Atlas 200I DK A2 开发者套件
23.0.RC3
智能语音台灯应用开发指南

文档版本 01

发布日期 2023-11-14

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

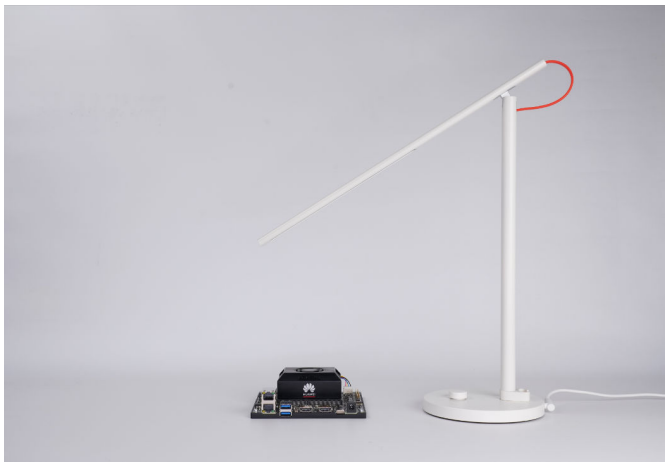
漏洞声明

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该政策可参考华为公司官方网站的网址：https://www.huawei.com/cn/psirt/vul-response-process。
如企业客户须获取漏洞信息，请访问：https://securitybulletin.huawei.com/enterprise/cn/security-advisory。

目 录

1 样例介绍

1.1 外观展示



1.2 功能与原理介绍

智能语音台灯应用通过将语音识别模型WeNet转换为OM模型，使其能够运行在开发者套件上的昇腾AI处理器进行加速，主要的工作流程是在网页前端收集用户语音输入，通过后端放入消息队列，后端进行语音识别模型推理解析为文本，并转换为控制命令发送给台灯，从而实现台灯开关控制。交互界面以聊天窗口的形式呈现，用户通过语音输入命令后，聊天机器人会以文本的形式返回控制结果。

图 1-1 实现原理图

2 样例组装

2.1 准备组件

表 2-1 智能语音台灯组件表

模块名称	数量/个	是否需要单独购买
开发者套件	1	否
SD卡	1	是
读卡器	1	是
路由器	1	是
RJ45网线	1	是
米家台灯1s增强版	1	是

2.2 组装步骤

步骤1 参见**一键制卡**章节烧录镜像，将SD卡插入开发者套件的卡槽，**当前样例仅在Ubuntu OS适配验证过，未在openEuler OS适配验证，推荐烧录Ubuntu OS镜像。**

步骤2 参考《**硬件接口使用指南**》”配置网络 > 配置路由器直连网络”章节，连接路由器与开发者套件以太网口（网口需DHCP模式）。

步骤3 使用手机连接与开发者套件相连的路由器网络，下载“米家”APP，登录小米账号，与小米台灯完成配对。

----结束

3 快速体验

查看智能台灯 IP 与令牌序列

步骤1 远程登录开发者套件。

1. PC和开发者套件接入路由器网络，并登录路由器管理后台查看开发者套件连接路由器的网口IP地址（方法请参见**通过路由器联网**）。
2. 使用**root**用户（密码：Mind@123）在PC端的MobaXterm远程连接工具登录开发者套件。

步骤2 执行以下命令登录已连接的小米账号，查询台灯IP与令牌序列token。

```
miiocli cloud
```

显示以下回显内容，表示开发者套件与台灯连接正常。

```
==Mijia LED Desj Lamp 1S (Enhanced Edition)（Device online）==
    Model：yeelink.light.lamp27
    Token：9f0c3ebbce287f8d0b8c2d6fb98ca9de
    IP：192.168.1.100（mac：CC:B5:D1:64:DB:CA）
    DID：590951119
    Locale：cn
```

----结束

运行代码

步骤1 获取代码。

1. 远程登录开发者套件，进入“/usr/local”目录运行脚本拉取代码。
   ```
   cd /usr/local
   ```
2. 运行脚本拉取代码。
   ```
   bash E2E_samples_download_tool.sh -d download_destination_path -s source_repository -b branch target_path
   ```
   参数说明：
   - -d：指定代码的下载路径。
   - -s：指定开源仓库的clone url。
   - -b：指定开源仓库分支名称及待下载的项目目录。
   - -f：强制更新下载路径中的目录。当样例目录已删除，但重新下载时提示“Already up to date”时可使用此参数。

   命令示例：


```
bash E2E_samples_download_tool.sh -d /home/HwHiAiUser/E2ESamples -s https://gitee.com/HUAWEI-ASCEND/ascend-devkit.git -b master src/E2E-Sample/Voice/
```

回显如下：

```
Download E2E samples successfully!
```

执行完成后，会在“/home/HwHiAiUser/E2Esamples”目录下生成“src/E2E-Sample/Voice/”目录。

代码解析参见**4 代码实现**。

步骤2 单击**链接**下载模型代码，将压缩包中“dist”文件夹上传至“/home/HwHiAiUser/E2Esamples/src/E2E-Sample/Voice/v2”目录，将“wenet”中的文件上传至“/home/HwHiAiUser/E2Esamples/src/E2E-Sample/Voice/v2/wenet”。

目录结构如下所示：

```
├── config.py
├── dist
│   ├── assets
│   │   ├── avtor-141cd8e9.jpg
│   │   ├── index-0c2196b0.css
│   │   ├── index-0f38e264.js
│   │   └── me-f369eebc.jpg
│   ├── index.html
│   └── vite.svg
├── main.py
└── wenet
    ├── model.py
    ├── offline_encoder.om
    └── vocab.txt
```

步骤3 进入代码目录执行以下命令修改文件，填写台灯IP与令牌序列Token。

```
cd /home/HwHiAiUser/E2Esamples/src/E2E-Sample/Voice/v2
vi config.py
```

输入“**i**”进入编辑模式。

```
#填写查询的IP与Token
lamp_ip = '192.168.1.100'
lamp_token = '9f0c3ebbce287f8d0b8c2d6fb98ca9de'
```

按"ESC"键退出编辑模式，输入“**:wq**”保存并退出。

步骤4 执行以下命令启动代码脚本。

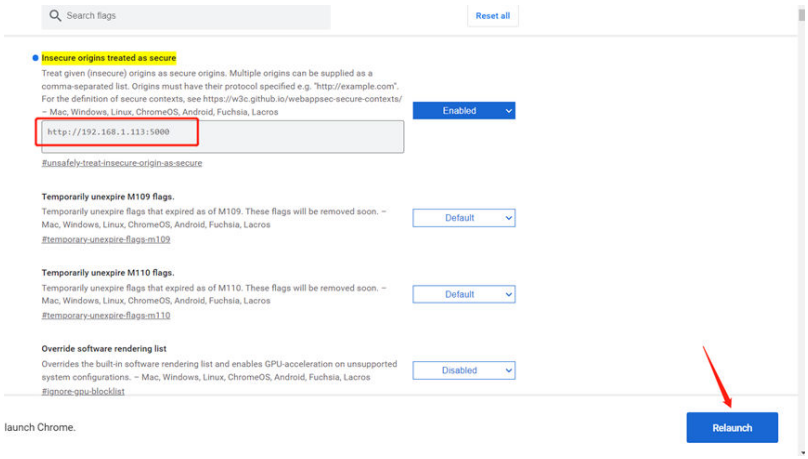
```
python main.py
```

显示以下回显内容，表示模型加载完成。

```
======= Cache cleared ========
Loading models...
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.113:5000
Press CTRL+C to quit
[INFO] acl init success
[INFO] open device 0 success
[INFO] load model wenet/offline_encoder.om success
[INFO] create model description success
asr ready
```

步骤5 打开chrome浏览器，输入“chrome://flags/#unsafely-treat-insecure-origin-as-secure”，将**步骤4**回显中加粗部分复制输入**图3-1**中所示文本框中，将选项配置为“Enabled”，单击“Relaunch”按钮重启浏览器。

图 3-1 浏览器页面



步骤6 在重启后的浏览器地址栏输入**步骤4**回显中加粗部分，按下键盘“Enter”键，进入语音交互界面。

步骤7 单击**图3-2**所示录音按钮开始录音，语音输入指令，再次单击按钮结束录音，语音执行结果参见**图3-3**。

说明

- 首次单击录音按钮浏览器会提示是否允许录音，需选择允许，提供录音权限。
- 若指令不是“开灯”或“关灯”，聊天机器人会回复“指令无法识别。当前可识别的指令有：“开灯”，“关灯”。

图 3-2 录音按钮

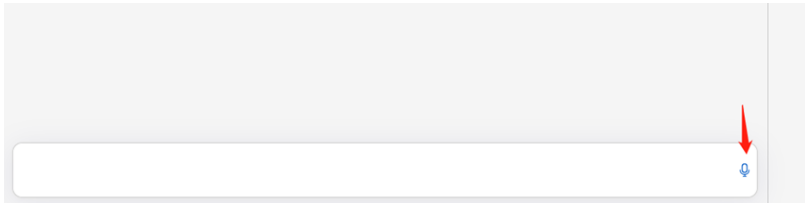
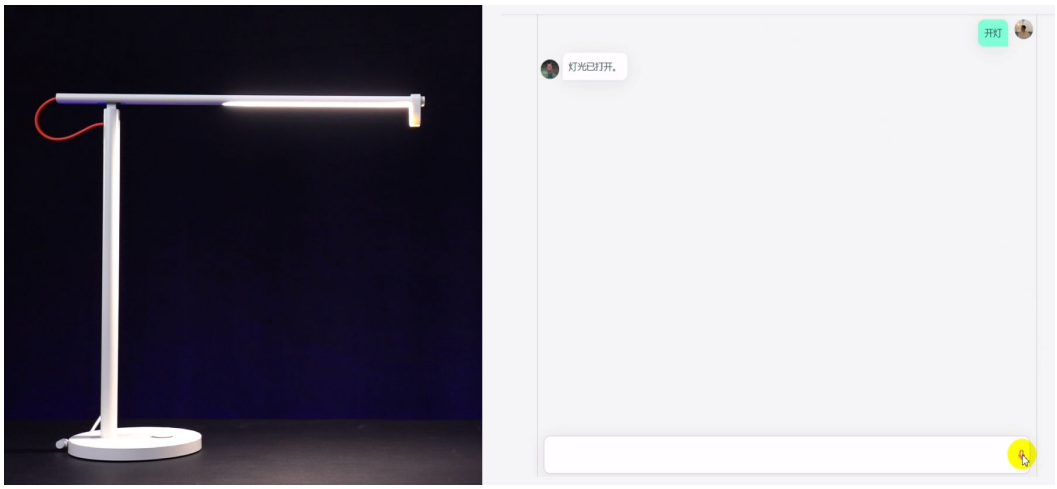
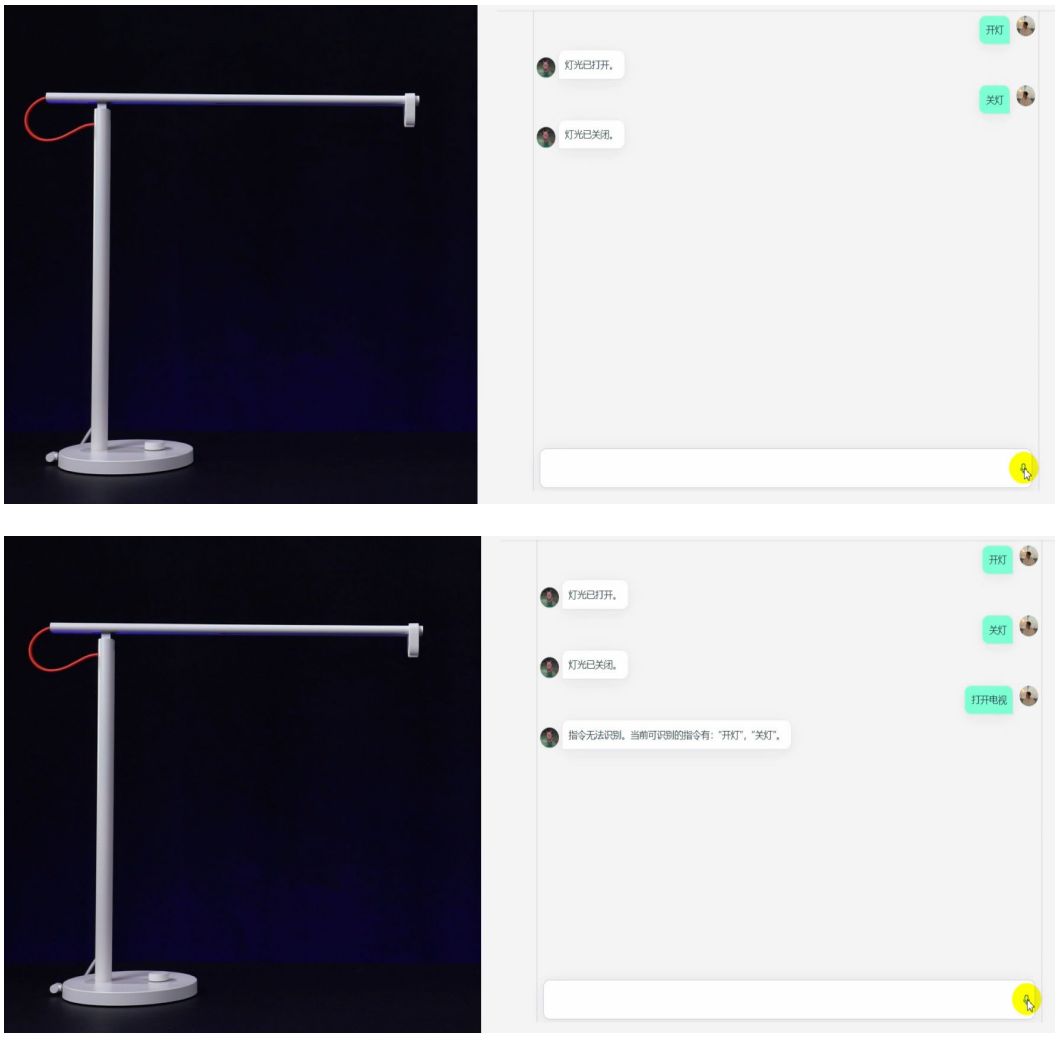


图 3-3 语音控制台灯效果展示图





步骤8 在远程登录软件命令行窗口按下键盘“Ctrl” +“C”键，即可退出程序。

----结束

4 代码实现

4.1 代码文件介绍

表 4-1 文件介绍

文件（夹）名称	说明
config.py	语音识别模型相关配置文件，包含模型文件路径、台灯的IP和token。
dist	网页前端代码存储文件夹。
main.py	语音识别逻辑运行代码脚本。
wenet	WeNet语音识别模型的相关文件。

4.2 台灯控制逻辑代码

“main.py”中的Python类LampController由两个函数组成。

- run_asr函数负责接收前端用户输入的录音文件，进行语音识别模型推理，并返回识别文本。
- process_lamp_cmd负责接收语音识别得到的文本，并将文本转换为控制指令发送给台灯。

```
class LampController:
    @staticmethod
    def run_asr(input_queue, output_queue):
        """语音识别模型推理，并返回识别文本。"""
```



```
        print('Loading models...')
        model = WeNetASR(wenet_model_path, wenet_vocab_path)
        print('asr ready')
        while True:
            if input_queue.empty():
                continue
            input_wav_file = input_queue.get()
            text = model.transcribe(input_wav_file)
            output_queue.put(text)

    @staticmethod
    def process_lamp_cmd(input_queue, output_queue):
        """接收语音识别文本，发送台灯控制指令。"""
        while True:
            if input_queue.empty():
                continue
            asr_output_txt = input_queue.get()

            cmd = None
            if asr_output_txt == '开灯':
                msg = '灯光已打开。'
                cmd = f'miiocli yeelight --ip {lamp_ip} --token {lamp_token} on'
            elif asr_output_txt == '关灯':
                msg = '灯光已关闭。'
                cmd = f'miiocli yeelight --ip {lamp_ip} --token {lamp_token} off'
            else:
                msg = '指令无法识别。当前可识别的指令有：“开灯”，“关灯”。'

            if cmd is not None:
                os.system(cmd)

            response = {
                "code": 200,
                "data": {
                    "isEnd": True,
                    "message": msg
                }
            }
            output_queue.put(response)
```

4.3 台灯控制执行代码

main.py中的执行入口“if __name__ == '__main__'”分别启动两个进程，一个负责处理台灯指令，一个负责语音识别模型推理，再创建并运行一个flask应用，就能够在网页端运行完整的语音识别控制程序。

```
if __name__ == '__main__':
    # 清理音频缓存文件
    clear_cache()

    # 启动台灯指令处理进程
    queue_sz = 1
    controller = LampController()
    lamp_cmd_in_q = Queue(queue_sz)
    lamp_cmd_out_q = Queue(queue_sz)
    lamp_process = Process(target=controller.process_lamp_cmd,
                           args=(lamp_cmd_in_q, lamp_cmd_out_q))
    lamp_process.start()

    # 启动语音识别模型推理进程
    asr_in_q = Queue(queue_sz)
    asr_out_q = Queue(queue_sz)
    asr_process = Process(target=controller.run_asr,
                          args=(asr_in_q, asr_out_q))
    asr_process.start()

    # 创建并运行flask应用
```



```
app = create_flask_app()
app.run(
    host="0.0.0.0",
    port=5000
)
```

4.4 模型推理代码

“wenet/model.py”中的python类WeNetASR封装了WeNet模型推理的主要代码。

- transcribe是推理主函数，将输入的录音文件经模型推理转换为文本。
- preprocess函数用于执行音频数据预处理，返回音频特征作为模型输入。
- post_process函数用于模型推理结果后处理，将数值型推理结果转换为文本输出。

```
class WeNetASR:
    def __init__(self, model_path, vocab_path):
        self.vocabulary = load_vocab(vocab_path)
        self.model = InferSession(0, model_path)
        self.max_len = self.model.get_inputs()[0].shape[1]

    def transcribe(self, wav_file):
        """执行模型推理，将录音文件转为文本。"""
        feats_pad, feats_lengths = self.preprocess(wav_file)
        output = self.model.infer([feats_pad, feats_lengths])
        txt = self.post_process(output)
        return txt

    def preprocess(self, wav_file):
        """数据预处理"""
        waveform, sample_rate = torchaudio.load(wav_file)
        # 音频重采样，采样率16000
        waveform, sample_rate = resample(waveform, sample_rate, resample_rate=16000)
        # 计算fbank特征
        feature = compute_fbank(waveform, sample_rate)
        feats_lengths = np.array([feature.shape[0]]).astype(np.int32)
        # 对输入特征进行padding，使符合模型输入尺寸
        feats_pad = pad_sequence(feature,
                                 batch_first=True,
                                 padding_value=0,
                                 max_len=self.max_len)
        feats_pad = feats_pad.numpy().astype(np.float32)
        return feats_pad, feats_lengths

    def post_process(self, output):
        """对模型推理结果进行后处理"""
        encoder_out, encoder_out_lens, ctc_log_probs, \
            beam_log_probs, beam_log_probs_idx = output
        batch_size = beam_log_probs.shape[0]

        num_processes = batch_size
        log_probs_idx = beam_log_probs_idx[:, :, 0]
        batch_sents = []
        for idx, seq in enumerate(log_probs_idx):
            batch_sents.append(seq[:encoder_out_lens[idx]].tolist())
        # 根据预置的标签字典将推理结果转换为文本
        txt = map_batch(batch_sents, self.vocabulary, num_processes, True, 0)[0]
        return txt
```

