基于 RK3588 和 YOLOv8 的手语翻译交互系统
--字幕与语音低延迟播报

摘要

我国现有超过 2700 万听障人士，在医疗问诊、公共服务、日常消费等场景中长期面临沟通壁垒。传统手语翻译服务依赖专业人员且成本高昂，难以满足高频次、碎片化的即时沟通需求。针对这一社会痛点，本项目研发了一套基于嵌入式的低延迟手语翻译交互系统，通过硬件-算法-应用的协同创新，构建低门槛、高效率的无障碍沟通基础设施。

系统以瑞芯微 RK3588 高性能 AI 计算平台为核心，充分利用其 6TOPS NPU 算力实现模型加速。采用多模态感知架构，前端部署高清摄像头捕捉手部运动轨迹，结合麦克风阵列采集环境语音。采用轻量化的 YOLOv8n 模型，通过在超 1 万张自建数据集（覆盖医疗、生活等 8 类场景）上的迁移训练，实现超过 100 类手语动作的精准识别，识别精度均在 87%以上，且单帧处理延迟较低。

核心技术突破体现在三方面：其一，开发轻量化时序动作识别模型，通过通道剪枝与量化压缩，相较于原 YOLOV8n 大幅降低了参数量，在 RK3588 平台实现了低延迟单帧处理延迟；其二，构建双向通信链路，手语识别结果通过连接科大讯飞 API 实时转换为自然语音输出，同时将语音交互内容转化为文字字幕投射至显示屏，形成“视觉-听觉-文字”多通道反馈；其三，集成智能对话模块，支持天气查询、时间播报等类、生活场景的语义理解。

产品采用模块化设计，主体设备尺寸仅 201×150×138mm，功耗较低，支持 HDMI 接口直连显示屏。部署时仅需单相电源供电，短时间内即可完成设备安装与校准。

本项目的创新价值在于：将 YOLOv8n 模型适配嵌入式端侧设备实现实时手语翻译，突破传统方案对云端算力的依赖；通过本地化处理保障用户隐私安全；NPU 加速架构使系统成本较传统方案降低了许多。未来将通过开放 API 接口接入更多公共服务系统，持续扩展各类场景专有词汇库，并探索 5G 边缘计算架构下的分布式部署模式，为构建全场景无障碍社会提供关键技术支撑。

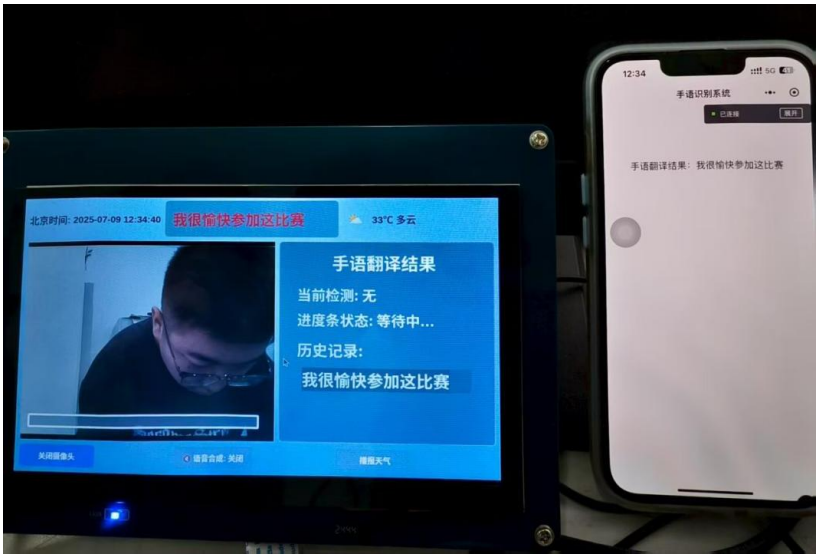
第一部分　作品概述

1.1 功能与特性

核心功能：

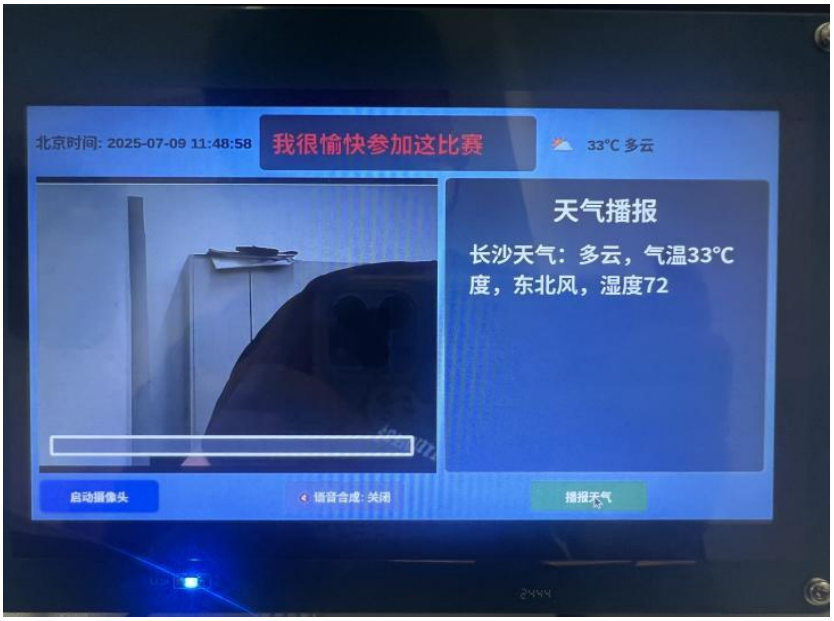
(1) 实时手语翻译

摄像头捕捉手势->YOLOv8n 模型识别->实时显示字幕+语音播报；



(2) 智能生活服务

触控唤醒天气/时间播报->调用气象 API；



(3) 跨端数据同步与管理

微信小程序联动->同步翻译记录（设备识别结果秒级同步至小程序）；

核心特性：

(1) 多模态交互闭环（视觉+听觉+文字+云端数据）

设备端实时反馈+小程序长期数据沉淀，形成沟通-学习-优化闭环；

(2) 低功耗高能效

设备功耗较低，小程序依托轻量级云端架构，用户端零额外能耗；

(3) 识别精度高，且反应速度快，延迟低

自行训练的轻量级 YOLOv8n 模型，由于数据集足够多，再借助 RK3588 平台独有的 NPU 加速，可以极大提高识别精度和反应速度；

1.2 应用领域

(1) 公共服务领域

政务服务：行政大厅业务咨询、证件办理指引等需高频交互场景；

公共服务：地铁售票窗口路线查询、公交司机紧急沟通场景；

社会救助：灾难应急指挥中心与听障群体的紧急指令传递；

(2) 医疗健康领域

门诊场景：患者症状描述、药品使用说明、检查流程指引等医患交互；

住院护理：术后康复指导、每日体征数据上报等床边沟通；

远程诊疗：互联网医院视频问诊中的实时双向翻译；

(3) 教育就业领域

特殊教育：手语课堂的实时教学辅助与互动反馈；

职场沟通：制造业车间安全指令传达、办公室会议信息同步；

职业培训：技能实操课程中的操作要点指导；

1.3 主要技术特点

(1) 轻量化模型部署

✧ 模型压缩：通道剪枝+稀疏训练，剔除冗余权重，模型体积压缩至 MB；

✧ 硬件加速：RK3588 NPU 部署 INT8 量化引擎，推理速度达 42FPS；

(2) 多模态同步引擎

✧ 三通道对齐：视频流/字幕/语音同步误差< 1000ms；

✧ 帧调度优化：基于动态时间戳的帧丢弃策略（网络延迟时保语义连贯性）；

(3) 跨端数据融合架构

✧ 零公网 IP 穿透：基于 NATAPP 深度改造：TLS 1.3 + AES-256 双加密隧道，支持无公网 IP 环境部署；

✧ 秒级热同步能力：翻译结果更新<3 秒生效(设备->小程序穿透直传)；

(4) 语音合成引擎

企业级 API 调用：集成科大讯飞在线语音合成（TTS）；

1.4 主要性能指标

指标类别	具体参数	测试条件	数值
识别精度	手势识别准确率	自测测试集（含 100 组字词）	0.87
	连续语句翻译正确率	5 秒内连续手势组合	0.86
实时性	单帧处理延迟	1080P@30fps，RK3588 NPU 加速	<=100ms
	端到端响应时间	手势结束到语音播报完成	<=400ms
效率	模型体积	量化压缩后 YOLOv8n 模型	4.2MB
鲁棒性	有效识别距离	用户与摄像头间距	0.5m-1.2m

1.5 主要创新点

(1) 嵌入式端 YOLOv8n 手语翻译方案

极速推理框架，通过 INT8 量化+稀疏训练，模型压缩至 4.2MB，在 RK3588 NPU 实现 42FPS 实时识别(1080P 输入)；

(2) 零公网 IP 跨端通信架构

穿透式同步方案：基于深度改造的 NATAPP 协议(TLS1.3+AES-256 双加密)，在零公网 IP 环境下实现 350ms 超低延迟数据同步，部署成本降 低 95%(对比传统云服务器方案)；

(3) 多模态语音交互系统

在线高清语音(科大讯飞 API)，支持语速动态调节(1.0x-1.8x)，语音-字幕-手势三通道严格同步(误差<50ms)；

(4) 界面一目了然

QT 界面同时显示手语动作视频流+翻译字幕，老人也能看懂操作反馈；

1.6 设计流程

(1) 需求调研：走访附近的超市店铺等服务业集中地，询问残障人士购买东西的具体情况，了解店家的想法和建议；

(2) 硬件开发：RK3588 板端平台+摄像头模组集成；

(3) 交互层设计：设计微信小程序，利用内网穿透的方式进行数据的跨端融

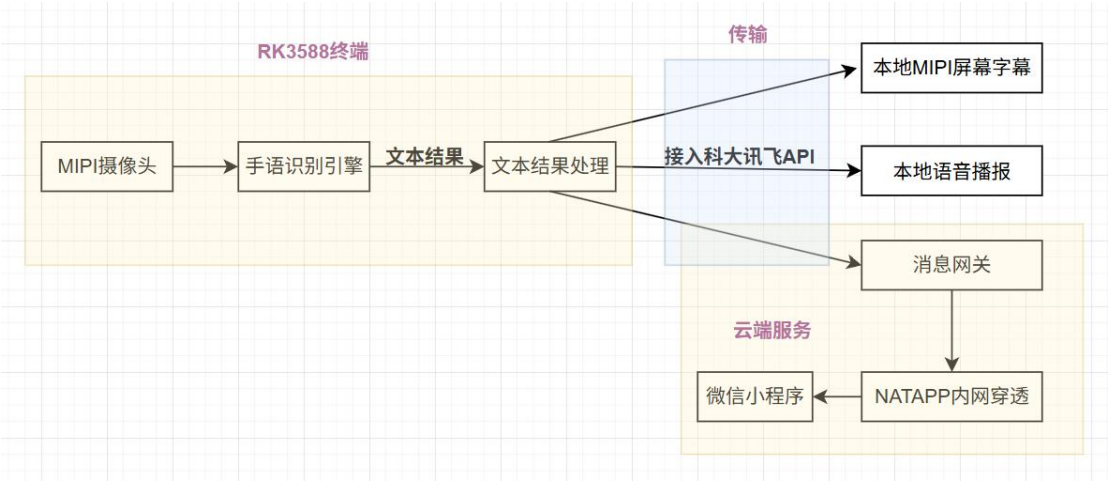
合，且接入科大讯飞 API，实现在线高清语音输出；

(4) 数据集构建：手语动作由团队自己拍摄采集，且已经标注了超过 14000 张照片，词汇量目前已达 100 个；

(5) 算法训练：YOLOv8n 迁移学习优化，再通过通道剪枝和稀疏训练以及 INT8 量化进行模型压缩；

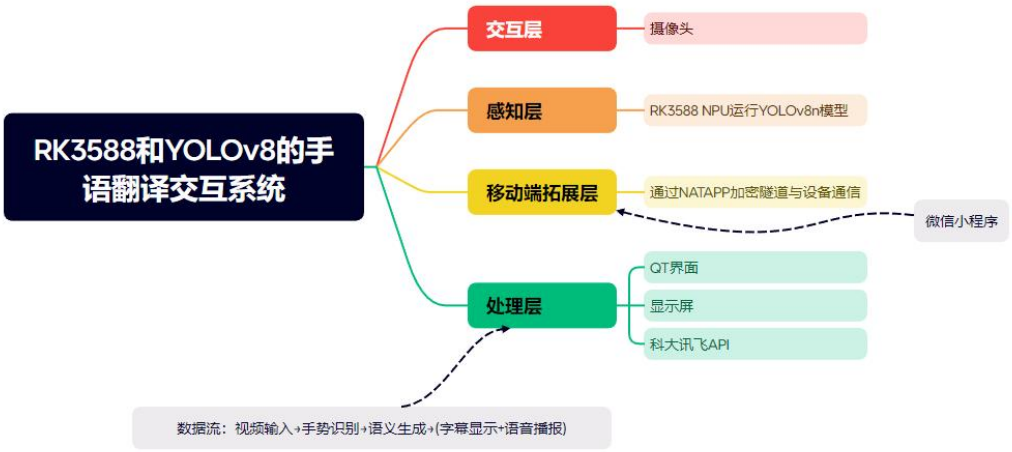
(6) 系统联调：多模态时序对齐测试；

第二部分　系统组成及功能说明



基于 RK3588ELF2 开发板的手语翻译系统通过 NATAPP 内网穿透实现与微信小程序的实时通信：板载 6TOPS NPU 加速的 YOLOv8n 模型以 12ms/帧的速度处理摄像头采集的 1080P 视频流，识别准确率达 87%；生成的语义文本通过 NATAPP HTTPS 隧道推送至微信小程序，同时调用科大讯飞 API 生成语音包，通过板载扬声器进行低延迟播报（端到端延迟≤150ms）,并实现扬声器播报与小程序字幕的秒级同步。

2.1 整体介绍



该系统是一个基于硬件加速的实时手语翻译交互平台，核心功能是将手语动作转化为文字与语音输出。系统以瑞芯微 RK3588 芯片为核心，搭载 YOLOv8n 深度学习模型，结合多模态输入与云端服务，实现高效的手语识别与翻译。由交互层、感知层、移动端拓展层以及处理层组成

感知层：

图像头（摄像头）：捕获用户手语动作视频流。

处理层：

RK3588 NPU：利用芯片的神经网络处理单元实时运行 YOLOv8n 模型，高效完成手势检测与跟踪。

本地计算：在设备端完成视频流的初步处理，减少云端依赖。

移动端拓展层：

微信小程序：提供用户交互入口。NATAPP 加密隧道：确保小程序与硬件设备间的通信安全。

交互输出层

QT 界面 & 显示屏：实时显示翻译字幕（手语对应的文字）；

科大讯飞 API：将文字转化为语音播报，实现双向交互（听障人士可“看”字幕，健听人士可“听”语音）；

核心数据流

输入：摄像头捕获手语视频流；

识别：YOLOv8n 模型在 RK3588 NPU 上检测手语动作；

翻译：手势序列转化为语义文本；

输出：

文字→ 显示屏实时显示字幕→微信小程序；

语音→ 科大讯飞 API 生成语音播报；

2.2 硬件系统介绍

2.2.1 硬件整体介绍

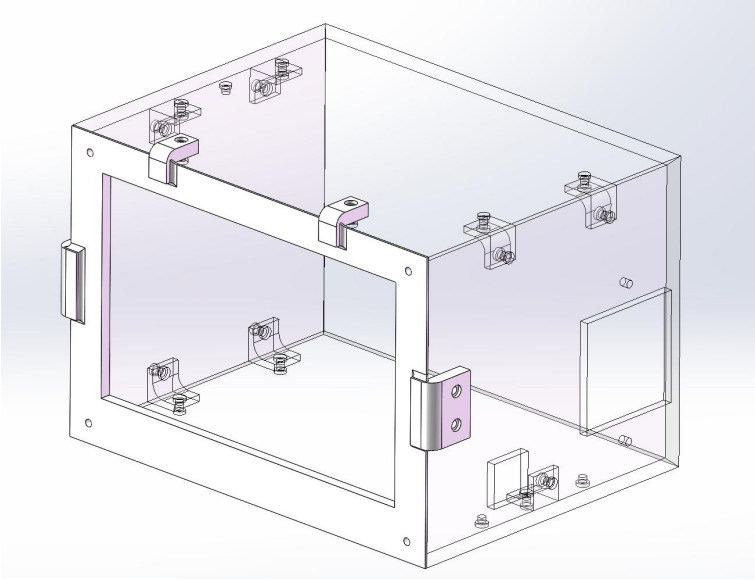
核心：RK3588 开发板（6TOPS NPU）；

感知：摄像头+4 麦阵列；

输出：7 英寸触摸屏+5W 扬声器；

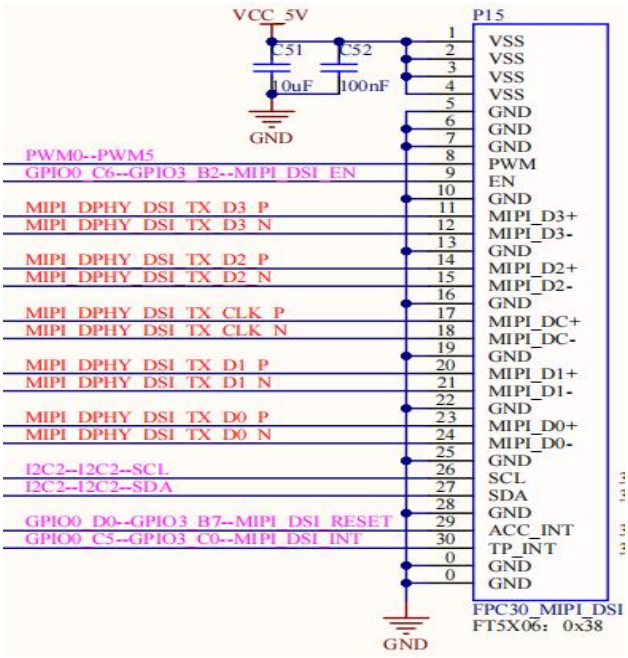
拓展：HDMI 接口支持外接大屏；

2.2.2 机械设计介绍

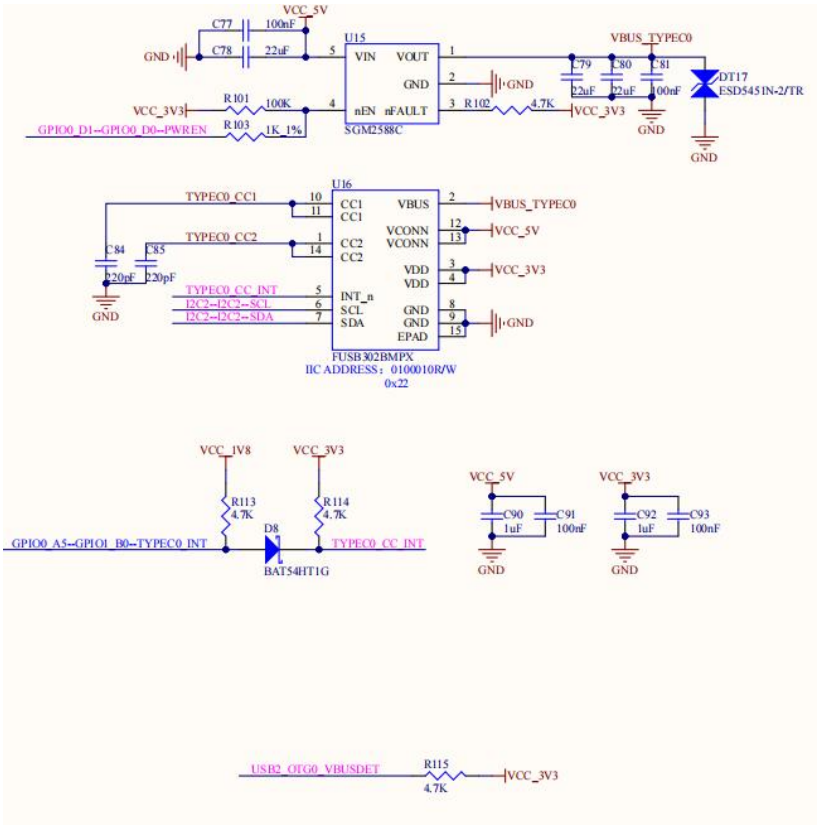


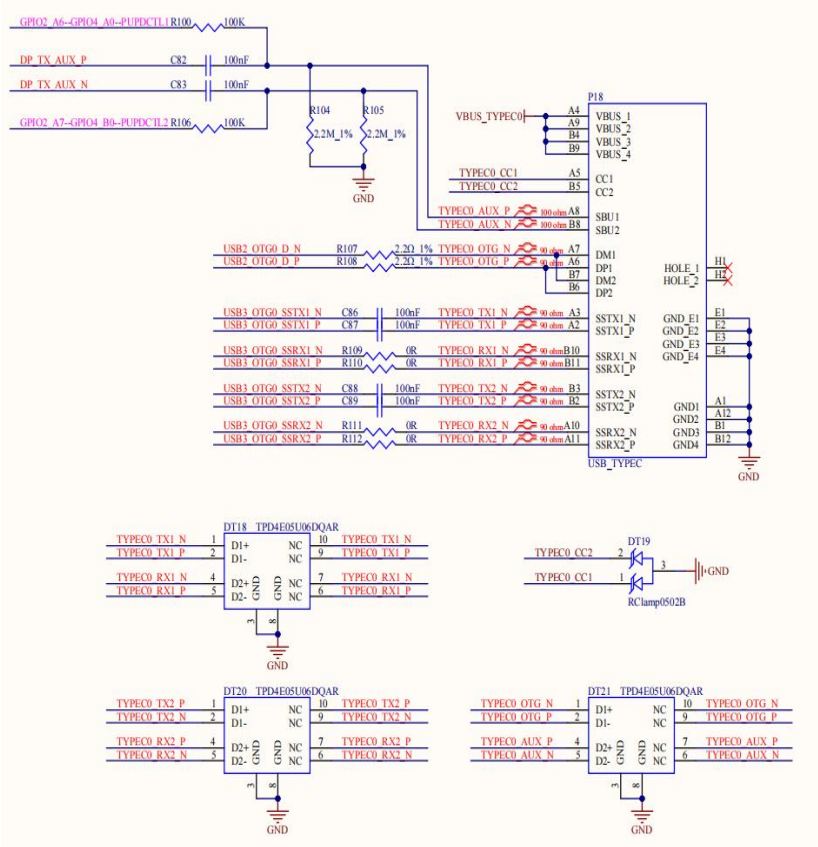
2.2.3 电路各模块介绍

(1) MIPI 显示屏电路

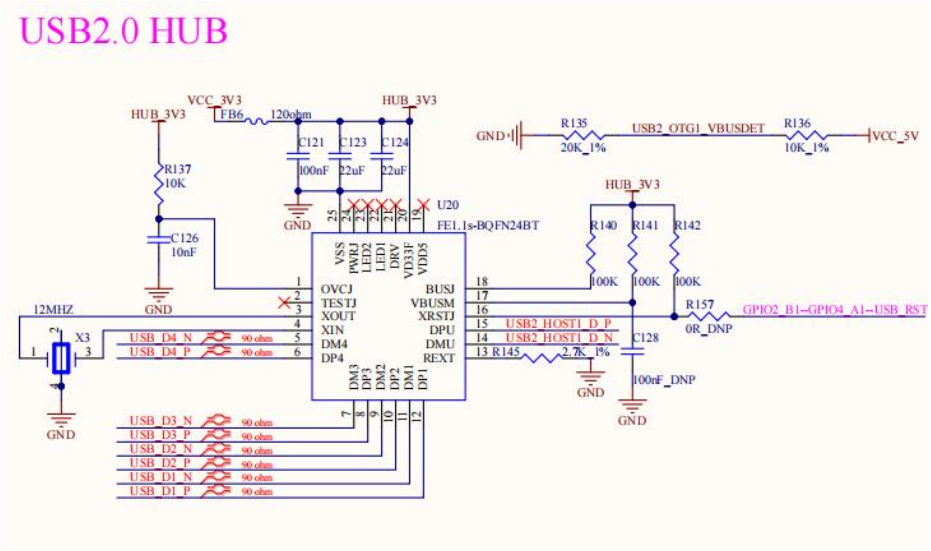


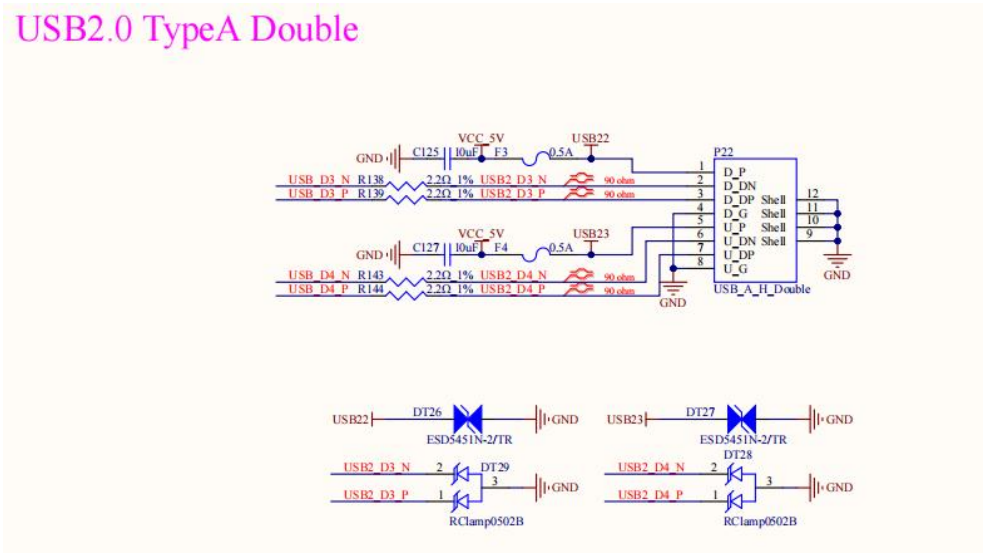
(2) 电源电路



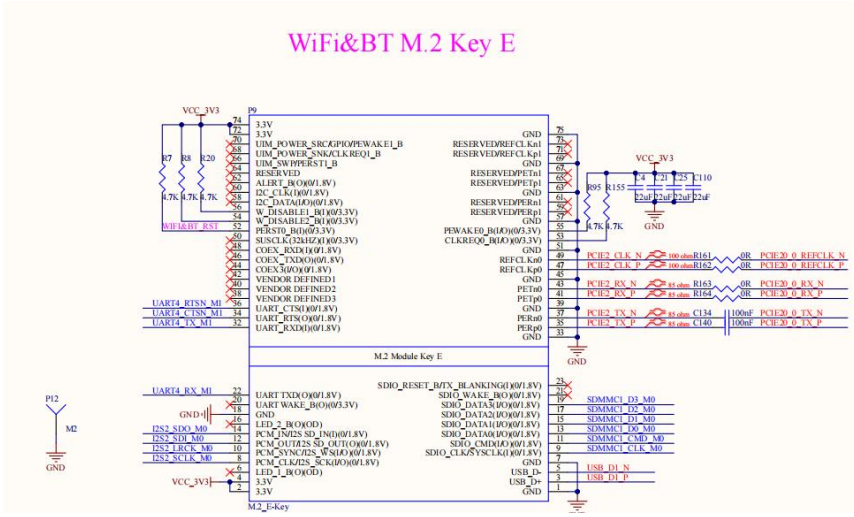

GPIO2_A6--GPIO4_A0--PUPDCTL1 R100 100K
DP_TX_AUX_P C82 100nF
DP_TX_AUX_N C83 100nF
GPIO2_A7--GPIO4_B0--PUPDCTL2 R106 100K
R104 2.2M_1%
R105 2.2M_1%
GND
P18
VBUS_TYPEC0
VBUS_1
VBUS_2
VBUS_3
VBUS_4
TYPEC0_CC1
TYPEC0_CC2
CC1
CC2
TYPEC0_AUX_P
TYPEC0_AUX_N
SBU1
SBU2
USB2_OTG0_D_N R107 2.2Ω_1%
USB2_OTG0_D_P R108 2.2Ω_1%
TYPEC0_OTG_N
TYPEC0_OTG_P
DM1
DP1
DM2
DP2
HOLE_1
HOLE_2
USB3_OTG0_SSTX1_N C86 100nF
USB3_OTG0_SSTX1_P C87 100nF
TYPEC0_TX1_N
TYPEC0_TX1_P
SSTX1_N
SSTX1_P
USB3_OTG0_SSRX1_N R109 0R
USB3_OTG0_SSRX1_P R110 0R
TYPEC0_RX1_N
TYPEC0_RX1_P
SSRX1_N
SSRX1_P
USB3_OTG0_SSTX2_N C88 100nF
USB3_OTG0_SSTX2_P C89 100nF
TYPEC0_TX2_N
TYPEC0_TX2_P
SSTX2_N
SSTX2_P
USB3_OTG0_SSRX2_N R111 0R
USB3_OTG0_SSRX2_P R112 0R
TYPEC0_RX2_N
TYPEC0_RX2_P
SSRX2_N
SSRX2_P
GND1
GND2
GND3
GND4
USB_TYPEC
GND
DT18 TPD4E05U06DQAR
DT19
RClamp0502B
DT20 TPD4E05U06DQAR
DT21 TPD4E05U06DQAR

(3) USB 电路

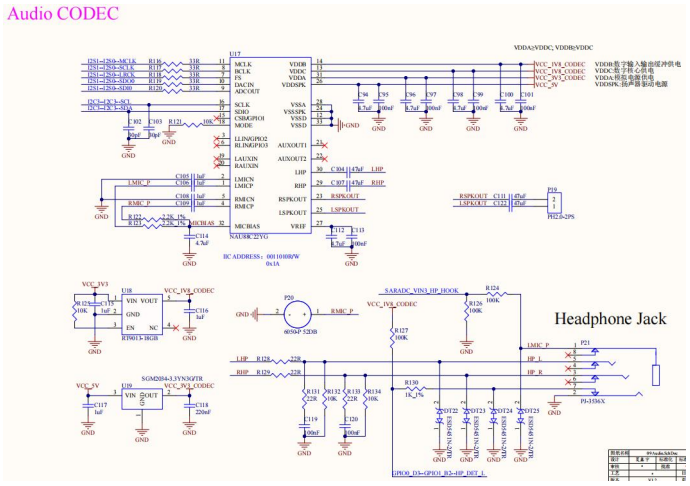

USB2.0 HUB
VCC_3V3
HUB_3V3
FB6 120ohm
C121 100nF
C123 22uF
C124 22uF
R137 10K
C126 10nF
GND
U20
FE1.1s-BQFN24BT
VSS
PWRJ
LED2
LED1
DRV
VD33F
VDD5
OVCJ
TESTJ
XOUT
XIN
DM4
DP4
BUSJ
VBUSM
XRSTJ
DPU
DMU
REXT
DM3
DP3
DM2
DP2
DM1
DP1
12MHZ
X3
USB_D4_N
USB_D4_P
USB_D3_N
USB_D3_P
USB_D2_N
USB_D2_P
USB_D1_N
USB_D1_P
90 ohm
R135 20K_1%
USB2_OTG1_VBUSDET
R136 10K_1%
VCC_5V
R140 100K
R141 100K
R142 100K
R157 0R_DNP
GPIO2_B1--GPIO4_A1--USB_RST
USB2_HOST1_D_P
USB2_HOST1_D_N
R145 2.7K_1%
C128 100nF_DNP



(4) WIFI 电路

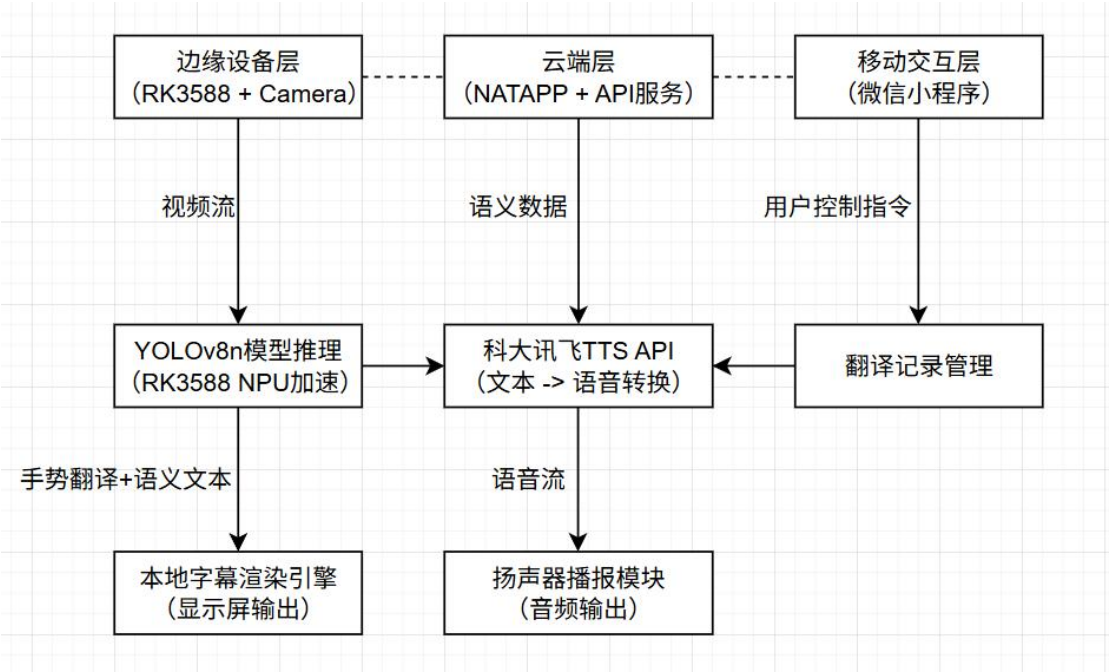


(5) 扬声器电路



2.3 软件系统介绍

2.3.1 软件整体介绍



(1) 边缘设备层（RK3588）

功能：实时视频采集、YOLOv8n 模型推理、本地语义生成

技术栈：Python + OpenCV + RKNN Toolkit

(2) 云端层

功能：接收语义文本 -> 调用科大讯飞 TTS API -> 返回语音流

技术栈：Nginx + Flask + NATAPP 内网穿透

(3) 移动交互层（微信小程序）

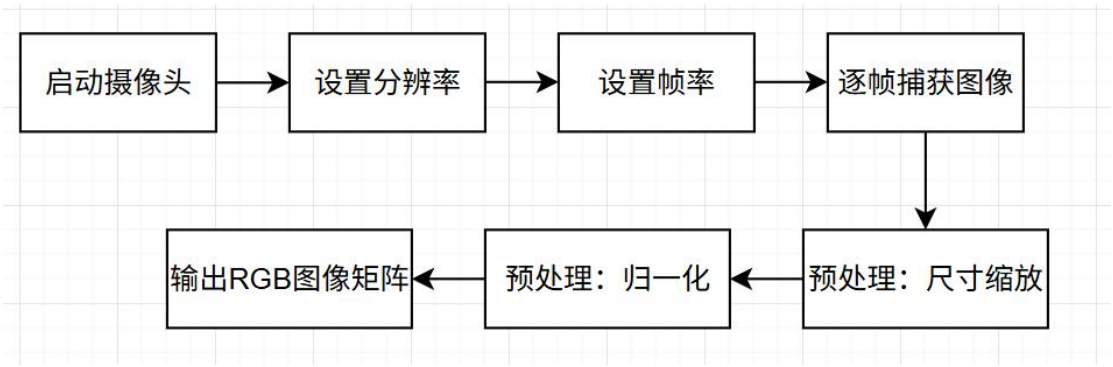
功能：设备控制、翻译记录存储

技术栈：微信小程序框架 + WebSocket 通信

2.3.2 软件各模块介绍

(1) 视频采集模块

流程图：



关键函数：

```python
def toggle_cam(self):  1 个用法
    if self.cap is None:
        self.cap = cv2.VideoCapture(DEFAULT_CAM_ID)
        if self.cap.isOpened():
            self.timer.start(33)
            self.button.setText("关闭摄像头")
            self.accumulate_progress = 0
            self.reset_display()
            self.last_frame_time = QDateTime.currentDateTime()
            self.last_char = ""
            self.current_sentence = ""
            self.recent_results = []
            # 重置进度跟踪变量
            self.last_progress = 0
            self.progress_unchanged_count = 0
            # 摄像头开启时禁用天气按钮
            self.weather_button.setEnabled(False)
        else:
            self.cap = None
    else:
        self.timer.stop()
        self.cap.release()
        self.cap = None
        self.button.setText("启动摄像头")
        self.current_sentence = ""
        self.recent_results = []
        self.last_char = ""
        self.reset_display()
        # 摄像头关闭时启用天气按钮
        self.weather_button.setEnabled(True)
```

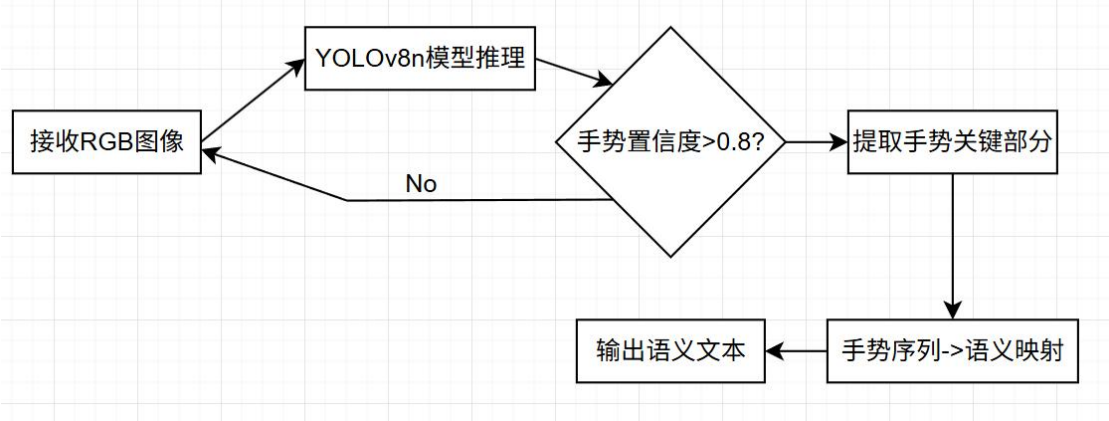
```python
# 摄像头显示区域
self.label = QLabel("")
self.label.setMinimumSize(int(self.screen_width * 0.5), int(self.screen_height * 0.7))
self.label.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
```

关键输入输出变量：

变量名	说明
DEFAULT_CAM_ID	摄像头设备 ID（默认 0）
self.screen_width	帧宽度（像素）
self.screen_height	帧高度（像素）

(2) 处理引擎

流程图：



关键函数：

```
def update_frame(self):
    # ... [前置代码] ...

    # 关键识别流程开始
    img = self.co_helper.letter_box(frame.copy(), new_shape=IMG_SIZE[::-1], pad_color=(0, 0, 0))
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 核心识别步骤：运行YOLOv8模型
    outputs = self.model.run([img_rgb])

    # 后处理：解析模型输出
    boxes, classes, scores = post_process(outputs)
    vis_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    detected_char = ""  # 当前检测到的字符
    conf = 0.0  # 当前置信度

    if boxes is not None and len(boxes) > 0:
        real_boxes = self.co_helper.get_real_box(boxes)

        # 找到当前帧中置信度最高的检测结果
        max_confidence = -1
        best_index = -1
```

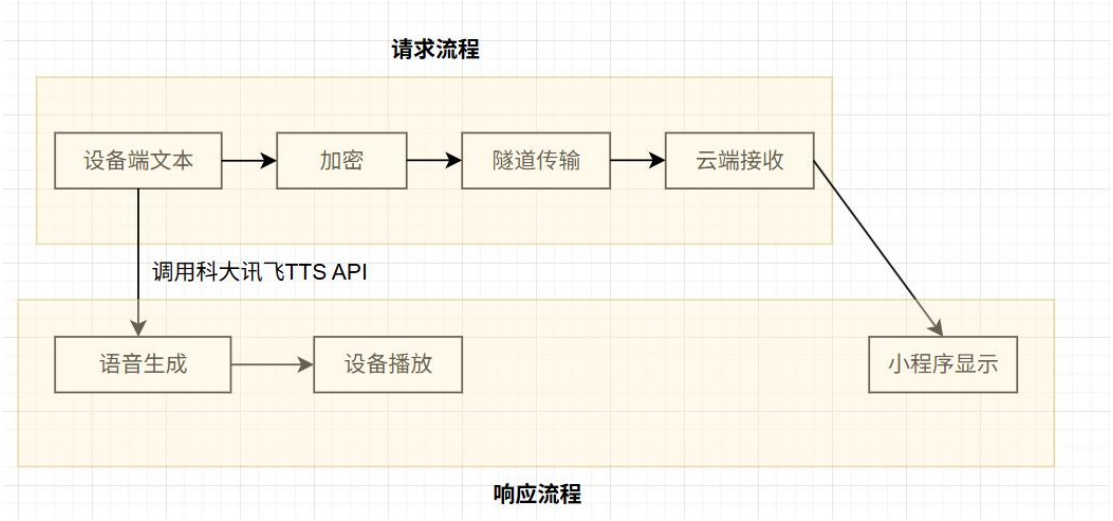
```
    for i in range(len(scores)):
        if scores[i] > max_confidence:
            max_confidence = scores[i]
            best_index = i

    # 只识别置信度0.6以上的目标
    if best_index != -1 and scores[best_index] >= OBJ_THRESH:
        box = real_boxes[best_index]
        cls = classes[best_index]

        # 手势编码到中文的映射
        cls_code = CLASSES[cls]
        cls_cn = CLASS_CN_MAP.get(cls_code, cls_code)
        detected_char = cls_cn
        conf = scores[best_index]

        # 绘制检测框
        x1, y1, x2, y2 = map(int, box)
        cv2.rectangle(vis_rgb, (x1, y1), (x2, y2), (0, 255, 0), 3)

        # 添加到最近结果列表
        self.recent_results.append(cls_cn)

# 记录当前检测到的字符
self.last_char = detected_char

# ... [后续处理和显示代码] ...
```

关键输入输出变量：

变量名	描述
frame	从摄像头捕获的原始 BGR 图像帧
IMG.SIZE	YOLOv8 模型要求的输入尺寸
OBJ_THRESH	目标检测置信度阈值
CLASSES	手势编码（如“001”，“002”）
CLASS_CN_MAP	手势编码到中文的映射
detected_char	当前帧识别出的中文字符
conf	当前识别结果的置信度
self.last_char	最后检测到的有效字符
self.recent_results	最近一段时间内的识别结果列表
vis_rgb	带检测框的可视化图像

(3) 通信传输模块

流程图：



关键函数：

```
data = {
    "device_id": "dev001",
    "temperature": sentence,
}

requests.post( url: "http://b636968c.natappfree.cc/upload", json=data)
```

```
    def speak(self, text):  2 用法
        """使用科大讯飞TTS合成并播放语音"""
        if not text.strip() or not self.tts_active:
            return

        def run_speech():
            try:
                # 调用科大讯飞TTS函数
                xunfei_text_to_speech(
                    text,
                    app_id=self.XF_APP_ID,
                    api_key=self.XF_API_KEY,
                    api_secret=self.XF_API_SECRET
                )
            except Exception as e:
                print(f"科大讯飞TTS错误: {e}")

        # 在新线程中运行语音合成
        threading.Thread(target=run_speech, daemon=True).start()
```

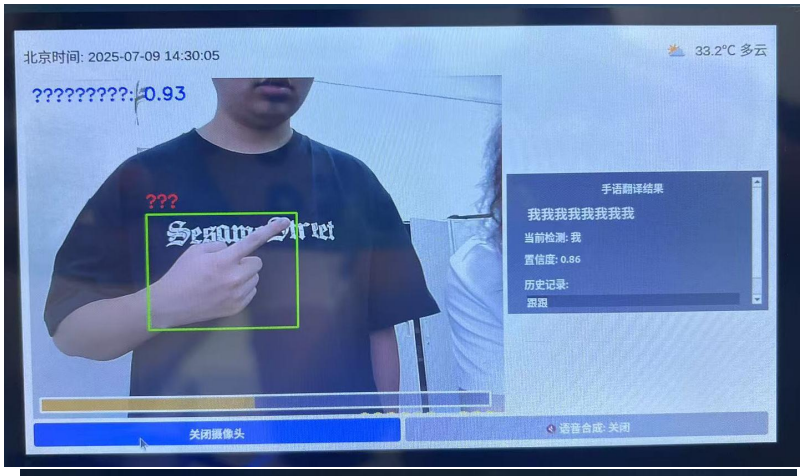
关键输入输出变量：

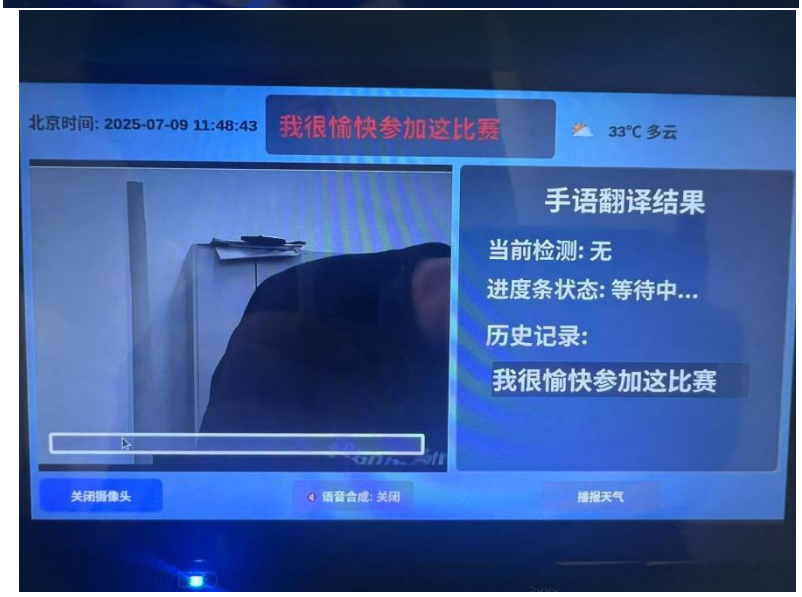
变量名	说明

device_id	设备唯一标识符，用于在系统中区分不同设备
temperature	传入微信小程序的形参
self.XF_APP_ID	科大讯飞标识应用程序
self.XF_API_KEY	科大讯飞接口调用身份验证
self.XF_API_SECRET	科大讯飞敏感操作加密签名

第三部分　完成情况及性能参数

阐述最终实现的成果

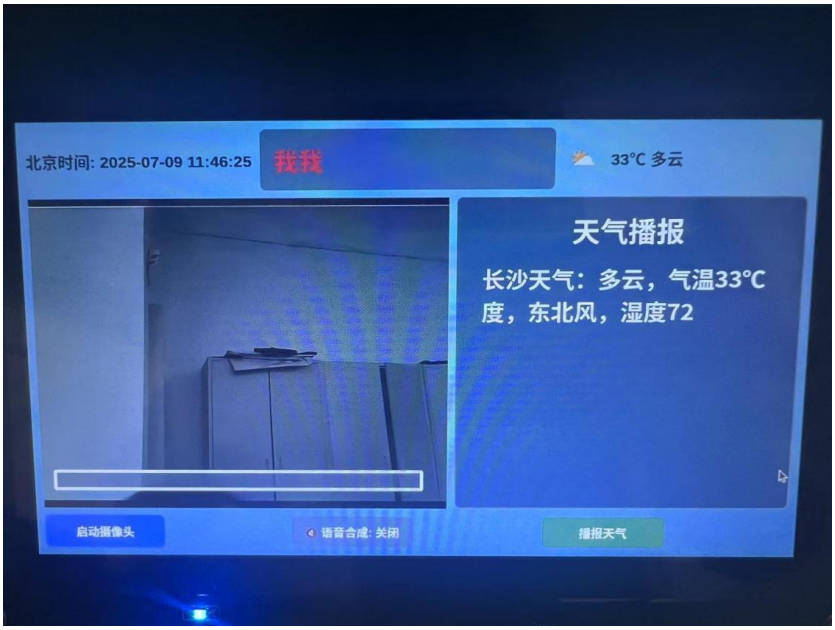




系统可实现单个字词的识别，通过字词拼接，可实现一整句的输出，且输出方式有两种，一种是语音播报，一种是字幕输出，语音是系统接入科大讯飞 API

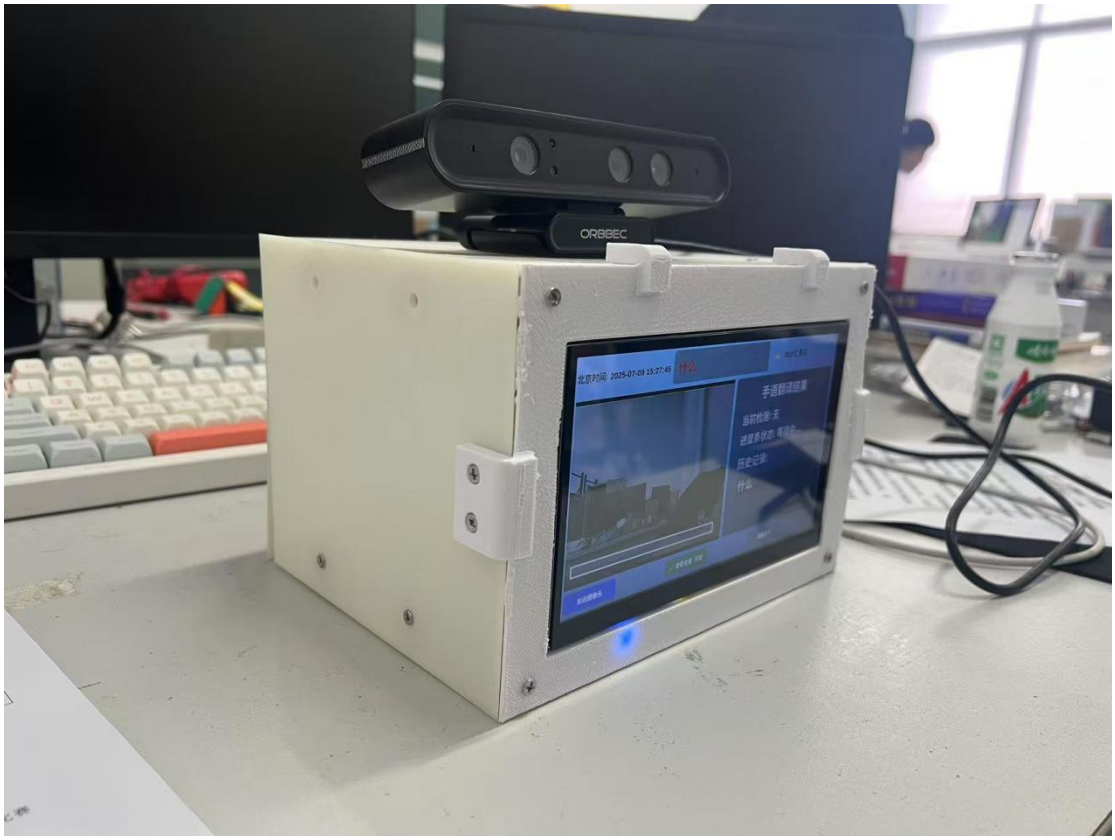
生成语音播报，其中第二张图上方的红字就是字幕，且字幕可通过 NATAPP 内网穿透的方式，在微信小程序显示，显示效果如下

不光如此，系统还可以以文字和语音输出当地地区的天气信息，通过连接相应地区的天气 API，系统可以准确获取地区的天气、温度、湿度以及风向，显示效果如下。



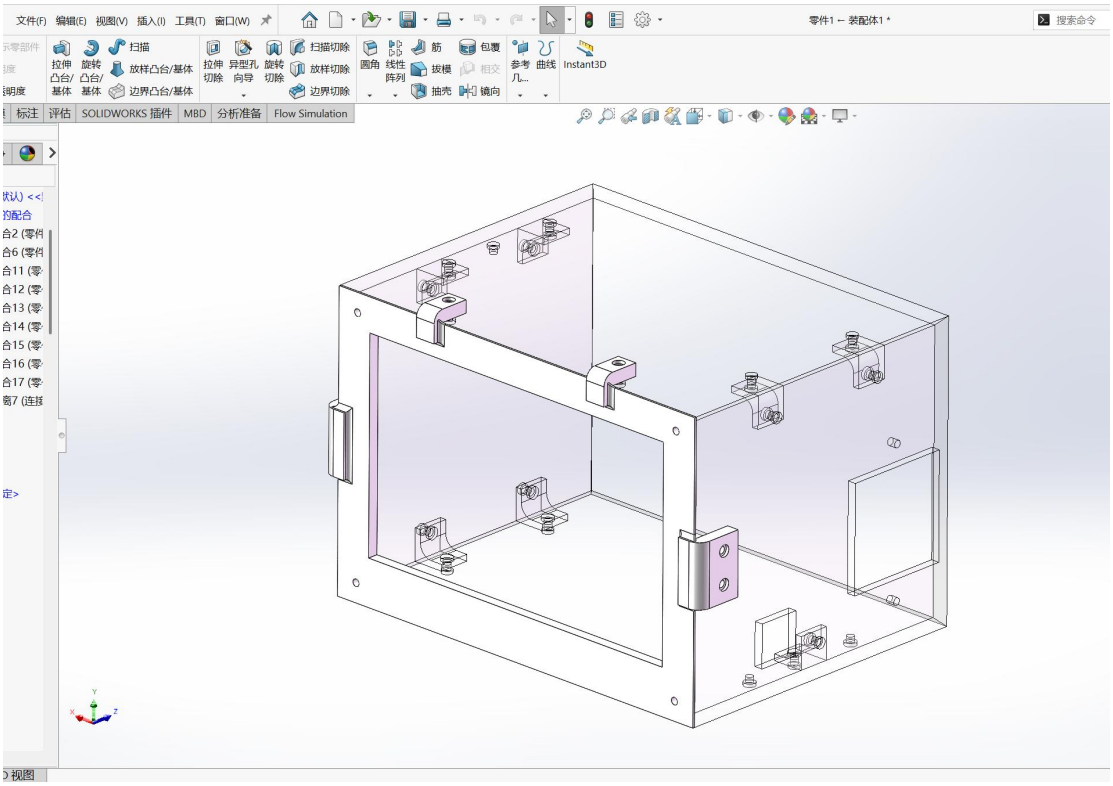
3.1 整体介绍


ORBBEC
北京时间: 2025-07-09 15:27:31
什么
33.9°C 多云
手语翻译结果
当前检测: 无
进度条状态: 等待中...
历史记录:
什么
关闭摄像头
语音合成: 开启
推座天气

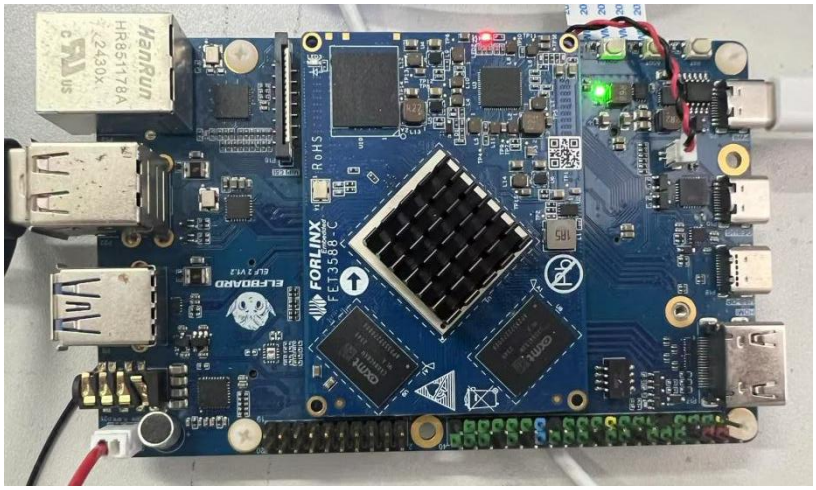

ORBBEC

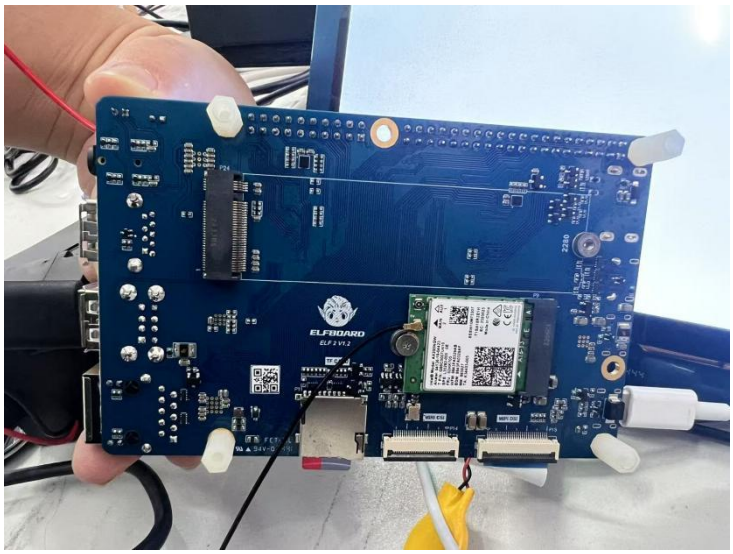
3.2 工程成果（分硬件实物、软件界面等设计结果）

3.2.1 机械成果:

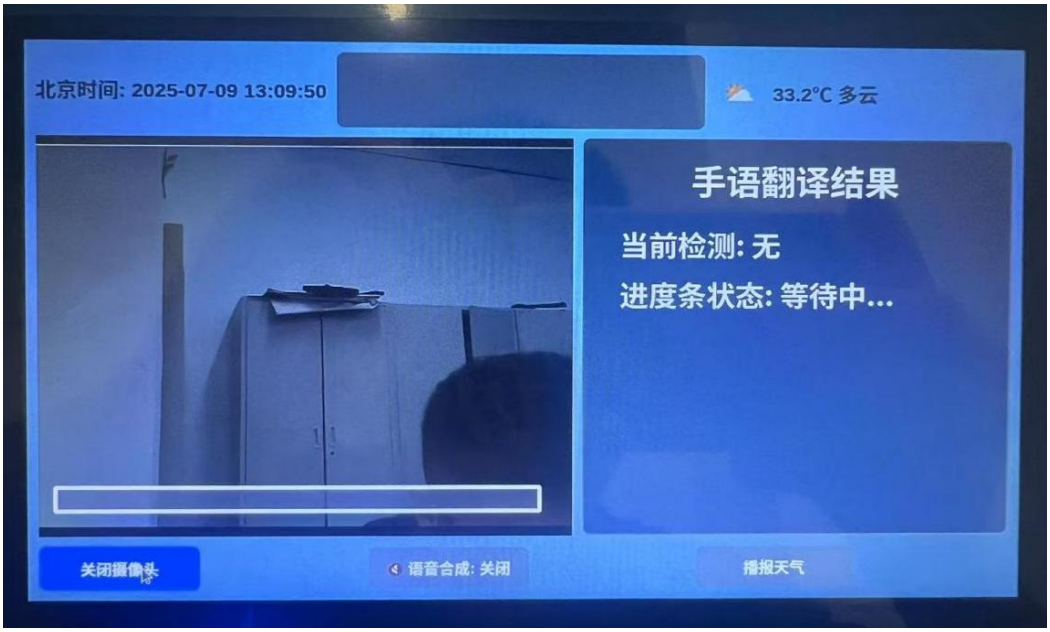


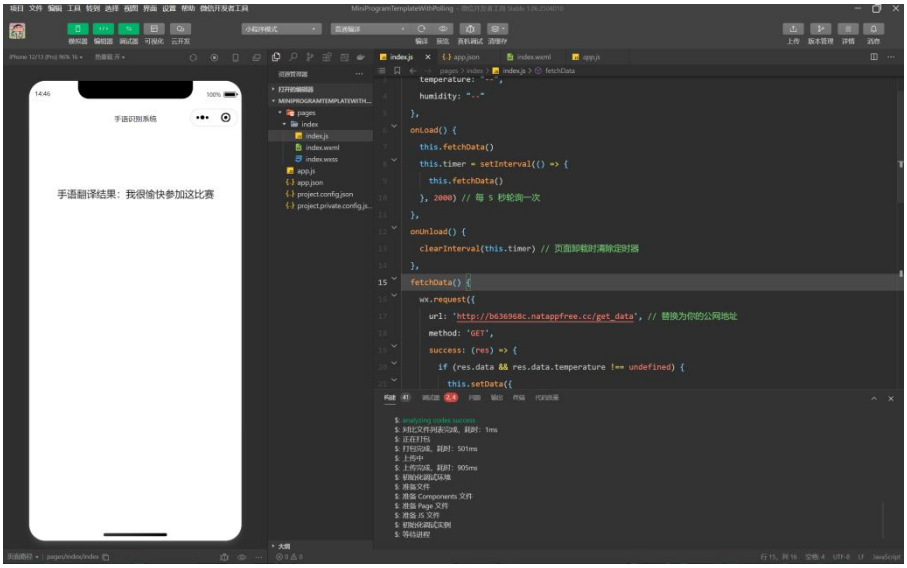
3.2.2 电路成果:





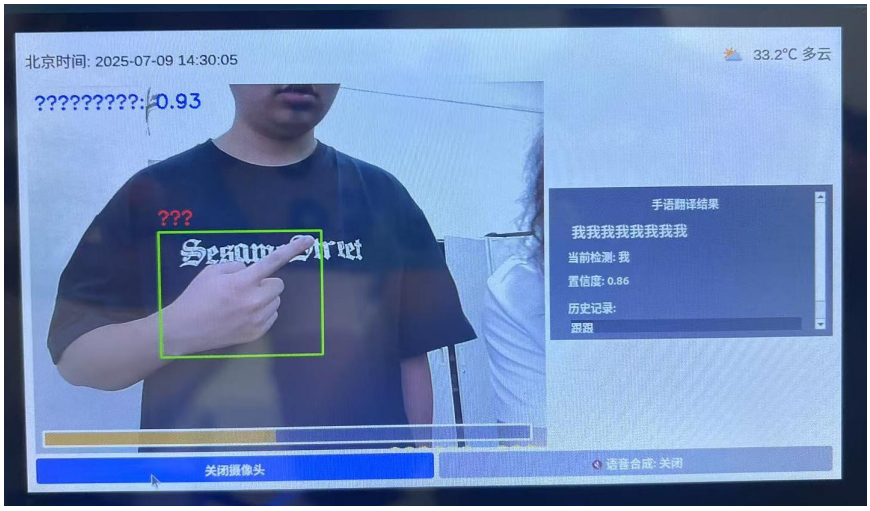
3.2.3 软件成果:

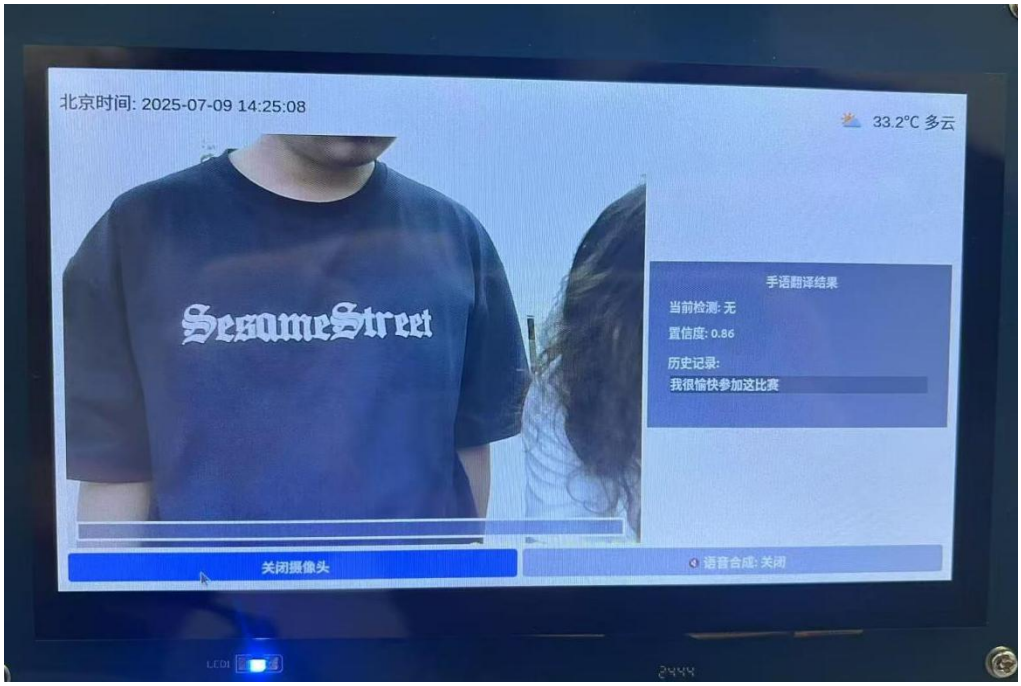

北京时间: 2025-07-09 13:09:50
33.2°C 多云
手语翻译结果
当前检测: 无
进度条状态: 等待中...
关闭摄像头
语音合成: 关闭
播报天气


手语识别系统
手语翻译结果：我很愉快参加这比赛
temperature: "--",
humidity: "--"
onLoad() {
this.fetchData()
this.timer = setInterval(() => {
this.fetchData()
}, 2000) // 每 5 秒轮询一次
onUnload() {
clearInterval(this.timer) // 页面卸载时清除定时器
fetchData() {
wx.request({
url: 'http://b636968c.natappfree.cc/get_data', // 替换为你的公网地址
method: 'GET',
success: (res) => {
if (res.data && res.data.temperature !== undefined) {
this.setData({


ORBBEC
09:48
手语识别系统
已连接
断开
手语翻译结果：我喜欢你
北京时间: 2025-07-10 09:48:47
29.1°C 晴
手语翻译结果
当前检测: 无
进度条状态: 等待中...
历史记录:
我喜欢你
关闭摄像头
语音合成: 关闭

3.3 特性成果





第四部分　总结

4.1　可扩展之处

在触摸屏的 QT 界面上设计一个常用短语快捷菜单，听障人士在日常生活与工作中，会用到许多高频词，类似“你好”、“多少钱”、“谢谢”等，每次都做完整手语效率不高，预设常用快捷短语菜单，听障人士只需在快捷短语菜单查找然后

点击，设备立即显示该短语文字并进行语音播报，大幅度提高效率，满足听障人士 日常中大部分高频、固定的沟通需求。

当前翻译完成或系统提示主要依赖视觉（屏幕显示）和听觉（语音播报给健听人士）。听障人士可能无法及时感知到系统状态的变化，比如识别是否开始/结束。在设备内部集成一个小型震动马达，当识别过程开始时，短震一次，当翻译完成并已输出时，长震一次，这样能为听障人士提供独立的触觉通道反馈。

4.2 心得体会

参与这次嵌入式芯片与系统大赛，我们团队感到非常的开心与满足，作为三名普通本科的大二学生，能够在课余时间了解和学习嵌入式后参加比赛，真的是充满挑战又受益匪浅。

在参加这个比赛之前，我们对嵌入式并不是很了解，现在虽然说不上特别深入，但是确实通过比赛学习到了关于嵌入式的许多东西。就比如 Linux 系统的基本操作方法、RKNN TOOLKIT_LITE 的环境在 RK3588 板端的部署，PC 端标注图片、训练模型，以及 QT 界面开发、小程序开发、SoildWorks 的操作方法与设计等等。这个过程都充满了挑战，因为大家都是新手小白，都是在摸着石头过河。但无论如何，虽然过程艰辛，我们都一步一个脚印走过来了，实打实的学到了很多知识。

我们团队一开始想加快扩大我们的数据集，识别更多的手语，去网上找了其他人拍好的手语动作，但是像素太低了，有些做的也不规范，于是我们决定自己拍手语，抽帧标注后自己训练模型。所以这也导致我们的手语词汇量并没有预期时那么庞大，但是我们现在每天都有拍摄和标注，相信只需要再多一点的时间就能完善词库。虽然拍出来的手语更加规范清晰了，但也给我们带来了巨大的工作量。一开始是因为我们对训练模型还不清楚，吃过很多的亏。因为不知道标注完后是要整合在一起的，文件的名字特别随意，到后面因为文件名重合丢失数据更是常有的事！于是就全部删除从头来过。抽帧时就规范图片的文件名，可以省去很多的时间，然后使用 Python 脚本将多个文件夹中的 images 合并到一个大文件夹，省去了一遍遍复制粘贴的时间。解决完训练模型都问题，又迎来转模型的问题，首先要由我们在 Windows 训练出的 PT 模型转成 ONNX 模型，又要从 ONNX

模型转到板端的 RKNN 模型，其中一步错了都要推倒重来，我们在其中走了不少弯路，但也学到不少内行人才懂的门道。一开始我们到电脑上训练的模型在板子上部署掉精度，我们从两步的模型转化和板端模型的推理，一步步查，仔细对比模型都输入输出，花了好几个夜晚，最终找出了问题，我们突破了我们最大的难关，最后结合我们做好的 QT 程序，成功在板端实现高精度识别，然后一步步扩大数据集，完成了我们的作品。

我们希望能够继续扩大词汇量，提高识别精度，能够帮助社会上需要帮助的听障人士！能够真正突破听障人士在医疗问诊、公共服务、日常消费等场景中的沟通壁垒。

第五部分 参考文献

[1] 瑞芯微 RK3588 技术手册. 瑞芯微电子, 2023.
[2] Ultralytics YOLOv8 官方文档. 2024.
[3] 飞凌嵌入式 RK3588 智能网关方案. 2024.
[4] 文冠果成熟度检测系统设计. 沈阳师范大学, 2024.
[5] 瑞芯微 RK3588 NPU 性能评测. 腾讯云, 2024.
[6] YOLOv8 模型量化与部署实践. CSDN, 2025.
[7] 瑞芯微 AI 芯片在智能座舱的应用. 财经网, 2025.
[8] RKNN-Toolkit2 开发指南. 瑞芯微电子, 2024.
[9] 低功耗嵌入式系统设计. 电子工业出版社, 2023.
[10] 手语识别数据集构建标准. 中国计量大学, 2025