing in three-axis machining centers. The International Journal of Advanced Manufacturing Technology, 106(5–6), 1–14. https://doi.org/10.1007/s00170-019-04626-w

3. Breaz, R., Racz, S.-G., Girjob, C., & Tera, M. (2024). Using open source software CNC controllers and modular multi-axis mechanical structure as integrated teaching environment for CAD/CAM/CAE training. IOP Conference Series: Materials Science and Engineering, 968(1), 012024. https://doi.org/10.1088/1757-899X/968/1/012024

4. Dai, Y., Li, Y., Li, Z., & Wen, W. (2022). Temperature measurement point optimization and experimental research for bi-rotary milling head of five-axis CNC machine tool. International Journal of Advanced Manufacturing Technology, 121, 309–322. https://doi.org/10.1007/s00170-022-09317-7

5. Fahmi, Z., Abidin, Z., & Nafis, O. Z. (2019). Comparative study of tool path strategies in CNC machining for part with B-spline surfaces. In iMEC-APCOMS 2019, Proceedings of the 4th International Manufacturing Engineering Conference (pp. 564–569). https://doi.org/10.1007/978-981-15-0950-6_86

6. García-Ruiz, M.-E., & Lena-Acebo, F. J. (2022). FabLabs: The Road to Distributed and Sustainable Technological Training through Digital Manufacturing. Sustainability, 14(7), 3938. https://doi.org/10.3390/su14073938

7. Huang, K. L. Mak, & P. G. Maropoulos (Eds.), Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology (pp. 469–482). Springer. https://doi.org/10.1007/978-3-642-10430-5_36

8. Jiang, P., & Li, P. (2019). Shared factory: A new production node for social manufacturing in the context of sharing economy. arXiv. https://arxiv.org/abs/1904.11377

9. Jung, H.-C., Hwang, J.-D., Park, K.-B., & Jung, Y.-G. (2011). Development of practical postprocessor for five-axis machine tool with non-orthogonal rotary axes. Journal of Central South University of Technology, 18(1), 159–164. https://doi.org/10.1007/s11771-011-0674-x

10. Krantz, M., & Niggemann, O. (2023). Diagnostic algorithms for a rotary indexing machine. arXiv preprint. https://arxiv.org/abs/2305.15934

11. Lee, R.-S., Lin, Y.-H., & Kuo, L.-A. (2010). Development of automated universal postprocessor for non-orthogonal multi-axis machine tools with modified D–H notation. In G. Q.

12. Lena-Acebo, F. J., & García-Ruiz, M. E. (2019). The FabLab movement: Democratization of digital manufacturing. In A. Guerra (Ed.), Organizational Transformation… Hershey, PA: Business Science Reference. https://www.researchgate.net/publication/330426056

13. Lin, W.-Z., Xiao, J.-X., Lin, Y.-C., Chang, B.-W., & Hung, J.-P. (2024). Innovative design and machining verification of a dual-axis swivel table for a milling machine. Advances in Science and Technology – Research Journal, 18(1), 306–319. https://doi.org/10.12913/22998624/178528

14. Lo Valvo, E., Licari, R., & Adornetto, A. (2012). CNC milling machine simulation in engineering education. International Journal of Online and Biomedical Engineering (iJOE), 8(2), 33–38. https://doi.org/10.3991/ijoe.v8i2.2047

15. Makera. (2025). Carvera Air Desktop CNC Machine with 4th Axis Bundle. Retrieved from https://www.makera.com/products/carvera-air

16. Moroşanu, G.-A., Moroșanu, F.-I., Susac, F., Teodor, V.-G., Păunoiu, V., & Baroiu, N. (2025). Implementation of an academic learning module for CNC manufacturing technology of the part "Double Fixing Fork." Inventions, 10(4), 63. https://doi.org/10.3390/inventions10040063

17. Sarguroh, S. S., & Rane, A. B. (2018, January 5). Using GRBL-Arduino-based controller to run a two-axis computerized numerical control machine. In 2018 International Conference on Smart City and Emerging Technology (ICSCET) (pp. 1–5). IEEE. https://doi.org/10.1109/ICSCET.2018.8537315

18. Srai, J. S., Kumar, M., Graham, G., & Phillips, W. (2020). Distributed manufacturing: A new form of localised production. International Journal of Operations & Production Management, 40(1), 1–20. https://doi.org/10.1108/IJOPM-08-2019-0600

19. Xu, S., Anwer, N., & Lavernhe, S. (2014). Conversion of G-code programs for milling into STEP-NC. arXiv. https://arxiv.org/abs/1412.5496

Appendix A: Source Code Excerpt:

The following excerpts include core routines from the Python-based post-processor described in Section 4.1. These functions illustrate how G-code is parsed, how rotary synchronization is computed, and how the modified toolpath is output. For brevity, auxiliary code (e.g., GUI layout and error handling) is omitted. The complete implementation is available at https://github.com/PedroPortugalS/Rotary-G-code-Post-Processor

```
def calculate_passes_and_angular_displacement():
    """Calculate number of passes and angular displacement based on toolpath and stock
        diameter"""
    try:
        stock_dia = float(stock_diameter_var.get())
        tool_dia = float(tool_diameter_var.get())

        # Calculate toolpath diameter (innermost radius reached by the tool)
        toolpath_dia = stock_dia - (2 * tool_dia)

        if toolpath_dia <= 0:
            messagebox.showerror("Error", "Tool diameter is too large for stock diameter")
            return None, None

        # Calculate number of passes needed for complete revolution
```

```
    # Based on tool diameter and desired overlap
    overlap_factor = 0.8  # Default: 80% overlap for better surface finish; can be adjusted by user
    pass_width = tool_dia * overlap_factor
    num_passes = max(1, int(math.ceil(math.pi * toolpath_dia / pass_width)))

    # Calculate angular displacement per pass
    angular_displacement = 360.0 / num_passes

    # Update display variables
    num_passes_var.set(str(num_passes))
    angular_displacement_var.set(f"{angular_displacement:.2f}")

    return num_passes, angular_displacement

except ValueError:
    return None, None
```

Appendix B: Web Interface Access.

The live web interface for the rotary G-code converter is available at: https://cnc-rotary-axis-g-code-converter-seven.vercel.app/