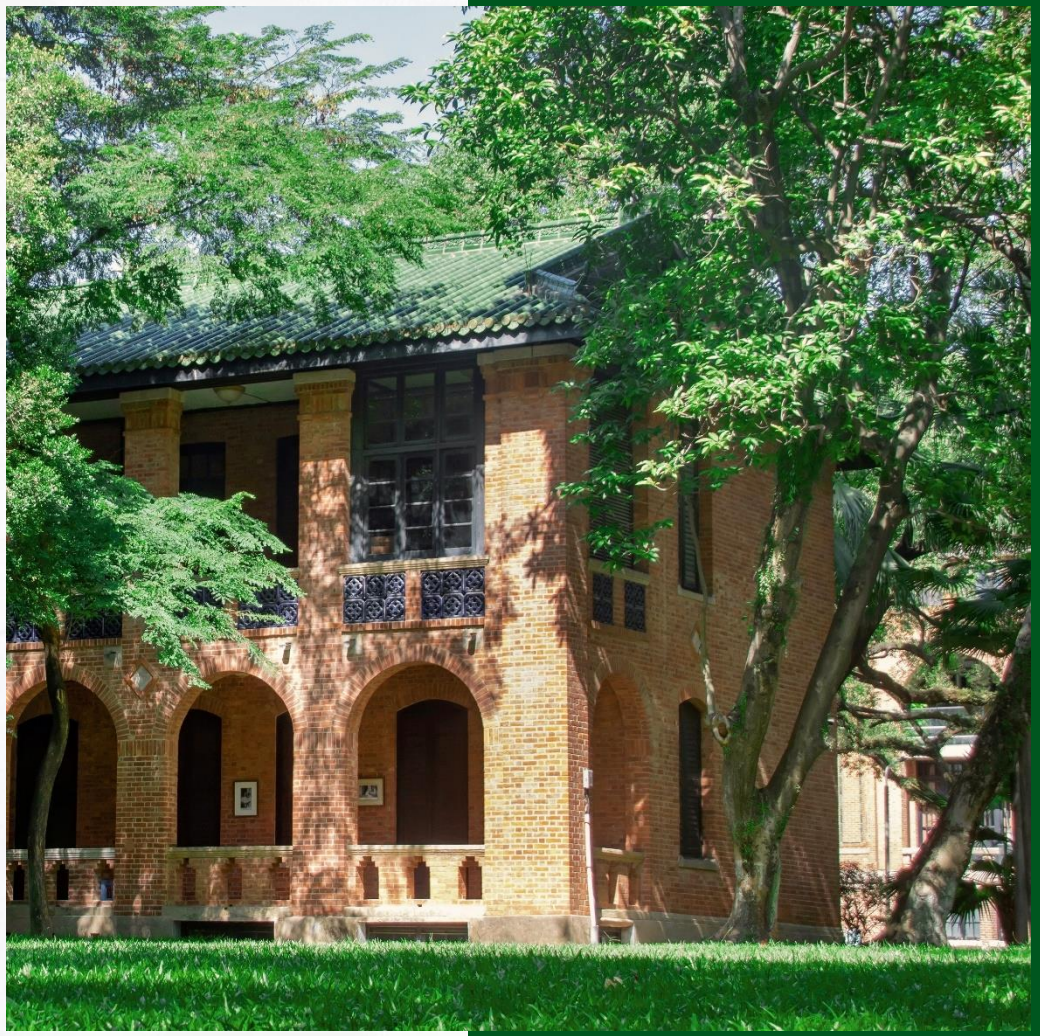
中山大學
SUN YAT-SEN UNIVERSITY

鸿蒙操作系统
设备开发基础



OpenHarmony技术架构

OpenHarmony整体遵从分层设计，从下向上依次为：内核层、系统服务层、框架层和应用层。系统功能按照“系统 > 子系统 > 组件”逐级展开，在多设备部署场景下，支持根据实际需求裁剪某些非必要的组件。

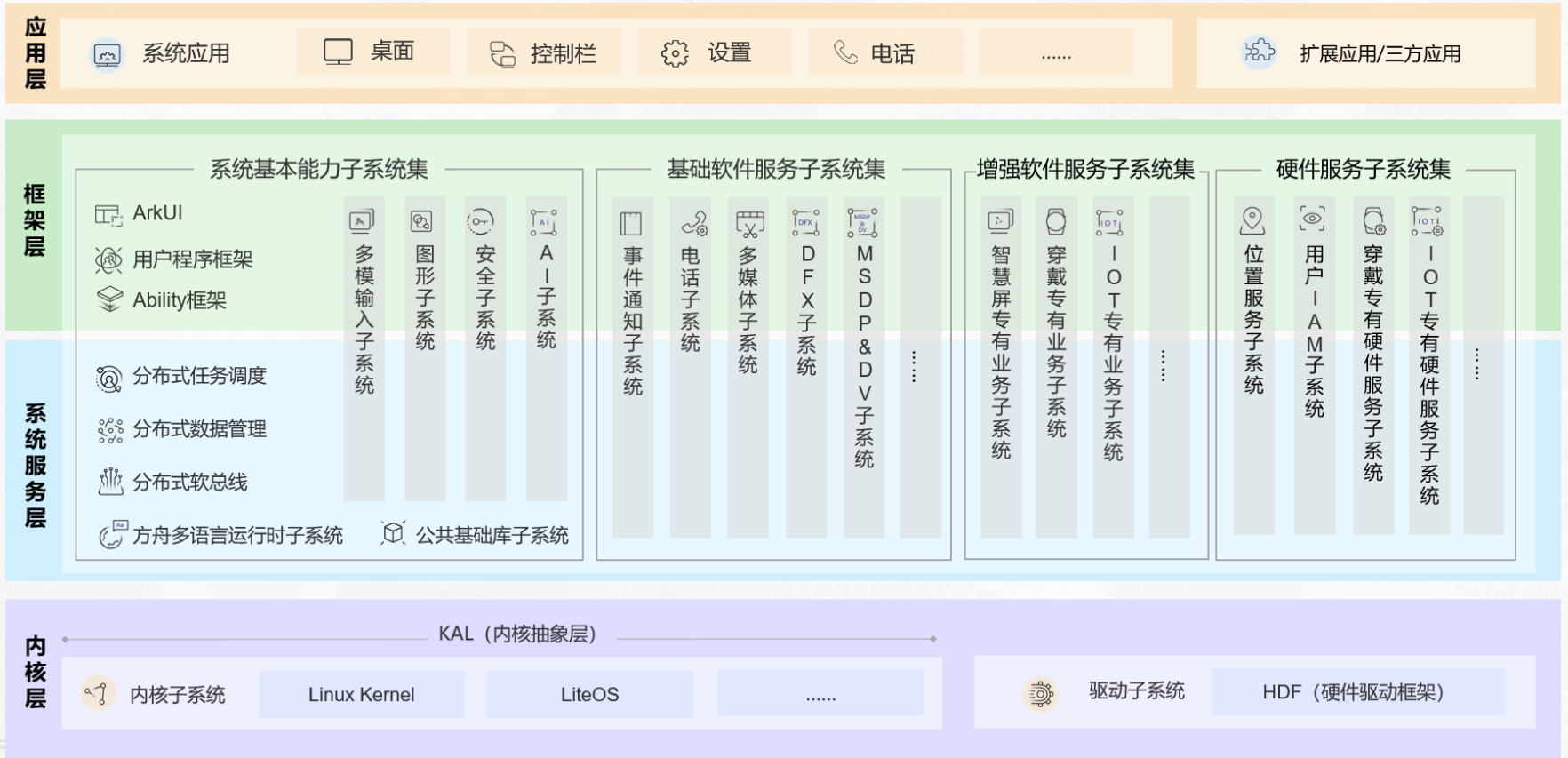


图10-1 OpenHarmony技术架构

子系统与组件

1. 子系统是一个逻辑概念，它具体由对应的组件构成。
2. 组件：是对子系统的进一步拆分，它是可复用的软件单元，它包含源码、配置文件、资源文件和编译脚本；能独立构建，以二进制方式集成，具备独立验证能力的二进制单元。
3. Harmony中的子系统有：
   - **内核**：支持适用于嵌入式设备及资源受限设备，具有小体积、高性能、低功耗等特征的LiteOS内核；支持基于linux kernel演进的适用于标准系统的linux内核。
   - **分布式文件**：提供本地同步JS文件接口。
   - **图形**：主要包括UI组件、布局、动画、字体、输入事件、窗口管理、渲染绘制等模块。
   - **驱动**：OpenHarmony驱动子系统采用C面向对象编程模型构建，通过平台解耦、内核解耦，兼容不同内核，提供了归一化的驱动平台底座。
   - **电源管理服务**：提供重启系统；管理休眠运行锁；系统电源状态管理和查询；充电和电池状态查询和上报；显示亮灭屏状态管理等功能。
   - **编译构建**：编译构建子系统提供了一个基于Gn和ninja的编译构建框架。
   - **分布式任务调度**：提供系统服务的启动、注册、查询及管理能力。
   - **包管理子系统**：提供包安装、卸载、更新、查询等能力。
   - **安全**：包括系统安全、数据安全、应用安全等模块，为OpenHarmony提供了保护系统和和用户数据的能力。

子系统与组件

以编译构建子系统为例，学习产品、子系统、部件和模块间关系

1. **平台**：开发板和内核的组合，不同平台支持的子系统和部件不同。
2. **产品**：产品是包含一系列部件的集合，编译后产品的镜像包可以运行在不同的开发板上。
3. **模块**：模块就是编译子系统的一个编译目标，部件也可以是编译目标。
4. 编译子系统的各部分关系
   - 子系统是某个路径下所有部件的集合，一个部件只能属于一个子系统。
   - 部件是模块的集合，一个模块只能归属于一个部件。
   - 通过产品配置文件配置一个产品包含的部件列表，部件不同的产品配置可以复用。
   - 部件可以在不同的产品中实现有差异，通过变体或者特性feature实现。
   - 模块就是编译子系统的一个编译目标，部件也可以是编译目标。

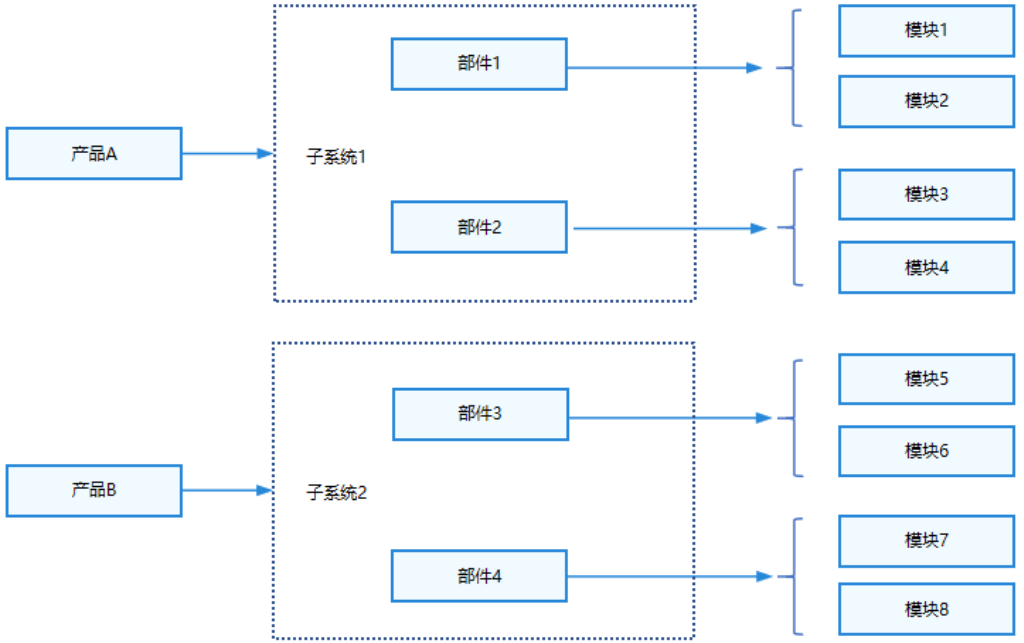


图10-2 产品、子系统、部件和模块间关系

基础类型系统

1. OpenHarmony是一款面向全场景的开源分布式操作系统，采用组件化设计，支持在128KiB到xGiB RAM资源的设备上运行系统组件，设备开发者可基于目标硬件能力自由选择系统组件进行集成。
2. OpenHarmony当前定义了三种基础系统类型，设备开发者通过选择基础系统类型完成必选组件集配置后，便可实现其最小系统的开发。这三种基础系统类型的参考定义如下：
   - **轻量系统（mini system）**：面向MCU类处理器，例如Arm Cortex-M、RISC-V 32位的设备，硬件资源极其有限，支持的设备最小内存为128KiB，可以提供多种轻量级网络协议，轻量级的图形框架，以及丰富的IOT总线读写部件等。可支撑的产品如智能家居领域的连接类模组、传感器设备、穿戴类设备等。
   - **小型系统（small system）**：面向应用处理器，例如Arm Cortex-A的设备，支持的设备最小内存为1MiB，可以提供更高的安全能力、标准的图形框架、视频编解码的多媒体能力。可支撑的产品如智能家居领域的IP Camera、电子猫眼、路由器以及智慧出行域的行车记录仪等。
   - **标准系统（standard system）**：面向应用处理器，例如Arm Cortex-A的设备，支持的设备最小内存为128MiB，可以提供增强的交互能力、3D GPU以及硬件合成能力、更多控件以及动效更丰富的图形能力、完整的应用框架。可支撑的产品如高端的冰箱显示屏。

芯片、模组和开发板

1. 芯片，也被称为集成电路，是微电子技术的核心。它集成了大量的电子元件，如晶体管、电阻和电容，在非常小的面积上实现了特定的功能。如Hi3516开发板中的处理器Hi3516DV300芯片、Hi3518开发板中的海思3518EV300芯片。
2. 模组是基于芯片开发的一种扩展硬件，它通过电路板和其他组件将芯片的功能转化为可实际应用的设备。模组将芯片的功能进行了专用化和具体化，使得开发者能够更方便地利用芯片实现各种应用。
3. 开发板则是为学习和测试目的而设计的电路板，它集成了芯片和必要的外围电路，提供了一系列的接口和扩展槽，方便开发者进行原型设计和测试。开发板通常是针对特定芯片或模组设计的。在开发过程中，开发者可以在开发板上进行编程和测试，验证自己的设计是否符合预期。华为设备开发中常用的开发板有Hi3861开发板、Hi3516开发板和Hi3518开发板。

Hi3861开发板

1. Hi3861 WLAN模组是一片大约2cm*5cm大小的开发板，是一款高度集成的2.4GHz WLAN SoC芯片，集成IEEE 802.11b/g/n基带和RF（Radio Frequency）电路。支持OpenHarmony，并配套提供开放、易用的开发和调试运行环境。主要适配轻型系统。
2. Hi3861开发板还可以通过与Hi3861底板连接，扩充自身的外设能力。

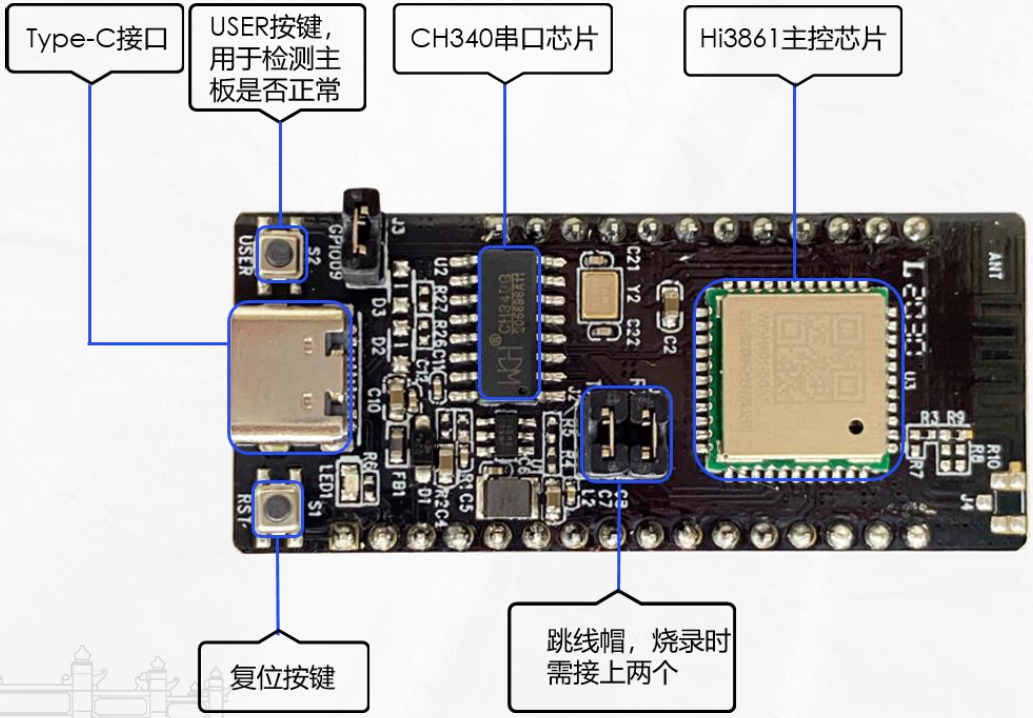


图10-3 Hi3861开发板外观图

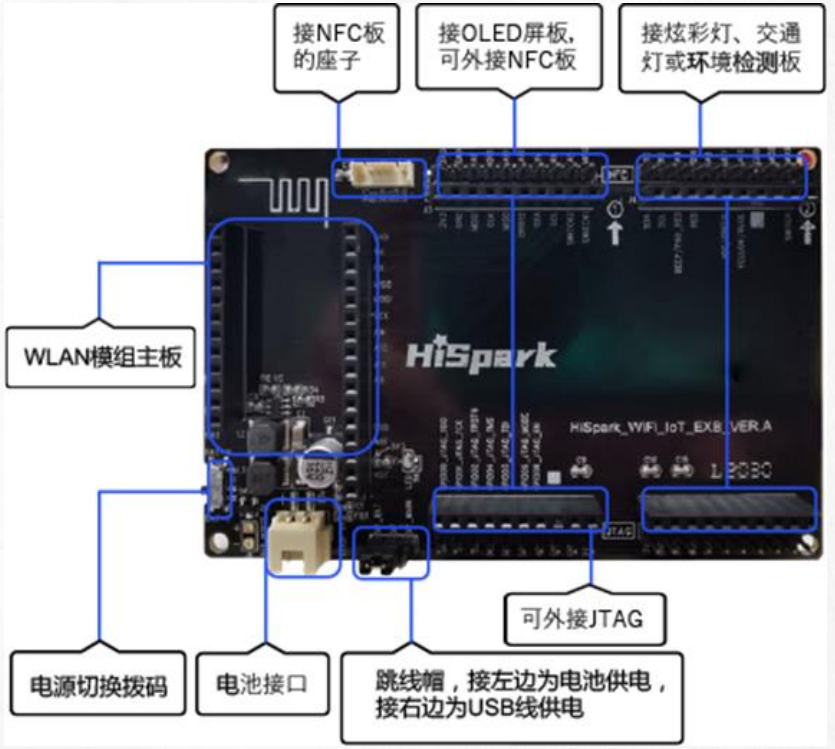


图10-4 Hi3861底板外观图

Hi3861开发板

3. Hi3861芯片适用于智能家电等物联网智能终端领域。
4. Hi3861 WLAN基带支持正交频分复用（OFDM）技术，并向下兼容直接序列扩频（DSSS）和补码键控（CCK）技术，支持IEEE 802.11 b/g/n协议的各种数据速率。
5. Hi3861芯片集成高性能32bit微处理器、硬件安全引擎以及丰富的外设接口，外设接口包括SPI、UART、I2C、PWM、GPIO和多路ADC，同时支持高速SDIO2.0接口，最高时钟可达50MHz；芯片内置SRAM和Flash，可独立运行，并支持在Flash上运行程序。
6. Hi3861平台提供多种关键组件，如WLAN服务、模组外设控制、分布式软总线、设备安全绑定、基础加解密、设备服务管理、启动引导、系统属性、基础库等。

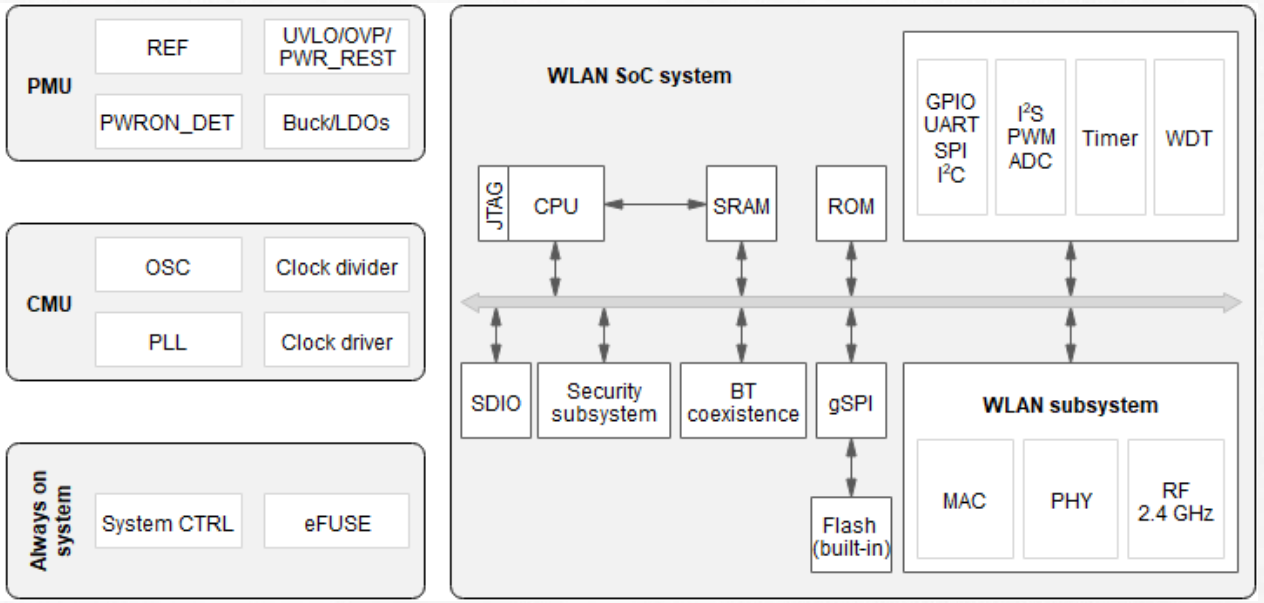


图10-5 Hi3861功能框图

Hi3516开发板

1. Hi3516DV300作为新一代行业专用Smart HD IP摄像机SOC，集成新一代ISP（Image Signal Processor）、H.265视频压缩编码器以及高性能NNIE引擎，具备低码率、高画质、低功耗等特点，并具备强劲的智能处理和分析能力。主要适配小型系统。
2. Hi3516DV300开发板使用Hi3516DV300芯片，内部存储空间为1GB的DDR3和8GB的eMMC4.5。

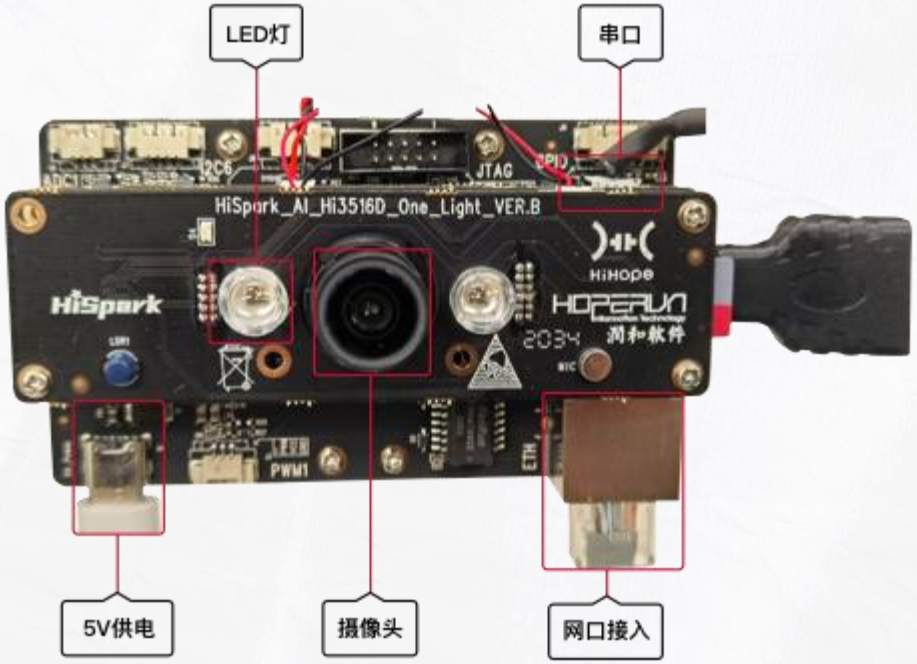


图10-6 Hi3516单板正面外观图

- **RK3568开发板**

1. RK3568开发板基于Rockchip RK3568芯片，集成双核心架构GPU以及高效能NPU；搭载四核64位Cortex-A55处理器，采用22nm先进工艺，主频高达2.0GHz；支持蓝牙、Wi-Fi、音频、视频和摄像头等功能，拥有丰富的扩展接口，支持多种视频输入输出接口；配置双千兆自适应RJ45以太网口，可满足NVR、工业网关等多网口产品需求。
2. RK3568开发板主要适配标准系统。

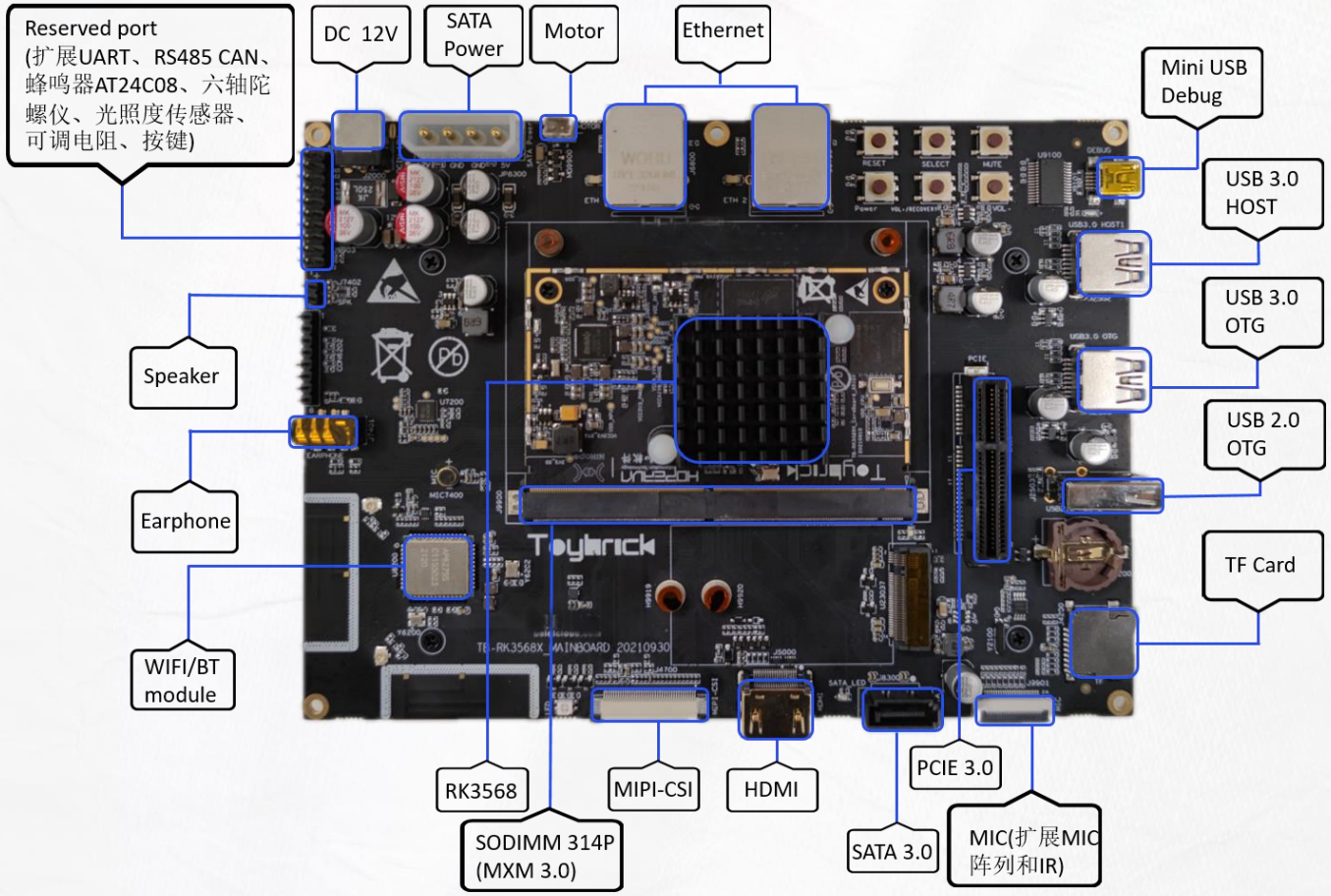


图10-7 RK3568开发板正面

HarmonyOS设备开发方式

1. 考虑到开发者的开发习惯，OpenHarmony为开发者提供了基于IDE开发和基于命令行开发两种开发方式。
2. 基于IDE开发
   - 完全采用IDE进行一站式开发，编译依赖工具的安装及编译、烧录、运行都通过IDE进行操作。
   - DevEco Device Tool采用Windows+Ubuntu混合开发环境。
   - 在Windows上主要进行代码开发、代码调试、烧录等操作。
   - 在Ubuntu环境实现源码编译。
   - DevEco Device Tool提供界面化的操作接口，提供更快捷的开发体验。
3. 基于命令行开发
   - 通过命令行方式下载安装编译依赖工具，在Linux系统中进行编译时，相关操作通过命令实现；
   - 在Windows系统中使用开发板厂商提供的工具进行代码烧录。
   - 命令行方式提供了简便统一的工具链安装方式。

开发工具

基于IDE开发

1. 当前阶段，大部分的开发板源码还不支持在Windows环境下进行编译，如Hi3861、Hi3516系列开发板。因此，建议使用Ubuntu的编译环境对源码进行编译。
2. DevEco Device Tool采用Windows+Ubuntu混合开发环境，在Windows上主要进行代码开发、代码调试、镜像烧录等操作，在Ubuntu环境实现源码编译。
3. DevEco Device Tool特点：
   - 支持代码查找、代码高亮、代码自动补齐、代码输入提示、代码检查等，开发者可以轻松、高效编码。
   - 支持丰富的芯片和开发板，包括基于华为海思芯片的Hi3516DV300/Hi3861V100/Hi3751V350开发板。
   - 支持自动检测各芯片/开发板依赖的工具链是否完备，并提供一键下载和安装缺失工具链。
   - 支持多人共享开发模式。
   - 支持源码级调试能力

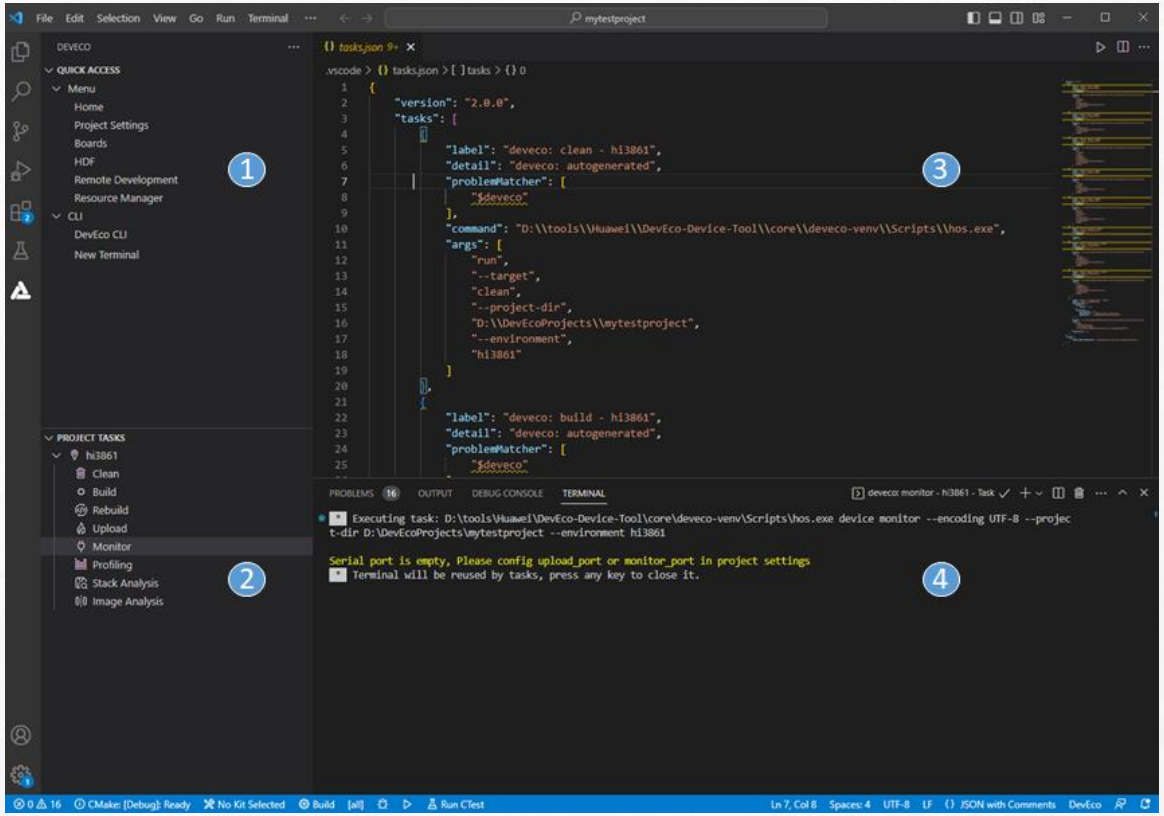


图10-8 DevEco Device Tool

开发工具

基于命令行开发

1. 使用命令行进行设备开发时，需要先安装编译OpenHarmony需要的库和工具。包括Python、Java、gcc等。
2. 配置完所需要的库之后，需要获取OpenHarmony的源码。
   - 注册码云gitee帐号。
   - 安装和配置安装码云repo工具。
   - 使用repo工具从OpenHarmony主干代码或发布分支上获取代码。
   - 在源码根目录下执行prebuilts脚本，安装编译器及二进制工具。
3. 安装编译工具hb和llvm。

Harmony设备开发基本流程

Harmony的设备开发流程主要分为搭建开发环境、获取源码、编写程序、编译、烧录和运行这几大步骤。

- **编写程序**：创建目录，编写业务代码。新建编译组织文件并修改配置文件。
- **编译**：基于命令行的开发方式可以使用hb和build.sh两种方式进行编译，DevEco Device Tool则支持源码一键编译功能，提供编译工具链和编译环境依赖的检测及一键安装。
- **烧录**：烧录是指将编译后的程序文件下载到芯片开发板上的动作，为后续的程序调试提供基础。DevEco Device Tool提供一键烧录功能，操作简单。除此之外也可以使用HiTool进行烧录。
- DevEco Device Tool提供串口工具进行运行；基于命令行的方式则在系统启动成功后，取源码out目录下的helloworld可执行文件放入系统的bin目录，然后运行。

图10-9 设备开发流程

OpenHarmony设备开发实例

智能门禁人脸识别系统

1. 人脸识别应用是基于OpenHarmony 3.2 Beta标准系统上开发的eTS应用，利用NAPI组件调用Seetaface2三方库实现人脸识别能力，主要功能有摄像头拍照、人脸模型录入、人脸框选和人脸识别。
2. 样例原理：通过NAPI组件调用SeetaFace2接口实现人脸录入、框选和识别功能。

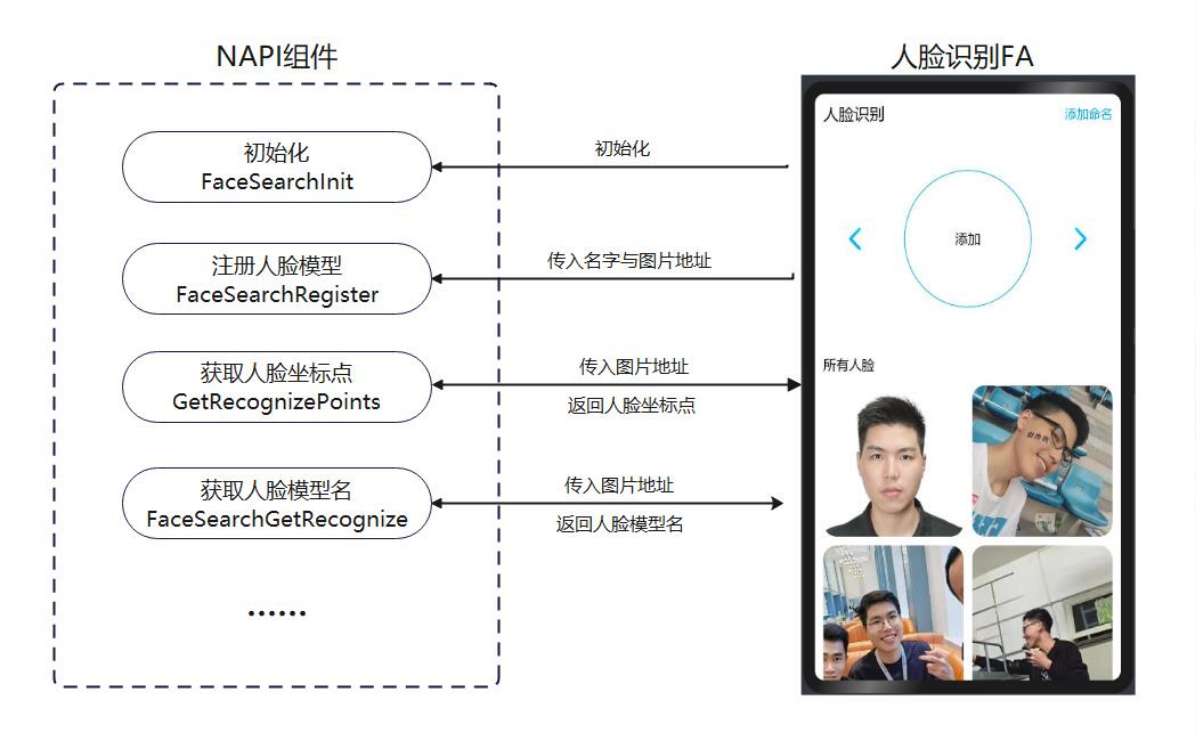


图10-10 样例原理

OpenHarmony设备开发实例

智能门禁人脸识别系统

3. SeetaFace2 是中科视拓开源的第二代人脸识别库。包括了搭建一套全自动人脸识别系统所需的三个核心模块，即：人脸检测模块 FaceDetector、面部关键点定位模块 FaceLandmarker 以及人脸特征提取与比对模块 FaceRecognizer。
4. 移植SeetaFace2： (具体命令可参照人脸识别库的移植)
   - 获取源码：包括获取SeetaFace2以及openharmony3.1beta的源码。
   - 配置源码：下载完openharmony源码后，将SeetaFace2的源码拷贝到openhamony的third_party目录下。
   - SeetaFace2编译gn化：在SeetaFace2顶层目录中编写的gn文件，只需要依赖所需要使用到的模块。写完顶层目录的gn后，还要在每个模块添加对应的gn。
   - 编译：使用./build.sh进行编译。
5. NAPI 接口开发：主要实现 FaceSearchInit 、 FaceSearchRegister 、 FaceSearchGetRecognize 、 GetRecognizePoints等接口。

OpenHarmony设备开发实例

智能门禁人脸识别系统接口实现

1. FaceSearchInit()初始化接口：主要是提供给人脸搜索以及识别调用的，初始化主要包含模型的注册以及识别模块的初始化。
2. FaceSearchDeinit()逆初始化接口：对已申请的内存做一些释放。

```
static  int FaceSearchInit(FaceSearchInfo *info)
{
    if (info == NULL) {
        info = (FaceSearchInfo *)malloc(sizeof(FaceSearchInfo));
        if (info == nullptr) {
            cerr << "NULL POINT!" << endl;
            return -1;
        }
    }

    seeta::ModelSetting::Device device = seeta::ModelSetting::CPU;
    int id = 0;
    // 模型初始化
    seeta::ModelSetting FD_model( "/system/usr/model/fd_2_00.dat", device, id );
    seeta::ModelSetting PD_model( "/system/usr//model/pd_2_00_pts5.dat", device, id );
    seeta::ModelSetting FR_model( "/system/usr/model/fr_2_10.dat", device, id );

    info->engine = make_shared<seeta::FaceEngine>(FD_model, PD_model, FR_model, 2, 16);
    info->engine->FD.set( seeta::FaceDetector::PROPERTY_MIN_FACE_SIZE, 80);

    info->GalleryIndexMap.clear();

    return 0;
}
```

OpenHarmony设备开发实例

智能门禁人脸识别系统接口实现

3. FaceSearchRegister()人脸搜索识别注册接口：读取人脸照片，获取照片对应的id并进行注册。
4. GetRecognizePoints()获取人脸框接口：通过应用层输入一张图片，通过OpenCV的imread接口获取到图片数据，并获得图片中所有的人脸矩形框（矩形框是以x,y,w,h的方式）并将人脸框矩形以数组的方式返回到应用层。

```
static int FaceSearchRegister(FaceSearchInfo &info, RegisterInfo &gegister)
{
    if (info.engine == nullptr) {
        cerr << "NULL POINT!" << endl;
        return -1;
    }
    // 注册人脸模型
    seeta::cv::ImageData image = cv::imread(gegister.path);
    auto id = info.engine->Register(image);
    if (id >= 0) {
        info.GalleryIndexMap.insert(make_pair(id, gegister.name));
    }

    return 0;
}
```

- **OpenHarmony设备开发实例**

智能门禁人脸识别系统接口实现

5. FaceSearchGetRecognize()：该接口实现了通过传入一张图片，调用识别引擎中的QualityAssessor进行搜索识别。如果识别引擎中有类似的人脸注册，则返回对应人脸注册时的名字，否则返回不识别(ignored)字样。该方法是通过异步回调的方式实现的。
6. 该方法是通过异步回调的方式实现的。具体会通过FaceSearchRecognizeExecuteCB来调用右图的FaceSearchSearchRecognizer()函数；最后再通过FaceSearchRecognizeCompleteCB()函数将识别结果返回到应用端。

```
static string FaceSearchSearchRecognizer(FaceSearchInfo &info, string filename)
{
    if (info.engine == nullptr) {
        cerr << "NULL POINT!" << endl;
        return "recognize error 0";
    }
    string name;
    float threshold = 0.7f;
    seeta::QualityAssessor QA;
    auto frame = cv::imread(filename);
    if (frame.empty()) {
        LOGE("read image %{public}s failed!", filename.c_str());
        return "recognize error 1!";
    }
    seeta::cv::ImageData image = frame;
    std::vector<SeetaFaceInfo> faces = info.engine->DetectFaces(image);

    for (SeetaFaceInfo &face : faces) {
        int64_t index = 0;
        float similarity = 0;

        auto points = info.engine->DetectPoints(image, face);

        auto score = QA.evaluate(image, face.pos, points.data());
        if (score == 0) {
            name = "ignored";
        } else {
            auto queried = info.engine->QueryTop(image, points.data(), 1, &index, &similarity);
            // no face queried from database
            if (queried < 1) continue;
                // similarity greater than threshold, means recognized
            if( similarity > threshold ) {
                name = info.GalleryIndexMap[index];
            }
        }
    }
    LOGI("name : %{public}s \n", name.length() > 0 ? name.c_str() : "null");
    return name.length() > 0 ? name : "recognize failed";
}
```

OpenHarmony设备开发实例

智能门禁人脸识别系统

最终效果

图10-11 人脸录入

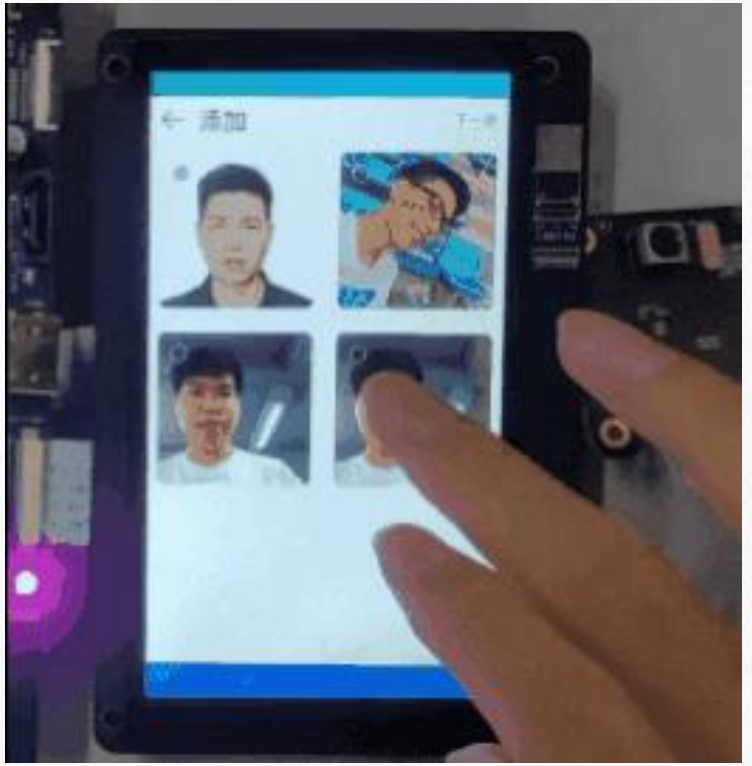


图10-12 人脸框选与识别

OpenHarmony设备开发实例

分布式音乐播放器

1. 该实例使用fileIo获取指定音频文件，并通过AudioPlayer完成了音乐的播放完成了基本的音乐播放、暂停、上一曲、下一曲功能；并使用DeviceManager完成了分布式设备列表的显示和分布式能力完成了音乐播放状态的跨设备迁移。
2. 在分布式音乐播放器中，分布式设备管理包含了分布式设备搜索、分布式设备列表弹窗、远端设备拉起三部分。
3. 首先在分布式组网内搜索设备，然后把设备展示到分布式设备列表弹窗中，最后根据用户的选择拉起远端设备。
   - **分布式设备搜索**：通过SUBSCRIBE_ID搜索分布式组网内的远端设备。
   - **分布式设备列表弹窗**：使用continueAbilityDialog弹出分布式设备列表弹窗。
   - **远端设备拉起**：通过startAbility(deviceId)方法拉起远端设备的包。

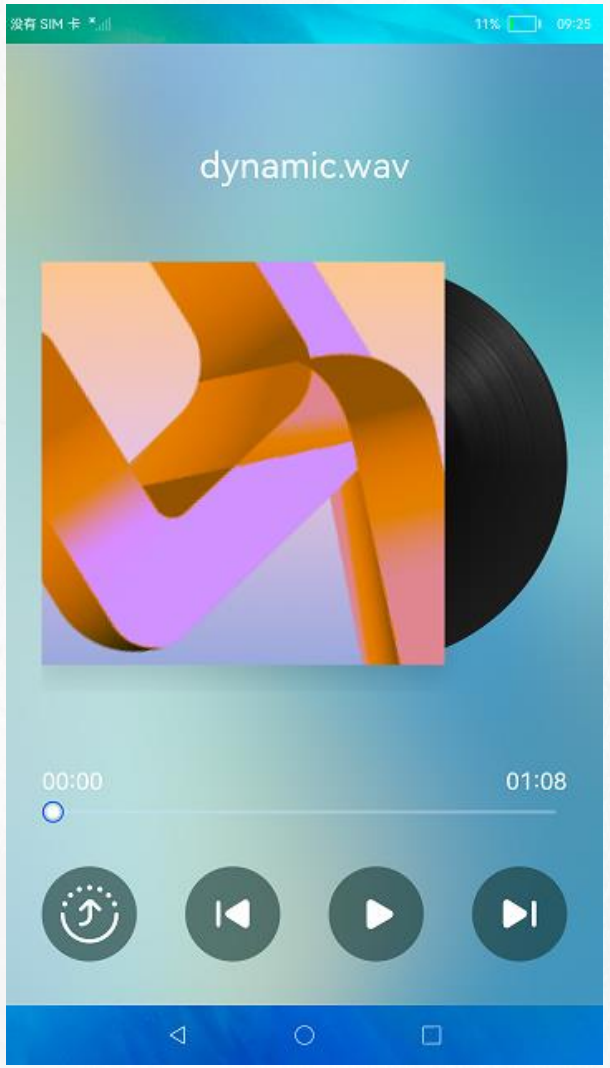


图10-13音乐播放器效果图

OpenHarmony设备开发实例

分布式音乐播放器

1. 分布式数据管理：主要分为管理分布式数据库和订阅分布式数据变化两部分。
   - **管理分布式数据库**：创建一个KVManager对象实例，用于管理分布式数据库对象。通过distributedData.createKVManager(config)，并通过指定Options和storeId，创建并获取KVStore数据库，并通过Promise方式返回，此方法为异步方法。
   - **订阅分布式数据变化**：通过订阅分布式数据库所有（本地及远端）数据变化实现数据协同。
2. 跨设备播放操作
   - 分布式设备管理器绑定应用包 deviceManager.createDeviceManager 。
   - 初始化播放器构造函数中通过‘@ohos.multimedia.media’组件对播放器进行实例化，并调用播放器初始化函数，通过播放器的on函数，监听error、finish、timeUpdate事件。
   - 同步当前播放数据播放器通过onRadioChange(),将当前播放的资源、时间、以及播放状态同步给选中的设备。
   - 接收当前播放数据播放通过onNewRequest()，调用this.restoreFromWant(),featureAbility获取want参数，kvstore组件获取播放列表，playerModel组件重新加载播放器状态和资源。

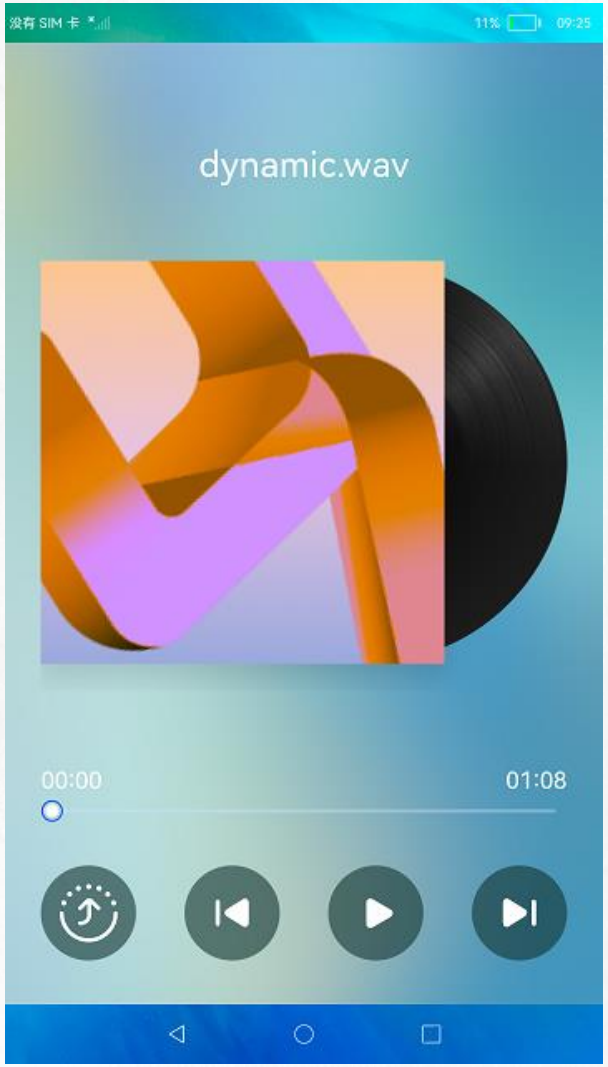


图10-13音乐播放器效果图

中山大學
SUN YAT-SEN UNIVERSITY

谢谢观看

SUN YAT-SEN UNIVERSITY