物联网开发平台

AI 智能眼镜



腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2026 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

AI 智能眼镜
功能介绍
快速体验
跑通应用端

版权所有：腾讯云计算（北京）有限责任公司

AI 智能眼镜
功能介绍

最近更新时间：2026-11-17 14:42:31

IoT Explorer · AI 智能眼镜

AI 眼镜零开发即接即用

普通蓝牙眼镜连上手机即可使用。App 集成 TRTC SDK，即可获得实时翻译、会议转录、AI 对话与语音通话能力。

立即体验 ↗　接入文档



眼镜零开发
标准蓝牙连接，品牌不限

四大 AI 能力
翻译、转录、对话、通话

低时延抗弱网
TRTC 全球实时音频网络

双端源码开放
Demo 源码可直接改造

核心功能与典型场景

四项功能共用眼镜收音链路，按场景单独或组合接入。

实时翻译 | 会议转录 | AI 对话 | 语音通话

版权所有：腾讯云计算（北京）有限责任公司

版权所有：腾讯云计算（北京）有限责任公司



实时翻译

对方开口，眼镜即刻收音，译文在手机上逐句呈现，也可直接播报到耳边。

典型场景 只需安静听讲，讲者的外语实时变成母语字幕。

外语讲座　国际会议　出境旅行

核心能力

实时识别 中英互译，逐句断句

逐句对照 原文与译文成对呈现，保存双语记录

译文播报 译文自动播放到眼镜，可随时开关

立即体验
运行 AI 眼镜 App，体验完整流程

接入到你的项目 选择对应端查看文档

iOS ›　Android ›

方案架构

三层分工：眼镜收音与播放，App 采集与呈现，云端识别与理解。

眼镜端 | 手机 App | 腾讯云

01 / DEVICE

眼镜只负责收音与播放

眼镜以标准蓝牙耳机方式连接手机：麦克风采集说话声音，扬声器播放译文、回答与通话声音。



这一层包含哪些能力

蓝牙连接
标准蓝牙耳机，连接即可使用

拾音
眼镜麦克风收音，说话人无需拿起手机

播放
译文播报、回答播报、通话声音

说明：
翻译、转录、纪要与问答内容由人工智能生成或辅助生成，可能存在偏差，仅供辅助参考，请结合原文核对后使用。接入方应向终端用户明确告知语音采集、云端 AI 处理及合成语音播报事项，依法取得授权，并遵守《生成式人工智能服务管理暂行办法》、《互联网信息服务深度合成管理规定》、《个人信息保护法》等相关法律法规。

版权所有：腾讯云计算（北京）有限责任公司

快速体验

最近更新时间：2026-11-17 14:58:01

本文将介绍如何运行 AI 眼镜 App，快速体验实时翻译、会议转录、语音通话与 AI 对话等功能。眼镜在本方案中作为标准蓝牙耳机使用，**无需任何设备端开发**。
完成服务开通并与手机连接后，即可在手机上运行 App，体验「眼镜拾音 → 云端识别与翻译 → 结果呈现与回放」的完整链路。

> **说明：**
> 建议在 Chrome/Edge 等浏览器下体验交互效果。

前提条件

- 准备一台 iOS 17 及以上或 Android 10 及以上的手机（需使用真机）；体验语音通话时，需要另一台设备作为对端。
- 准备开发环境：iOS 需安装 Xcode 与 CocoaPods，Android 需安装 Android Studio。
- 准备已实名认证的腾讯云账号。
- 如需体验 AI 对话、AI 会议纪要或「识别 + 大模型翻译」，还需准备一个兼容 OpenAI Chat Completions 的大模型服务，并获取服务地址、API Key 与模型名。

开通服务

1. 登录 实时音视频控制台，单击**创建应用**。如果您已经完成创建，可以跳过该操作。
2. 在创建应用成功后，您可以在**应用管理**中获取到您的 **SDKAppID** 和 **SDKSecretKey**。



说明：
本方案的实时识别、转写翻译与语音合成能力基于 TRTC AI 智能语音，请参见 TRTC AI 智能语音 了解能力说明与计费详情。

运行 App

下载 App 源码

AI 眼镜 App 源码已在 GitHub 开源，位于 Tencent_IoT_App 仓库的 `solutions/iot_ai_glasses` 目录，其中 `iOS` 与 `Android` 分别为两端工程：

```
git clone https://github.com/Tencent-RTC/Tencent_IoT_App.git
cd Tencent_IoT_App/solutions/iot_ai_glasses
```

填写配置参数

编译运行 App 前需要填写两类参数：**TRTC 应用参数**（必填）与**第三方大模型服务参数**（按需填写）。源码中的密钥均已清空，请按下面两个小节填写后再构建。

1. TRTC 应用参数（必填）

登录与进房参数使用开通服务中获取的 SDKAppID 与 SDKSecretKey。填写位置如下：

Android

文件： `Android/app/src/main/java/com/tencent/iot/aiglasses/services/trtc/TRTCCredentials.kt`

```
object TRTCCredentials {
    const val SDK_APP_ID = 0      // 替换为您的 SDKAppID
    const val SECRET_KEY = ""     // 替换为您的 SDKSecretKey
}
```

iOS

文件： `iOS/IoTAIGlasses/Services/TRTC/TRTCCredentials.swift`

```
enum TRTCCredentials {
    static let sdkAppId = 0       // 替换为您的 SDKAppID
    static let secretKey = ""     // 替换为您的 SDKSecretKey
}
```

2. 第三方大模型服务参数（按需填写）

AI 对话、AI 会议纪要，以及「语音识别 + 大模型翻译」这条翻译链路，会调用您自己的大模型服务。请填写服务地址、API Key 与模型名，三项填写完整后相关功能才会启用。未配置时 App 不会向大模型服务发起请求，实时翻译仍可使用 TRTC AI 转写机器人链路，会议转录也仍可正常录音与转写。

Android

文件： `Android/app/src/main/java/com/tencent/iot/aiglasses/services/ai/AIServiceConfig.kt`

```
object AIServiceConfig {
    const val ENDPOINT = ""    // 完整请求地址，例如 https://your-host/v1/chat/completions
    const val API_KEY = ""     // 替换为您的 API Key
    const val MODEL = ""       // 替换为您的模型名
```

```
}
```

iOS

文件： `iOS/IoTAIGlasses/Services/AI/AIServiceConfig.swift`

```
enum AIServiceConfig {
    static let endpoint = ""   // 完整请求地址，例如 https://your-host/v1/chat/completions
    static let apiKey = ""     // 替换为您的 API Key
    static let model = ""      // 替换为您的模型名
}
```

`ENDPOINT` 必须是 `https://` 开头的完整 Chat Completions 接口地址，而不是服务域名。App 以 `Authorization: Bearer <API Key>` 发起非流式请求，并读取 `choices[0].message.content`；请求体中会带上 `thinking: {"type":"disabled"}` 用于关闭深度思考以降低延迟，如果您的服务不接受该字段，可在 `ChatCompletionClient` 中去掉。

编译运行

填写参数后，按以下方式编译并安装到真机：

Android

使用 Android Studio 打开 `Android` 目录，Gradle 同步时会自动从 Maven 拉取 `LiteAVSDK_Professional` 与 `LiteAVSDK_VoiceAI`。同步完成后连接 Android 手机，单击 **Run** 安装运行。Demo 仅打包 `arm64-v8a` 架构。

iOS

```
cd iOS
pod install
open IoTAIGlasses.xcworkspace
```

请打开 `IoTAIGlasses.xcworkspace`（而不是 `.xcodeproj`）。在 **Signing & Capabilities** 中选择您自己的 Team，并将 Bundle Identifier 改为您账号下可用的值，然后连接 iPhone 运行。

在手机系统蓝牙设置中连接眼镜后，即可开始体验以下功能。

体验实时翻译

适合单向听译场景：眼镜持续拾音，外语被实时识别并翻译为您的语言。



实时翻译界面说明

体验会议转录

适合会议记录场景：实时转写 + 说话人分离 + AI 会议纪要。



会议转录界面说明

体验语音通话

眼镜即耳机，与对端进行低时延语音通话。开始前请准备另一台设备作为对端。下图左侧为输入房间号进入通话，右侧为通话中界面。



语音通话界面说明

体验 AI 对话

随身语音助手：对眼镜说出问题，大模型作答。使用前请先完成上方 第三方大模型服务参数 的配置，否则会提示“请先在 AIServiceConfig 中配置大模型服务地址、API Key 与模型名”。


1:15
AI 对话
AI 眼镜已连接
深圳有推荐的徒步路线吗?
你的提问
有的。轻松点可以走深圳湾公园滨海栈道，看海很舒服；想爬一点选塘朗山郊野公园，交通方便；想看山海就去盐田海滨栈道或大南山。你偏好海景、山景，还是难度低一点的?
AI 回答
深圳湾公园全程有多长?
深圳湾公园滨海休闲带大约 15 公里左右，从福田红树林一带一直延伸到蛇口附近。全程走完大概 3 到 4 小时，也可以只选一段轻松逛。
回答播报中
正在眼镜中播报
播报开关
正在聆听
停止
开始 / 暂停聆听

AI 对话界面说明

跑通应用端

最近更新时间：2026-11-17 14:42:31

本文将介绍如何在自有 App 中集成 TRTC SDK，实现 AI 眼镜方案的实时翻译、会议转录、AI 对话与语音通话。眼镜作为标准蓝牙耳机使用，**全部接入工作都在 App 端完成**，眼镜端无需开发。

前提条件

- 已开通腾讯云 TRTC 服务，获取 SDKAppID 和 SDKSecretKey，详见 快速体验 中的开通服务介绍。
- 如需 AI 对话与会议纪要，准备一个兼容 OpenAI Chat Completions 协议的大模型服务。
- 下文代码均摘自 AI 眼镜 Demo，完整源码见 GitHub Tencent_IoT_App/solutions/iot_ai_glasses 。

功能与接口对照

功能	使用的接口
实时翻译	`TRTCCloud` 进房 + `AITranscriberManager` 云端转写翻译。
译文 / 回答播报	`TXRealtimeTTS` 语音合成。
会议转录	`TXRealtimeASR`（说话人分离 + 录音笔降噪）。
AI 对话	`TXRealtimeASR`（AI 对话降噪）+ 大模型 + `TXRealtimeTTS` 。
会议纪要	大模型。
语音通话	`TRTCCloud` 进房。

接入步骤

步骤 1：集成 SDK

通过 Maven / CocoaPods 集成两个 SDK：`LiteAVSDK_Professional` 提供进房、音频采集与云端转写（`AITranscriberManager`），VoiceAI 包提供 `TXRealtimeASR` 与 `TXRealtimeTTS` 。两个包需配套使用，版本号以 Demo 工程为准。

Android

```
// app/build.gradle
dependencies {
```

```
    implementation
'com.tencent.liteav:LiteAVSDK_Professional:13.6.0.20747'
    implementation 'com.tencent.liteav:LiteAVSDK_VoiceAI:13.6.0.255'
}
```

iOS

```
# Podfile
target 'YourApp' do
  pod 'TXLiteAVSDK_Professional', '13.6.21648'
  pod 'TXLiteAVSDK_VoiceAI_iOS', '13.6.0.255'
end
```

执行 `pod install` 后，使用生成的 `.xcworkspace` 打开工程。

步骤 2：声明权限

App 需要麦克风权限；Android 12 及以上还需要“附近的设备”权限，SDK 才能使用蓝牙耳机收音。

Android

```
<!-- AndroidManifest.xml -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission
android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH"
android:maxSdkVersion="30" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"
/>
```

iOS

```
<!-- Info.plist -->
<key>NSMicrophoneUsageDescription</key>
<string>需要使用麦克风，通过眼镜收音进行实时翻译、会议转录和语音通话。</string>
```

眼镜以蓝牙耳机方式连接后，SDK 会自动通过眼镜采集和播放声音，无需额外代码。Android 端 `RECORD_AUDIO` 需要在运行时动态申请；Android 12 及以上，`BLUETOOTH_CONNECT` 也需要在运行时动态申请。

步骤 3：生成 UserSig

进房、语音识别与语音合成都需要 UserSig 鉴权，由 SDKAppID、SDKSecretKey 和 UserID 计算得出，详见 用户鉴权 。Demo 中的计算代码位于 `GenerateTestUserSig`，调用方式如下：

Android

```
val userSig = GenerateTestUserSig.genTestUserSig(SDKAppID, userId,
SDKSecretKey)
```

iOS

```
let userSig = GenerateTestUserSig.genTestUserSig(sdkAppId: SDKAppID,
userId: userId, secretkey: SDKSecretKey)
```

后续代码中的 `genUserSig(id)` 即表示用上述方式为指定 UserID 计算 UserSig。

> **注意：**
> 本地计算 UserSig 仅适用于调试。正式上线请在业务服务端计算 UserSig，App 通过接口获取，切勿把 SDKSecretKey 打包进 App。

步骤 4：实时翻译

进入房间并开启收音：实时翻译基于 TRTC 房间，云端转写机器人会订阅本端音频完成识别与翻译。

Android

```
val cloud = TRTCCloud.sharedInstance(context)
cloud.addListener(cloudListener)

val params = TRTCCloudDef.TRTCParams().apply {
    sdkAppId = SDKAppID
    userId = myUserId
    userSig = myUserSig
    strRoomId = "room_$myUserId"
    role = TRTCCloudDef.TRTCRoleAnchor
}
cloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_AUDIOCALL)
```

iOS

```
let cloud = TRTCCloud.sharedInstance()
cloud.addDelegate(self)

let params = TRTCParams()
params.sdkAppId = UInt32(SDKAppID)
params.userId = myUserId
params.userSig = myUserSig
params.strRoomId = "room_\(myUserId)"
params.role = .anchor
cloud.enterRoom(params, appScene: .audioCall)
```

启动转写翻译：在进房成功回调 `onEnterRoom` 中开启本地音频，并启动转写机器人，指定源语言与译文语言。

Android

```
override fun onEnterRoom(result: Long) {
    if (result <= 0) return
    cloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH)
    val params = AITranscriberManager.TranscriberParams().apply {
        sourceLanguage = "en"
        translationLanguages = listOf("zh")
        userIdsToTranscribe = listOf(myUserId)
    }
    cloud.aiTranscriberManager.addListener(transcriberListener)
    cloud.aiTranscriberManager.startRealtimeTranscriber(params)
}
```

iOS

```
func onEnterRoom(_ result: Int) {
    guard result > 0 else { return }
    cloud.startLocalAudio(.speech)
    let params = TranscriberParams()
    params.sourceLanguage = "en"
    params.translationLanguages = ["zh"]
    params.userIdsToTranscribe = [myUserId]
    cloud.getAITranscriberManager().add(self)
    cloud.getAITranscriberManager().startRealtimeTranscriber(params)
}
```

接收原文与译文：同一句话会多次回调，`isCompleted` 为 true 时表示该句已说完；可按 `segmentId` 更新同一条字幕。

Android

```
override fun onReceiveTranscriberMessage(roomId: String?, message:
AITranscriberManager.TranscriberMessage) {
```

```
    val source = message.sourceText
    val translated = message.translationTexts?.get("zh")
    // 按 message.segmentId 刷新字幕；message.isCompleted 为 true 时该句
定稿
}
```

iOS

```
func onReceiveTranscriberMessage(_ roomId: String, message:
TranscriberMessage) {
    let source = message.sourceText
    let translated = (message.translationTexts as? [String:
String])?["zh"]
    // 按 message.segmentId 刷新字幕；message.isCompleted 为 true 时该句
定稿
}
```

步骤 5：把译文播报到眼镜

使用 `TXRealtimeTTS` 将定稿的译文合成为语音，声音会通过当前蓝牙路由在眼镜中播放。

Android

```
val tts = TXRealtimeTTS()
tts.setListener(ttsListener)

val ttsUserId = "tts_" + UUID.randomUUID().toString().take(8)
val params = TXRealtimeTTS.TXRealtimeTTSParams().apply {
    online.appId = SDKAppID.toString()
    online.userId = ttsUserId
    online.userSig = genUserSig(ttsUserId)
    voiceName = "v-male-s5NqE0rZ"
}
tts.start(params)
```

```
tts.appendText(UUID.randomUUID().toString(), translated, true)
```

iOS

```
let tts = TXRealtimeTTS()
tts.setListener(self)

let ttsUserId = "tts_" + UUID().uuidString.prefix(8)
let online = TXRealtimeTTSOnlineCredential()
online.appId = String(SDKAppID)
online.userId = ttsUserId
online.userSig = genUserSig(ttsUserId)
let params = TXRealtimeTTSParams()
params.online = online
params.voiceName = "v-male-s5NqE0rZ"
tts.start(params)
tts.appendText(UUID().uuidString, text: translated, isEnd: true)
```

步骤 6：会议转录

会议转录直接使用 `TXRealtimeASR` 。启动前通过实验性 API 开启说话人分离，并将降噪策略设为录音笔场景（ `clientDenoiseStrategy` 为 `1` ）。

Android

```
val asr = TXRealtimeASR()
asr.addListener(asrListener)
asr.callExperimentalAPI(
    """{"api":"setExtraParams","params":
{"extraRequestParams":"speaker_diarization=1","clientDenoiseStrategy
":1}}"""
)

val voiceId = UUID.randomUUID().toString()
```

```
val params = TXRealtimeASR.Params().apply {
    sdkAppId = SDKAppID.toString()
    this.voiceId = voiceId
    userSig = genUserSig(voiceId)
    sourceLanguage = "zh"
}
asr.startRealtimeASR(params)
```

iOS

```
let asr = TXRealtimeASR()
asr.addListener(self)
_ = asr.callExperimentalAPI(
    #"{"api":"setExtraParams","params":
{"extraRequestParams":"speaker_diarization=1","clientDenoiseStrategy
":1}}"#
)

let voiceId = UUID().uuidString
let params = TXRealtimeASRParams()
params.sdkAppId = String(SDKAppID)
params.voiceId = voiceId
params.userSig = genUserSig(voiceId)
params.sourceLanguage = "zh"
asr.startRealtimeASR(params)
```

识别结果中的 `speakerId` 用于区分发言人（ `-1` 表示未知），可显示为“发言人 1”、“发言人 2”。

Android

```
override fun onReceiveRealtimeASRMessage(message:
TXRealtimeASR.Message) {
    // message.segmentId、message.sourceText、message.speakerId、
message.isCompleted
```

```
}
```

iOS

```
func onReceiveRealtimeASRMessage(_ message: TXRealtimeASRMessage) {
    // message.segmentId、message.sourceText、message.speakerId、
message.isCompleted
}
```

说明：
`voiceId` 参与 UserSig 计算，即需要以 `voiceId` 作为 UserID 生成 UserSig。建议每次启动识别时重新生成 `voiceId`。

步骤 7：AI 对话与会议纪要

AI 对话：识别同样使用 `TXRealtimeASR`，降噪策略设为 AI 对话场景（`clientDenoiseStrategy` 为 `2`），`sourceLanguage` 留空自动识别语言。一句话定稿后发给大模型，再用 步骤 5 的 `TXRealtimeTTS` 把回答播报到眼镜。

Android

```
asr.callExperimentalAPI("""{"api":"setExtraParams","params":
{"clientDenoiseStrategy":2}}""")
```

iOS

```
_ = asr.callExperimentalAPI(#"{"api":"setExtraParams","params":
{"clientDenoiseStrategy":2}}"#)
```

调用大模型：Demo 通过 OpenAI 兼容的 Chat Completions 接口请求大模型，请求示例：

```
POST https://<大模型服务地址>/v1/chat/completions
{
  "model": "<模型名>",
  "messages": [
    { "role": "system", "content": "你是 AI 眼镜的语音助手，回答简洁口语化，不使用 Markdown。" },
    { "role": "user", "content": "今天深圳天气怎么样？" }
  ],
  "stream": false
}
```

会议纪要：会议结束后，将带发言人的转写文本交给大模型，要求输出包含 `headline`（一句话总结）、`consensus`（共识）、`questions`（待确认问题）、`todos`（待办）的 JSON，App 解析后展示。提示词可参考 Demo 中的 `MeetingSummarizer`。

> **注意：**
> 大模型 API Key 请保存在业务服务端并由服务端转发请求，不要写在 App 中。

步骤 8：语音通话

双方进入同一个房间并开启本地音频即可通话，眼镜作为耳机完成收音与播放。

Android

```
val params = TRTCCloudDef.TRTCParams().apply {
    sdkAppId = SDKAppID
    userId = myUserId
    userSig = myUserSig
    roomId = 8866
}
cloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_AUDIOCALL)
cloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH)

cloud.muteLocalAudio(true)    // 静音
cloud.exitRoom()              // 挂断
```

iOS

```
let params = TRTCParams()
params.sdkAppId = UInt32(SDKAppID)
params.userId = myUserId
params.userSig = myUserSig
params.roomId = 8866
cloud.enterRoom(params, appScene: .audioCall)
cloud.startLocalAudio(.speech)

cloud.muteLocalAudio(true)    // 静音
cloud.exitRoom()              // 挂断
```

步骤 9：结束与释放

功能结束时停止识别与转写并退房；`TXRealtimeASR` 使用完毕需先停止再销毁。

Android

```
asr.stopRealtimeASR()
asr.destroy()
tts.stop()
cloud.aiTranscriberManager.stopRealtimeTranscriber(robotId)
cloud.exitRoom()
```

iOS

```
asr.stopRealtimeASR()
tts.stop()
cloud.getAITranscriberManager().stopRealtimeTranscriber(robotId)
```

```
cloud.exitRoom()
```

其中 `robotId` 来自转写启动成功回调 `onRealtimeTranscriberStarted`。

常见问题

声音没有从眼镜采集或播放?

请确认眼镜已在系统蓝牙中连接，且 App 已获得麦克风权限；Android 12 及以上还需授予“附近的设备”权限。

识别没有结果?

请检查 SDKAppID 与 SDKSecretKey 是否正确、应用是否已开通 AI 智能语音服务；使用 `TXRealtimeASR` 时，UserSig 需以 `voiceId` 作为 UserID 生成。

会议中所有人都显示为同一个发言人?

请确认启动识别前已通过 `setExtraParams` 传入 `speaker_diarization=1`。

实时翻译收不到译文?

请确认在进房成功后再启动转写机器人，`translationLanguages` 与 `sourceLanguage` 不同，且 `userIdsToTranscribe` 包含本端 UserID。

相关文档

- 实时音视频 Android SDK（实时语音识别）
- 用户鉴权
- AI 眼镜 Demo 源码（GitHub）