Arm 中国生态技术市场经理
郑亚斌

一、概述

随着边缘计算和深度学习领域的不断发展，越来越多的端侧 AI 设备开始出现在我们的视野中。本次提出的这一方案着眼于边缘计算与深度学习场景，提出了一款应用于无人值守的仓储、居民社区或危险禁入区域的智能监控方案，具有成本低、功耗低等优势，可以大量投放且基于具体场景需求修改模型功能。此处以人体检测为例，对于长时间徘徊于某区域的行人进行智能识别，并且将异常现象及时上报云端，从而实现边缘 AI，TencentOS-tiny 以及腾讯云完美结合的智能安防案例。

二、系统的创新点

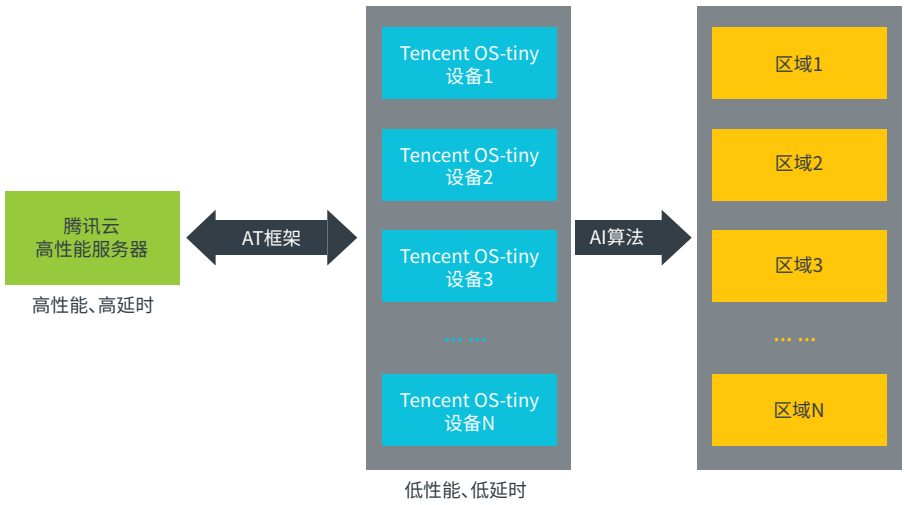
1. **本系统采用了 TencentOS-tiny 这一物联网实时操作系统来实现片上资源的智能化管理。**一方面它可以灵活高效的管理系统的片上资源，为系统并发执行多个任务提供了基础；另一方面，搭载 TencentOS-tiny 的端侧系统可以方便的与腾讯云对接，依托于腾讯云的丰富资源为端云结合带来了更多的可能性。

2. **本系统搭载了 Tensorflow Lite Micro 超低功耗深度学习 AI 推理框架以及 Arm CMSIS-NN 加速库。**Tensorflow Lite Micro 是 Tensorflow 针对微控制器应用场景所专门设计的深度学习推理框架。它占用的资源少，运行时内存最低只需要 16KB。同时其依托于 Tensorflow 平台强大的生态背景，使得更多开发者可以方便的集成、使用 AI 算法。Tensorflow Lite Micro 通过使能 Arm 开源加速库 CMSIS-NN，为端侧带来人工智能的新活力。

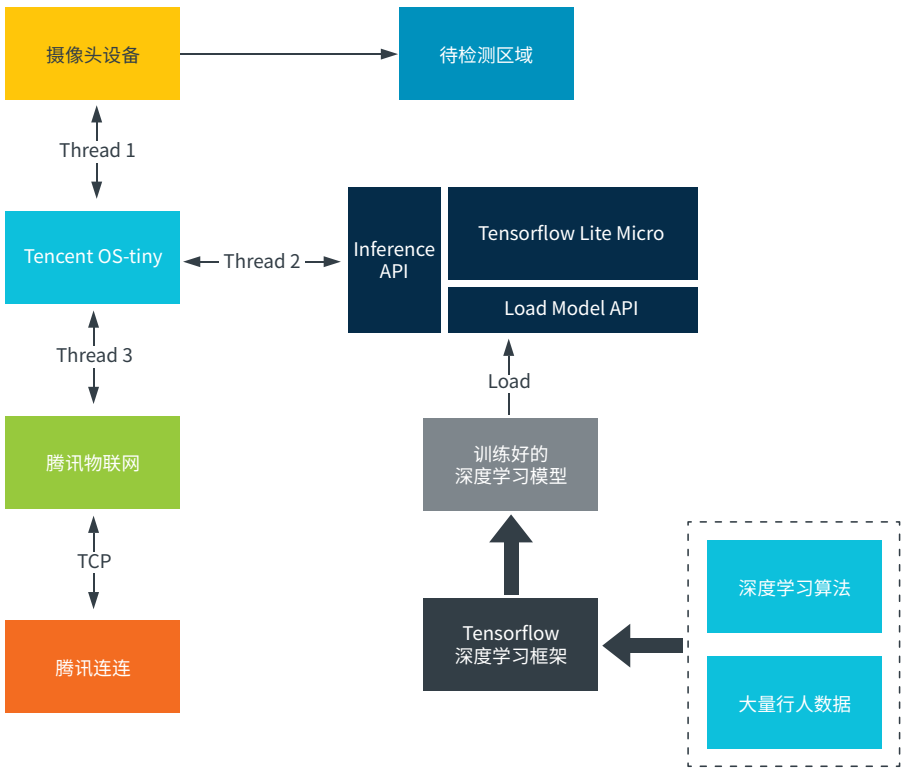
目前 Tensorflow Lite Micro 及 CMSIS-NN 组件已经进入 TencentOS-tiny GitHub 仓库，其中提供了 Tensorflow Lite Micro 源码以及针对 Arm Cortex-M 系列的 MDK lib 库文件，可以方便开发者集成到 MCU 开发环境中。

3. **本系统将边缘 AI 和 TencentOS-tiny 结合在一起，面向区域安防管理，提出了一种新的端云系统架构。**通过对系统的验证并配合详细的用户移植文档，让整个系统具备在多领域多场景的可迁移性与易用性。

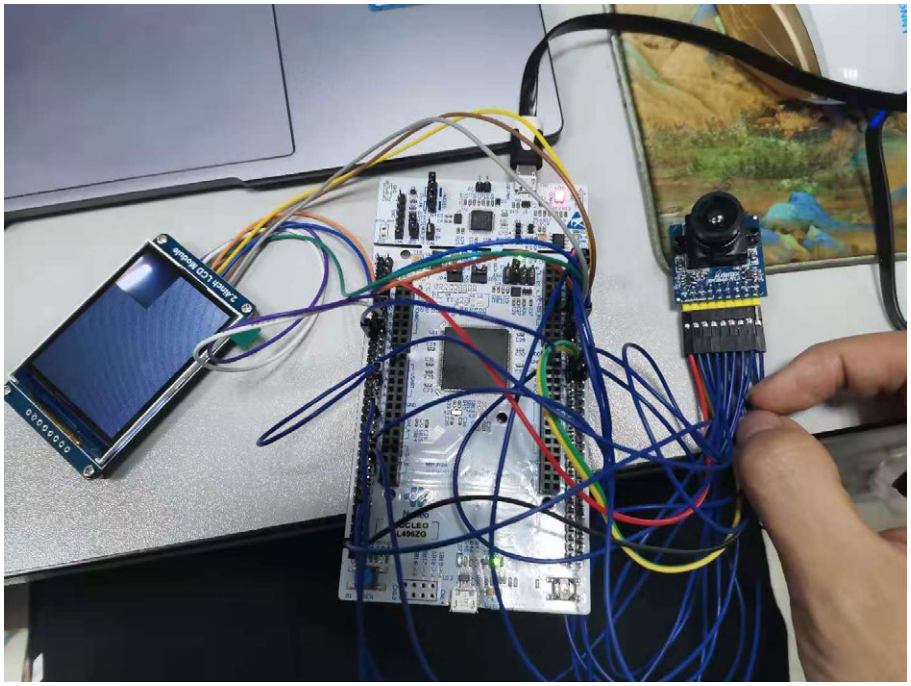
系统拓扑结构图：



单个 TencentOS-tiny 内部结构图：



系统 Demo 展示：



根据我们实际的测试，在 MCU 端运行的行人检测例程正确率达到 84%。端侧检测到人后，MCU 将有关行人的数据上传到云端，云端根据收到的数据来唤醒其他高性能设备，进一步获取行人的真实数据。

相比与传统的社区管理员通过大量连接了高清摄像头的屏幕来完成社区安全管理的方法相比，这种基于 TencentOS-tiny 和 CMSIS-NN 实现的端云协同智能安防系统仅会在端侧检测到人后才会触发相应事件，不仅减轻了管理员的工作负担，更重要的是端侧实现的智能化应用将极大的节省系统的带宽资源，在提升系统响应的同时也降低了成本，是一种综合能力较强的解决方案。

三、系统移植概述

我们已将相关代码和文档开源至官方主仓库，开发者可以按照指南进行移植和应用。

1. 准备目标硬件（开发板 / 传感器 / 模组）

需准备以下硬件 :

- 开发板：NUCLEO-L496ZG，MCU 为 STM32L496ZG
- Camera：获取 RGB 图像，本例程使用 OV2640 摄像头
- 通信模组：负责 MCU 与云端之间的通信，本例程选用的乐鑫 ESP8266

2. 准备系统软件

- 首先，参考 TencentOS-tiny 基于 Keil 的移植教程进行移植：
  https://github.com/Tencent/TencentOS-tiny/blob/master/doc/10.Porting_Manual_for_KEIL.md
  为了方便初始化 MCU 外设，后续要继续使用 STM32CubeMX 软件，请确保正确安装。在系统移植完成后，工程可以进行线程任务切换，通过串口打印”hello world”，表明基础 Keil 工程代码准备完毕。

- 准备 Tensorflow Lite Micro 组件。本次我们使用 Tensorflow Lite Micro 推理框架来实现行人检测任务，用户可以直接采用主仓库中 `TencentOS-tiny\components\ai\tflite_micro` 路径下对应的 lib 库文件来集成到系统中，然后调用相关的 API 即可将 AI 组件部署在 MCU 平台上。

- 如何制作 lib 库文件以及如何使能 CMSIS NN 加速
  请参考 Tensorflow Lite Micro 组建使用说明。

3. 系统移植流程

在获得基础工程后，我们首先移植驱动代码

- 添加与本例程相关的 Necluo STM32L496RG 的摄像头驱动代码 (此处采用官方仓库中的驱动代码)

在 mcu_init 函数重写 DCMI 帧中断回调函数。值得注意的是，当使用 CubeMX 重新配置外设并生成代码时，代码需要写在 CubeMx 生成的注释语句内，这样添加的代码才不会被覆盖。如下所示，代码添加在 /* USER CODE BEGIN 4 */ 和 /* USER CODE END 4 */ 注释语句之间：

```
/* USER CODE BEGIN 4 */
void HAL_DCMI_FrameEventCallback(DCMI_HandleTypeDef *hdcmi)
{
  if(hdcmi->State == 2 && frame_flag != 1){
    frame_flag = 1;
  }
}
/* USER CODE END 4 */
```

添加 LCD 程序显示摄像头图像

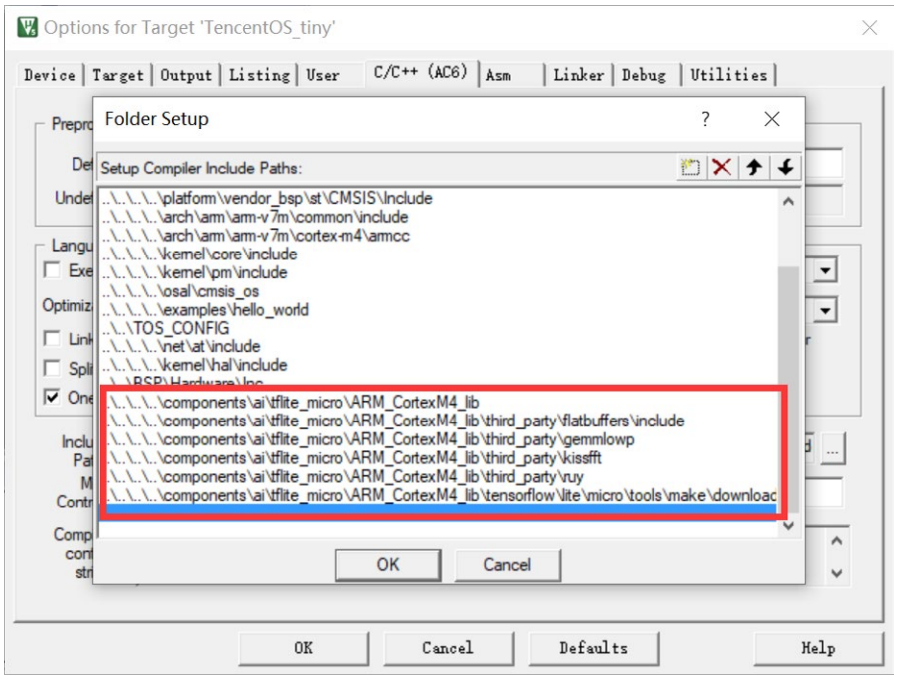
```
void task1(void *arg)
{
  while (1) {
   if(frame_flag == 1){
```

```
        if(HAL_DCMI_Stop(&hdcmi))Error_Handler(); //stop DCMI
        LCD_2IN4_Display(camera_buffer,OV2640_PIXEL_WIDTH,OV2640_PIXEL_HEIGHT);
        //display
        frame_flag = 0;
        if(HAL_DCMI_Start_DMA(&hdcmi,DCMI_MODE_CONTINUOUS,\  //restart DCMI
                    (uint32_t)camera_buffer ,\
                    (OV2640_PIXEL_WIDTH*OV2640_PIXEL_HEIGHT)/2))
          Error_Handler();
        osDelay(50);
  }
}
```

将 Tensorflow Lite Micro 的模型和数据接入代码添加到工程中，同时添加

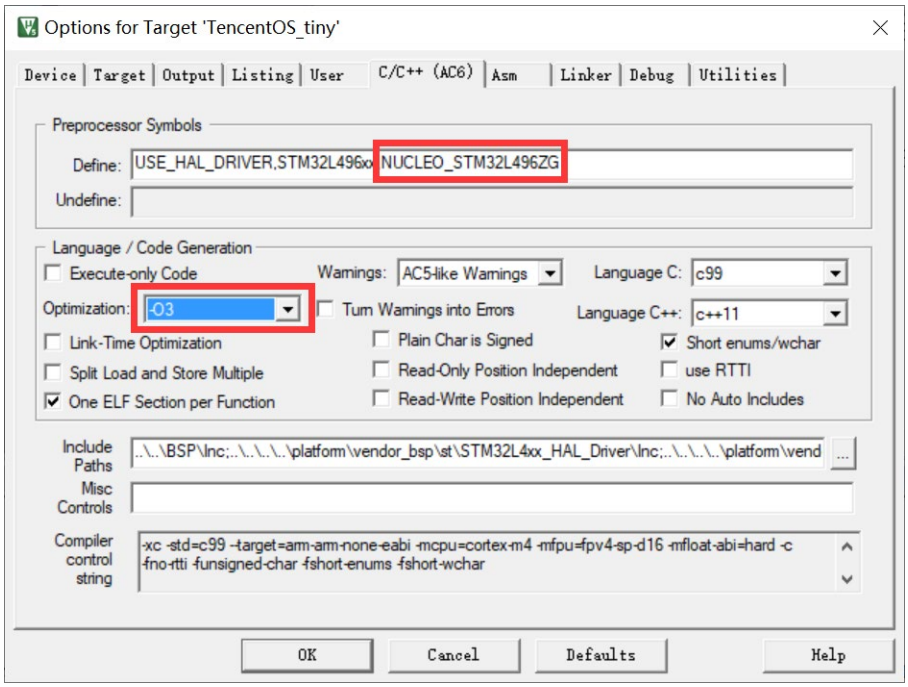
- `TencentOS-tiny\components\ai\tflite_micro\KEIL\retarget.c`
- `TencentOS-tiny\components\ai\tflite_micro\ARM_CortexM4_lib\tensorflow_lite_micro.lib`

最后关闭 Keil 的 Microlib 库，添加相关的 include 文件



注：最下方的路径为：

```
TencentOS- tiny\components\ai\tflite_micro\ARM_CortexM4_lib\tensorflow\lite\micro\tools\make\downloads
```



调整优化等级和 tflite_micro 的交互信息输出串口：

其中宏 `NUCLEO_STM32L496ZG` 指定 Nucleo STM32L496 的 hlpuart1 为系统 printf 函数的输出串口，具体定义在 Nucleo STM32L496 BSP 文件夹中的 `mcu_init.c` 中。

至此 Tensorflow Lite Micro 已经成功的移植到 MCU 平台中，可以开始设计与行人检测有关的应用层代码了。

- 设计图像预处理函数。在本例程中，模型的输入为灰度图，为完成摄像头获取的 RGB 图像到灰度图转换，需要解析出 RGB565 像素格式中 R、G、B 通道的数据，再根据公式计算出单个像素点的灰度，具体代码如下：

```
uint8_t rgb565_to_gray(uint16_t bg_color)
{
  uint8_t bg_r = 0;
  uint8_t bg_g = 0;
  uint8_t bg_b = 0;
  bg_r = ((bg_color>>11)&0xff)<<3;
  bg_g = ((bg_color>>5)&0x3f)<<2;
  bg_b = (bg_color&0x1f)<<2;
  uint8_t gray = (bg_r*299 + bg_g*587 + bg_b*114 + 500) / 1000;
  return gray;
}
```

```
void input_convert(uint16_t* camera_buffer , uint8_t* model_buffer)
{
  for(int i=0 ; i<OV2640_PIXEL_WIDTH*OV2640_PIXEL_HEIGHT ; i++)
  {
    model_buffer[i] = rgb565_to_gray(camera_buffer[i]);
  }
}
```

- 编写行人检测线程任务函数：

```
void task1(void *arg)
{
  while (1)
  {
   if(frame_flag == 1)
   {
    if(HAL_DCMI_Stop(&hdcmi))
      Error_Handler(); //stop DCMI
    frame++;
    input_convert(camera_buffer,model_buffer);
    res_p = person_detect(model_buffer);
    LCD_2IN4_Display(camera_buffer,OV2640_PIXEL_WIDTH,OV2640_PIXEL_HEIGHT);
    if(res_p == 1)
    {
      HAL_GPIO_WritePin(GPIOB, LCD_DC_Pin|LED_Pin, GPIO_PIN_SET);
      if(count == 0)
        frame = 0;
      if(frame-count > 5)
        count = 0;
      else
      {
          count ++;
      }
      if(count == 5)
      {
        person_flag = 1;
        count = 0;
        printf(“detect a dangerous person!!\n”);
      }
    }
    else
    {
      HAL_GPIO_WritePin(GPIOB, LCD_DC_Pin|LED_Pin, GPIO_PIN_RESET);
    }
```

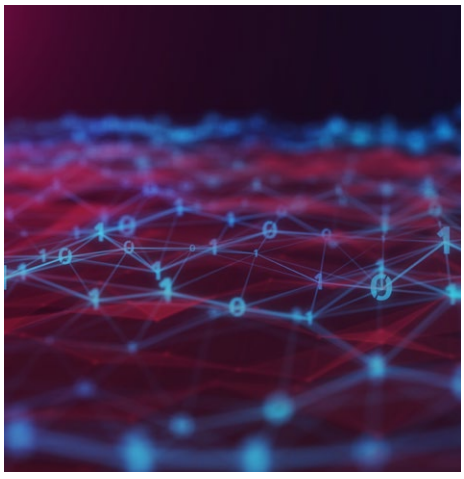
```
      frame_flag = 0;
      person_flag = 0;
      if(HAL_DCMI_Start_DMA(&hdcmi,DCMI_MODE_CONTINUOUS,(uint32_t)camera_buffer ,\
        (OV2640_PIXEL_WIDTH*OV2640_PIXEL_HEIGHT)/2))
        Error_Handler(); //restart DCMI
      }
      osDelay(50);
    }
}

void task2(void *arg)
{
   while (1)
   {
      printf(“***task2\r\n”);
      osDelay(50);
   }
}
```

应用程序主体若在 10 帧中检测到超过半数的人像，就判定为异常并上报云端。根据实际的测试结果，执行一帧图像推理耗时约 633ms。

- 在 MCU 端处理完行人检测后，MCU 与云端建立数据连接并传输数据。部分代码如下：

```
int deal_up_stream_user_logic(DeviceProperty *pReportDataList[], int *pCount)
{
   int i, j;
      static  int person_count_last, person_flag_last, dagerous_flag_last = 0;

   // 上传数据逻辑处理
      if(person_count != person_count_last || person_flag != person_flag_last){

         sg_DataTemplate[0].state = eCHANGED;
         sg_DataTemplate[3].state = eCHANGED;

         sg_ProductData.m_count = person_count;
         sg_ProductData.m_person = person_flag;

         person_count_last = person_count;
         person_flag_last = person_flag;

      }
```

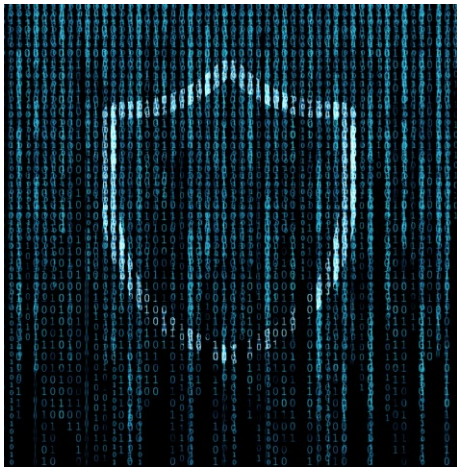
```
    if(dagerous_flag != dagerous_flag_last)
    {
      sg_DataTemplate[1].state = eCHANGED;
      sg_ProductData.m_dagerous = dagerous_flag;
      dagerous_flag_last = dagerous_flag;
    }


  for (i = 0, j = 0; i < TOTAL_PROPERTY_COUNT; i++) {
    if(eCHANGED == sg_DataTemplate[i].state) {
      pReportDataList[j++] = &(sg_DataTemplate[i].data_property);
            sg_DataTemplate[i].state = eNOCHANGE;
    }
  }
  *pCount = j;

  return (*pCount > 0)?QCLOUD_RET_SUCCESS:QCLOUD_ERR_FAILURE;
}
void deal_down_stream_user_logic(void *client, ProductDataDefine   * pData)
{
    // 处理下发逻辑
  if (sg_ProductData.m_warning == 1) {
      /* 开警示 */
      HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_SET);
  } else {
      /* 关警示 */
      HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_RESET);
  }
}
```



3. 腾讯物联网开发平台 - 腾讯连连小程序开发

为了方便用户实时查看端侧上传的信息（是否有异常报警、人流量计数等）以及控制设备端发出报警提示，我们利用腾讯云 IoT Explorer 开发平台，开发腾讯连连小程序。

开发过程如下，登录腾讯云开发平台（图片较大，可下载后查看）：



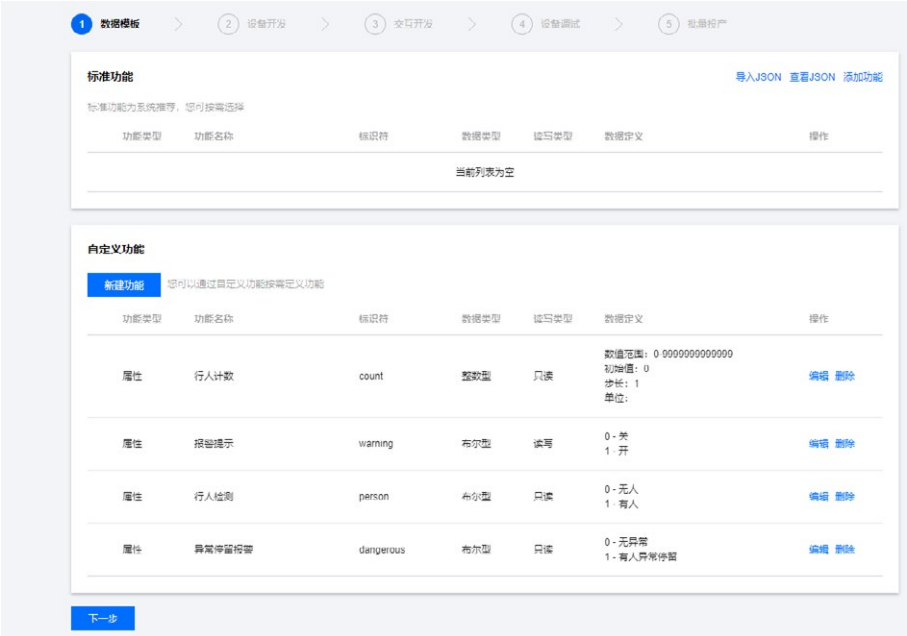
- 步骤一：新建产品
- 步骤二：跟据场景和应用定义数据模板

设备端上传的只读数据：

1. **行人检测**：设备端检测到行人时，标志位置为 1；
2. **异常停留报警**：当设备端持续检测到行人时，触发异常停留报警，标志位置为 1；
3. **行人计数**：当设备端的行人检测结果从无人变化为有人时，人流量计数值 +1。

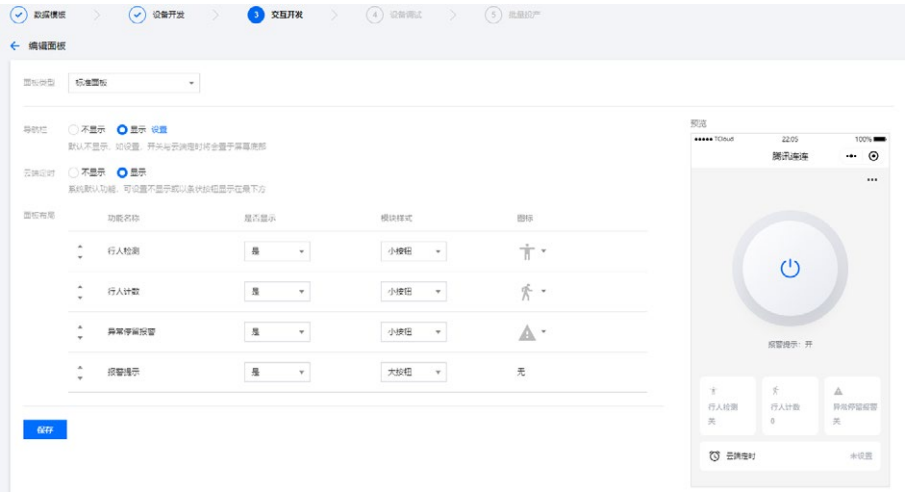
设备端接收的控制指令：

报警提示：当用户看到有异常停留时，可以控制设备端发出报警提示，也可以关闭报警提示。



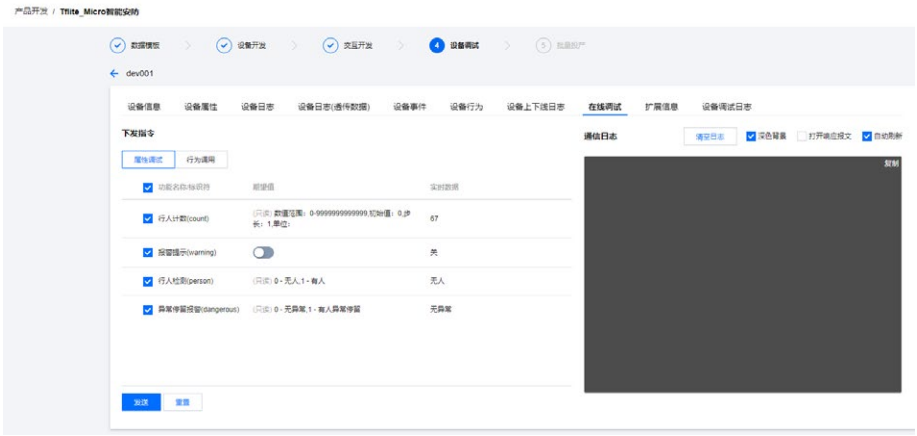
- 步骤三：编辑小程序的面板

由于目前腾讯连连提供的模板较少，暂时使用按钮显示设备状态信息。



- 步骤四：设备调试

1. 将产品 ID、密钥、设备号等信息填入设备端程序；
2. 移植基于 TencentOS-tiny 的 AT 组件和 ESP8266 适配的 SAL 框架；
3. 设备端编写上行和下行数据处理程序。



四、结语

我们提出了一种基于边缘 AI+TencentOS-tiny 的新架构，虽然在用户前端还有很多改进的空间，但通过对整体方案的验证并且配合详细的用户移植文档，使我们的工作具备了可迁移性和扩展性，同时也打开了 TencentOS-tiny 对于人工智能领域的支持。在未来我们会继续完善 Tensorflow Lite Micro 组件并不断更新应用，致力于丰富整个 TencentOS-tiny 以及 Arm 生态。随着越来越多的厂商采用 Arm Cortex-M55 和 Ethos-U NPU IP 方案，相信未来端侧 AI 的应用会更加广阔。

TencentOS-tiny AI 组件：
https://github.com/Tencent/TencentOS-tiny/tree/master/components/ai/tflite_micro
Arm Cortex-M4：
https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m4
Arm Cortex-M55：
https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m55
Arm Ethos-U55：
https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-u55
Arm Ethos-U65：
https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-u65

五、致谢

感谢 TencentOS-tiny 以及 Google 团队的大力支持，感谢个人开发者邓可笈，杨庆生，刘恒言的贡献。