Project Experience Summary - moOde Audio Custom Build

■ Project Overview

This project involves creating a custom build of moOde Audio Player for Raspberry Pi 5, specifically configured for the "Ghettoblaster" audio system with:
• HiFiBerry AMP100 amplifier
• Waveshare DSI touchscreen display (portrait mode)
• Custom systemd services
• Network configuration
• Remote access via WireGuard VPN

Base System: moOde Audio Player 10.0.1 on Debian GNU/Linux 13 (Raspberry Pi OS)

■ Biggest Problems Encountered

1. Boot Hangs - The Persistent Nightmare

Problem:
The Pi would hang during boot, consistently failing at various stages:
• "Kernel file systems..." stage
• "fix-ssh-service" or "fix-ssh-sudoers" services
• Network initialization
• Systemd service dependencies

Why It Was So Difficult:
• Multiple interdependent services causing cascading failures
• Services waiting for dependencies that didn't exist (`moode-startup.service`)
• Network services blocking on `network-online.target` which could hang indefinitely
• No timeouts on services, causing infinite waits
• Hard to debug without physical access during boot

Impact:
• Weeks of development time lost
• Constant SD card mounting/unmounting
• Couldn't test changes without full boot cycle
• Very frustrating development experience

---

2. Systemd Service Dependency Hell

Problem:
Services were configured with dependencies that:
• Referenced non-existent services (`After=moode-startup.service`)
• Waited for network targets that could hang (`Wants=network-online.target`)
• Had no timeouts, causing indefinite blocking
• Were redundant (multiple SSH services doing the same thing)

Examples of Problematic Services:

```
[Unit]
After=moode-startup.service  # This service doesn't exist!
Wants=network-online.target   # Can hang forever
Before=multi-user.target

[Service]
ExecStart=/bin/bash -c "visudo -c && systemctl enable ssh"  # Blocking commands
# No TimeoutStartSec - can hang forever
```

Why It Was Hard:
• Systemd dependency resolution is complex
• Services in both `/etc/systemd/system/` and `/lib/systemd/system/`
• Build scripts install services, but fixes needed to be in source files
• Hard to test without full boot cycle

---

3. Network Configuration Conflicts

Problem:
Multiple services and scripts trying to configure network:
• `fix-network-ip.service` setting static IP
• `03-network-configure.service` also setting IP
• `NetworkManager` vs `systemd-networkd` conflicts
• IP addresses hardcoded in multiple places (192.168.10.2, 192.168.2.3, etc.)

Why It Was Hard:
• Different services running at different times
• Race conditions between network managers
• Hard to track which service "won" the configuration
• IP changes required updates in multiple files

---

4. Display Configuration Issues

Problem:
• Display showing only 1/3 of screen (cut-off)
• Portrait mode not working correctly
• Rotation settings conflicting between `cmdline.txt`, `config.txt`, and `.xinitrc`
• Chromium not starting due to `.xinitrc` syntax errors

Why It Was Hard:
• Multiple configuration layers (kernel, X11, moOde database)
• Native portrait display requires specific resolution (400x1280)
• Rotation settings in multiple places that could conflict
• Syntax errors in `.xinitrc` preventing X server from starting

---

5. Audio Chain Configuration

Problem:
• Audio not routing correctly through ALSA chain
• MPD not finding correct audio device
• `_audioout.conf` not configured correctly
• PeppyMeter integration issues

Why It Was Hard:
• Complex ALSA routing: MPD → `_audioout` → `camilladsp`/`peppy` → hardware
• moOde database controls routing logic
• Multiple configuration files need to be in sync

• Hard to test without actual audio hardware

---

6. SSH Host Key Changes

Problem:
After re-flashing SD card, SSH would fail with "REMOTE HOST IDENTIFICATION HAS CHANGED" error. Lost 3 weeks of debugging time because this wasn't recognized.

Why It Was Hard:
• Error message not immediately obvious
• SSH keys change when SD card is re-flashed
• Need to remove old keys: `ssh-keygen -R `
• Easy to miss when focused on other issues

■■ Solutions Tried & What Worked

1. Boot Hang Solutions

What We Tried:
1. **Added timeouts to all services:**
```
TimeoutStartSec=10
TimeoutStopSec=5
DefaultDependencies=no
```
■ **This worked** - Prevents infinite hangs

2. **Removed problematic dependencies:**
• Removed `After=moode-startup.service` (doesn't exist)
• Removed `Wants=network-online.target` (can hang)
• Changed to `After=network.target` (less blocking)
■ **This worked** - Services start faster

3. **Made commands non-blocking:**
```
ExecStart=/bin/bash -c "timeout 5 visudo -c && timeout 5 systemctl enable ssh && exit 0"
```
■ **This worked** - Commands can't hang forever

4. **Disabled redundant services:**
• Multiple SSH services doing the same thing
• Disabled all but one unified service
■ **This worked** - Reduced conflicts

5. **Fixed cmdline.txt:**
• Changed `fsck.repair=yes` → `fsck.mode=skip`
• Removed `quiet`, added `loglevel=7`
■ **This worked** - Boot faster, more verbose

---

2. Systemd Service Fixes

What We Tried:
1. **Systematic audit of all services:**
• Checked every service in `/lib/systemd/system/` and `/etc/systemd/system/`
• Found and fixed all problematic dependencies
■ **This worked** - Comprehensive fix

2. **Source file fixes:**
• Fixed services in `moode-source/lib/systemd/system/` (not just SD card)
• Ensures fixes persist in future builds
■ **This worked** - Permanent solution

3. **Build script updates:**
• Modified `imgbuild/moode-cfg/stage3_03-ghettoblaster-custom_00-run-chroot.sh`
• Disabled redundant services during build
■ **This worked** - Clean builds from start

---

3. Network Configuration Fixes

What We Tried:
1. **Unified network service:**
• Created `00-unified-boot.service` for network + SSH
• Single service handles everything
■ **This worked** - No more conflicts

2. **Standardized IP addresses:**
• Changed all references to single IP (192.168.2.3)
• Updated all scripts and services
■ **This worked** - Consistent configuration

3. **NetworkManager detection:**
• Scripts check if NetworkManager is enabled
• Skip systemd-networkd if NetworkManager is active
■ **This worked** - Prevents conflicts

---

4. Display Configuration Fixes

What We Tried:
1. **Fixed cmdline.txt for portrait:**
```
video=HDMI-A-1:400x1280M@60
```
(Removed `rotate=90` - display is native portrait)
■ **This worked** - Correct resolution

2. **Fixed .xinitrc:**
• Moved `SCREEN_RES` after shebang
• Fixed if-else syntax errors
• Set correct rotation for portrait
■ **This worked** - X server starts correctly

3. **Updated moOde database:**
• Set `hdmi_scn_orient='portrait'` in default SQL
■ **This worked** - moOde knows it's portrait

---

5. Audio Chain Fixes

What We Tried:
1. **Comprehensive audio fix script:**
• Detects AMP100 card number
• Updates moOde database
• Creates/updates ALSA configs (`_audioout.conf`, `pcm.default`)
• Regenerates MPD config

■ **This worked** - Complete audio chain

2. **Routing logic:**
• CamillaDSP → PeppyMeter → Hardware
• Proper fallback chain
■ **This worked** - Flexible routing

---

6. Development Workflow Improvements

What We Tried:
1. **Docker-based testing:**
• Test fixes in Docker before applying to SD card
• Much faster iteration
■ **This worked** - Professional workflow

2. **Diagnostic scripts:**
• `check-pi-status.sh` - Comprehensive status
• `auto-check-pi-when-ready.sh` - Auto-detect Pi
■ **This worked** - Better debugging

3. **Location-independent scripts:**
• All scripts work from any directory
• Use `SCRIPT_DIR` and `WORKSPACE_ROOT`
■ **This worked** - User-friendly

■ Key Lessons Learned

1. Always Add Timeouts to Services

```
TimeoutStartSec=10
TimeoutStopSec=5
DefaultDependencies=no
```

Never create a service without timeouts. Services can hang forever otherwise.

2. Check Both Service Directories

• `/lib/systemd/system/` - System services
• `/etc/systemd/system/` - Custom services
Both need to be checked and fixed.

3. Fix Source Files, Not Just SD Card

• Fix files in `moode-source/` so fixes persist in builds
• Don't just patch the SD card

4. Test in Docker First

• Use Docker simulation before touching real hardware
• Much faster iteration
• Can test boot process without SD card

5. Remove SSH Host Keys After Re-flash

```
ssh-keygen -R <ip>
```
Always do this after re-flashing SD card.

6. One Service Per Function

• Don't create multiple services doing the same thing
• Use unified services instead

7. Never Claim "Root Cause" Without Verification

• Make changes, document them, wait for verification
• Don't claim fixes work until tested

■ Development Hints for Your Friend

Getting Started

1. **Understand the Build Process:**
• Base: moOde Audio Player (pi-gen based)
• Custom components: `custom-components/` directory
• Build script: `imgbuild/moode-cfg/stage3_03-ghettoblaster-custom_00-run-chroot.sh`
• Source files: `moode-source/` directory

2. **Key Directories:**
```
moodeaudio-cursor/
■■■ imgbuild/              # Build configuration
■■■ moode-source/          # Source files (copied into image)
■■■ custom-components/     # Custom services, scripts, overlays
■■■ tools/                 # Development and fix tools
■■■ docs/                  # Documentation
```

3. **Build Workflow:**
```
# 1. Make changes to source files
#    - moode-source/lib/systemd/system/*.service
#    - custom-components/services/*.service
#    - imgbuild/moode-cfg/stage3_03-ghettoblaster-custom_00-run-chroot.sh

# 2. Test in Docker (if possible)
./tools/test.sh --docker

# 3. Build image (if needed)
# (Build process - see moOde documentation)

# 4. Burn to SD card
./scripts/deployment/burn-v1.0-robust.sh
```

Common Development Tasks

Adding a New Systemd Service

1. **Create service file:**
```
# In: custom-components/services/my-service.service
[Unit]
Description=My Service
After=network.target
DefaultDependencies=no
[Service]
```

```
Type=oneshot
ExecStart=/usr/local/bin/my-script.sh
TimeoutStartSec=10
TimeoutStopSec=5
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

2. **Add to build script:**
```
# In: imgbuild/moode-cfg/stage3_03-ghettoblaster-custom_00-run-chroot.sh
cp "$CUSTOM_SERVICES/my-service.service" /lib/systemd/system/
systemctl enable my-service.service
```

3. **Always include:**
• `TimeoutStartSec` and `TimeoutStopSec`
• `DefaultDependencies=no` (if not needed)
• Non-blocking commands in `ExecStart`

Fixing Network Configuration

1. **Check for conflicts:**
```
# Is NetworkManager enabled?
systemctl is-enabled NetworkManager

# Is systemd-networkd enabled?
systemctl is-enabled systemd-networkd
```

2. **Update IP in all places:**
• `custom-components/scripts/fix-network-ip.sh`
• `moode-source/lib/systemd/system/00-unified-boot.service`
• `moode-source/lib/systemd/system/02-eth0-configure.service`
• Any other network scripts

3. **Test network:**
```
# On Pi after boot
ip addr show eth0
ping 8.8.8.8
```

Testing Changes

1. **Use Docker simulation:**
```
./tools/test.sh --docker
```

2. **Check service status:**
```
./tools/fix/check-pi-status.sh <IP>
```

3. **Monitor boot:**
```
./tools/fix/monitor-pi-boot.sh
```

Debugging Tips

1. **Check service dependencies:**
```
systemctl list-dependencies <service>
```

2. **Check service status:**
```
systemctl status <service> -l
```

3. **Check logs:**
```
journalctl -u <service> -n 50
```

4. **Check boot process:**
```
journalctl -b -p err
```

5. **Check for hanging services:**
```
systemctl list-jobs
```

Critical Rules

1. **NEVER create a service without timeouts**
2. **ALWAYS check both `/lib/systemd/system/` and `/etc/systemd/system/`**
3. **FIX source files, not just SD card**
4. **TEST in Docker before real hardware**
5. **ONE service per function - no redundancy**
6. **NEVER claim "root cause" without verification**

■■ Custom Build Process

Overview

The custom build is based on moOde Audio Player's pi-gen build system, with custom components integrated during the build process.

Build Stages

1. **Stage 0-2:** Base Raspberry Pi OS setup
2. **Stage 3:** moOde Audio installation
• `stage3_03-ghettoblaster-custom_00-run-chroot.sh` - Custom components
• `stage3_03-ghettoblaster-custom_02-display-cmdline.sh` - Display config

Custom Components Integration

Location: `imgbuild/moode-cfg/stage3*03-ghettoblaster-custom*00-run-chroot.sh`

What it does:
1. Creates user `andre` with UID 1000 (moOde requirement)
2. Installs custom systemd services
3. Compiles device tree overlays
4. Applies patches
5. Sets permissions
6. Configures network
7. Disables/enables services

Key Build Files

```
imgbuild/moode-cfg/
■■■ stage3_03-ghettoblaster-custom_00-run-chroot.sh  # Main custom integration
■■■ stage3_03-ghettoblaster-custom_02-display-cmdline.sh  # Display config

custom-components/
■■■ services/          # Systemd service files
■■■ scripts/           # Custom scripts
■■■ overlays/          # Device tree overlays
■■■ configs/          # Configuration templates

moode-source/
```

```
■■■ lib/systemd/system/  # Service files (copied to image)
■■■ etc/                 # Configuration files
■■■ home/                # User home files (.xinitrc, etc.)
■■■ var/                 # moOde database, web files
```

Making a Custom Build

1. **Modify source files:**
• Edit files in `moode-source/` or `custom-components/`
• Update build script if needed

2. **Test changes:**
• Use Docker simulation if possible
• Or test on SD card

3. **Build image:**
• Follow moOde build process
• Custom components are integrated automatically

4. **Burn to SD card:**
```
./scripts/deployment/burn-v1.0-robust.sh
```

■ Important Files Reference

Build Scripts

• `imgbuild/moode-cfg/stage3_03-ghettoblaster-custom_00-run-chroot.sh` - Main build script
• `imgbuild/moode-cfg/stage3_03-ghettoblaster-custom_02-display-cmdline.sh` - Display config

Service Files

• `moode-source/lib/systemd/system/*.service` - Services in image
• `custom-components/services/*.service` - Custom services

Configuration

• `moode-source/home/xinitrc.default` - X server startup
• `moode-source/var/local/www/db/moode-sqlite3.db.sql` - moOde database defaults
• `boot-config/cmdline.txt.example` - Kernel command line
• `boot-config/config.txt` - Raspberry Pi config

Tools

• `tools/fix/` - Fix and diagnostic scripts
• `tools/test.sh` - Test suite
• `scripts/deployment/` - SD card burning scripts

■ Recommendations for Future Development

1. **Use Docker workflow:**
• Test everything in Docker first

• Much faster iteration

2. **Comprehensive testing:**
• Test boot process
• Test all services
• Test network configuration
• Test audio chain

3. **Documentation:**
• Document all changes
• Keep session summaries
• Document problems and solutions

4. **Version control:**
• Use git for all changes
• Tag working versions
• Keep changelog

5. **Incremental changes:**
• Make small, testable changes
• Test after each change
• Don't make multiple changes at once

■ Final Notes

This project has been a learning experience in:
• Systemd service management
• Raspberry Pi boot process
• Network configuration
• Audio system integration
• Build system customization

The biggest takeaway: **Always add timeouts to services, and test in Docker before real hardware.**

Good luck with your development! ■

Document Version: 1.0
Last Updated: 2026-01-14
Project: moOde Audio Custom Build for Raspberry Pi 5