参考文献

Application Note

在 TI Sitara 器件上启用 PipeWire

Paresh Bhagat, Vishnu Singh

摘要

PipeWire 是一种现代、低延迟的多媒体框架，已成为 Linux 应用中音频和视频处理的标准，使用统一架构提供基于图形的处理和实时功能。本文档演示了如何在具有双核/四核 Arm Cortex-A53 处理器的德州仪器 (TI) Sitara 系列器件上启用 PipeWire。PipeWire 的多进程架构允许多个应用程序无缝共享多媒体内容，而不会发生冲突或资源争用。

本应用手册详细介绍了使用 PipeWire 集成、音频处理配置和性能基准测试构建基于 Yocto 的嵌入式 Linux 映像的过程。

目标应用：

- 专业音频设备
- 智能扬声器和条形音箱
- 汽车信息娱乐系统
- 具有多媒体功能的工业 HMI
- 具有音频功能的物联网设备

支持的平台：

- SK-AM62B-P1
- SK-AM62-LP
- AUDIO-AM62D-EVM
- TMDS62LEVM
- SK-AM62-SIP
- SK-AM62P-LP
- SK-AM62A-LP

内容

商标

所有商标均为其各自所有者的财产。

1 简介

Linux 音频环境历来呈现碎片化，不同的子系统用于满足不同的需求：

- 用于底层硬件访问的 ALSA
- 用于桌面音频的 PulseAudio
- 用于专业音频的 JACK
- 用于多媒体流水线的 GStreamer

这种碎片化给音频开发人员带来了挑战，不得不支持多个 API 并管理不同音频子系统之间的复杂路由。

PipeWire 是一种现代多媒体框架和基于图形的处理架构，旨在革新 Linux 系统中的音频和视频处理方式。它采用统一的设计，将 PulseAudio、JACK 和其他多媒体框架的功能整合到单个高效的处理引擎中。

1.1 主要亮点：

- 统一多媒体框架：适用于专业音频 (JACK)、消费类音频 (PulseAudio) 和视频处理的一站式解决方案。
- 低延迟性能。
- 安全模型：内置沙盒，支持容器化应用程序。
- 多进程架构，让应用程序共享多媒体内容。
- 对音频和视频进行实时多媒体处理。

1.2 基本概念：

1.2.1 PipeWire 服务器

该服务器是核心守护程序，用于管理基于图形的多媒体处理引擎。它负责处理媒体图形的创建和执行，音频、视频或 MIDI 数据在不同的处理组件之间流动。

1.2.2 PipeWire 客户端

客户端是连接到 PipeWire 服务器以生成或使用媒体流的应用程序。它们在媒体图形中创建节点来发送或接收音频或视频数据。

1.2.3 会话管理器

PipeWire 不单独处理设备路由或策略决策。这些任务由会话管理器管理，会话管理器会监视设备并自动连接各个流。在本文档中，WirePlumber 用作会话管理器。

1.2.4 节点、端口和链接

PipeWire 处理图由节点、端口和链路组成。节点表示处理元件，端口充当输入或输出接口，并链接节点之间的端口以允许媒体数据流经图形。

1.3 PipeWire 主要组件

- PipeWire 守护程序，用于实现 IPC 和图形处理。
- 示例 PipeWire 会话管理器，用于管理 PipeWire 守护程序中的对象。
- 一组程序，用于自省和使用 PipeWire 守护程序。
- PipeWire 库，用于开发 PipeWire 应用程序和插件。
- PipeWire 守护程序和 PipeWire 库中使用的 SPA (Simple Plugin API)。

2 Linux 音频栈

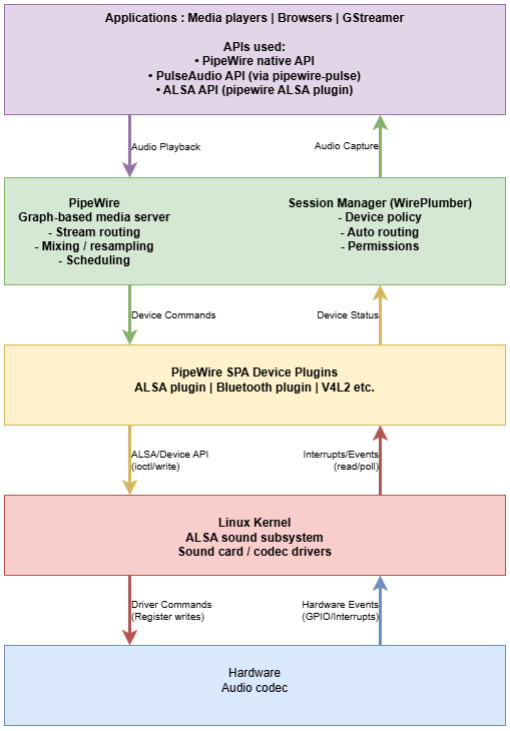

图 2-1. Linux 音频栈

- **应用程序**（媒体播放器、浏览器、基于 GStreamer 的工具等）会生成音频流并通过本机 PipeWire API 或其他系统（Jack、PulseAudio、ALSA）的兼容层进行通信。
- **PipeWire 守护程序**会接收这些音频流并构造一个处理图形，以同时处理来自多个来源的音频混合、路由和调度。
- **会话管理器** (WirePlumber) 会应用策略决策，包括设备选择、流路由规则以及应用程序和硬件端点之间的自动连接管理。
- **SPA (Simple Plugin API) 插件**提供硬件抽象。ALSA 插件专门与内核的 ALSA 子系统连接，以访问物理音频设备。
- **内核驱动程序**与实际硬件层（音频编解码器、声卡、I2S 总线、HDMI 音频接口）进行通信，以生成物理音频输出或采集输入。

3 通过 Yocto 构建支持 PipeWire 的 SDK 映像

本节提供了通过 Yocto 构建支持 PipeWire 的可擦写映像的步骤。Yocto Project 是一个开源协作项目，用于帮助开发人员创建基于 Linux 的自定义系统，而不考虑硬件架构。处理器 Linux SDK 构建版基于 Arago 项目，该项目为 OpenEmbedded 提供多个层和针对 TI 平台的 Yocto 项目。

3.1 在主机上运行 Yocto 构版建的步骤

3.1.1 必要条件（一次性设置）

推荐的 Linux 发行版是 Ubuntu 22.04。以下命令将在 Ubuntu Linux 发行版中安装所需的工具。

```
$ sudo apt-get update

# Install packages required for builds
$ sudo apt-get -f -y install \
    git build-essential diffstat texinfo gawk chrpath socat doxygen \
    dos2unix python3 bison flex libssl-dev u-boot-tools mono-devel \
    mono-complete curl python3-distutils repo pseudo python3-sphinx \
    g++-multilib libc6-dev-i386 jq git-lfs pigz zstd liblz4-tool \
    cpio file lz4 debianutils iputils-ping python3-git python3-jinja2 \
    python3-subunit locales libacl1 unzip gcc python3-pip \
    python3-pexpect xz-utils wget \

$ sudo locale-gen en_US.UTF-8
```

默认情况下，Ubuntu 使用 *dash* 作为 /bin/sh 的默认 shell。通过运行以下命令，重新配置为使用 bash：

```
$ sudo dpkg-reconfigure dash
```

3.2 克隆 oe 层设置

```
$ cd $HOME

$ git clone https://git.ti.com/git/arago-project/oe-layersetup.git tisdk

$ cd tisdk

$ ./oe-layertool-setup.sh -f configs/processor-sdk/processor-sdk-master-12.00.00.07.04-config.txt
```

3.3 下载并应用 PipeWire 补丁

需要补丁才能在映像中启用 PipeWire 和 WirePlumber 支持。这些补丁需要应用于：

- meta-arago - 用于 Sitara 器件 Arago 分布配置的 Yocto 层。
- meta-tisdk — 用于 Sitara 器件的 TI 基础 SDK 的 Yocto 层。

下表介绍了这两个层所需的补丁：

表 3-1. Yocto PipeWire 补丁

补丁编号	补丁	说明
1	0001-recipes-multimedia-Add-pipewire-configuration-files.patch	为 8 通道和 2 通道音频添加参考 PipeWire 配置文件。
2	0002-recipes-multimedia-Add-wireplumber-audio-configurati.patch	使用音频默认服务添加 WirePlumber 配置。
3	0003-recipes-core-arago-default-image-Add-pipewire-audio-.patch	在 Arago-default-image 中启用 PipeWire 音频栈。
4	0001-ti-apps-launcher-Remove-pulseaudio-service-dependenc.patch	对于在 OOB 演示中集成了 PulseAudio 的 EVM，从映像中删除 PulseAudio 服务。

以下是下载和应用补丁的步骤。

```
$ cd $HOME
```

```
$ git clone https://github.com/TexasInstruments/Beyond-SDK.git -b main

$ cd $HOME/tisdk/sources/meta-arago

$ git am $HOME/Beyond-SDK/collaterals/appnotes/sdaa320-Enable_PipeWire_on_TI_Sitara_Devices/0001-recipes-multimedia-Add-pipewire-configuration-files.patch

$ git am $HOME/Beyond-SDK/collaterals/appnotes/sdaa320-Enable_PipeWire_on_TI_Sitara_Devices/0002-recipes-multimedia-Add-wireplumber-audio-configurati.patch

$ git am $HOME/Beyond-SDK/collaterals/appnotes/sdaa320-Enable_PipeWire_on_TI_Sitara_Devices/0003-recipes-core-arago-default-image-Add-pipewire-audio-.patch

$ cd $HOME/tisdk/sources/meta-tisdk

$ git am $HOME/Beyond-SDK/collaterals/appnotes/sdaa320-Enable_PipeWire_on_TI_Sitara_Devices/0001-ti-apps-launcher-Remove-pulseaudio-service-dependenc.patch
```

3.4 构建 PipeWire 映像

下面的最后一个命令会构建 tisdk-default-image，这是启用了 Arago 文件系统和 PipeWire 支持的 Processor SDK 映像。

```
$ cd $HOME/tisdk

$ cd build

$ . conf/setenv

# For RT (Real Time) Linux build
$ MACHINE=<machine> ARAGO_RT_ENABLE=1 bitbake -k tisdk-default-image

# For Non-RT Linux Build
$ MACHINE=<machine> bitbake -k tisdk-default-image
```

对于延迟敏感型应用，建议使用 RT Linux 版本。

其中 MACHINE 为以下数值之一：

表 3-2. MACHINE 数值

MACHINE	支持的 EVM
am62xx-evm	SK-AM62B-P1
am62xx-lp-evm	SK-AM62-LP
am62dxx-evm	AUDIO-AM62D-EVM
am62lxx-evm	TMDS62LEVM
am62xxsip-evm	SK-AM62-SIP
am62pxx-evm	SK-AM62P-LP
am62axx-evm	SK-AM62A-LP

最终的 wic 映像在 ***deploy-ti/images/<machine>/*** 目录中生成。

4 在 Sitara 器件上设置 PipeWire

本指南以 SK-AM62B-P1、TMDS62LEVM 和 AUDIO-AM62D-EVM 为例，演示如何构建 SDK 映像及随后如何配置和使用 PipeWire。

4.1 硬件

4.1.1 SK-AM62B-P1

SK-AM62B-P1 开发套件通过 HDMI 和 LVDS 支持双显示器，而且具有一整套工业通信接口，因此非常适合 HMI、PLC 和自动化应用。

- SK-AM62B-P1
- Micro-SD 卡（最小 32GB）
- USB Type-C 供电 (20W)
- USB 转 UART 线缆
- 用于刷写和控制台访问的 Windows 或 Linux 主机 PC
- 带 3.5mm 插孔的耳机

4.1.2 TMDS62LEVM

TMDS62LEVM 评估模块为使用 AM62L 系列应用处理器进行开发提供了一个低成本平台，提供可扩展的性能、丰富的嵌入式功能、广泛的连接功能以及用于电源和热管理的工具。

- TMDS62LEVM
- Micro-SD 卡（最小 32GB）
- USB Type-C 供电
- USB 转 UART 线缆
- 用于刷写和控制台访问的 Windows 或 Linux 主机 PC
- 带 3.5mm 插孔的耳机

4.1.3 AUDIO-AM62D-EVM

AUDIO-AM62D-EVM 评估模块 (EVM) 是一个低成本的可扩展平台，专为开发人员设计，用于对各种应用场景中的多通道音频应用进行原型设计和评估。

- AUDIO-AM62D-EVM
- Micro-SD 卡（最小 32GB）
- USB Type-C 供电
- USB 转 UART 线缆
- 用于刷写和控制台访问的 Windows 或 Linux 主机 PC
- 音频输出设备（扬声器，TRS 兼容）
- 音频输入设备（麦克风，TRS 兼容）

4.2 配置 EVM 引导模式

4.2.1 SK-AM62B-P1

- 图 4-1 显示了一些重要的线缆连接、端口和交换机。
- 请注意 SD 卡引导模式的“BOOTMODE”开关的位置。
- 设置 EVM SD 卡引导模式：
  - BOOTMODE [8 : 15] (SW2) = 0100 0000
  - BOOTMODE [0 : 7] (SW1) = 1100 0010
- 有关更多详细信息，请参阅快速入门指南。

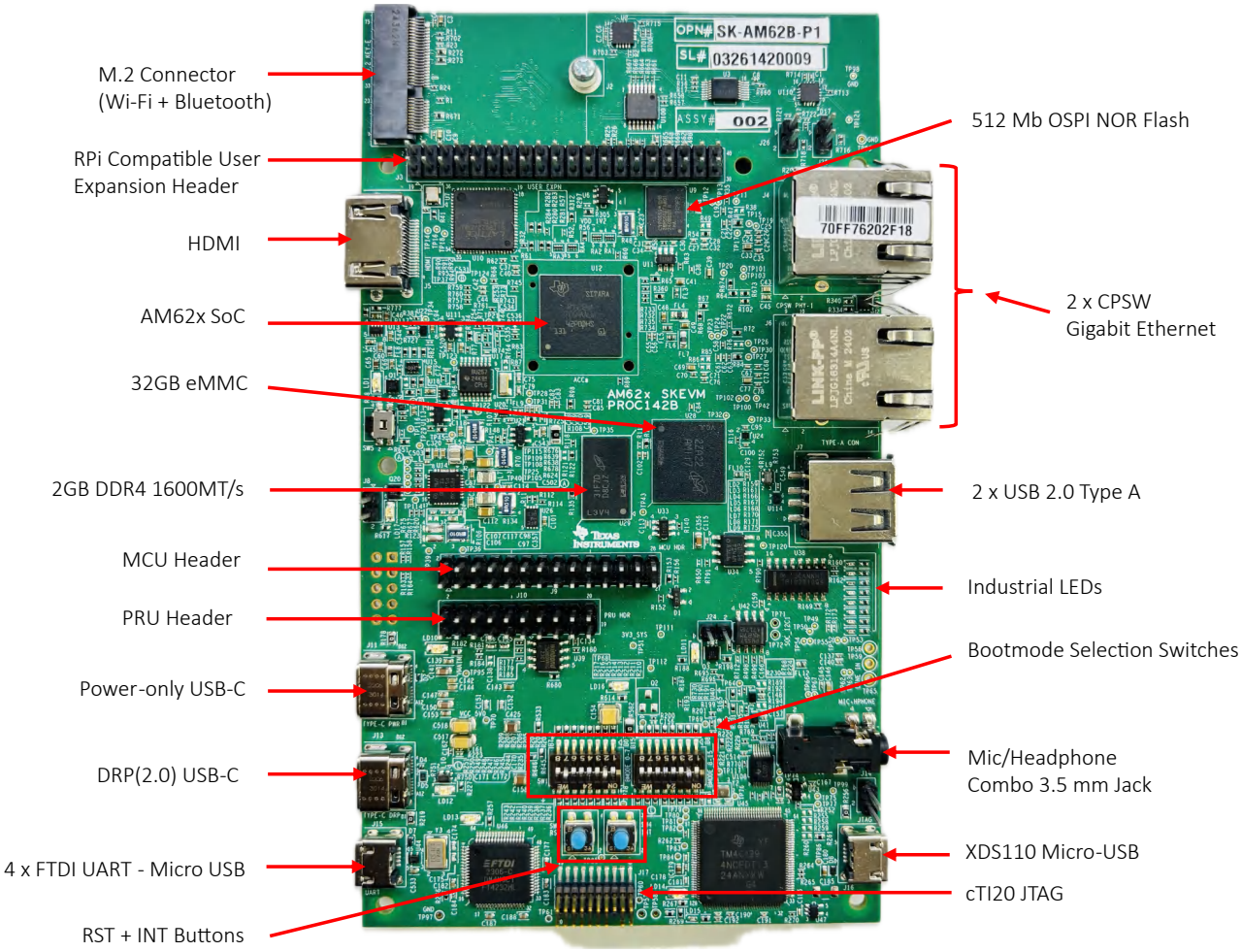

图 4-1. SK-AM62B-P1

4.2.2 TMDS62LEVM

- 图 4-2 显示了一些重要的线缆连接、端口和交换机。
- 请注意 SD 卡引导模式的 *BOOTMODE* 开关的位置。
- 设置 EVM SD 卡引导模式：
  - BOOTMODE [8 : 11] (SW2) = 0100
  - BOOTMODE [11: 15] (SW3) = 0000
  - BOOTMODE [0 : 7] (SW4) = 1100 0010
- 有关更多详细信息，请参阅快速入门指南。

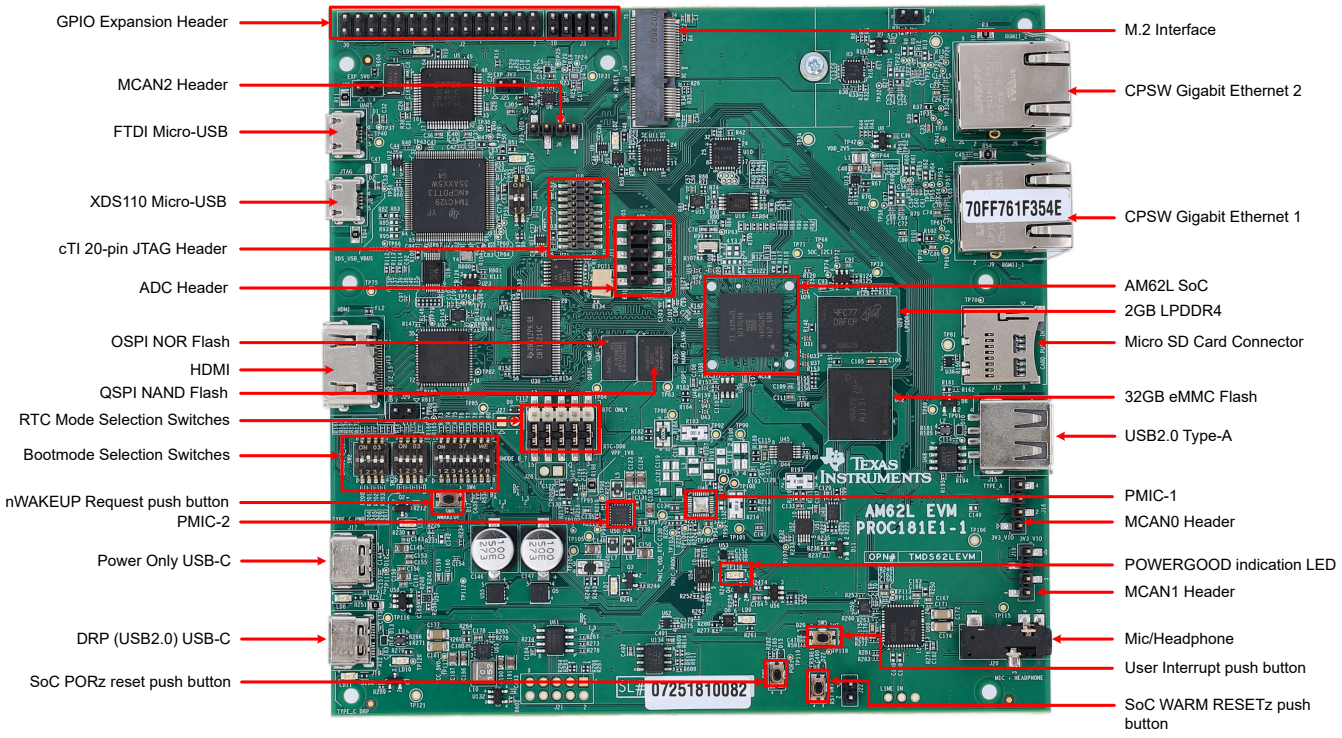

图 4-2. TMDS62LEVM

4.2.3 AUDIO-AM62D-EVM

- 下面的图 4-3 显示了一些重要的线缆连接、端口和交换机。
- 请注意 SD 卡引导模式的“BOOTMODE”开关的位置。
- 设置 EVM SD 卡引导模式：
  - BOOTMODE [8 : 15] (SW1) = 0100 0000
  - BOOTMODE [0 : 7] (SW2) = 1100 0010

- 有关更多详细信息，请参阅快速入门指南。

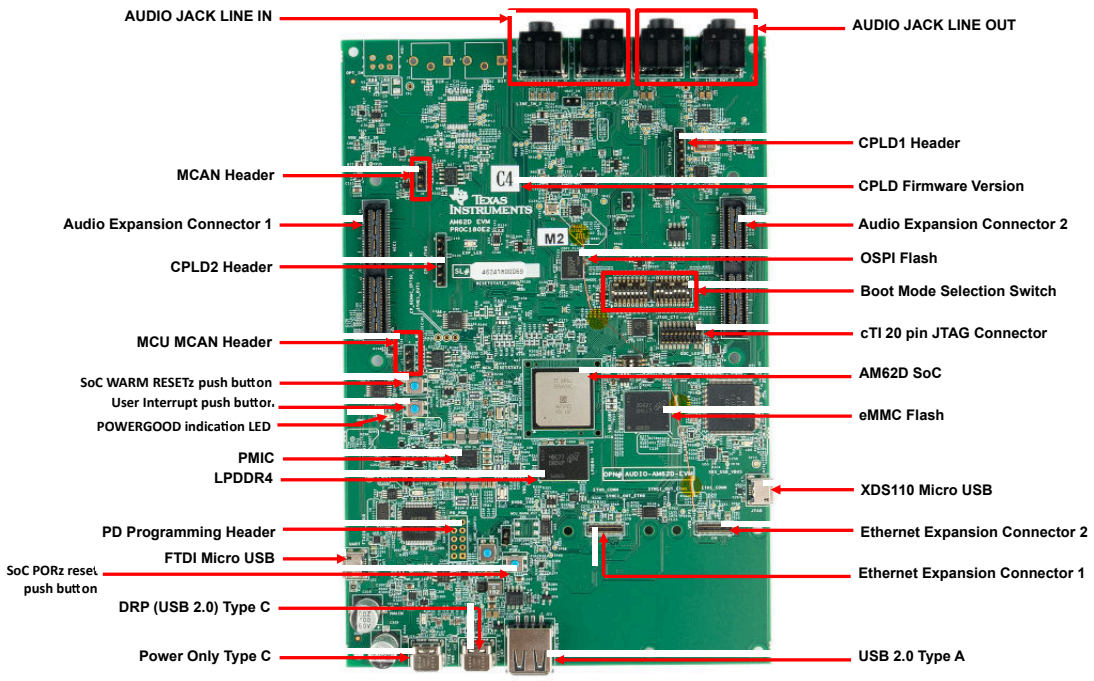

Figure 2-1. AM62D Audio EVM Top Side

SoC WARM RESETz push button

图 4-3. AUDIO-AM62D-EVM

4.3 UART 控制台设置

- 将 UART 转 USB 线缆连接到 EVM。
- 识别主机上枚举的 UART COM 端口（Windows 设备管理器端口 (COM 和 LPT)）。
- 如果设备管理器中的端口 (COM & LPT) 下没有列出 USB 串行端口，请从 FTDI 安装 UART 转 USB 驱动程序。
- 终端设置
  - 波特率：115200
  - 8-N-1
- 打开终端并等待器件引导。

4.4 刷写 SD 卡映像

- 支持 PipeWire 的默认 wic 映像将在 ***deploy-ti/<machine>/images*** *中生成，名称为* ***tisdk-default-image-<machine>-evm.rootfs.wic.xz***。
- 使用 Balena Etcher 刷写 wic 映像
  - 将 micro-SD 卡插入 USB SD 读卡器并启动 Etcher。
  - 选择 wic.xz 映像
  - 选择 SD 卡
  - 点击*刷写*

4.5 使用 SD 卡引导 EVM

- 确保电路板上的引导模式引脚用于 SD 卡引导。
- 将 SD 卡插入 SD 卡插槽。
- 将主机 PC 连接到 USB Micro-B 接口，以访问系统控制台和查看 UART 日志。
- 为电路板上电。
- 以“root”身份登录，不使用密码。

```
Trying to boot from MMC2
Authentication passed
Authentication passed
Authentication passed
Authentication passed
Authentication passed
Starting ATF on ARM64 core...
.
.
.
Arago Project <machine> -
Arago 2025.01 <machine> -
<machine> login:
```

5 使用 PipeWire

5.1 检查服务状态

```
root@<machine>: systemctl status pipewire

root@<machine>: systemctl status wireplumber
```

5.2 启用 PipeWire 和 Wireplumber

PipeWire 和 WirePlumber 旨在作为用户空间服务运行。在单个用户 Arago 映像（仅限 ***root***）中，可以将 PipeWire 作为系统服务运行。但是，在多用户映像（例如一些 Sitara 中的 ***weston***），则强烈建议启用每用户会话以隔离多媒体访问，避免与 D-Bus 会话实例冲突，ALSA 器件保留，并防止用户会话之间的权限不匹配。

使服务在引导时自动启动

```
root@<machine>: systemctl enable pipewire

root@<machine>: systemctl enable wireplumber
```

为了简化 AUDIO-AM62D-EVM 的设置，补丁中包含一项新服务，可自动在 WirePlumber 中设置默认音频器件。要启用它，请使用以下命令：

```
root@<machine>: systemctl enable set-audio-defaults
```

5.3 启动 PipeWire 和 WirePlumber

启动 PipeWire 和 WirePlumber（如果未启用）

```
root@<machine>: systemctl start pipewire

root@<machine>: systemctl start wireplumber

# For AUDIO-AM62D-EVM
root@<machine>: systemctl start set-audio-defaults
```

5.4 常规 PipeWire 命令

5.4.1 列出 PipeWire 服务器中当前的所有对象

使用 ***pw-cli list-objects*** 命令列出所有对象。

```
root@<machine>: pw-cli list-objects
id 0, type PipeWire:Interface:Core/4
object.serial = "0"
core.name = "pipewire-0"
id 1, type PipeWire:Interface:Module/3
object.serial = "1"
module.name = "libpipewire-module-rt"
id 2, type PipeWire:Interface:Module/3
object.serial = "2"
module.name = "libpipewire-module-protocol-native"
id 3, type PipeWire:Interface:SecurityContext/3
object.serial = "3"
id 4, type PipeWire:Interface:Module/3
object.serial = "4"
module.name = "libpipewire-module-profiler"
id 5, type PipeWire:Interface:Profiler/3
object.serial = "5"
```

5.4.2 仅列出节点

使用 ***pw-cli list-objects Node*** 列出所有节点

```
root@<machine>: pw-cli list-objects Node
id 29, type PipeWire:Interface:Node/3
object.serial = "29"
factory.id = "11"
priority.driver = "200000"
node.name = "Dummy-Driver"
id 30, type PipeWire:Interface:Node/3
object.serial = "30"
factory.id = "11"
priority.driver = "190000"
node.name = "Freewheel-Driver"
id 31, type PipeWire:Interface:Node/3
object.serial = "31"
factory.id = "19"
node.description = "Audio Output"
node.name = "alsa_audio_sink"
media.class = "Audio/Sink"
id 32, type PipeWire:Interface:Node/3
object.serial = "32"
factory.id = "19"
node.description = "Audio Input"
node.name = "alsa_audio_source"
media.class = "Audio/Source"
```

5.4.3 检查特定对象

使用 ***pw-cli info <object-id>*** 命令检查特定对象。

```
root@<machine>: pw-cli info 31
id: 31
permissions: rwxm-
type: PipeWire:Interface:Node/3
* input ports: 0/0
* output ports: 8/129
* state: "suspended"
* properties:
* factory.name = "api.alsa.pcm.source"
* node.name = "alsa_audio_source"
* node.description = "Audio Input"
* media.class = "Audio/Source"
* api.alsa.period-size = "1024"
* node.driver = "true"
* api.alsa.disable-mmap = "false"
* api.alsa.disable-batch = "false"
* api.alsa.path = "hw:0,0"
* audio.rate = "48000"
* audio.channels = "8"
```

5.5 播放和录制立体声音频

通过 ***pw-play*** 或 ***aplay*** 播放音频：

```
root@<machine>: pw-play --target=alsa_audio_sink <path to wav file>

root@<machine>: aplay -r 48000 -f S32_LE -c 2 <path to wav file>
```

通过 ***pw-record*** 或 ***arecord*** 录制音频：

```
root@<machine>: pw-record --target=alsa_audio_source record_stereo.wav

root@<machine>: arecord -r 48000 -f S32_LE -c 2 record_stereo.wav
```

6 配置

PipeWire 设置包括：

- PipeWire 守护程序
- 会话管理器（通常为 WirePlumber）
- 兼容性服务器（PulseAudio、JACK）
- 客户端配置

其中每一个都有自己的配置文件，配置文件使用 SPA JSON 格式，这是一种宽松 JSON 语法。

表 6-1. 配置文件

文件	用途
pipewire.conf	配置 PipeWire 守护程序
client.conf	配置 PipeWire 客户端
pipewire-pulse.conf	PulseAudio 兼容型服务器
filter-chain.conf	音频处理滤波器

PipeWire 建议不要修改主文件，而是使用直接插入式文件来覆盖特定的设置。使用以下目录中的示例：

```
/etc/pipewire/pipewire.conf.d/
```

6.1 接收端和源端配置

为 AUDIO-AM62D-EVM 添加了两个参考配置文件，即接收端和源端，可在 3.5mm 插孔中提供 8 通道支持。

对于其他仅需要两个通道的 Sitara EVM，无需额外配置。但是，基准配置可以根据需要修改采样率、通道数、周期大小和其他参数。

90-pipewire-sink.conf

```
# PipeWire sink configuration for AM62D.
context.objects = [
    {
        factory = adapter
        args = {
            factory.name = api.alsa.pcm.sink
            node.name = "alsa_audio_sink"
            node.description = "Audio Output"
            media.class = "Audio/Sink"
            api.alsa.period-size = 1024
            api.alsa.headroom = 0
            api.alsa.disable-mmap = false
            api.alsa.disable-batch = false
            api.alsa.path = "hw:AM62D2EVM,0"
            audio.rate = 48000
            audio.channels = 8
            audio.position = [ FL FR FC LFE RL RR SL SR ]
        }
    }
]
```

91-pipewire-source.conf

```
# PipeWire source configuration for AM62D.
context.objects = [
    {
        factory = adapter
        args = {
            factory.name = api.alsa.pcm.source
            node.name = "alsa_audio_source"
            node.description = "Audio Input"
            media.class = "Audio/Source"
            api.alsa.period-size = 1024
            api.alsa.headroom = 0
            api.alsa.disable-mmap = false
```

```
            api.alsa.disable-batch = false
            api.alsa.path = "hw:AM62D2EVM,0"
            audio.rate = 48000
            audio.channels = 8
            audio.position = [ FL FR FC LFE RL RR SL SR ]
        }
    }
]
```

context.objects

主配置数组，用于定义在 PipeWire 上下文中创建的对象。该数组中的每个对象都成为 PipeWire 图形中的一个节点。

factory = adapter

指定使用“adapter”工厂创建此对象。PipeWire 中的适配器用于桥接不同的 API (在本例中为 ALSA 到 PipeWire) 。两个配置文件都使用适配器工厂将 ALSA 硬件桥接至 PipeWire 节点。

factory.name

确定方向：输出还是输入

node.name

内部 PipeWire 节点标识符。创建 alsa_audio_sink 用于播放，创建 alsa_audio_source 用于采集。

node.description

音频应用程序中的人类可读名称。

media.class

PipeWire 媒体分类“Audio/Sink”用于播放，“Audio/Source”用于录制。

api.alsa.path

用于直接硬件访问。只有 PipeWire 才能访问音频硬件，ALSA 应用程序必须经由 PipeWire。

audio.channels

对于 AUDIO-AM62D-EVM，可配置 8 通道音频的输入和输出。

这些配置会在 PipeWire 的音频图形中创建两个基本节点：

- 接收端节点：用于 8 通道音频播放的终端端点
- 源端节点：用作 8 通道音频采集的起点

有关更多信息，请参阅 Alsa 配置。

6.2 WirePlumber 配置

使用 ***wpctl status*** 命令，列出所有可用的音频接收端和源端。

```
root@<machine>: wpctl status
PipeWire 'pipewire-0' [1.6.0, root@am62dxx-evm, cookie:3333499771]
└─ Clients:
34. WirePlumber [1.6.0, root@am62dxx-evm, pid:9716]
58. WirePlumber [export] [1.6.0, root@am62dxx-evm, pid:9716]
94. wpctl [1.6.0, root@am62dxx-evm, pid:9753]
Audio
├─ Devices:
│  59. Built-in Audio [alsa]
│
├─ Sinks:
│  * 31. Audio Output [vol: 1.00]
│  68. Built-in Audio Stereo [vol: 0.40]
│
├─ Sources:
│  * 32. Audio Input [vol: 1.00]
│  69. Built-in Audio Stereo [vol: 1.00]
│
├─ Filters:
│
└─ Streams:
Video
├─ Devices:
│
├─ Sinks:
│
├─ Sources:
│
├─ Filters:
│
└─ Streams:
Settings
└─ Default Configured Devices:
0. Audio/Sink alsa_audio_sink
1. Audio/Source alsa_audio_source
```

使用 ***wpctl Inspect <id>*** ，显示有关指定对象的信息。

```
root@<machine>: wpctl inspect 31
```

使用接收端和源端的 ID 数量，手动设置默认源端：

```
root@<machine>: wpctl set-default 30

root@<machine>: wpctl set-default 31
```

WirePlumber 变更通过以下两个关键文件，为 AUDIO-AM62D-EVM 引入了自动音频器件配置系统：

- ***set-audio-defaults.sh***

此脚本实现了一个初始化序列，等待长达 30 秒 WirePlumber 准备就绪，然后使用 PipeWire 的命令行工具（pw-cli 和 wpctl）找到 PipeWire 配置文件中定义的 alsa_audio_sink 和 alsa_audio_source 节点，并将它们明确设置为系统的默认音频设备。这种自动化是必要的，因为当 PipeWire 根据配置文件创建音频节点时，它不会自动将它们指定为应用程序将使用的默认设备。

- ***set-audio-defaults.service***

随附的 systemd 服务确保 set-audio-defaults.sh 脚本在 WirePlumber 启动后自动运行。

> 默认情况下，由于 set-audio-defaults.service 的原因，AUDIO-AM62D-EVM 使用 alsa_AUDIO_sink 和 alsa_AUDIO_source。wpctl 命令可用于更改到其他器件（例如 USB）的音频路由。

7 性能基准测试

测试设置如下所示：

- 硬件
  - SK-AM62B-P1
  - TMDS62LEVM
  - AUDIO-AM62D-EVM
- 内核 - ti-linux-6.18.y - 12.00.00.07
- Yocto - 12.00.00.07.04

将同时启动多个音频应用程序，以测量和比较 PulseAudio 和 PipeWire 的平均性能。

PulseAudio 已打包用于所有 Sitara 平台。但是，由于本文档中提到的补丁，服务文件未打包。要将 PulseAudio 作为后台进程启动，请使用以下命令：

```
root@<machine>: pulseaudio --daemonize
```

使用以下命令停止 PulseAudio

```
root@<machine>: pulseaudio --kill
```

7.1 延迟

该测试可测量 PulseAudio 和 PipeWire 的播放延迟。

对于 PulseAudio，延迟是使用 ***pactl*** 报告的接收端延迟来测量的，反映了在音频输出中观察到的有效播放延迟。

```
root@<machine>: pactl list sinks | grep Latency
Latency: 1370744 usec, configured 1837500 usec
```

对于 PipeWire，对通过 ***pw-top*** 命令获得的活动流和接收节点的缓冲区持续时间（时间片）求和来估算延迟，表示应用程序和器件缓冲。

```
root@<machine>: pw-top
S ID QUANT RATE WAIT BUSY W/Q B/Q ERR FORMAT NAME
S 29 0 0 --- --- --- --- 0 Dummy-Driver
S 30 0 0 --- --- --- --- 0 Freewheel-Driver
R 31 2048 48000 19.1us 504.7us 0.00 0.01 0 S32LE 8 48000 alsa_audio_sink
R 77 4800 48000 28.9us 125.1us 0.00 0.00 0 S16LE 8 48000 = pw-play
S 32 0 0 --- --- --- --- 0 alsa_audio_source
S 68 0 0 --- --- --- --- 0 alsa_output.platform-sound.stereo-fallback
S 69 0 0 --- --- --- --- 0 alsa_input.platform-sound.stereo-fallback
```

使用下方公式，根据时间片和速率计算延迟：

$$\textbf{Latency(ms)} = \frac{\textbf{quantum}}{\textbf{rate}} \times \textbf{1000} \quad (1)$$

表 7-1. 默认延迟

器件	音频服务器	延时 (ms)
SK-AM62B-P1	PulseAudio	大约 1837ms
	PipeWire	大约 143ms
TMDS62LEVM	PulseAudio	大约 1837ms
	PipeWire	大约 143ms
AUDIO-AM62D-EVM	PulseAudio	大约 1837ms
	PipeWire	大约 143ms

通过更改配置（例如时间片、时钟速率、片段数量、片段大小等），可以进一步减小这些延迟值。

7.2 CPU 和内存使用情况

此测试可测量当多个音频流同时使用音频服务器时的 CPU 利用率和内存使用情况。通过更改 PULSE_LATENCY_MSEC，将 PulseAudio 配置为以接近 143ms（PipeWire 的延迟）的延迟工作。

```
root@<machine>: export PULSE_LATENCY_MSEC=325

root@<machine>: pactl list sinks | grep Latency
Latency: 141702 usec, configured 142500 usec
```

在 PulseAudio 中，无法使用 PULSE_LATENCY_MSEC 将延迟设置为准确的值，因为它被视为目标而不是严格配置。因此，观察到的延迟可能与请求值有很大差异。实际延迟由内部缓冲区碎片、硬件限制和调度器行为决定。

还有其他方法可以控制延迟，例如直接修改影响延迟的变量（例如片段、片段大小等）。

有关更多详细信息，请参阅 Pulseaudio 文档。

表 7-2. CPU 负载（相同延迟）

器件	音频服务器	CPU 利用率（平均）
SK-AM62B-P1	PulseAudio	约 5%
	PipeWire	约 1%
TMDS62LEVM	PulseAudio	约 6%
	PipeWire	约 1%
AUDIO-AM62D-EVM	PulseAudio	约 5%
	PipeWire	约 1%

表 7-3. 内存使用情况（相同延迟）

器件	音频服务器	内存使用情况（平均）
SK-AM62B-P1	PulseAudio	约 22500KB
	PipeWire	约 46570KB（Wireplumber - 约 22910KB）
TMDS62LEVM	PulseAudio	约 25000KB
	PipeWire	大约 33400KB（Wireplumber - 约 30190KB）
AUDIO-AM62D-EVM	PulseAudio	约 21600KB
	PipeWire	大约 55000KB（Wireplumber - 约 22990KB）

备注

由于多进程设计，核心守护程序和 WirePlumber 会话管理器独立运行，因此 PipeWire 使用的内存更多。这种分离提供了故障隔离：如果 WirePlumber 崩溃，音频播放将不会中断。PulseAudio 的单进程设计具有更高的内存效率，但对组件故障的恢复能力较差。

7.3 重新采样后的 CPU 和内存使用情况

此测试可测量当音频服务器执行采样率转换并以相同的延迟进行配置时引入的 CPU 开销和内存使用情况。此测试会评估重采样的效率。PulseAudio 和 PipeWire 均配置为将音频重新采样为 44.1kHz（对比原生 48kHz）。

对于 PulseAudio，使用以下命令重新采样为 44.1kHz：

```
root@<machine>: echo "default-sample-rate = 44100" >> /etc/pulse/daemon.conf
root@<machine>: echo "alternate-sample-rate = 44100" >> /etc/pulse/daemon.conf
```

对于 PipeWire，使用以下命令重新采样为 44.1kHz（仅适用于当前会话）

```
root@<machine>: pw-metadata -n settings 0 clock.force-rate 44100
```

要进行永久配置，请使用以下内容创建新的自定义 ***/etc/pipewire/pipewire.conf.d/99-custom.conf***

```
context.properties = {
    default.clock.rate = 44100
    default.clock.allowed-rates = [ 44100 ]
}
```

表 7-4. 重新采样后的 CPU 利用率

器件	音频服务器	CPU 利用率（平均）
SK-AM62B-P1	PulseAudio	约 24%
	PipeWire	约 3%
TMDS62LEVM	PulseAudio	约 29%
	PipeWire	约 3%
AUDIO-AM62D-EVM	PulseAudio	约 23%
	PipeWire	约 3%

表 7-5. 重新采样后的内存使用情况

器件	音频服务器	内存使用情况（平均）
SK-AM62B-P1	PulseAudio	约 21600KB
	PipeWire	约 52750KB (Wireplumber - 40000)
TMDS62LEVM	PulseAudio	约 25500KB
	PipeWire	大约 33310KB (Wireplumber - 33800)
AUDIO-AM62D-EVM	PulseAudio	约 21500KB
	PipeWire	大约 32250KB (Wireplumber - 23000)

7.4 观察结果

基准测试结果在很大程度上与平台无关，因为测试的工作负载（5 个并发音频流）相对轻量化，不会显著影响 CPU、多核扩展和 DDR 带宽。因此，CPU 频率、内核数和 DDR 速度的差异对 CPU 利用率、延迟和内存占用的影响极小。由于调度器时序、后台活动、缓存等因素，观察到的微小差异可能属于正常的批次间波动。

8 总结

本文档提供了在 TI Sitara 器件上使用 PipeWire 的全面指南，涵盖从构建 Yocto 映像到配置音频框架并进行基准测试的各个方面。这些基准测试使用 SK-AM62B-P1、TMDS62LEVM 和 AUDIO-AM62D-EVM 作为参考平台，演示了 PipeWire 能够作为嵌入式应用的低延迟、高性能音频设计。

9 参考资料

1. PipeWire，*PipeWire 文档*，网页。
2. Pulseaudio，*Pulseaudio 文档*，网页。
3. 德州仪器 (TI)，*SK-AM62B-P1* 产品页面。
4. 德州仪器 (TI)，*SK-AM62B-P1 Processor SDK 文档*，软件开发指南。
5. 德州仪器 (TI)，*SK-AM62B-P1 快速入门指南* 快速入门指南。
6. 德州仪器 (TI)，*TMDS62LEVM* 产品页面。
7. 德州仪器 (TI)，*TMDS62LEVM Processor SDK 文档*，软件开发指南。
8. 德州仪器 (TI)，*TMDS62LEVM 快速入门指南* 快速入门指南。
9. 德州仪器 (TI)，*AUDIO-AM62D-EVM* 产品页面。
10. 德州仪器 (TI)，*AUDIO-AM62D-EVM Processor SDK 文档*，软件开发指南。
11. 德州仪器 (TI)，*AUDIO-AM62D-EVM 快速入门指南* 快速入门指南。

10 重要声明和免责声明

TI “按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款、TI 通用质量准则或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。除非 TI 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

上次更新日期：2026 年 6 月

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月