ESP32+FluidNC+Raspberry Pi 4B+CNCjs セットアップ草案

草案者: Wood Burned

2025 年 8 月 11 日

1　概念設計図

本章では、Raspberry Pi 4B 上で CNCjs 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 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）
- 電源（モーター駆動用 24V 系、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
- 電力経路: 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(Operating Software)。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

G-code CNC 機械を制御するための標準命令セット。例: G0/G1 は直線移動、などの規則がある。

ステッピングモーター 一定角度ごとに回転するモーター。精密位置決めに用いる。

STEP/DIR/EN ドライバに送る基本制御信号。STEP はパルス数で移動量、DIR は方向、EN は有効化。

ステップ ステッピングモーターが動作する最小単位。

ホーミング 原点位置を決定する動作。現状省略。

Linux オープンソースのある条件を満たす OS の総称。Raspberry Pi OS も Linux 系。

CLI(Command Line Interface) 文字入力でコンピュータを操作する方式。ターミナルやシェルで実行。「コマンドプロンプト」だと思って良い。

シェル CLI の実行環境。bash や zsh など。

USB シリアル通信 USB 経由でシリアルデータを送受信する方式。ESP32 との接続に用いる。

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]
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。

[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。

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、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 上に乗る。設定変更や更新時は一時的に無効化して再起動する運用にする。

Listing 2: OverlayFS の有効化（対話/非対話の例）

```
対話（raspi-config）

sudo raspi-config # Performance Options -> Overlay File System -> Enable
非対話（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
```

2.4　この草案の要点（実施順）

1. Google Chrome で FluidNC Web Installer を使い、ESP32 へ書き込み後、**インストーラの Config 欄**から `config.yaml` を作って編集する（配布 YAML は用いない）。
2. 各軸のピン割付は ESP32 の GPIO 制約を満たす組合せにする（ストラッピング・フラッシュ・入力専用ピンは避ける）。

3. Raspberry Pi OS は**デスクトップ版**を Raspberry Pi Imager で導入。導入時のみ Wi-Fi 有効、以後は無効化。電プチ対策で OverlayFS を有効化。

4. CNCjs は**公式 Dockerfile** でローカルビルドし、コンテナ運用する（デフォルト Expose は 8000、例では 8080 へ割当）。

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）.