ESP32+FluidNC+Raspberry Pi 4B+CNCjs レシピ本

文責: 小木勇輝

2025 年 8 月 11 日

目 次

1 概念設計図

本章では、Raspberry Pi 4B 上で CNCjs(用語等は後述) を動作させ、USB 経由で ESP32(FluidNC) を直接制御する構成の全体像を示す。高校卒業直後の初学者を想定し、Linux やモーション制御の基礎から説明する。

1.1 全体構成

Raspberry Pi 4B 上で CNCjs を起動し、USB 接続された ESP32(FluidNC) に対し G-code を送信する。ESP32 はモーター制御信号 (STEP/DIR/EN) をステッピングドライバへ出力し、ドライバはステッピングモーターを駆動する。

図 1: Raspberry Pi 直結の最小構成

1.2 必要構成要素

- Raspberry Pi 4B(4GB) 本体（CNCjs 動作用）
- ESP32 ボード（FluidNC ファームウェア書き込み済み）
- ステッピングドライバ（TB6600 など、3A4 線バイポーラ出力に対応したもの）
- ステッピングモーター（NEMA17/23 など）
- スピンドルモーター (E25 コレットに対応したもの)
- 電源（モーター駆動用 24V 系 2 つ、Raspberry Pi/ESP32 用 5V 系）
- USB ケーブル（Raspberry Pi と ESP32 の接続用）

1.3 信号と電力の流れ

- 信号経路: Raspberry Pi (CNCjs) → ESP32 (FluidNC) → ステッピングドライバ → モーター
- 電力系統: 24V 系 (モーター/ドライバ駆動), 5V 系 (Raspberry Pi/ESP32 駆動)

1.4 ソフトウェア構成

- Raspberry Pi OS:Raspberry Pi の OS。Mac OS の遠い親戚。
- CNCjs: ブラウザベースの CNC 操作 UI。Raspberry Pi 上で動作。
- FluidNC: ESP32 上で動作する CNC 制御ファームウェア。G-code を解釈しモーション制御信号を生成。

1.5 用語集

末尾の「おまけ」欄を参照すること。

1.6 最低限調べておいてほしいこと

本構成を安全かつ効率的に理解し運用するため、以下の基礎事項は事前に調べておくことを推奨する。いずれも高度な専門書を読む必要はないが、概要と主要用語を把握しておくべきである。

Linux とは何か オープンソースの UNIX 系オペレーティングシステムであり、Raspberry Pi OS もこれに属する。ファイルシステムの階層構造、ユーザーと権限の概念、パッケージマネージャ (例: `apt`)、コマンドライン操作 (CLI) などの基本を理解しておくこと。

Raspberry Pi とは何か ARM 系プロセッサを搭載した小型シングルボードコンピュータであり、低消費電力で Linux を動作させられる。本構成では CNCjs のホストとして使用する。GPIO 端子や USB ポートなどの基本的なハード構造も確認しておくこと。

高校電磁気の復習 直流回路のオームの法則 ($V = IR$)、抵抗・コイル・コンデンサの基本特性、電力計算 ($P = VI$)、直列/並列接続の違い、電流と磁界の関係、誘導起電力の概念など。ステッピングモーターやドライバの動作理解に不可欠である。

ステッピングモーターとは何か 電気パルスの入力に応じて一定角度ずつ回転するモーター。1 回転あたりのステップ数、マイクロステップ制御、保持トルク、脱調の概念を把握すること。STEP/DIR 信号と回転方向・回転量の関係も確認する。

CNC による死亡事故例やヒヤリハット CNC 装置は強力な駆動系を持ち、加工対象や工具が高速で移動するため、不適切な操作や安全管理不足により死亡事故や重大な負傷が発生している。過去には回転工具への巻き込まれ、破損工具の飛散、固定不良材料の射出、制御系誤動作による衝突などが報告されている。実際の事故事例やヒヤリハット事例を調べ、危険要因とその防止策を理解しておくこと。非常停止 (E-Stop) や安全カバーの有効性についても確認する。

1.7 安全留意事項

- モーター動作中は手や物を可動部に近づけない。
- 電源投入前に配線の正誤と固定状態を確認する。
- 電源系は系統ごとにヒューズを設ける。
- 手袋をつけて NC を稼働させない。

2 セットアップ編

2.1 ESP32 ボードの準備とファーム書き込み

CNC 用モーションコントローラとして ESP32 を用いる。ファームウェアは FluidNC を採用する。初期導入は PC 上の Google Chrome から FluidNC 公式のオンラインインストーラを用いる。Web Serial API を使うため Chrome を前提とする。

手順の概観は次の通り。

1. ESP32 を USB で PC に接続する（給電は PC 側/基板側の二冗長系で行う）。
2. Google Chrome で FluidNC Web Installer を開き、Connect で該当シリアルポートを選択する。
3. Install から Latest ビルド（Wi-Fi 有効版）を選択して書き込む。[1]

[1] 運用では Wi-Fi は使わない。導入や更新時のみ使う方針とする。

4. 書き込み完了後、**インストーラの「Config」（または「Files」）欄**から `config.yaml` を新規作成する。基本パラメータは後で対話的に調整する（手入力で YAML を記入せず、インストーラ側で編集・保存する）。

注意点として、各軸の STEP/DIR/EN のピン割付は、ESP32 で**出力に使ってよい GPIO** を選ぶ。ブートストラップ用のストラッピングピン（GPIO0, 2, 5, 12, 15）やフラッシュ接続ピン（GPIO6〜11）、入力専用ピン（GPIO34〜39）は避ける。UART の TX/RX（GPIO1/3）も原則避ける。割付は後述の設定方針と合わせて決定する。ピン番号は必ず最新の情報を確認すること。

2.2 FluidNC の基本設定（Config 欄で行う方針）

本書の**基本パラメータを信用せず**、FluidNC Web Installer の **Config 欄から対話的に調整**する。理由は、機械固有の要素（スクリューピッチ、マイクロステップ、剛性、負荷、ドライバ仕様）が初期値を大きく左右するためである。次の考え方で進める。

設定の流れ（対話的チューニング）

1. 軸とモータの論理構成を定義する（X/Y/Z/A、スレーブ軸の有無など）。
2. 各軸の**ピン割付**を決める。前節の**使用可能 GPIO の制約**を満たす組合せを選ぶ。
3. **steps_per_mm**(例:320steps/mm) は実測で校正する。適当な手段での測長を繰り返し、誤差を詰める。
4. **max_rate** と **acceleration** は安全側から立ち上げ、脱調・共振・発熱を観察しつつ段階的に上げる。
5. 方向反転やリミット/プローブ極性は、実機の挙動に合わせて修正する。

補足: Config 欄は Web UI での保存を前提とする。エディタでのテキスト貼付は避ける。

2.3 Raspberry Pi 4B への CNCjs 導入（デスクトップ版 OS + 公式 Dockerfile）

CNCjs は Web ベースの G-code 送受信/UI であり、Raspberry Pi 4B で常用する。OS は **Raspberry Pi OS（64-bit, Desktop 版）**を用いる。導入時のみ Wi-Fi を有効化し、**運用は有線（USB/有線 LAN）**とする。「電プチ」（突然の電源断）対策として Overlay File System を有効化する運用を推奨する。

OS インストール（Raspberry Pi Imager）

1. PC で Raspberry Pi Imager を起動し、OS は **Raspberry Pi OS (64-bit) with desktop** を選ぶ。
2. Advanced Settings（歯車）で、ホスト名／ロケール、**SSH 有効化**、**Wi-Fi 設定（導入時のみ使用）**など好みの設定を事前指定して書き込む。
3. 初回起動後、必要なパッケージを更新する。

Listing 1: 初期パッケージ導入（APT）

```
# Raspberry Pi OSの更新
sudo apt update
sudo apt -y full-upgrade

# 導入に使うツール群
sudo apt -y install git curl build-essential

#Dockerのインストール、公式HPを参照
curl -sSL https://get.docker.com | sh
sudo docker ps

#再起動
sudo reboot now
```

電プチ対策（Overlay File System）

Raspberry Pi OS は `raspi-config` から **Overlay File System** を有効化できる。運用中は rootfs を実質 Read-Only 化し、変更は RAM 上に乗せる。設定変更や更新時は一時的にこれを無効化して再起動する運用にする。
***Raspberry Pi* は正常にシャットダウンされないと高確率で *OS* のデータが吹き飛ぶ。悲しい思いをしたくなければこの項を実行すること。**

Listing 2: OverlayFS の有効化（対話/非対話の例）

```
# 対話UIで行う方法（raspi-config）

sudo raspi-config # Performance Options -> Overlay File System -> Enable
# 非対話UIで行う方法（新しめのraspi-configで有効）

sudo raspi-config nonint enable_overlayfs
sudo reboot
```

CNCjs の導入（公式 Dockerfile を用いる）

公式リポジトリの Dockerfile を用いて **Docker イメージをローカルビルド**し、コンテナとして常駐運用する。公式 Dockerfile は `dist/cncjs` の成果物を前提とするため、**先に Node/Yarn でビルドする**。

1) CNCjs のビルド（dist 生成）　このように行う。

Listing 3: Node/Yarn でフロントをビルド（Pi 上で可）

```
Node 18系を前提（bookworm既定のnodejsで可。必要ならNodeSource等を使用）

sudo apt -y install nodejs npm
sudo npm -g install yarn

git clone https://github.com/cncjs/cncjs.git
cd cncjs
yarn install
yarn run build # dist/cncjs が生成される
```

2) 公式 Dockerfile でイメージ化　このように行う。

Listing 4: Docker ビルドと起動

```
リポジトリ直下にある公式Dockerfileを使用

docker build -t cncjs:local -f Dockerfile .
デフォルトExposeは 8000。ホスト8080に割り当てる例。
/dev/ttyUSB* (or /dev/ttyACM*) をコンテナへ渡す

docker run -d --name cncjs
--restart=unless-stopped
--device /dev/ttyUSB0
-p 8080:8000 cncjs:local
```

3) 起動・アクセス・運用方針

- ブラウザから `http://<raspi-ip>:8080/` にアクセスし、CNCjs UI を操作する。ブラウザは Google Chrome を推奨する。
- ESP32 は USB 直結し、CNCjs のポート選択で `/dev/ttyUSB*` または `/dev/ttyACM*` を選ぶ。
- 運用は有線前提、Wi-Fi は基本無効化（導入・更新時のみ有効）。

Listing 5: systemd で自動起動（コンテナを常駐させる例）

```
/etc/systemd/system/cncjs.service

[Unit]
Description=CNCjs container
After=network-online.target docker.service
Wants=docker.service

[Service]
Restart=always
ExecStart=/usr/bin/docker start -a cncjs
ExecStop=/usr/bin/docker stop cncjs

[Install]
WantedBy=multi-user.target
有効化

sudo systemctl daemon-reload
sudo systemctl enable --now cncjs
```

3　機械構築・接続編

本節では、本 CNC 制御システムの機械構築と配線について述べる。

3.1　必要部品と仕様確認

本システムは制御系と駆動系が分離される。制御系には ESP32 マイコン、ステッピングモータドライバ、Raspberry Pi 4B（CNCjs 実行用）、制御系電源

ユニット (DC5V、DC3.3V のもの) を用いる。駆動系にはステッピングモータと伝達機構（リードスクリュー）、駆動系電源ユニット (DC12-24V) を用いる。ステッピングモータは株式会社オオツカと相談して仕様（トルク、定格電流、保持トルク）を決定する。安全系には非常停止スイッチ、ヒューズを備えるべきである。

現状、スピンドルは制御系と完全に分離されており、独立電源で運用されている。リミットスイッチは物理的には設置済みだが未接続であり、Z プローブは搭載していない。

3.2 配線計画と電源構成

配線計画では「系統分離」と「耐環境性」を重視する。モータ用電源、制御系電源、スピンドル電源は物理的に分離し、相互にノイズが回り込まないようにする。モータドライバは典型的にフォトカプラ入力を備えるため、ESP32 側の GND とモータ側の GND を共通化する必要はない。

事実上の屋外設置である点を考慮し、耐候性のある屋外用ケーブル（ビニルキャブタイヤケーブル VCT、耐 UV シース付きケーブルなど）を使用する。信号線とモータ線は別ルートで配線し、筐体に導入する。整備性、可用性、信頼性の観点から、可能な限りコネクタの数が少ないことが好ましい。

3.3 ステッピングモータとドライバ接続

モータには 4 本の電源線があり、A+/A-/B+/B-の順に接続する。配線色はメーカーにより異なるため、付属資料または導通試験により確認する。片方の相を逆接続することにより回転方向が逆転する。

ドライバ側では、電源入力（例: 24VDC）とモータ出力端子のほか、STEP/DIR/EN 信号入力端子がある。これらは ESP32 の GPIO ピンから出力され、ドライバのフォトカプラ入力に接続する。フォトカプラの＋端子に信号線、－端子に ESP32 側の GND を接続する。マイクロステップ設定は DIP スイッチで行い、機械分解能と加工速度に応じて選択する（例: 1/32 ステップ設定）。電流制限も DIP スイッチで設定し、モータ定格電流の 90 %程度に設定するのが一般的である。

3.4 リミットスイッチ

X/Y/A 軸にリミットスイッチが設置されているが、現状未接続であるため、FluidNC ではソフトリミットまたは手動位置決めで運用する。将来接続する場合は NC（常閉）接点を用い、配線には屋外用シールドケーブルを採用する。ノイズ防止のため、信号線はツイストペア構造が望ましい。

3.5　Z プローブ

現状 Z プローブは搭載していない。ツール長補正は手動で行い、相対座標系の Z 位置を調整する。

3.6　スピンドル

スピンドルは完全に制御系から分離され、独立電源で運用する。制御系からの ON/OFF 信号は現状送らない。将来的に制御を導入する場合は、ESP32 から PWM 信号を出力し、適切な変換回路を介して可変電源のアナログ入力に接続する。この場合、信号線は光絶縁するのが好ましい。

3.7　非常停止回路

非常停止（E-Stop）スイッチはモータドライバの EN ラインにプルダウンまたはプルアップして接続し、押下時にモータ駆動を即時停止させる。加えて、制御系電源を遮断する回路を組み込むことで、より高い安全性を確保できる。屋外用の E-Stop スイッチは IP65 以上の防水/防塵仕様かつ B 接点のものを採用するのが好ましい。この場合、ESP32 からの EN 出力は適切な抵抗器を介して接続する、または絶縁する必要がある。

3.8　ノイズ対策と配線整理

モータ線と信号線は物理的に離して配線する。屋外用ケーブルを用いる場合も、必要に応じてシールドケーブルやフェライトコアを追加し、ノイズの進入を防ぐ。配線はケーブルクランプや配線ダクトで固定し、振動による断線を防止する。

3.9　接続テストと初動確認

全配線が完了したら、以下の手順で動作確認を行う。

1. モータが正方向・逆方向に正しく回転するか確認する。X/A 軸が逆方向である可能性があるため、Y 軸アームが取り外されている状態であることが好ましい。
2. リミットスイッチ未接続の前提で動作範囲を確認する。
3. スピンドルを別系統で安全に起動確認する。
4. CNCjs から手動ジョグ操作を行い、座標指令と実際の移動方向が一致しているか確認する。

3.10　トラブルシュート

モータが動作しない場合は、ドライバ入力電圧、フォトカプラ入力電圧、GPIO 出力状態を順に確認する。ノイズによる誤動作が疑われる場合は、信号線シールド強化や配線経路変更を行う。スピンドルが起動しない場合は、独立電源側のスイッチと安全回路を確認する。オシロスコープがあるのが好ましいが、電圧計単体でも信号の確認は行えるので、適切なツールを用いること。

4　制御パラメータ

CNC マシンの動作精度および加工効率は、制御パラメータの適切な設定によって大きく左右される。制御パラメータとは、モータ駆動信号の特性や移動時の加減速、座標系の基準、機構のガタ（バックラッシュ）補正など、機械の運動挙動を規定するものである。本節では、FluidNC を用いた本環境における主要な制御パラメータの設定方針を述べる。

4.1　ステップパルスと加速度設定

ステッピングモータは、ドライバから与えられるパルス信号に同期して回転する。このパルス信号の幅（pulse width）および間隔は、モータの応答性と最大速度に影響する。パルス幅が短すぎるとモータが信号を取りこぼし、長すぎると最高速度が低下する他、誤作動の可能性が高まる。初期設定ではモータドライバの推奨値を用い、安定動作を確認した上で調整を行うと良い。大抵の場合、初期設定で問題ない。

加速度（acceleration）は、停止状態から目標速度に到達するまでの速度増加率である。過大な加速度設定は脱調を引き起こし、過小な設定は加工時間を延長させる。初期設定では低めに設定し、加工中に問題が発生しない範囲で徐々に引き上げる。

4.2　機械原点およびワーク原点の設定

機械原点（Machine Zero）は、CNC 機械が認識する絶対座標系の基準位置である。一般に、各軸のリミットスイッチに到達した位置を原点とする。電源投入後は原点復帰（homing）を行い、機械座標を確定させるのが基本である。現状、リミットスイッチが存在しないので無意味な概念である。

ワーク原点（Work Zero）は、加工対象物（ワーク）の座標系の基準位置である。加工する図面の基準点（例：材料左下角や中央）を決定し、その位置を機械に記憶させる。FluidNC では G54 などのワーク座標系コマンドを用いて設定する事ができるが、手動で設定するのが好ましい。

4.3　バックラッシュ補正

バックラッシュは、送りねじやギアの機構的遊びによって発生する位置誤差である。本環境では、CAM ソフトのオフセット設定により補正を行う。補正値は以下の方法で求める。

1. 長辺と短辺 (例:30*40) を持つ長方形の輪郭を切削する。
2. 実測値と設計値を比較し、各軸ごとの寸法誤差を算出する。
3. CAM ソフトにバックラッシュ補正オフセットを記入し、再度切削して誤差が最小になるよう調整する。

この工程を繰り返し、可能な限りバックラッシュの影響を低減する。

4.4　加工前の確認事項

制御パラメータを適切に設定しても、加工対象が傾いていたり固定が不十分であれば、寸法精度は保証されない。加工を行う前に、以下を必ず確認すること。

- ワーク設置面が水平であること（水平器等で確認）
- ワークが加工中に動かないよう確実に固定されていること（ネジ等の締結状態を確認）

フラットベッドは湿度変化に伴う木材の反り・歪み、および過切削による表面凹凸の発生が避けられない構造である。このため、年に一度程度を目安に水平出しを実施し、加工精度を維持することが望ましい。

4.5　フラットベッドの水平出し

本機のフラットベッドはスタイロフォーム製であり、湿度による変形や過切削による凹凸の影響を受けやすい。そのため、年に一度程度を目安に水平出しを行い、加工精度を維持することが望ましい。対象ベッドの標準サイズは幅 910 mm、長さ 1820 mm である。

作業目的

水平出しは、ベッド全体を浅く切削して平坦性を回復させる作業である。スタイロフォーム製ベッドでは切り込み量を 3〜5 mm 程度まで許容できるため、凹凸の修正を一度で行える場合が多い。

事前準備

- 加工前に必ずベッド上の釘やネジ、その他金属部品をすべて撤去する。これを怠ると工具の破損や事故につながる。
- 面取り用またはサーフェスカッター（直径 16〜25 mm 程度、2〜3 枚刃超硬）を準備する。
- 集塵機やエアブローにより切削粉の排出経路を確保する。

手順

1. **原点設定**：ベッド左前端をワーク原点とする。Z 原点は適切に設定する。
2. **加工範囲設定**：実寸に基づき、X 方向 910 mm、Y 方向 1820 mm の範囲を指定する。安全マージンとして外周 20 mm 程度を追加してもよい。
3. **切削条件設定**：
    - 切り込み量（Z 方向）：3.0〜5.0 mm ／パス
    - 送り速度：500〜800 mm/min
    - スピンドル回転数：10,000〜16,000 rpm
4. **パス生成**：CAM ソフトにて全面を覆う走査パスを作成する。各パスの間隔は工具径の 80〜90%程度とし、パスをオーバーラップさせることで段差を低減する。X 方向と Y 方向の交互切削を推奨するが、非常に時間がかかるのでやらなくても良い。
5. **切削実行**：初回は小さい領域を浅めに加工し、ベッド全体の高低差を確認する。必要に応じて深さを追加して全面を削る。
6. **仕上げ確認**：加工後、凹凸が許容範囲内であれば作業を終了する。

注意事項

- 切り込み量が大きい場合でも、ベッド全体を一度で削り切らず、2〜3 回に分けて行う方が精度を確保しやすい。
- 水平出し後は必ず座標系および固定治具の高さを再設定する。

適切な設定を行わないと、30 時間以上かかる可能性がある。太いエンドミルを使うことや、サーフェスカッターを購入することを検討すると良い。「裏技」として、6〜8 時間程度でこれを完了させる方法が存在するが、再現性に乏しいこと、あまり好ましくない方法であることからここには記載しない。

5　おまけ

用語集（機械／加工）

ステッピングモーター パルス信号で一定角度ずつ回転するモーター。位置制御に用いる。

ステッピングドライバ ステッピングモーターを駆動する回路／モジュール。電流制御やマイクロステップ生成を行う。

NEMA17・NEMA23 モーターの取付け寸法規格。数字はフレーム外形（インチ）。NEMA17 ≈ 1.7 in（約 42 mm）、NEMA23 ≈ 2.3 in（約 57 mm）。

バイポーラ モーター巻線に双方向の電流を流す駆動方式。一般にユニポーラより高トルクだが電流制御が必要。

リードスクリュー 回転を直線運動に変えるねじ機構。

マイクロステップ 巻線電流を細かく制御して 1 ステップを更に分割する駆動。滑らかさ・振動低減に有効（絶対精度は機械剛性や負荷に依存）。

保持トルク 通電して静止を保てる最大トルク。大きいほど外力に強い。

脱調 指令ステップと実際の回転がずれる現象。過負荷・加減速過大・共振などが原因。

共振 機構の固有振動数と駆動周波数が一致して振動が増える現象。速度帯の回避や減衰で対策。

バックラッシュ ねじやギアの遊びによる方向反転時のガタ。オフセット補正やプリロードで低減。

リミットスイッチ 移動端を検出する機械スイッチ。原点復帰や過走行防止に用いる。

NC（常閉）接点 通常は導通し、作動で開放する接点。断線を検知しやすい。

Z プローブ 工具先端の Z=0 位置（ワーク上面など）を自動検出する仕組み。

スピンドル 工具を回転させる主軸（モーター）。回転数や剛性が切削品質に影響。

エンドミル 側面・底面で切削できる回転工具。材質・刃数・コーティングで用途が分かれる。

サーフェスカッター ベッドや材料の面出し（平面出し）に使う広幅工具。

送り速度 工具とワークの相対移動速度（mm/min）。切削負荷や仕上げ面に影響。

ツール長補正 工具ごとの長さ差を Z オフセットで補正する機能（TLO）。

ツイストペア（線） 2 本の導線を撚って外来ノイズを相殺する配線方法。

フェライトコア ケーブルに装着して高周波ノイズを減衰させる磁性体部品。

ケーブルクランプ ケーブルを固定し、引張応力や振動から保護する部品。

配線ダクト ケーブルを収めて整理する樹脂製トラフ（スリット入りも多い）。

用語集（電気／信号）

STEP ステッピングモーターの回転指令パルス信号。

DIR ステッピングモーターの回転方向を指定する信号。

EN モーター駆動の有効／無効を切り替える信号（Enable）。

GPIO 汎用入出力ピン（General Purpose Input/Output）。

UART（TX/RX） シリアル通信方式の一種。TX は送信、RX は受信。

フォトカプラ 光を介して信号を伝える絶縁素子。ノイズ耐性・安全性向上に使う。

GND 電気回路の基準電位（グラウンド）。

PWM パルス幅変調。デジタル信号の ON/OFF 比でアナログ量を表現。

DIP スイッチ 小型のレバー式切り替えスイッチ。設定変更に使う。

ストラッピングピン 起動時の設定を決定するための固定配線端子。

EN ライン Enable 信号線。機器や回路の動作を有効化する。

用語集（CNC 固有）

G-code CNC 機械を制御するための標準的な命令コード。

機械原点（Machine Zero） 機械の基準座標系の原点位置。

ワーク原点（Work Zero） 加工対象物に対する基準位置。

Homing（原点復帰） リミットスイッチ等を使って機械原点に戻す動作。

ジョグ操作 手動 (ソフト上) で軸を少しずつ動かす操作。

ソフトリミット 制御ソフト上で設定された移動範囲制限。

用語集（安全／規格）

E-Stop（非常停止） 緊急時に機械の動作を即時停止させる装置（Emergency Stop）。

安全カバー 作業者を切削くずや工具から保護する覆い。安全規格に準拠する場合も多い。

IP65 国際防水・防塵規格。粉塵の侵入を完全防止し、あらゆる方向からの噴流水にも耐える。

B 接点 作動すると導通が切れる接点（NC と同義）。安全検出回路で使用されることが多い。

ヒヤリハット 事故には至らないが、危険につながりかけた事象の記録・共有。

用語集（数量／単位／制御パラメータ）

steps per mm 1 mm 移動するのに必要なステップ数。機構寸法とマイクロステップ設定で決まる。

max rate 最大移動速度。加工や機構に応じて設定する。

acceleration 加速度。加減速のスムーズさや脱調防止に影響する。

rpm 回転毎分（revolutions per minute）。モーターやスピンドルの回転速度の単位。

mm/min 移動速度や送り速度の単位（毎分ミリメートル）。

参考文献

[1] FluidNC Web Installer（公式）.

[2] FluidNC Config file Overview（公式 Wiki）.

[3] FluidNC Motion Setup（公式 Wiki）.

[4] Espressif ESP-IDF: GPIO & RTC GPIO（公式）.

[5] Espressif ESP32 Series Datasheet（公式）.

[6] Raspberry Pi Documentation: Wi-Fi/Imager Advanced Settings（公式）.

[7] Raspberry Pi Whitepaper: Making a more resilient file system（公式）.

[8] CNCjs Installation / Raspberry Pi Setup Guide（公式）.

[9] CNCjs 公式 Dockerfile（master）.

[10] Docker Hub: cncjs/cncjs（公式イメージ）.