Virtual Raspberry Pi 5 (Moode Audio) Plan — for Cursor on Mac

Goal

I want a virtual environment that is Raspberry-Pi-like (ARM64 Linux) so I can develop + test Moode Audio style software using Cursor.

Important: true Raspberry Pi 5 hardware (EEPROM boot, GPU, I2S DAC HAT) is not fully emulatable. So we use:
• Devcontainer (fast dev) ■
• Optional QEMU ARM64 VM (closest Pi-like validation) ■
• Real Pi 5 (final hardware test) ■

A) Best workflow for Cursor learning: Devcontainer (recommended)

This makes Cursor “learn” the environment because dependencies and scripts live inside the project.

A1) Create these files in the repo

.cursorrules

Create a file called `.cursorrules` at repo root:

You are working on a project that mimics Moode Audio behavior on Raspberry Pi.
Goals:
• Compatibility with Debian/Raspberry Pi OS (Linux)
• Assume deployment on ARM64 even if dev runs on x86 Docker
• Prefer CLI tooling and config files
• Use mpd/nginx/php-fpm/alsa patterns where needed
When writing code:
• Prefer bash scripts for install/start/diagnostics
• Provide exact file paths and systemd service names
• Add comments for Pi-specific assumptions and hardware limitations

---

.devcontainer/devcontainer.json

Create folder `.devcontainer/` then create file `.devcontainer/devcontainer.json`:

```
{
  "name": "Moode-like Dev",
  "build": { "dockerfile": "Dockerfile" },
  "runArgs": ["--privileged"],
  "forwardPorts": [80, 3000],
  "postCreateCommand": "bash .devcontainer/postCreate.sh",
  "customizations": {
    "vscode": {
      "extensions": [
        "ms-azuretools.vscode-docker"
```

```
            ]
          }
        }
      }
```

.devcontainer/Dockerfile

Create file `.devcontainer/Dockerfile`:

```
FROM debian:bookworm

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y \
    nginx php-fpm php-cli php-curl php-xml php-mbstring php-zip \
    git curl unzip ca-certificates \
    mpd mpc alsa-utils \
    sudo \
    && apt-get clean

RUN useradd -m dev && echo "dev ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
USER dev
WORKDIR /workspace
```

.devcontainer/postCreate.sh

Create file `.devcontainer/postCreate.sh`:

```
#!/usr/bin/env bash
set -e
echo "Devcontainer ready."
echo "Installed: nginx, php-fpm, mpd, alsa-utils"
```

A2) Add scripts Cursor can run (so it understands how to operate the project)

scripts/start.sh

Create folder `scripts/` then file `scripts/start.sh`:

```
#!/usr/bin/env bash
set -e

echo "Starting services..."
sudo service nginx start || true
sudo service php*-fpm start || true
sudo service mpd start || true

echo "Status:"
sudo service nginx status || true
sudo service mpd status || true
```

---

scripts/diagnose_audio.sh

Create file `scripts/diagnose_audio.sh`:

```
#!/usr/bin/env bash
set -e

echo "### ALSA devices"
```

```
aplay -l || true

echo
echo "### MPD version"
mpd --version || true

echo
echo "### Kernel"
uname -a || true

echo
echo "### Notes"
echo "- In devcontainer/QEMU, real Raspberry Pi I2S DAC hardware will not exist."
echo "- This is for software + config development."
```

A3) How to run in Cursor (Mac)

1) Open repo in Cursor
2) Command Palette -> "Dev Containers: Reopen in Container"
3) In terminal inside Cursor:
• `chmod +x scripts/*.sh .devcontainer/postCreate.sh`
• `./scripts/start.sh`
• `./scripts/diagnose_audio.sh`

Result: Cursor now has a stable, reproducible environment and the AI can reason about it.

B) Closest to Raspberry Pi in a VM: ARM64 Linux using QEMU (optional)

Use this when you want ARM64 validation similar to Raspberry Pi OS.

B1) Install QEMU on Mac

```
brew install qemu
```

B2) Create a VM disk

```
qemu-img create -f qcow2 arm64.img 32G
```

B3) Get an ARM64 installer ISO

Download an ARM64 ISO (Debian ARM64 or Ubuntu Server ARM64), save as:
• `debian-arm64.iso` OR `ubuntu-arm64.iso`

B4) Boot installer (first boot)

```
qemu-system-aarch64 \
  -machine virt \
  -cpu cortex-a72 \
  -m 4096 \
  -smp 4 \
  -cdrom debian-arm64.iso \
  -drive if=virtio,file=arm64.img,format=qcow2 \
  -netdev user,id=net0,hostfwd=tcp::2222-:22 \
  -device virtio-net-device,netdev=net0 \
  -nographic
```

Install the OS, enable SSH server during install.

B5) Boot VM after install

```
qemu-system-aarch64 \
  -machine virt \
  -cpu cortex-a72 \
  -m 4096 \
  -smp 4
```

Note

This document was recovered from an **unsaved Cursor editor buffer** (`Untitled-1`) and Cursor's extraction truncated parts of the later VM section. If anything is missing after **B5**, reopen the original tab and we'll append the remaining sections.