

AI 執事 ミケ

補足資料

氏森 充

本補足資料では、AI 執事ミケ（以下、AI ミケ）を実際に動作させるための環境構築手順、起動方法、およびプログラム構成について解説します。AI ミケは、LLM（Large Language Model）と MCP（Model Context Protocol）を組み合わせで構築した AI コミュニケーションツールです。

本誌面では紙幅の制約上十分に説明できなかった、システム全体のアーキテクチャ構成や各コンポーネントの役割分担、さらに動作環境の具体的な構築手順について、本資料では補足的に整理しています。

実際に手を動かして検証する読者が、再現性をもって AI ミケを構築できることを目的としています。



1.	初めに	2
2.	プログラム説明（補足）	3
3.	環境構築（サーバ側環境設定）	10
4.	環境構築（ラズベリーパイの設定）	14
5.	環境構築（Windows の設定）	25
6.	各種設定ファイル説明	26
7.	プログラムインストール（アプリ版）	32
8.	プログラムインストール（Docker 版）	34
9.	開発メモ	43

1. 初めに

本資料で紹介する AI ミケのプログラムは、用途や検証目的に応じて、次に示す **2 種類**のシステム構成を用意しています

1.1. 2つのシステム構成

1 : Docker 版（本誌で紹介した構成）

Docker 版は、**Raspberry Pi** 上での動作を前提としており、様々な設置環境での運用を想定した構成です。

限られたリソース環境でも安定して動作させるため、機能を明確に分離した Docker 構成を採用しています。Docker の構成は、次の **2 つの Docker コンポーネントを連携**させて AI ミケを構成する方式です。

- **UI 機能**をクライアント側コンテナ
- **応答判定機能**をサーバ側コンテナ

なお、お天気 MCP サーバは、応答判定機能と同じサーバ側コンテナ内に配置していますが、**実行プロセスとしては分離**されています。これにより、応答判定処理と外部情報取得処理を疎結合に保ち、構成の見通しと保守性を向上させています。

MCP クライアントと MCP サーバ間のトランスポートには **Streamable HTTP** を使用しています。この方式を採用することで、将来的に AI ミケの外部へ **新たな MCP サーバを追加**することも可能とする、拡張性を重視した設計としています。

2 : アプリ版（Windows 上で動作する簡易構成）

アプリ版は、Windows 環境上で Python の単一アプリケーションとして AI ミケを起動できる簡易構成です。

音声認識エンジンには、Windows 用にビルドされた Julius を使用します。

このため、Docker 環境の構築やソースコードのコンパイル作業は不要で、**短時間で導入・検証**できる点が特徴です。

この構成では、セットアップを簡略化するために、MCP サーバと MCP クライアント間のトランスポート方式として **STDIO（標準入出力）** を使用しています。

1.2. 構成選択の指針

まずは、導入が容易な **アプリ版**を利用して、MCP の基本的な動作や AI ミケの全体像を体験することを推奨します。

その後、より**実運用に近い構成や拡張性の検証**を行いたい場合には、Docker 版を利用することで、MCP を用いたシステム構築の全体像を段階的に理解することができます。

2. プログラム説明（補足）

2.1. シーケンス図（AI ミケ 全体）

本誌に掲載した機能関連図に対応する シーケンス図を図 2.1 に示します。

図 2.1

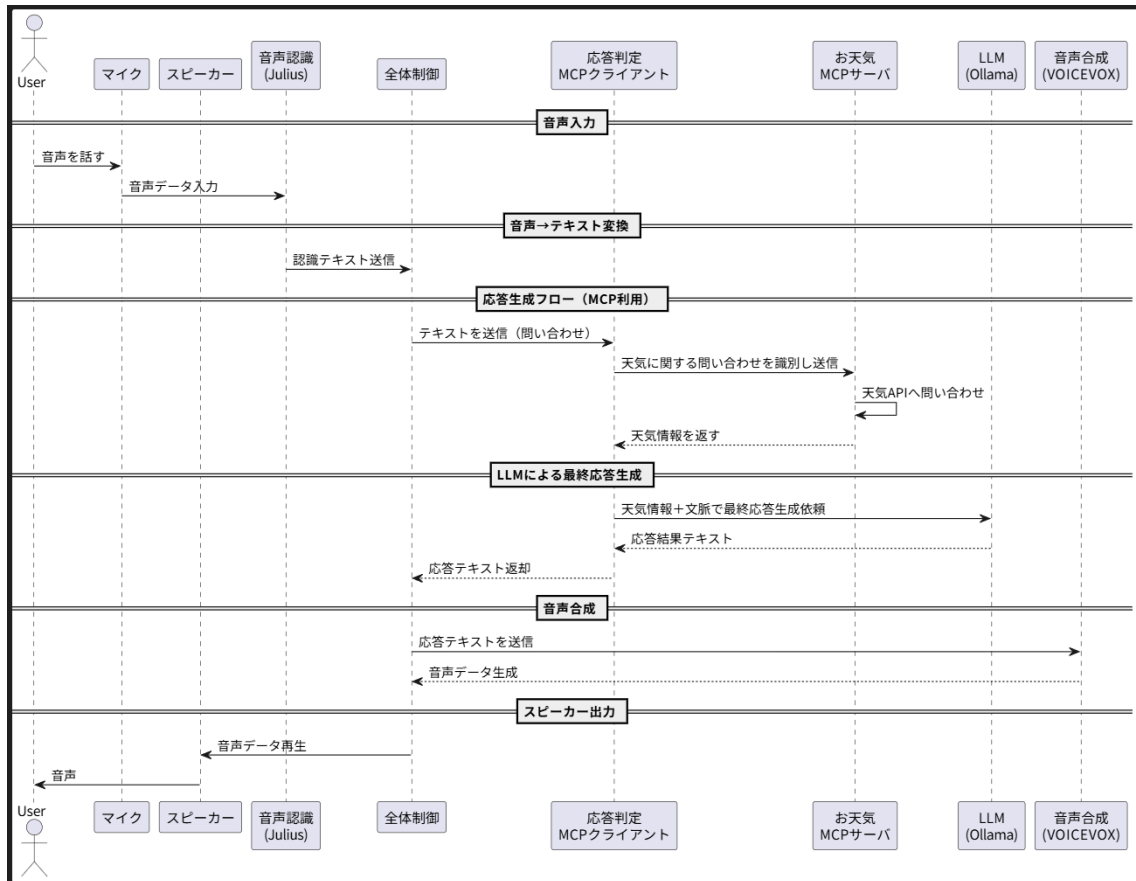


図 2.1 は、AI ミケにおける音声入力から音声応答出力までの一連の処理の流れを示したものです。音声認識、応答判定、外部情報取得、LLM による文章生成、音声合成といった各処理が、どのコンポーネントで、どの順序で実行されるかを時系列で表現しています。

2.1.1. 処理の流れ

ユーザーがマイクに向かって発話した音声は、音声認識エンジン **Julius** によりテキストへ変換されます。生成された認識テキストは **全体制御モジュール**に渡され、応答生成フローへと進みます。応答内容の判定は **MCP クライアント**が担当し、問い合わせ内容に応じて適切な MCP サーバへ処理を委譲します。**MCP サーバ**から取得した情報は、元の問い合わせテキストと合わせて **LLM** に送信されます。LLM はこれらの情報を文脈として利用し、ユーザーに対する **自然な文章形式の最終応答文**を生成します。

生成された応答テキストは **音声合成処理**により音声データへ変換され、最終的にスピーカーから出力されます。これにより、ユーザーは音声による対話形式で AI ミケの応答を受け取ることができます。

2.2. シーケンス図（応答判定）

本誌に掲載した機能関連図（応答判定）に対応する、詳細なシーケンスを図 2.2 に示します。

図 2. 2

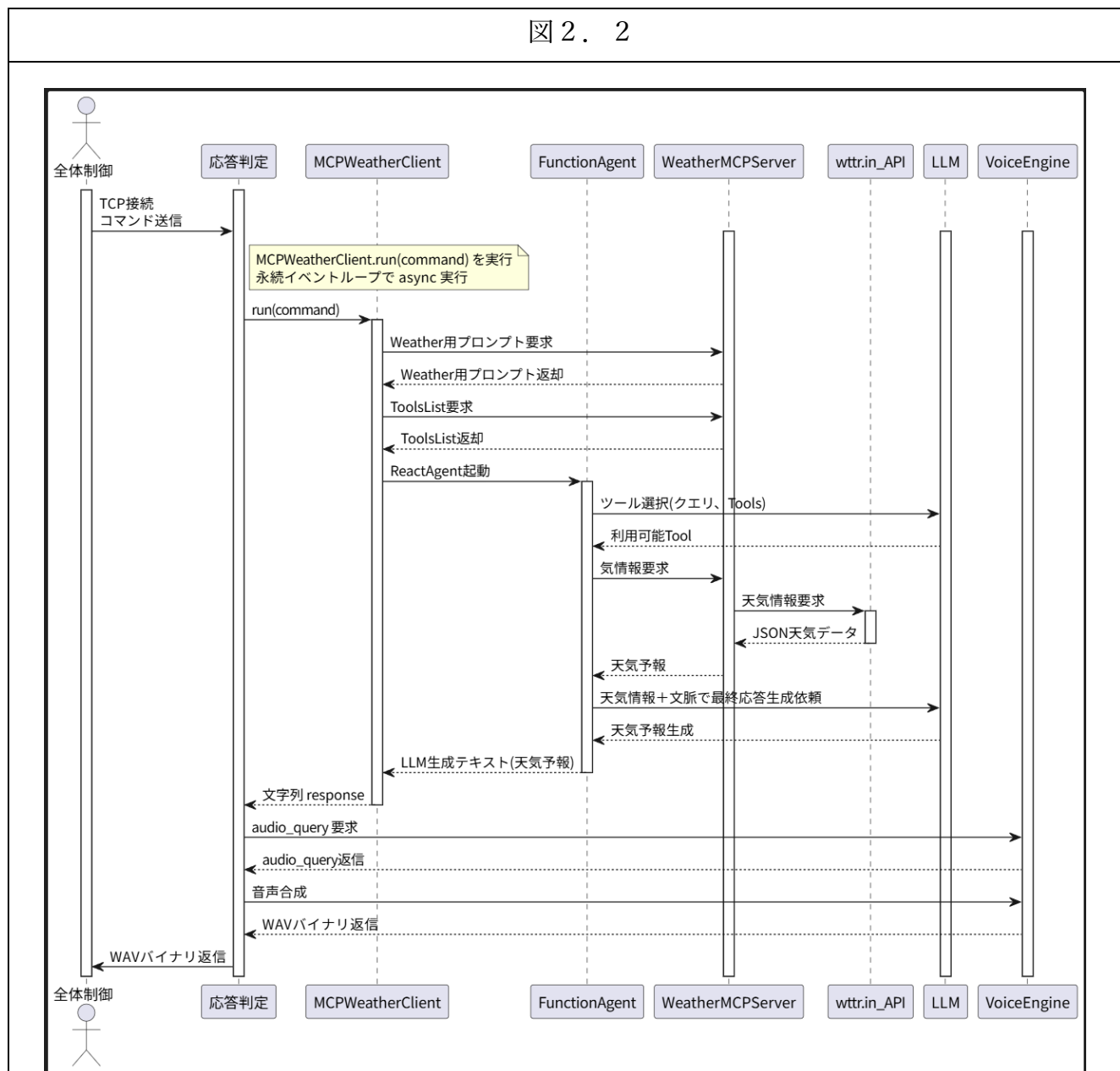


図 2.2 は、AI ミケにおいて 音声認識結果として得られたコマンド文字列が、MCP と LLM を用いて最終的な音声応答へ変換されるまでの内部処理フローを示したものです。

特に、MCP クライアントと MCP サーバ、FunctionAgent、LLM の役割分担と連携関係に着目して整理しています。

2.2.1. 処理の流れ

音声認識によって得られた コマンド文字列は、全体制御モジュールから応答判定処理へ渡されます。応答判定では、MCP クライアントである MCPWeatherClient が起動され、非同期イベントループ上で処理が進行します。

MCPWeatherClient は、WeatherMCPServer に対して

- 天気用プロンプト
- 利用可能なツール一覧

を要求し、これらの情報を取得します。

取得した情報をもとに、**FunctionAgent (ReactAgent)** が起動されます。

FunctionAgent は、問い合わせ内容から天気情報が必要であると判断すると、適切なツールを選択し、外部の **wtrr.in API** に対して天気情報の取得要求を行います。

外部 API から取得した JSON 形式の天気データは、FunctionAgent により整理され、後続処理に引き渡されます。

整理された天気情報と元の問い合わせ文は **LLM** に渡され、文脈情報として統合されたうえで、**自然な文章形式の最終応答文**が生成されます。生成された応答テキストは、音声合成エンジンに送信され、**WAV 形式の音声データ**として変換されます。

最終的に、生成された音声データは全体制御モジュールへ返却され、スピーカー出力処理へと引き渡されます。

2.3. ハードウェア・ソフトウェア構成

本節では、本稿で紹介する AI ミケを構成するハードウェアおよびソフトウェア環境についてまとめます。

Docker 版では、AI ミケを Raspberry Pi 上で動作させ、PC 側に LLM による応答生成および音声合成処理を担当させる構成としています。これにより、エッジ側の負荷を抑えつつ、高性能な計算資源を必要とする処理を PC 側へオフロードしています。

一方、**Windows 版 AI ミケ**では、音声入力から応答生成、音声合成までの **すべての処理**を **Windows PC** 上で完結させる構成としています。

表 2.1 Docker 版 AI ミケ (Raspberry Pi)

No	機器名	Version/商品名 (参考)・備考
1	Raspberry Pi 4B	本体 (Zero 2W でも動作は可能ですが、スピーカーフォンのように電力を必要とするデバイスを動作させる事ができない場合がありますので、4B を推奨します)
2	Raspberry Pi OS (64bit) Lite	Release date: October 1 2025 System: 64-bit Kernel version: 6.12
3	USB メモリー	USB メモリ 64GB (SanDisk)
4	AC アダプタ	Rasp berry Pi 用 AC アダプター
5	大容量モバイルバッテリー	モバイルバッテリー (5V/3A 以上)
6	スピーカーフォン WEB 会議用	PR-SK95CK マイク・スピーカー (比較的マイク感度が安定していました)
6'	USB オーディオアダプター	ENVEL ヘッドセットアダプター 3.5mm Zero 2W では、スピーカーフォン を安定して動作させる事ができない場合があります。その場合こちらを使用してください
7	USB-HUB & LAN (micro-B 対応)	有線 LAN 搭載 3 ポート OTG ハブ (RUH-OTGU3HE)
8	Keyboard	USB 接続 インストール時の言語設定に注意してください
9	モニター	HDMI 接続

表 2.2 Windows 版 AI ミケ / LLM & 音声合成用 PC

No	機器名	Version
1	OS	Windows11 Pro
2	CPU	AMD Ryzen 5 3600
3	メモリー	DDR4 266 32GB
4	GPU	RTX-3060Ti 12GB
5	ストレージ	M2 SSD 2T NVMe 2TB

表 2.4 使用するソフト/ライブラリ

No	名称	Version
1	DockerDesktop(Windows)	4.55.0
2	Docker	28.5.2
3	Ollama	0.13.5
4	LLM : gpt-oss:20b	---
5	ffmpeg	7.1.3-0+deb13u1+rpt1
6	ffmpeg(Windows)	N-117495-ge347b4ff31-20241014
7	Python	3.13.5-1
8	python3-dev	3.13.5-1
9	libasound2-dev	1.2.14-1+rpt1
10	portaudio19-dev	19.6.0-1.2+b3
11	alsa-utils	1.2.14-1+rpt1
12	flac	1.5.0+ds-2
13	Raspberry Pi Imager	https://www.raspberrypi.com/software/

※ 音声入出力および音声認識関連ライブラリは、Julius・PyAudio・音声合成処理で使
す

表 2.5 使用するモジュール

No	名称	Version
1	SpeechRecognition	3.14.4
2	fastmcp	2.14.1
3	numpy	2.3.5
4	llama-index-core	0.14.10
5	llama-index-llms-ollama	0.9.0
6	pyaudio	0.2.14
7	ffmpeg-python	0.2.0
8	requests	2.32.5
9	scipy	1.16.3

2.4. プログラムファイル一覧

表 2.6 プログラムファイル一覧

No	ファイル名	役割	役割・機能概要
1	mike_mcp_client.py	MCP クライアント	AI ミケが外部 MCP サーバを利用するための MCP クライアント実装。FastMCP を用いて MCP サーバへ接続し、取得したツールを FunctionTool として LlamaIndex ReAct Agent に統合。天気 API などの外部ツール呼び出しを、LLM から自然言語で行う仕組みを提供します。
2	weather_mcp_server.py	MCP サーバ	天気予報機能を提供する MCP サーバ。wttr.in API による天気取得、日次キャッシュ、MCP 標準プロンプト (Weather) の提供、および get_weather / clear_cache / get_cache_stats 各 MCP ツールを実装します。AI ミケの外部知識源として動作します。
3	voicevox_handler.py	音声合成	VOICEVOX への音声合成リクエスト処理を担当。audio_query/synthesis API を利用した音声生成を行います。
4	julius_client.py	音声認識	Julius を用いた WAV 音声データからのテキスト変換処理を担当します。
5	julius_handler.py	音声認識	Julius による音声コマンド認識の制御を担当。録音処理と認識 API 呼び出しを統合します。
6	record_with_vad.py	音声認識	VAD (Voice Activity Detection) を用いて無音区間を除去した音声録音を行います。音声認識前処理として使用します。
7	speech_recognition_client.py	音声認識	SpeechRecognition (Google API) を利用した WAV → テキスト変換処理を担当します。
8	speech_recognition_handler.py	音声認識	SpeechRecognition を用いた自由会話向け音声認識処理を担当。録音処理と API 認識を統合します。
9	voice_recognition_interface.py	音声認識	音声認識エンジンの抽象インターフェース。Julius と SpeechRecognition を同一のインターフェースで扱うための基底クラスです。
10	audio_stream_manager.py	音声 入出力制御	PyAudio による音声入出力ストリームをシングルトンとして管理。排他制御、最適サンプリングレート選択、リサンプリング処理など、音声 I/O の中核を担います。
11	main.py	クライアント	アプリ版 AI ミケのエントリーポイント。設定読み込み、ローグ初期化、MikeClient の起動を担当します (GUI なし)。
12	mike_client.py	クライアント	AI ミケ本体クラス。モード制御 (Sleep / Command / Chat)、音声入出力、AI 処理、MCP 呼び出しを統括する中核モジュールです。
13	mike_server.py	サーバ	メインサーバプログラム。クライアント接続処理、MCP・LLM を用いた応答生成、音声合成処理を管理します。
14	file_handler.py	ツール	WAV ファイルの保存・変換・送信を行うユーティリティ。ffmpeg を用いた音声フォーマット変換処理を提供します。
15	logger_handler.py	ツール	ロギング機能を提供。ログレベル設定、フォーマット管理、ストリームハンドラーの初期化を行います。
16	mike_tools.py	ツール	コマンドと WAV ファイルの対応表 (CSV) の読み出し、WAV 読み込み、URL 解析、エラー整形など、ミケ内部ロジックを補助するユーティリティ群です。
17	weather_cache.py	ツール	天気情報キャッシュ管理クラス。日次キャッシュ管理、ファイル永続化、データクリーニング、統計情報取得などを行い、weather_mcp_server を補助します。

2.5. 主要コンポーネント詳細

本節では、AI ミケを構成する主要コンポーネントについて、それぞれの役割と機能を詳しく説明します。各コンポーネントは責務を明確に分離しています。

No	コンポーネント名	区分	役割概要	主な機能
1	MikeMcpClient	MCP	MCP ReAct エージェント	Ollama LLM 接続、FastMCP クライアント初期化、ツール一覧取得・実行、システムプロンプト取得、FunctionAgent 推論ループ
2	WeatherMCPServer	MCP	天気情報提供用 MCP サーバ	Streamable HTTP 通信、get_weather ツール提供、Weather プロンプト提供、wttr.in API 連携、日次キャッシュ管理
3	MikeClient	クライアント	音声対話システム全体を統括する中核クラス	モード管理(待機／コマンド／会話)、音声認識エンジン切替、サーバ通信、音声 I/O 制御、会話状態管理、スレッドセーフ制御、タイムアウト管理
4	MikeServer	サーバ	サーバサイド処理の統括	TCP サーバ起動(0.0.0.0:5000)、クライアント接続管理、マルチスレッド処理、MCP クライアント管理
5	AudioStreamManager	音声 I/O	オーディオデバイスの統一管理	PyAudio 初期化、マイク入力管理、排他制御、タイムアウト付きロック、音声データ取得、エラーハンドリング、複数デバイス対応
6	VoiceVoxHandler	音声合成	VOICEVOX 音声合成連携	HTTP API 通信、audio_query 生成、synthesis 実行、複数話者対応、FileHandler 連携による音声変換
7	JuliusHandler	音声認識	短いコマンド・ウェイクワード認識	Julius プロセス管理、WAV 入力、認識結果パース、cmscore1 による品質判定、自動再試行、文字列整形、設定ファイル利用
8	SpeechRecognitionHandler	音声認識	自由会話向け音声認識	Google Cloud Speech API 連携、日本語対応、長文音声認識、エラーハンドリング、設定ファイル連携
9	RecordWithVAD	音声録音	VAD による発話区間のみの録音	RMS 音量検出、閾値判定、音声区間検出、無音自動停止、PCM 蓄積、WAV 保存、品質判定、AudioStreamManager 連携

2.6. 設定・データファイル一覧

本節では、AI ミケで使用する設定ファイルおよびデータファイルについてまとめます。
これらのファイルは、動作モードの切り替えや外部サービス連携、音声認識・音声合成の挙動を制御する重要な要素です。

表 2.8 設定・データファイル一覧

ファイル名	種別	機能概要
communication.csv	CSV	コマンドおよび通信データ定義ファイル。ID、名前、応答テキスト、スピーカー ID、ファイル名、読み(音素列)を列として管理し、応答用音声ファイル生成および固定応答処理に使用します。
config.json	JSON	システム全体の設定ファイル。VOICEVOX、MIKE_SERVER、MIKE_CLIENT、OLLAMA の接続先や各種パラメータ、ログレベルなど、主要モジュールの動作設定を定義します。
mcp_servers.json	JSON	STDIO 構成で使用する MCP サーバ定義ファイル。ローカル実行する MCP サーバの起動設定やコマンドを記述します。
mike_mcp_servers.json	JSON	Streamable HTTP 構成で使用する MCP サーバ設定ファイル。天気 MCP サーバのアドレスやポート番号など、外部 MCP サーバ接続情報を定義します。
mike.dic	辞書(dic)	Julius 用音声認識辞書。認識対象となる単語および音素列を定義します。
mike.jconf	設定 (Config)	Julius 音声認識エンジン用設定ファイル。音声モデル、辞書、認識パラメータなどを定義します。

2.7. ツール関連

2.7.1. プログラムファイル一覧

AI ミケの運用・構築・停止を補助するために用意しているツールスクリプトの一覧を表 2.9 に示します。これらのスクリプトは主に Raspberry Pi 側での運用管理や Docker 環境の再構築を目的としています。

表 2.9 ツールスクリプト一覧

スクリプト名	目的・役割	主な処理内容
chmod.sh	実行権限の付与	各 スクリプトに対して実行権限を付与します。
build_mike.sh	Docker 環境の完全再構築	コンテナの新規イメージのビルド、Docker ネットワークの作成を行います。
julius_make.sh	Julius のビルドおよび動作確認	Julius の再ビルドまたは起動を実行します。マイク入力による簡易動作テストを含みます。
make_data.sh	MIKE 用データ生成	既存データの削除後、mike_data.sh を実行し、音声ファイル生成・コピーおよび生成日時の確認を行います。
mike_start.sh	MIKE の起動	composition.json を切り替えたうえで既存サービスを停止し、サーバ → クライアントの順で MIKE を起動しログも表示します。
mike_stop.sh	Raspberry Pi 側 MIKE の停止	Docker 上で動作している全コンテナを停止し、PC 側で動作している MIKE の停止を促します。

3. 環境構築（サーバ側環境設定）

AI ミケをインストールする前に、応答生成用 PC の音声合成環境および LLM 実行環境を準備します。

3.1. VoiceVox のインストール

VOICEVOX 公式サイト (<https://voicevox.hiroshiba.jp/>) にアクセスし、利用規約を確認したうえでインストーラーをダウンロードしてください。

インストールは、ダウンロードしたインストーラーの指示に従って進めます。
特別な設定は不要で、標準設定のままで問題ありません。

3.1.1. VOICEVOX エンジンの起動（Windows の例）

インストール完了後、コマンドプロンプトを開き、以下のコマンドで VOICEVOX エンジンを実行します。

※ 実際のパスは、VOICEVOX のインストール先に合わせて変更してください。

```
#=====起動コマンド=====  
C:\Program Files\VOICEVOX\vv-engine>run.exe --host 0.0.0.0
```

※：セキュリティ上の注意

--host 0.0.0.0 を指定すると、外部からの接続を無制限に受け付ける状態になります。
本設定は、家庭内ネットワークなどの限定された環境でのみ使用してください。

● 起動ログの確認

以下のようなログが表示されれば、VOICEVOX エンジンは正常に起動しています。

```
INFO: Started server process [14672]  
INFO: Waiting for application startup.  
INFO: Application startup complete.  
INFO: Uvicorn running on http://0.0.0.0:50021 (Press Ctrl+C to quit)
```

※VOICEVOX は、デフォルトで TCP の 50021 ポートを使用します。

3.1.2. Windows Defender ファイアウォールの設定

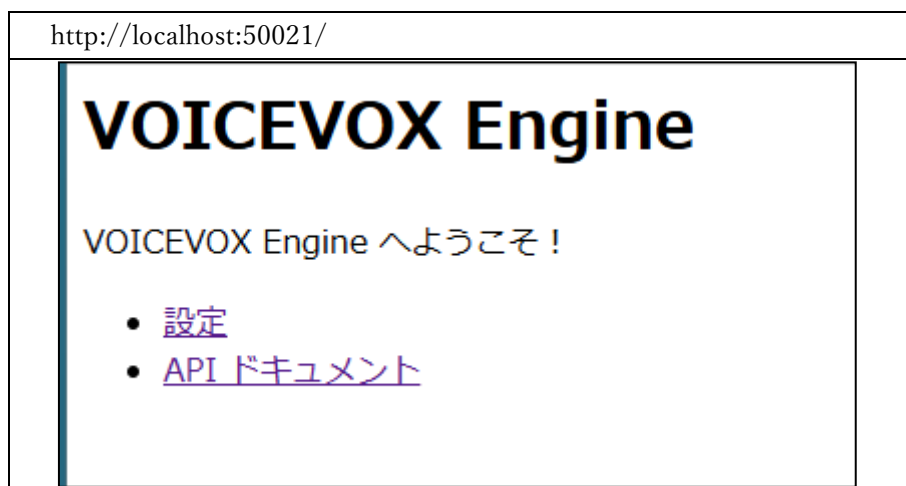
初回起動時に、Windows Defender ファイアウォールから通信許可を求められる場合があります。

TCP ポート 50021 の通信は必須となるため、必ず許可してください。

3.1.3. エンジン起動の確認（ローカル）

PC のブラウザを起動し、以下の URL にアクセスします。

VOICEVOX のトップページ (index.html) が表示されれば、ローカル環境での起動は正常です。



3.1.4. クライアント（Raspberry Pi）からの接続確認

次に、Raspberry Pi 側から VOICEVOX を実行している PC へネットワーク越しに接続できるかを確認します。

Raspberry Pi のターミナルで、次のコマンドを実行します。

```
$ wget http://192.168.0.10:50021/
```

※ : 192.168.0.10 は VOICEVOX がインストールされた PC の IP アドレスに置き換えてください。

index.html が取得できれば、正常に接続できています。

3.1.5. 注意点

1. ネットワーク接続

Raspberry Pi と VOICEVOX を実行する PC は、必ず同一ネットワーク内に配置してください。

2. ファイアウォール設定

Windows Defender 以外のセキュリティソフトを使用している場合、TCP ポート 50021 の通信許可を別途設定する必要があります。

3. IP アドレス管理

PC の IP アドレスは、ipconfig コマンドで確認できます。

DHCP 環境では、再起動時に IP アドレスが変更されることがあるため、固定 IP を設定しておくことで安定した運用が可能です。

3.2. Ollama のインストール

Ollama の公式サイトからインストーラーをダウンロードし、画面の指示に従ってインストールします。

Ollama 本体は GitHub 上で MIT ライセンスとして公開されています。

- Ollama ライセンス

<https://github.com/ollama/ollama/blob/main/LICENSE>

ただし、Ollama で使用する LLM (Llama、Code Llama など) には、モデルごとに異なるライセンスが適用される場合があります。

実運用や再配布を行う場合は、必ず各モデルのライセンス条件を個別に確認してください。

3.2.1. インストール時の注意点

1. Windows 版 Ollama のインストール形態

Windows 版 Ollama は、ユーザーごとの領域にインストールされます。

そのため、サーバ用途で利用する場合であっても、

Ollama を実行するユーザーでログインした状態で使用する必要があります。

サービスとして常駐させる構成ではない点に注意してください。

2. CUDA 環境について

Ollama のインストーラーには **CUDA ランタイム**が同梱されています。

GPU ドライバーが正しく導入されていれば、別途 **CUDA Toolkit** をインストールする必要はありません。

3.2.2. インストール後の設定

インストールが完了すると、Ollama は自動的にバックグラウンドで起動し、タスクバーに Ollama のアイコンが表示されます。

この状態で、CLI や WebUI から LLM モデルのロードおよび実行が可能になります。

3.2.3. 設定画面

設定画面は、タスクバーに表示された Ollama のアイコンを **右クリック**し、**[Settings]** を選択することで表示できます。

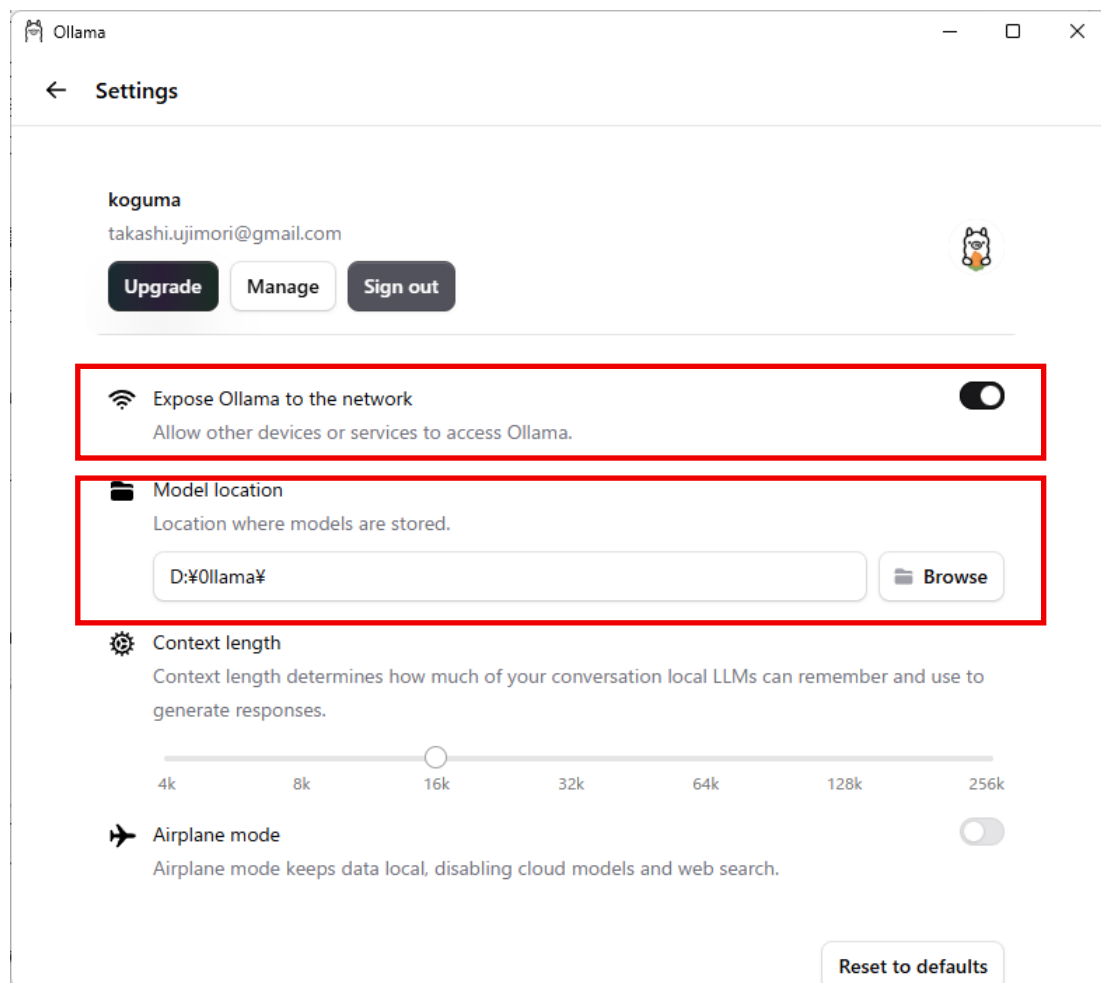
図タスクバーに表示された ollama のアイコン



3.2.4. 初期設定で実施する 2 つの項目

Docker 版 AI ミケから Ollama を利用するために、インストール直後に次の 2 項目を設定します。

図 Ollama 設定画面



1. Expose Ollama to the network

初期状態では、Ollama は localhost (127.0.0.1) からのアクセスのみを受け付ける設定になっています。

Docker 版 AI ミケでは、Raspberry Pi から Ollama に接続するため、ネットワーク経由での API アクセスを許可する必要があります。

この項目を有効にすることで、外部ホストから Ollama の API にアクセスできるようになります。

※：注意

本設定は、Ollama をネットワーク上に公開することを意味します。

家庭内 LAN など、信頼できる閉じたネットワーク内でのみ使用してください。

2. Model Location

Ollama が使用する LLM モデルファイルの保存先フォルダを指定します。

デフォルト設定では OS ドライブにモデルが保存されるため、大規模モデルを使用するとシステム領域を圧迫する可能性があります。

そのため、容量に余裕のある 別ドライブ (例：D: ドライブ) へ保存先を変更することを推奨します。

4. 環境構築（ラズベリーパイの設定）

本章では、Raspberry Pi 側に AI ミケの実行環境を構築する手順について説明します。
はじめに、Raspberry Pi OS を USB メモリに書き込み、Raspberry Pi を起動できる状態を準備します。

4.1. Raspberry Pi OS の書き込み

Raspberry Pi Imager を使用して、Raspberry Pi OS を USB メモリへ書き込みます。
本稿では、軽量でサーバ用途に適した Raspberry Pi OS (64-bit) Lite を使用します。

4.1.1. RaspberryPiImager の操作

Raspberry Pi Imager を起動し、OS イメージの書き込み設定を行います。
Imager では OS の書き込みに加えて、初回起動時に必要な各種設定を事前に行うことが可能です。

AI ミケのセットアップを円滑に進めるため、以下の項目は必ず設定しておくことを推奨します。

1. 事前に設定する項目（重要）

- SSH の有効化

AI ミケのセットアップ作業をリモートから行うため、必須の設定です。

- ユーザー名・パスワードの設定

Raspberry Pi OS の初期ユーザーを指定します。

以降の作業は、このユーザーでログインして進めます。

- ネットワーク設定 (Wi-Fi または有線 LAN)

ネットワーク接続が不安定な場合、初回セットアップ中に処理が失敗することがあります。可能であれば、有線 LAN の使用を推奨します。

-

2. 初回起動時の注意点

Imager で事前設定を行っていても、以下の理由により設定が正しく反映されない場合があります。

- Wi-Fi の SSID やパスワードの入力間違い
- キーボード配列の誤設定
(特に記号入力がずれるケース)
- ネットワーク機器や回線側のトラブル

そのため、初回起動時はキーボードとモニターを接続した状態で起動し、
raspi-config を使用して設定内容を確認することを強く推奨します。

Raspberry Pi Imager OS の設定

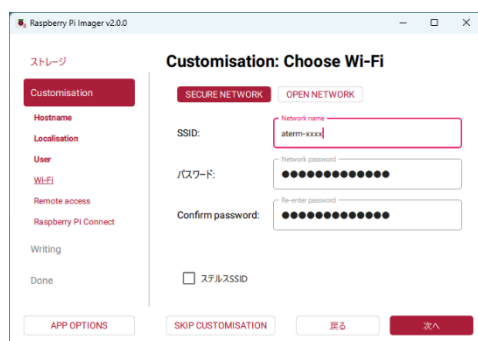
Raspberry Pi Imager における OS 書き込み時の設定画面を示します。

本画面では、OS イメージの選択に加え、SSH の有効化、ユーザー名・パスワードの設定、ネットワーク設定など、初回起動時に必要となる各種設定を事前に行うことができます。

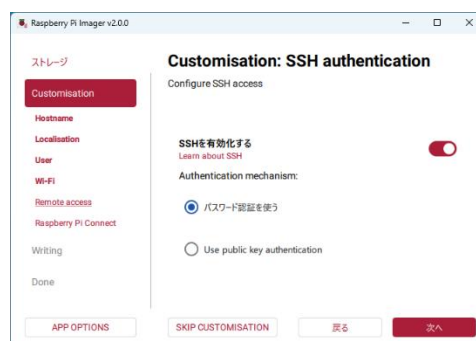
図 4.1 Raspberry Pi Imager OS の設定画面

<h3>デバイス選択</h3> 	<h3>OS 選択</h3>  <p>Raspberry Pi OS Lite (64-bit)を選択</p>
<h3>OS を書き込むメディアを選択</h3> 	<h3>ホスト名の入力</h3> 
<h3>国（首都）とキーボードレイアウト選択</h3> 	<h3>ユーザ名とパスワード入力</h3> 

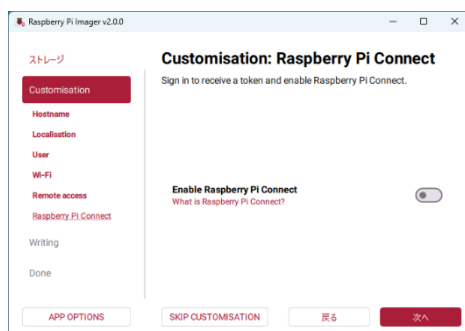
Wifi の設定



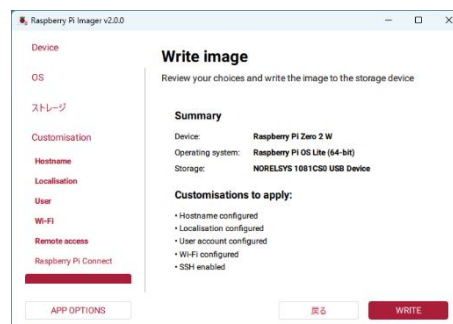
SSH の有効化



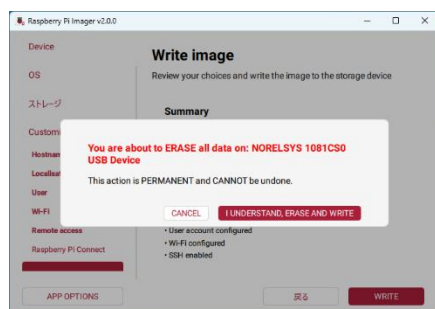
RaspberryPi Connect の有無



確認画面

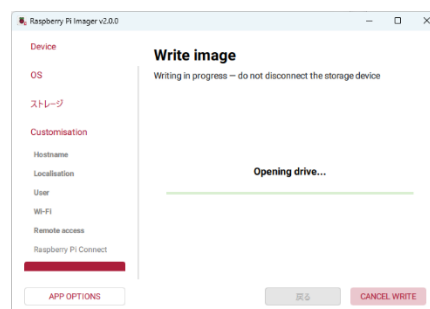


書き込み開始 (確認)

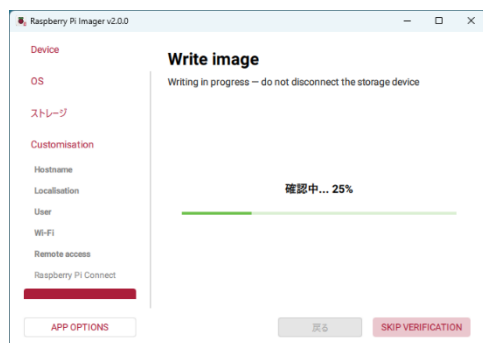


選択可能になるまで少し待ち時間が設定されています。

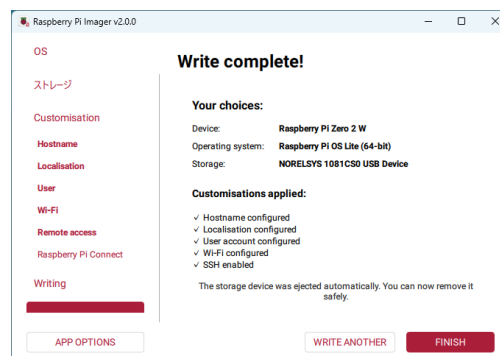
書き込み中



確認中



終了

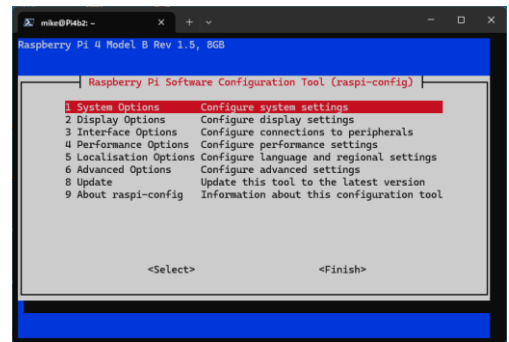


OS イメージを事前にダウンロードする場合の設定方法

Raspberry Pi Imager の オンライン機能を使用せず、事前にダウンロードした OS イメージファイルを用いて Raspberry Pi OS を書き込むことも可能です。ただし、この方法では Imager が提供する初期設定機能を利用できないため、

- ホスト名
- SSH の有効化
- ユーザー名・パスワード
- ネットワーク設定

図 4.2 raspi-config の設定画面



といった項目を 事前にカスタマイズすることができません。

そのため、OS の初回起動時に、以下の設定を 対話的に行う必要があります。

- キーボードレイアウト
- ユーザー名
- パスワード
- ネットワーク設定(必要な場合)

raspi-config による追加設定

OS のインストール完了後、必要な設定は raspi-config を用いて行います。

Raspberry Pi にログイン後、次のコマンドを実行してください。

```
• $ sudo raspi-config
```

図 4.2 は、Raspberry Pi OS に標準で用意されている設定ツール **raspi-config** の画面例を示しています。raspi-config を使用することで、キーボード設定やロケール、SSH の有効化、ネットワーク関連設定などを確認・変更できます。

Imager で事前設定を行っていない場合は、本画面を用いて初期設定内容の確認および修正を必ず行ってください。

参考情報

- ダウンロード URL :
https://downloads.raspberrypi.com/raspios_lite_arm64/images/raspios_lite_arm64-2025-11-24/2025-11-24-raspios-trixie-arm64-lite.img.xz
- OS のイメージファイル :

4.2. OS の設定

Raspberry Pi に、キーボード、モニター、ネットワーク（有線または Wi-Fi）、USB オーディオデバイス、および Raspberry Pi OS を書き込んだ USB メモリを接続した状態で電源を投入します。

初回起動時は、Raspberry Pi OS が内部で各種初期設定を自動的に実行するため、ログイン画面が表示されるまで数分程度かかる場合があります。

この間、画面が一時的に暗転したり、自動的に再起動が行われることがありますが、いずれも正常な動作ですので、そのまま待ってください。

4.2.1. OS の更新

初回起動後は、Raspberry Pi OS を 最新の状態に更新します。

これにより、パッケージ管理情報の更新や最新ドライバの適用が行われ、後続の AI ミケ環境構築時のトラブルを未然に防ぐことができます。

以下のコマンドを順に実行します。

```
# APT の 情報を更新
$ sudo apt update
# OS を情報を更新
$ sudo apt upgrade
# アップグレード完了後、再起動します。
$ sudo reboot
```

再起動後、通常どおりログインできれば OS の更新は完了です。

4.2.2. 状態確認

OS の更新後、Raspberry Pi が 正しいバージョンの Raspberry Pi OS（Debian Trixie ベース）で動作しているかを確認します。

以下のコマンドで OS のリリース情報を表示できます。

```
# リリース情報確認
$ cat /etc/os-release
```

```
#表示例：
PRETTY_NAME="Debian GNU/Linux 13 (trixie)"
NAME="Debian GNU/Linux"
VERSION_ID="13"
VERSION="13 (trixie)"
VERSION_CODENAME=trixie
DEBIAN_VERSION_FULL=13.2
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

VERSION="13 (trixie)" または VERSION_CODENAME=trixie が表示されていれば、Debian Trixie ベースの Raspberry Pi OS が正しく起動しています。

4.2.3. タイムゾーン・Locale(フォント)

Raspberry Pi を日本国内で利用する場合は、タイムゾーンを Asia/Tokyo に設定し、必要に応じて Locale (文字コード) 設定を確認します。

これらの設定が不適切な場合、ログやファイルのタイムスタンプのずれ、文字化けなどの原因になります。

# タイムゾーンの確認 \$ timedatectl	# Locale フォント \$ locale
<div>#表示例</div> <div>Local time: Fri 2025-11-28 08:22:24 JST</div> <div>Universal time: Thu 2025-11-27 23:22:24 UTC</div> <div>RTC time: n/a</div> <div>Time zone: Asia/Tokyo (JST, +0900)</div> <div>System clock synchronized: yes</div> <div> NTP service: active</div> <div> RTC in local TZ: no</div> <div># タイムゾーンのと設定</div> <div>\$ sudo timedatectl set-timezone Asia/Tokyo</div>	<div>#表示例</div> <div>LANG=en_GB.UTF-8</div> <div>LANGUAGE=</div> <div>LC_CTYPE="en_GB.UTF-8"</div> <div>LC_NUMERIC="en_GB.UTF-8"</div> <div>LC_TIME="en_GB.UTF-8"</div> <div>LC_COLLATE="en_GB.UTF-8"</div> <div>LC_MONETARY="en_GB.UTF-8"</div> <div>LC_MESSAGES="en_GB.UTF-8"</div> <div>LC_PAPER="en_GB.UTF-8"</div> <div>LC_NAME="en_GB.UTF-8"</div> <div>LC_ADDRESS="en_GB.UTF-8"</div> <div>LC_TELEPHONE="en_GB.UTF-8"</div> <div>LC_MEASUREMENT="en_GB.UTF-8"</div> <div>LC_IDENTIFICATION="en_GB.UTF-8"</div> <div>LC_ALL=</div>

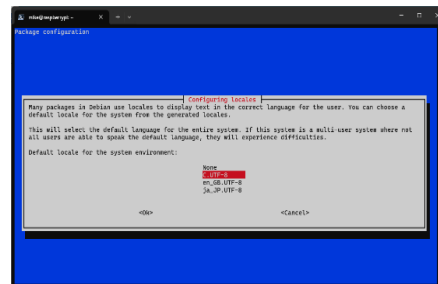
■ Locale 設定に関する注意点 (重要)

日本語環境にする場合は、次のコマンドで Locale を再設定できます。

\$ sudo dpkg-reconfigure locales

ja_JP.UTF-8 を選択することで日本語の環境になりますが、注意点があります。

- HDMI 接続時の 黒いコンソール画面 (CLI) では、日本語表示は基本的にできません
- 日本語表示を行うには、日本語フォント (Noto 系など) の追加インストールが必要です
- 設定ファイルやフォルダ名が日本語になる場合があります、スクリプト処理で扱いにくくなる可能性があります
- 一部の CLI ツールで 文字化けが発生する場合があります



Raspberry Pi の HDMI 接続コンソールは、日本語フォントに対応していないため、ソースコードやログに日本語が含まれていても、正しく表示されません。

推奨作業環境

日本語を含むログや表示を扱う場合は、以下のいずれかの方法を推奨します。

- SSH 接続した端末 (Windows / macOS / Linux) から作業する
- GUI 版 Raspberry Pi OS のターミナル (LXTerminal など) を利用する

本稿の AI ミケ環境構築では、SSH 経由での作業を推奨します。

SSH クライアント側のターミナルは UTF-8 表示に対応しており、日本語ログや文字列も正しく確認できるためです。

4.2.4. Audio 設定

Raspberry Pi には複数のオーディオデバイスが組み込まれており、初期状態でも音声の入出力は動作します。しかし環境によっては、**USB オーディオデバイスでの録音・再生が不安定になる**場合があります。

特に、標準状態では以下のデバイスが優先的に認識されます。

- bcm2835 (オンボードオーディオ)
- vc4-hdmi (HDMI オーディオ)

そのため、USB オーディオデバイスを利用する構成では、**意図しないデバイスが選択されてしまう**ケースがあります。

本稿で扱う AI ミケでは、USB オーディオデバイスを確実に利用するため、使用するオーディオデバイスを固定・制限する設定を推奨します。

再生デバイスの確認

USB オーディオデバイスを接続した状態で、現在認識されている再生デバイスを確認します。

```
# Audio デバイスリスト
```

```
$ aplay -l
```

```
**** List of PLAYBACK Hardware Devices ****
```

```
card 0: Headphones [bcm2835 Headphones], device 0: bcm2835 Headphones [bcm2835 Headphones]
```

```
Subdevices: 8/8
```

```
Subdevice #0: subdevice #0
```

```

Subdevice #1: subdevice #1
Subdevice #2: subdevice #2
Subdevice #3: subdevice #3
Subdevice #4: subdevice #4
Subdevice #5: subdevice #5
Subdevice #6: subdevice #6
Subdevice #7: subdevice #7
card 1: TSK95K [TSK95K], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: vc4hdmi0 [vc4-hdmi-0], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 3: vc4hdmi1 [vc4-hdmi-1], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0

```

この例では、以下のデバイスが認識されています。

- **card 0** : オンボードオーディオ (bcm2835)
- **card 1** : USB オーディオ (TSK95K)
- **card 2 / 3** : HDMI オーディオ

不要なオンボードオーディオデバイスを無効化

USB オーディオデバイスを安定して利用するため、オンボードオーディオおよび HDMI オーディオを無効化します。

設定は `/boot/firmware/config.txt` を編集して行います。

以下のように `snd_bcm2835` と `vc4` 関連の設定をコメントアウトします。

```

$ sudo vi /boot/firmware/config.txt

# Enable audio (loads snd_bcm2835)
#dtparam=audio=on          <— snd_bcm をコメントアウト

# Enable DRM VC4 V3D driver
#dtoverlay=vc4-kms-v3d     <— vc4 (をコメントアウト
#max_framebuffers=2       <— snd_bcm をコメントアウト

```

設定後、Raspberry Pi を再起動し、利用できるサウンドデバイスが少なくなっていることが確認します。

```

$ aplay -l

**** List of PLAYBACK Hardware Devices ****
card 0: TSK95K [TSK95K], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0

```

USB オーディオデバイスのみが表示されていれば、設定は完了です。

4.2.5. USB オーディオの音量調整

USB オーディオデバイスを接続した直後は、再生音量や録音レベルが 0（ミュートに近い状態）になっている場合があります。

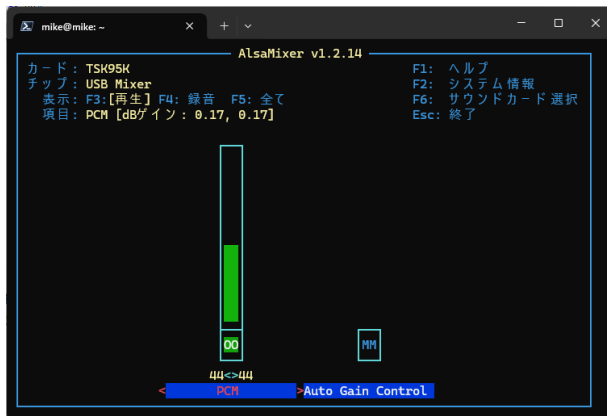
そのため、AI ミケを起動する前に音量設定を確認しておく必要があります。

alsamixer コマンドによる音量設定

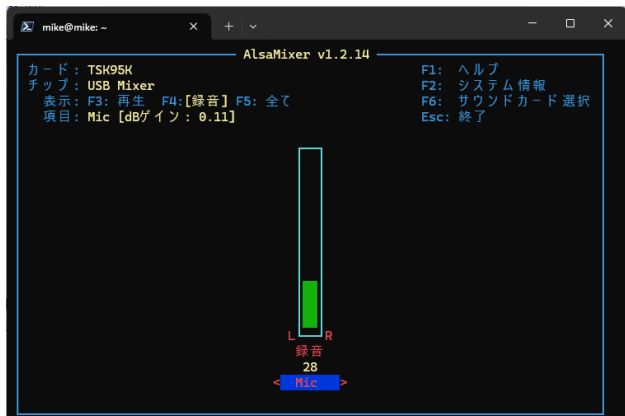
```
$ alsamixer
```

- 方向キー（↑↓）で音量調整
- M キーでミュート ON/OFF
- F6 キーで操作対象のデバイスを切り替え

再生音量



録音レベル



amixer による音量設定（CUI 完結）

CUI のみで設定したい場合は、amixer コマンドを使用できます。

```
$ amixer
```

```
Simple mixer control 'PCM',0
  Capabilities: pvolume pswitch pswitch-joined
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 100
  Mono:
    Front Left: Playback 44 [44%] [0.17dB] [on]
    Front Right: Playback 44 [44%] [0.17dB] [on]
Simple mixer control 'Mic',0
  Capabilities: cvolume cvolume-joined cswitch cswitch-joined
  Capture channels: Mono
  Limits: Capture 0 - 100
  Mono: Capture 30 [30%] [0.11dB] [on]
Simple mixer control 'Auto Gain Control',0
  Capabilities: pswitch pswitch-joined
  Playback channels: Mono
  Mono: Playback [off]
```

#注: デバイス名は大文字小文字が識別されます。

#再生音量 設定例

```
$ amixer sset PCM 30%
```

#録音音量 設定例

```
$ amixer sset Mic 30%
```


4.3. Docker サービスのインストール (Docker 版のみ)

本節では、AI ミケを Docker 版構成で実行するために必要な Docker 環境を、Raspberry Pi に導入する手順を説明します。

Raspberry Pi への Docker インストールは比較的シンプルで、いくつかのコマンドを実行するだけでセットアップが完了します。

ただし、既存パッケージや設定との不整合を避けるため、可能な限り初期状態に近い Raspberry Pi OS 環境に Docker を導入することを推奨します。

1. Docker のインストール：(参考：https://docs.docker.jp/linux/step_one.html)

以下のコマンドを実行すると、**Docker Engine** と関連ツールがまとめてインストールされます。

```
$ sudo curl -fsSL https://get.docker.com/ | sh
# Executing docker install script, commit: 7d96bd3c5235ab2121bcb855dd7b3f3f37128ed4
:
=====
To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:
    dockerd-rootless-setuptool.sh install
Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.
To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/
=====
```

このログが表示されれば、Docker のインストールは正常に完了しています。

※ 本稿では、rootless モードは使用せず、後述する docker グループによる運用を前提とします。

2. Docker グループにユーザーを追加

インストール直後は、**sudo** を付けないと Docker コマンドが実行できません。利便性のために、現在のユーザーを **docker** グループへ追加します。

```
$ sudo usermod -aG docker ${USER}
```

3. 再ログインと動作確認

グループ設定を反映するため、一度**ログアウトして再ログイン**します。

再ログイン後、以下のコマンドで Docker の動作確認を行います。

以下のようなメッセージが表示されれば、Docker は正しく動作しています。

```
$ docker --version
$ docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
198f93fd5094: Pull complete
95ce02e4a4f1: Download complete
```

```
Digest: sha256:f7931603f70e13dbd844253370742c4fc4202d290c80442b2e68706d8f33ce26
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(arm64v8)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

hello-world コンテナが正常終了すれば、**Docker** のインストールは成功です。

4. Docker network の作成

AI ミケでは、システム内部で **Docker** コンテナ間通信を行うため、専用の Docker ネットワークを使用します。

ここでは `mike-net` という名前でブリッジネットワークを作成します。

```
docker network create mike-net -d bridge
```

このネットワークにより、ミケの各コンテナ（API サーバ、音声処理、応答生成など）が安定して相互通信できるようになります。

5. 環境構築（Windows の設定）

本章では、Windows 環境で AI ミケ（アプリ版）を動作させるための事前準備について説明します。

5.1. インストール準備

AI ミケを Windows 環境で動作させるには、音声データの変換処理に使用する `ffmpeg.exe` が必要です。

`ffmpeg` がインストールされていない場合は、AI ミケのインストールを行う前に、以下の手順で `ffmpeg` をインストールしてください。

管理者権限でのコマンド実行

まず、管理者権限で PowerShell を起動します。

1. `winget`（Windows Package Manager）の確認・有効化

`winget` が利用できない環境では、以下のコマンドを実行して Windows Package Manager を有効化します。

```
# winget（Windows Package Manager）インストール
PS> Add-AppxPackage -RegisterByFamilyName -MainPackage
Microsoft.DesktopAppInstaller_8wekyb3d8bbwe
```

※ すでに `winget` が使用可能な場合、この操作は不要です

2. `ffmpeg` のインストール

次に、`winget` を使用して `ffmpeg` をインストールします。

```
PS>winget install --id=Gyan.FFmpeg -e
```

インストールが完了すると、`ffmpeg.exe` が PATH に登録されます。

PowerShell 新たに起動しすると コマンドラインから直接利用できるようになります。

3. インストール確認

以下のコマンドを実行し、バージョン情報が表示されれば `ffmpeg` のインストールは正常に完了しています。

```
PS> ffmpeg -version
```

6. 各種設定ファイル説明

本章では、AI ミケのクライアントおよびサーバ動作を制御する設定ファイルについて説明します。中心となる設定ファイルは `config.json` であり、音声認識、音声合成、MCP、LLM など各コンポーネントの接続先や挙動を定義しています。

6.1. config.json

6.1.1. WEATHER_MCP (天気情報サービス)

天気情報を提供する MCP サーバの設定です。

設定項目	値	説明
SERVER_NAME	weather-mcp	サーバー名または、IP アドレス
SERVER_HOST	0.0.0.0	バインドするホストアドレス (全インターフェース)
SERVER_PORT	8000	天気 API のリッスンポート
TRANSPORT	streamable-http	MCP (Model Context Protocol) の通信方式
WEATHER_API_BASE_URL	https://wttr.in	天気情報取得元の外部サービス
API_REQUEST_TIMEOUT	10	API リクエストのタイムアウト(秒)
CACHE_FILE_PATH	cache/weather_cache.json	天気情報のキャッシュファイルの保存先

6.1.2. VOICEVOX (音声合成)

VOICEVOX エンジンとの接続設定です。

設定項目	値(例)	説明
Host	localhost	VOICEVOX サーバーの IP アドレス
Port	50021	VOICEVOX サーバーのポート番号
Speaker	45	音声合成に使用する話者 ID 例: Speaker: 櫻歌ミコ, ロリ id: 45

6.1.3. MIKE_SERVER (応答判定)

応答判定を行うサーバの設定です。

設定項目	値	説明
Host	mike-server	サーバーホスト名(アドレス)
Port	5000	サーバーポート番号
S_Address	0.0.0.0	バインドするアドレス

6.1.4. MIKE_CLIENT (音声認識・入力)

クライアント側の音声認識および対話制御に関する設定です。

設定項目	値	説明
RECOGNITION	0	音声認識エンジン選択
DeviceIndex	0	マイクデバイスのインデックス
Commands_File	config/communication.csv	コマンド定義ファイル
WAKE_WORD	みけおはよう	呼び出しワード
SLEEP_WORD	おやすみ	スリープ以降ワード
CHAT_START_WORD	お話し	会話モード開始ワード
CHAT_END_WORD	ありがとう	会話終了ワード
END_WORD	またね	ミケ終了ワード

6.1.5. MIKE_DATA (データ作成)

音声データを作成する設定です。

設定項目	値	説明
Communication_File	input/communication.csv	元となる音声コマンド一覧 CSV
OutPut_Voice_Folder	voice	出力音声ファイルの保存フォルダ

6.1.6. OLLAMA (LLM 設定)

LLM 実行環境 (Ollama) との接続および初期プロンプト設定です。

設定項目	値	説明
Host	http://localhost:11434	Ollama サーバーのアドレスとポート番号
Model	gpt-oss:20b	使用する AI モデル名
Content	You are a wonderful butler. Please reply in 100 characters or less.	キャラクター設定 (初期プロンプト)

6.1.7. その他の設定

ログ出力に関する設定です。

設定項目	値	説明
LOG_LEVEL	INFO	ログ出力レベル

6.2. mcp_servers.json (アプリ用)

6.2.1. ファイル概要

mcp_servers.json は、Model Context Protocol (MCP) サーバの接続設定を定義するファイルです。AI ミケ (アプリ版) では、MCP を通じて AI モデルが外部ツールやサービス (天気情報など) と連携するために使用します。

アプリ版では、MCP サーバをローカルプロセスとして起動するため、通信方式として STDIO を使用します。

6.2.2. mcpServers セクション

利用する MCP サーバの定義および接続情報をまとめて記述します。

6.2.3. mcp_servers (天気情報 MCP サーバー)

設定項目	値	説明
transport	stdio	通信プロトコル (標準入出力)
url	_____	STDIO 通信のため不要
command	python	MCP サーバ起動に使用するコマンド
args	["weather_mcp_server.py"]	実行する MCP サーバスクリプトのパス

6.3. mcp_servers.json (Docker 用)

6.3.1. ファイル概要

Docker 用 mcp_servers.json は、Docker 版 AI ミケで使用する MCP サーバ接続設定を定義するファイルです。Docker 版では、MCP サーバは別プロセスとして起動されネットワーク経由でアクセスします。

6.3.2. mcpServers セクション

このセクションでは、Docker 環境で利用する MCP サーバの接続方式およびエンドポイント情報を定義します。

6.3.3. mcp_servers (天気情報 MCP サーバー)

設定項目	値	説明
transport	streamable-http	通信プロトコル (HTTP ベースの MCP)
url	http://mike-server:8000/mcp	MCP サーバーのエンドポイント URL
command	_____	外部サーバのため不要
args	_____	外部サーバのため不要

6.4. communication.csv

6.4.1. ファイル概要

communication.csv は、ミケの音声コマンドと応答音声に対応付ける辞書ファイルです。

ミケが音声認識によって取得したキーワードに対し、どの応答音声（WAV ファイル）を再生するかを定義します。

本ファイルで定義される応答は、事前に生成された静的な音声データとして扱われ、MCP や LLM を用いた動的応答とは独立して処理されます。

そのため、

- 定型あいさつ
- システム制御用コマンド
- 即時応答が必要なフレーズ

などを 高速かつ安定して処理する目的で使します。

6.4.2. コマンド一覧表

番号	音声コマンド	音声応答	スピーカ	ファイル名	問い合わせよみ
err	えらー	わからないにゃ～！	45	error	はて
end	またね	ばいばいだにゃ～！	45	end-nata	またね
wake	みけおはよう	なにかにゃ～	45	wake-mike	みけおはよう
chat	お話ししよう”	よろこんでにゃ～	45	chat-ohanashi	お話し”
cend	ありがとう	いつでもいいにゃ～！	45	cend-ariga	ありがとう
tenki	天気予報	それじゃーてんきよほうをしらべてみるにゃ～！ちょっとまってね	45	tenki-tenki	てんきよほう
1	おはよう	おはようにゃ～	45	1-ohayou	おはよう
2	こんにちは	こんにちはだにゃ～！	45	2-konniti	こんにちはわ
3	こんばんは	こんばんはだにゃ～！	45	3-konban	こんばんわ
4	お名前は	名前はミケだにゃ～！	45	4-onamaewa	おなまえわ
5	なにができるの	たのしいおはなしとおてんきのはなしだにゃ～！	45	5-naniga	なにができる

※ 表中の文言は例です。実際の応答内容は自由に追加変更できます。

6.4.3. 各列の説明

列名	説明
番号	コマンドの識別 ID。err や end は特殊コマンド、数値(1～)は通常の音声応答用 ID です
音声コマンド	ユーザーが発話するキーワード。音声認識結果と照合されます
音声応答	システムが返す音声応答テキスト(事前に音声合成されます)
スピーカ	VOICEVOX で使用する話者 ID(config.json の設定に対応)
ファイル名	生成される WAV ファイルの識別子。事前の音声データ生成時に使用されます
問い合わせよみ	音声認識補助用の読み表記。認識精度向上のために使します

6.5. Julius 音声辞書

6.5.1. ファイル概要

本ファイルは、Julius で認識対象とする単語およびその発音（音素列）を定義する音声辞書です。Julius はこの辞書を参照して音声認識を行い、認識結果として「単語表記」を返します。

AI ミケでは、

- 認識結果の文字列をキーとして処理分岐を行う
- communication.csv や制御ロジックと連携する

といった用途で、この辞書を使用します。

6.5.2. 音声辞書一覧表

単語表記	音素列
えらー	h a t e
またね	m a t a n e
みけおはよう	m i k e o h a y o u
おやすみ	o y a s u m i
お話し	o h a n a s h i
ありがとう	a r i g a t o u
天気予報	t e N k i y o h o u
おはよう	o h a y o u
こんにちは	k o N n i c h i w a
こんばんは	k o N b a N w a
お名前は	o n a m a e w a
なにができるの	n a n i g a d e k i r u

6.5.3. 各列の説明

列名	説明
単語表記	音声認識が成功した際に、Julius から認識結果として返される文字列です。プログラム側では、この文字列をキーとして処理分岐や応答生成に使用します。
音素列	単語の発音を Julius 用の音素記号で表したものです。日本語音響モデルに基づき、母音・子音・撥音(N)などを空白区切りで記述します。

6.5.4. 補足：Julius 音声辞書運用上のポイント

- 単語表記はプログラム側の判定キーになるため、表記ゆれを避ける必要があります（例：「こんにちは」と「こんにちわ」を混在させない）
- 音素列は必ず Julius の音響モデルに対応した記法を使用してください
- 認識語彙を必要最小限に絞ることで、Raspberry Pi 環境でも高速かつ高精度な認識が可能になります

6.6. Julius 設定ファイル

6.6.1. ファイル概要

本設定ファイルは、Julius 音声認識エンジンの動作条件を定義する設定ファイルです。認識に使用する辞書、音響モデル、言語モデル、出力形式などを指定します。

AI ミケでは、

- 短い定型コマンドを高精度に認識する
- Raspberry Pi 環境でも 軽快に動作させる

ことを目的として、認識語彙を意図的に制限した構成を採用しています。

6.6.2. 設定内容 解説 一覧表

オプション	説明
<code>-w mike.dic</code>	ユーザー定義の単語辞書を指定します。この辞書に登録された単語のみが認識対象となります。
<code>-v ../dictation-kit/model/lang_m/bccwj.60k.htkdic</code>	大語彙用の音素辞書を指定します。Julius 標準の約 60,000 語規模の辞書で、言語モデルと組み合わせて使用されます。
<code>-h ../dictation-kit/model/phone_m/jnas-tri-3k16-gid.binhmm</code>	音響モデル(HMM)を指定します。入力音声と音素列の対応関係を確率的に判定するためのモデルです。
<code>-hlist ../dictation-kit/model/phone_m/logicalTri-3k16-gid.bin</code>	音響モデルで使用する音素(phone)の一覧を指定します。音響モデルと辞書の整合性を取るために必要です。
<code>-n 3</code>	言語モデルの n-gram 次数を指定します。3 はトリグラムを意味し、直前 2 語までの文脈を考慮します。
<code>-output 1</code>	認識結果の出力形式を指定します。1 は通常のテキスト形式で認識結果を出力します。
<code>-nolog</code>	実行時のログ出力を抑制し、認識結果のみを出力します。

6.6.3. 補足 : AI ミケにおける Julius 設定の設計意図

- `-w` による語彙制限
自由発話を狙わず、制御コマンドに特化することで誤認識を減らし、処理負荷も低減しています。
- 標準音響モデルの流用
独自学習を行わず、Julius 標準モデルをそのまま使用することで、導入の手軽さと再現性を重視しています。
- `-nolog` の指定
Raspberry Pi 環境ではログ I/O がボトルネックになりやすいため、実運用を意識して抑制しています。

7. プログラムインストール（アプリ版）

本章では、Windows 環境で AI ミケ（アプリ版）をインストールし、起動するまでの手順を説明します。アプリ版は、Docker を使用せず、Windows 上で単体アプリケーションとして AI ミケを実行する構成です。

7.1. インストール先フォルダ作成と解凍

1. エクスプローラーで以下のフォルダを作成します。
2. 配布された `mike_app_xxx.zip` を、上記フォルダに解凍します。

7.2. 設定

解凍後、設定ファイルを環境に合わせて編集します。設定項目の詳細については、「6. 各種設定ファイル説明」を参照してください。

特に以下の項目は、環境に応じて必ず確認してください。

- VOICEVOX サーバのアドレス・ポート
- Ollama サーバのアドレス・使用モデル
- 呼び出しワード・終了ワード

7.3. インストール ・ 起動

```
# インストール
PS> ./install.ps1
```

`install.ps1` により、AI ミケの実行に必要な Python ライブラリや依存モジュールが自動的にインストールされます。

7.3.1. 仮想環境の有効化と起動

Ollama および VOICEVOX を サーバモードで起動していることを確認したうえで、Python の仮想環境を有効化します。

```
# ollama と voicevox をサーバモードで起動
# python の仮想環境に設定
PS> source venv313/bin/activate
# 起動
PS> python main.py
```

7.3.2. 初回起動時の注意点

インストール直後は、AI ミケの発声に必要な音声ファイル（WAV）が存在しません。そのため、

- 初めて音声コマンドを受け取った際に
→ 対応する音声データを生成・保存
 - 初回応答時は
→ 音声合成処理のため、応答に時間がかかる場合あり
- という挙動になります。

一度生成された音声データは 静的データとして再利用されるため、

2 回目以降はスムーズに応答するようになります。

7.3.3. 操作方法について

AI ミケの起動後の操作方法、モード切替、終了手順については、「9. プログラム起動・操作・停止」を参照してください。

8. プログラムインストール (Docker 版)

本章では、AI ミケ (Docker 版) のアプリケーション本体を Raspberry Pi に導入する手順を説明します。Docker 版では、プログラム本体、設定ファイル、ログなどを 専用ディレクトリ `/opt/mike` 配下に集約して配置します。

8.1. インストール先フォルダ作成

AI ミケでは、実行に必要なプログラム・設定ファイル・ログ類を `/opt/mike` ディレクトリ以下にまとめて管理します。

まず、以下のコマンドを実行してディレクトリを作成し、所有者をユーザー `mike` に変更します。

```
$ sudo mkdir /opt/mike
$ sudo chown mike:mike /opt/mike
$ ls -l /opt/
drwxr-xr-x 2 mike mike 4096 Nov 28 10:16 mike
```

作成後、所有権を確認します。

`/opt/mike` の所有者が `mike:mike` になっていれば準備完了です。

このディレクトリに、以降の手順で Docker ベースの AI ミケのプログラム一式を配置していきます。

8.2. 配布パッケージの展開

続いて、AI ミケの配布パッケージを `/opt/mike` に展開します。

ここでは、あらかじめ用意した `mike_xxx.tar.gz` を使用します。

まず、アーカイブを `/opt/mike` にコピーし、ディレクトリに移動して解凍します。

```
$ cp mike_doc_xxx.tar.gz /opt/mike
$ cd /opt/mike
$ tar xvzf mike_xxx.tar.gz
```

`tar xvzf` を実行すると、プログラム本体、設定ファイル、Docker 用各種ファイルが `/opt/mike` 配下に展開されます。

以降の章では、この展開されたファイルを用いて Docker イメージの構築や設定調整を行います。

8.3. Docker イメージのビルド

本節では、AI ミケの動作に必要な Docker イメージを構築します。

前章で /opt/mike に展開したファイル一式には、Dockerfile や各コンポーネント用の設定が含まれており、これらを基にコンテナ環境を作成します。

- AI ミケのシステムは、次のような **複数サービス構成**になっています。
- 全体制御（ミケクライアント）
 - 応答判定（MCP サーバを含む）
 - データ作成・管理
- このような構成により Docker による **サービス分離と一括管理**を実現しています。

8.3.1. ビルドスクリプトの実行

各種コンテナは、以下のビルドスクリプトを実行することで一括構築できます。

```
$ tools/build_mike.sh
```

```
=====
==== MIKE[ラズパイ] を DOCKER にインストールします ====
==== (注) 現在の CONTAINER と IMAGE は 削除されます====
=====
ok? (y/N): y
docker: 'docker stop' requires at least 1 argument

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

See 'docker stop --help' for more information
=====
==== 現在の CONTAINER を 削除します      =====
==== 初めての時は。CONTAINER ありませんので =====
==== ERR 表示されますが問題ありません  =====
=====
Error response from daemon: No such container: mike_client
Error response from daemon: No such container: mike_data-cont
Error response from daemon: No such container: mike_srv-cont
=====
==== 現在の IMAGE を 削除します      =====
==== 初めての時は。IMAGER はありませんので =====
==== ERR 表示されますが問題ありません  =====
=====
ok? (y/N): y
Error response from daemon: No such image: mike_client-img:latest
Error response from daemon: No such image: mike_server-img:latest
Error response from daemon: No such image: mike_data-img:latest
=====
==== MIKE[ラズパイ] を DOCKER にインストールします ====
=====
ok? (y/N): y
0 MAKE_NETWORK
Error response from daemon: network with name mike-net already exists
1 MIKE_DATA
:
```

ネットワークは先に作成してありますので、問題ありません

3 つのコンテナが作成されていることを確認する

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3415b5c41d32	mike_client-img	"sh -c '¥n sh /opt/m..."	16 seconds ago	Created		mike_client
3b7402d389b8	mike_server-img	"/bin/bash ./start.sh"	9 minutes ago	Created		mike_srv-cont
337a6af12fad	mike_data-img	"/bin/bash"	21 minutes ago	Created		mike_data-cont

8.3.2. ビルド完了後の確認

ビルドが完了すると、以下の 3 つのコンテナが作成されますので、各コンテナが存在することを確認してください。

IMAGE	NAMES
mike_client-img	mike_client
mike_server-img	mike_srv-cont
mike_data-img	mike_data-cont

8.4. 設定（接続先設定）

ビルド完了後、設定ファイルを環境に合わせて編集します。設定項目の詳細については、「6. 各種設定ファイル説明」を参照してください。

特に以下の項目は、環境に応じて必ず確認してください。

- VOICEVOX サーバのアドレス・ポート
- Ollama サーバのアドレス・使用モデル

\$ vi mike_data/opt_mike/config/config.json	
"VOICEVOX": { "Host": "voicevox", "Port": 50021, "Speaker": 45 },	<ul style="list-style-type: none">• Host: VOICEVOX を動かしている PC(または Docker コンテナ)のホスト名/IP• Port: VOICEVOX Engine のポート番号(標準は 50021)• Speaker: 使用する話者 ID 例: 45 は「四国めたん(あまあま)」 ここで指定した話者 ID が使用されま
"OLLAMA": { "Host": "http://ollama_local", "Port": 11434, "Model": "gpt-oss:20b", "Content": "You are a wonderful butler. Please reply in 100 characters or less." },	<ul style="list-style-type: none">• Host: OLLAMA を稼働させている PC の IP/ホスト名• Port: OLLAMA の標準ポート(11434)• Model: 使用するローカル LLM 名(例: llama3.2:1b)• Content: ミケのキャラクター設定(初期プロンプト) <p>この設定により、ミケは外部 PC の LLM 推論と音声合成を組み合わせ対話応答を生成できるようになります。</p>

8.5. コマンド用音声認識ソフト julius の作成

次に、コマンド認識用の音声認識エンジン Julius を作成します。

実行中、ネットワークやストレージの影響で処理が途中で止まっているように見える場合がありますが、慌てずに待ってください。

8.5.1. Julius 作成スクリプトの実行

以下のコマンドを実行します。

\$ tools/julius_make.sh
mike@Pi4b2:/opt/mike \$ tools/julius_make.sh =====
==== MIKE[ラズパイ]で起動しているサービスを停止します =====
=====
ok? (y/N): y 76e1349aaa25 eb887068e7a3 830c44419beb =====

```

===== MIKE[PC]で起動しているサービスも停止してください =====
=====
===== MIKE[PC]のターミナルで、MIKE_HOME に移動し =====
===== このコマンドで起動してください =====
===== PS> tools/mikeS_stop.ps1 =====
=====
ok? (y/N): y
=====
===== Julius をインストールします =====
===== 起動後は、マイクで確認してください =====
===== 確認後は Ctrl-C で停止してください =====
=====
ok? (y/N): y
[+] Building 3.2s (31/31) FINISHED
=> [internal] load local bake definitions
:
:
mike_client | ----- System Information end -----
mike_client |
mike_client | Notice for feature extraction (01),
mike_client | *****
mike_client | * Cepstral mean normalization for real-time decoding: *
mike_client | * NOTICE: The first input may not be recognized, since *
mike_client | * no initial mean is available on startup. *
mike_client | *****
mike_client |
mike_client | -----
mike_client | ### read waveform input
mike_client | Stat: capture audio at 16000Hz
mike_client | Stat: adin_alsa: current latency time: 64 msec
mike_client | Stat: adin_alsa: latency set to 64 msec (chunk = 1024 bytes)
mike_client | Stat: "default": TSK95K [TSK95K] device USB Audio [USB Audio] subdevice #0
mike_client | STAT: AD-in thread created
<input rejected by short input> >>>

```

ここまで表示されれば、Julius のビルドと起動が正常に完了している状態です。

ビルドが進むと、最終的に Julius が起動し、リアルタイム音声認識モードに入ります。

8.5.2. Julius 実行後の動作

Julius の起動後は、自動的に 音声入力モード（リアルタイム認識モード）になります。

この状態でマイクに向かって、以下のような短い発話を行います。

- 「おはよう」
- 「テスト」
- 何かしらの短い発話

音声認識されると、以下のようなログが表示されます。

認識精度は、

- USB マイクの性能
- 環境ノイズ
- 発話の長さ

などに左右されますが、この時点では 何かしら文字列が認識されれば正常動作と判断して問題ありません。

この段階では、完全な認識精度を求める必要はありません。

```

<input rejected by short input> >>>
mike_client | STAT: skip CMN parameter update since last input was invalid
mike_client |
pass1_best: おはよう 。 e speak >>>
mike_client | pass1_best_wordseq: <s> おはよう+感動詞 </s>

```



```

mike_client | pass1_best_phonemeseq: silB | o h a y o: | silE
mike_client | pass1_best_score: -2131.077148
mike_client | ### Recognition: 2nd pass (RL heuristic best-first)
mike_client | STAT: 00_default: 24088 generated, 1592 pushed, 285 nodes popped in 90
mike_client | sentence1: おはよう。
mike_client | wseq1: <s> おはよう+感動詞 </s>
mike_client | phseq1: silB | o h a y o: | silE
mike_client | cmscore1: 0.729 0.117 1.000
mike_client | score1: -2150.133789
mike_client |
<input rejected by short input> >>>

```

8.5.3. Julius の停止方法

確認が終わったら、Ctrl+C で Julius を停止してください。

```
Ctrl + C
```

8.6. 音声コマンド用音声データ作成

AI ミケでは、Julius による音声認識と VOICEVOX による音声合成を組み合わせ、
「音声コマンド → 返答音声」の対応データを事前に生成します。

この音声データ生成処理は、付属のスクリプト **tools/make_data.sh** により自動化されています。

8.6.1. 音声データ作成スクリプトの実行

以下のコマンドを実行し、音声データ作成処理を開始します。

スクリプトは、安全のために次の点を順に確認します。

- Julius の作成が完了しているか
 - config.json に設定した接続先アドレスが正しいか
 - VOICEVOX
 - MIKE_SERVER
 - Ollama
 - PC 側で VOICEVOX をサーバモードで起動しているか
 - Raspberry Pi 側で稼働中の MIKE サービスを一時停止してよいか
- それぞれ問題なければ、y を入力して処理を進めます。

```

tools/make_data.sh
=====
====   MIKE の DATA を作成します。   =====
=====
ok? (y/N): y
=====
====   Julius の make は済みましたか？   =====
=====
ok? (y/N): y
=====
====   MIKE_DATA の config/配下のファイルを編集済ですか？   =====
=====
====   下記のフィルに接続先のアドレスが正しいですか   =====
=====
====   config.json VOICEVOX,mike_serve,ollama のアドレス   =====
=====

```

```
ok? (y/N): y
```

```
=====MIKE{PC}側で VOICEVOX をサーバモードで起動していますか？=====
```

```
ok? (y/N): y
```

```
===== MIKE[ラズパイ]で起動しているサービスを停止し =====
```

```
===== MIKE の DATA を作成します。 =====
```

```
ok? (y/N): y
```

8.6.2. 音声コマンド用音声データ作成（完了確認）

音声データ作成処理が正常に完了すると、スクリプトの最後に次のようなメッセージが表示されます。

あわせて、生成された音声ファイルおよび設定ファイルの一覧が表示されます。

さらに、音声生成に使用した各種設定ファイルについても、同一時刻で更新されていることが確認できます。

```
=====
===== DATA 作成が完了しました =====
===== 現在時刻は、Sat Nov 29 11:14:49 JST 2025 です =====
===== 各ファイルの日時がこの時刻になっていますか？ =====
=====
-rw-r--r-- 1 mike mike 57468 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/0-mike.wav
-rw-r--r-- 1 mike mike 61230 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/1-ohayou.wav
-rw-r--r-- 1 mike mike 68286 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/2-ohanashi.wav
-rw-r--r-- 1 mike mike 80046 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/3-konban.wav
-rw-r--r-- 1 mike mike 81928 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/4-konniti.wav
-rw-r--r-- 1 mike mike 93216 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/5-
onamaewa.wav
-rw-r--r-- 1 mike mike 183064 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/6-naniga.wav
-rw-r--r-- 1 mike mike 70638 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/end-nata.wav
-rw-r--r-- 1 mike mike 73462 Nov 29 11:14 /opt/mike/mike_client/opt_mike/snd/error.wav
-rw-r--r-- 1 mike mike 868 Nov 29 11:14
/opt/mike/mike_client/opt_mike/config/communication.csv
-rw-r--r-- 1 mike mike 1219 Nov 29 11:14 /opt/mike/mike_client/opt_mike/config/config.json
-rw-r--r-- 1 mike mike 259 Nov 29 11:14 /opt/mike/mike_client/opt_mike/config/mike.dic
-rw-r--r-- 1 mike mike 219 Nov 29 11:14
/opt/mike/mike_client/opt_mike/config/mike_mcp_servers.json
-rw-r--r-- 1 mike mike 1219 Nov 29 11:14 /opt/mike/mike_server/opt_mike/config/config.json
-rw-r--r-- 1 mike mike 219 Nov 29 11:14
/opt/mike/mike_server/opt_mike/config/mike_mcp_servers.json
=====
===== データ作成が完了しました =====
===== ミケ は tools/mike_start.sh で起動します =====
===== 終了は tools/mike_stopt.sh です =====
=====
```

8.6.3. 正常完了の判断基準

次の条件をすべて満たしていれば、音声コマンド用音声データの作成は正常に完了しています。

- snd/ ディレクトリ配下に 複数の WAV ファイルが生成されている
- ファイルの更新日時が スクリプト実行時刻と一致している
- エラーメッセージが表示されていない

この時点で、communication.csv に定義されたコマンドに対応する音声データがすべて揃った状態になります。

8.7. プログラム 起動・操作・停止

本章では、Docker 版 AI ミケの **起動・操作・停止方法** について説明します。

AI ミケは複数の Docker コンテナで構成されており、付属の管理スクリプトを使用することで、簡単に操作できます。

8.7.1. 起動

ミケを起動するには、次のスクリプトを実行します。

```
$ tools/mike_start.sh
```

このスクリプトにより、以下のコンテナが起動します。

■ 起動するコンテナ

- **mike_client**
 - MikeClient (音声入力、音声認識、ユーザーインタフェース処理)
- **mike_server**
 - mike_server (応答判定機能：応答生成・コマンド処理)
 - mike_weather_mcp_server (天気情報 API サーバ)
が起動します。

起動後、ターミナルには **mike_client** のログがリアルタイムで表示されます。

この状態で AI ミケは **音声認識待機状態** に入り、呼びかけを待ちます。

8.7.2. 音声トラブル発生時の対処方法（重要）

稀に、AI ミケの音声が大音量のノイズとして出力される 場合があります。

これは、音声再生システムや USB オーディオデバイスが何らかの理由で異常状態に陥ったことが原因です。次の手順で復旧できます。

- 1 : AI ミケを終了する。
\$ cd /opt/mike
\$ toools/mike_stop.sh
- 2 : 音声デバイスを USB ソケットから**抜く**
- 3 : 音声デバイスを USB ソケットに**挿す**
- 4 : 音量を確認する
- 5 : AI ミケを起動する。
\$ toools/mike_start.sh

△ 注意

このような場合でも、**電源を抜くなどの強制終了は絶対に行わないでください。**

8.7.3. 操作

AI ミケの操作は、音声コマンドによる状態遷移によって行います。ミケは内部的に複数の動作モードを持っており、ユーザーの発話内容に応じてモードが切り替わります。

コマンド一覧	応答判定機能の状態遷移図										
<table border="1"> <thead> <tr> <th>遷移コマンド</th><th>音声コマンド</th></tr> </thead> <tbody> <tr> <td>①：開始モード遷移</td><td>みけおはよう</td></tr> <tr> <td>②：会話モード遷移</td><td>お話し</td></tr> <tr> <td>③：待機モード遷移</td><td>ありがとう</td></tr> <tr> <td>④：終了</td><td>またね</td></tr> </tbody> </table> <p>コマンド定義ファイル (config.json) で設定した語句が音声コマンドとして認識されます。</p> <pre> "MIKE_CLIENT": { "RECOGNITION": 0, "DeviceIndex": 0, "Commands_File": "config/communication.csv", "WAKE_WORD": "みけおはよう", "SLEEP_WORD": "ありがとう", "CHAT_START_WORD": "お話し", "END_WORD": "またね", }, </pre>	遷移コマンド	音声コマンド	①：開始モード遷移	みけおはよう	②：会話モード遷移	お話し	③：待機モード遷移	ありがとう	④：終了	またね	<pre> graph TD Start([起動]) -- ① --> Standby[待機モード] Standby -- ③ --> Command[コマンドモード] Standby -- ③ --> Chat[会話モード] Command -- ② --> Chat Command -- ④ --> End([終了]) Chat -- ④ --> End Standby -- ④ --> End </pre>
遷移コマンド	音声コマンド										
①：開始モード遷移	みけおはよう										
②：会話モード遷移	お話し										
③：待機モード遷移	ありがとう										
④：終了	またね										

■ 音声の作成用ファイル (communication.csv) の一部

番号	音声コマンド	音声応答	スピーカ	ファイル名	問い合わせよみ
"err"	"えらー"	"わからないにゃ～！"	"45"	"error"	"はて"
"end"	"またね"	"ばいばいだにゃ～！"	"45"	"end-nata"	"またね"
"wake"	"みけおはよう"	"なにかにゃ～"	"45"	"wake-mike"	"みけおはよう"
"chat"	"お話ししよう"	"よろこんでにゃ～"	"45"	"chat-ohanashi"	"お話し"
"cend"	"ありがとう"	"いつでもいいにゃ～！"	"45"	"cend-ariga"	"ありがとう"

この CSV を元に VOICEVOX で音声ファイルが自動生成され、音声コマンドと応答音声のペアが作成されます。

8.7.4. 操作上のポイント

- すべての操作は音声のみで完結します
- 明確なウェイクワード (例:「みけおはよう」) により、誤動作を防止しています
- 会話モードでは、Julius (短文認識) から SpeechRecognition (自由会話) へ自動的に切り替わります
- 「またね」は アプリケーション終了コマンドのため、使用後は再度 tools/mike_start.sh による起動が必要です

8.7.5. 停止

AI ミケの停止方法には、音声による停止 と スクリプトによる停止 の 2 種類があります。運用状況に応じて、適切な方法を選択してください。

音声による停止 (AI ミケの終了)

最も簡単な停止方法は、ミケに向かって次のように話しかけることです。

「またね」

この終了コマンドを AI ミケが認識すると、mike_client コンテナ内で動作している MikeClient が正常終了し、同時に 各コンテナ自体も終了します。

この方法では、

- 音声入力ストリーム
- USB オーディオデバイス
- 内部スレッドやバッファ

といった 関連リソースが安全に解放されるため、通常の運用では この方法を推奨します。

スクリプトによる停止

Docker 版 AI ミケのスクリプトによる停止は次のスクリプトにより、AI ミケを構成する以下の Docker コンテナが順に安全に停止します。

```
$ tools/mike_stop.sh
```

- **mike_client**
(音声入力・音声認識・ユーザーインタフェース処理)
- **mike_server**
(応答生成・コマンド処理を担当するメインサーバ)
- **weather_mcp_server**
(天気情報 API を提供する MCP サーバ)

すべてのコンテナが停止することで、AI ミケの全サービスは完全に終了した状態になります。

8.8. 会話の例

実際に試した会話の内容を次にしめします。

問い合わせ	AI ミケの回答
みけおはよう	なにかにや
なにができるの	たのしいおはなしとおてんきのはなしだにや～！
お話しましょ	よろこんでにや～
東京の天気を教えて	それじゃ一天気予報を調べてみるにや～！ちょっとまってね今日の東京は曇りです。気温は 10 度で、涼しいでしょう。軽めの上着を持って行くと安心です
遊びにけるかな	今日の東京は曇りです。気温は 10 度で、外出は快適です
どこに行くといい	東京の天気は曇りで 10 度、公園やカフェ、浅草寺や渋谷のショッピングがおすすめです。
渋谷にはどんなお店があるの	渋谷にはファッション、カフェ、レストラン、電気製品、雑貨店など多彩なお店があります
ありがとう	いつでもいいにや～！
またね	ばいばいだにや～！

注：AI ミケの回答は、常に変化します。

9. 開発メモ

本章では、AI ミケを 開発・運用する過程で得られた知見や、実際に構築して分かった **注意点・設計上の判断理由** をまとめます。これから同様のシステムを構築する際の参考として活用してください。

9.1. Raspberry Pi 利用時の注意点

- Raspberry Pi は CPU 処理能力が限られているため、**LLM 推論や音声合成は外部 PC にオフロードする設計が必須**です。
 - USB オーディオデバイスは認識順序が不安定な場合があるため、**不要なオンボードデバイス(bcm2835 / vc4-hdmi)を無効化**すると安定します。
 - 書き込み頻度の高いログは SD カード寿命を縮める可能性があります。実運用では **ログレベル INFO 程度に抑える**ことを推奨します。
-

9.2. Julius の辞書サイズと最適化

- Julius は辞書に登録する単語数が増えると、**認識精度が低下しやすい**特性があります。
 - AI ミケのような コマンド用途では、**5~10 語程度に制限する構成が最も安定**しました。
 - 起動直後に認識率が低くなる現象は、**Cepstral Mean Normalization (CMN) が未収束**なために起こります。
 - 対策として、**起動後に 1 秒程度の無音時間を設けると安定**します。
-

9.3. VOICEVOX の話者 (Speaker) について

- 生成するセリフ数が多い場合、VOICEVOX エンジン **CPU / GPU を一定時間占有**します。
 - そのため、**別 PC での実行が最適**です。
 - 話者 ID は **固定(例:45)** にしておくと、音声ファイル再生時の **差分管理が容易**になります。
-

9.4. Ollama のモデル選定とメモリ要件

- ローカル LLM は **GPU メモリ容量に強く依存**します。
- **1B~3B クラスのモデル**は応答速度は高速ですが、**ReAct Agent での推論には 能力不足になる場合があります**。
- 大規模モデルは生成時間が長くなり、実利用では **応答遅延がストレス**になることがあります。

- 実用面では、
小型モデル+手書きルールのハイブリッド構成が現実的でした。

9.5. Docker 構成とネットワーク設計

- コンテナ間通信には **専用ネットワーク(mike-net)**を使用しています。
- 外部 PC (VOICEVOX / Ollama)との通信は、
IP アドレスを固定するとトラブルが減ります。
- Raspberry Pi 上での Docker イメージビルドは時間がかかるため、
外部 PC で buildx によりビルド → Pi へ転送する方法が高速です。

9.6. 音声コマンド CSV (communication.csv) の管理

- communication.csv は、
AI ミケの「性格」と「会話品質」を決める重要ファイルです。
- VOICEVOX 音声を再生成する際、
ファイル名を変更すると旧音声が残るため注意が必要です。
 - 「問い合わせよみ」列は
Julius 辞書生成に使用されるため削除してはいけません。

9.7. トラブルシューティング

現象	確認ポイント
認識しない	USB マイクの DeviceIndex を確認
応答しない	VOICEVOX / Ollama の IP 設定を確認
終了しない	「またね」が認識されたかログを確認

9.8. Swap サイズについて

Raspberry Pi 4B (8GB) では、
通常 AI ミケの動作において **Swap** の拡張は不要です。

ただし、

- メモリ容量が小さいモデル (例: Zero 2W)
 - 他のサービスを同時に動かす場合
- には、Swap 設定の調整が有効な場合があります。

9.9. Swap サイズの変更 (zramswap)

最近の Raspberry Pi OS (Bookworm 以降) では、従来の `/etc/dphys-swapfile` を使った Swap ではなく、**zramswap (ZRAM Swap)** が標準で利用されています。

9.9.1. zramswap とは

zramswap は、RAM 上に圧縮されたスワップ領域を作成する仕組みです。物理ストレージを使用しないため、Raspberry Pi との相性が良い方式です。

9.9.2. 通常の Swap との違い

項目	通常の Swap	zramswap
使用場所	SD / SSD	RAM (圧縮)
書き込み	発生する	発生しない
速度	遅い	比較的高速
SD 寿命	消耗する	消耗しない

9.9.3. メリット

1. メモリ不足に強くなる
2. SD カードへの書き込みが減る
3. 体感性能が向上する場合がある

9.9.4. デメリット

- 圧縮・展開により CPU 負荷がわずかに増加
- 物理メモリを一部消費するため、
極端にメモリが少ない環境では調整が必要

9.9.5. 設定方法 (Raspberry Pi OS)

Bookworm 以降では、`/etc/rpi/swap.conf` で zramswap を設定します。

9.9.6. 設定例 (Raspberry Pi 4B)

```
$ cat /etc/rpi/swap.conf
[Zram]
MaxSizeMiB=4096
FixedSizeMiB=2048
```

必要に応じて数値を調整し、再起動してください。

以上