阿里云

阿里云AIoT开发手册(1.0版本)

全栈物联网开发鸟瞰

- 物联网开发全路径：从设备端到云服务到应用端
- 适合嵌入式、服务开发、应用开发等各种背景的开发者学习
- 助你迅速了解物联网解决方案的架构和实现

阿里云开发者社区

Alibaba Group
阿里巴巴集团

阿里技术

扫一扫二维码图案，关注我吧

「阿里技术」微信公众号

AIoT 开发互动手册

阿里云开发者社区

目录

第一章　开发者操作大图

如何让我的设备连上云


端-设备端开发
Link Kit SDK
AliOS Things
WiFi设备
以太网设备
蜂窝网设备
子设备通过网关代理连云
DTU
选择设备类型和连接方式
获取源码
Link Kit SDK for C
Link Kit SDK for Java
Link Kit SDK for Python
Link Kit SDK for NodeJS
Link Kit SDK for Andriod
Link Kit SDK for iOS
选择需要的SDK
选择开发方式
使用命令性工具开发方式
使用Docker开发
使用AliOS Studio开发
使用Keil MDK开发
使用IAR开发


管-LoRa网络管理
开发者
Link WAN
购买LoRa认证网关
注册网关
创建凭证
分享网络
通过物联网平台关联LoRaWAN节点
物联网平台
创建产品
定义物模型
购买LoRa认证设备
中转枢纽

如何实现边缘智能


边-边缘计算
开发者
边缘计算
创建边缘实例和网关
生成下载命令
本地登录网关，复制命令，一键下载配置并启动 Link Edge
边缘实例查看网关状态
函数计算
边缘流数据分析
子设备接入
消息路由
场景联动
云端函数计算
边缘函数计算
边缘场景联动
云端场景联动
部署实例
远程运维管理
中转
枢纽

第二章　从零开始——5 分钟简易实践

基于阿里云物联网平台实现的简易出入监控

作者：擎天

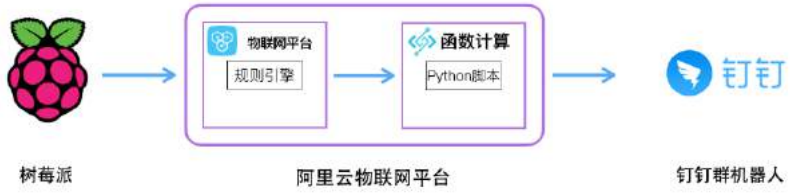
摘要

本文通过一个简单实例，主要介绍了如何使用树莓派快速接入阿里云 iot platform，并实现了一个简易的监控人员出入并拍照上送钉钉群的场景

本文通过一个简单实例，主要介绍了如何使用树莓派快速接入阿里云 iot platform，并实现了一个简易的监控人员出入并拍照上送钉钉群的场景

场景

在公司大门入口处布点树莓派和红外感应，实现出入口人员出入时，自动拍照并上送钉钉群机器人



准备

物料准备

- 树莓派
- HC-SR501 人体红外感应器

- 树莓派摄像头
- 母对母杜邦线三根

阿里云环境准备

- 物联网平台
- 对象存储 OSS
- 函数计算
- 日志服务（可选）

操作步骤

1. 云端开发

1.1　物联网平台

登录阿里云控制台，进入物联网平台控制面板

1.1.1　新建产品

进入设备管理，创建产品，选择基础版或高级版都可以，本实例使用基础版就可以满足基本要求。

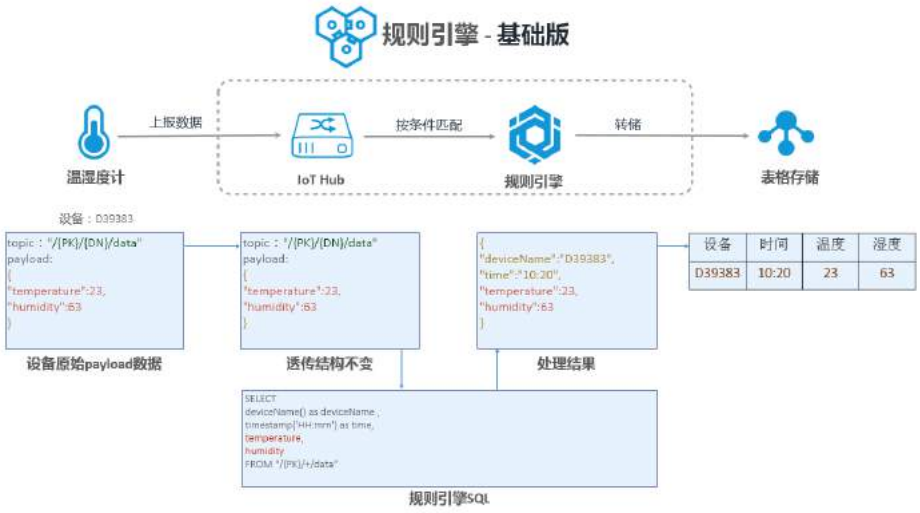
系统会自动创建 3 个 Topic, 我们需要使用 /ProductName/${deviceName}/update，作为设备告警消息的上送的 Topic。

1.1.2 设备管理

在产品中新增设备，并获得设备的 3 元组，在 2.3 节的设备代码的编写时需要使用此 3 元组。*设备三元组是设备的唯一标示*

1.1.3 新建规则引擎

设置规则引擎的意义在于，可以将设备上送的消息数据，通过配置转发规则将处理后的数据转发到阿里云其他服务，例如 RDS、TBS 和函数计算等等。我们需要注意的是从设备端到规则引擎处理后的 JSON 数据格式的变化，下图中是基础版的演变过程：



我们在设备端消息上送定义的 JSON 的格式是：

```
{
    'photo': 'xxxxxxx.jpg'
}
```

新创建一个规则，数据格式选择 JSON。编写处理数据的 SQL

```
SELECT deviceName() deviceName, photo FROM "/a1O4b4XcICc/+/update"
```

配置完成后，我们可以模拟调试一下，检查规则是否正确：

接着，配置数据转发，把数据转发到 FC 函数计算中。分别选择在 1.3 步骤中创建好的服务和函数。

1.2 对象存储

由于设备端拍摄的照片需要在钉钉中展示，因此把照片存储在 OSS 上是一个解决方案。

1.2.1 新建 bucket

新建一个 bucket 用于存储设备上送的照片。读写权限选择**公共读**然后在 bucket 中创建 photo 目录。

1.3 函数计算

经过物联网平台规则引擎转发过来的 JSON 数据，我们通过建立函数，把它转发到钉钉机器人接口上，实现钉钉群中的消息通知

1.3.1 新建服务

新创建服务，如果需要记录和回溯函数执行的日志，则需要开通日志服务，配置日志仓库。

1.3.2　新建函数

使用空白模版新建函数，不需要触发器，运行环境选择 **python2.7**

1.3.3　函数代码

```
# -*- coding: utf-8 -*-

import logging
import json
import requests

# 钉钉消息发送实现
def post(data):
    webhook_url='https://oapi.dingtalk.com/robot/send?access_token=${Token}'
# 钉钉群机器人的 webhook 的 URL
    headers = {'Content-Type': 'application/json; charset=utf-8'}
    post_data = json.dumps(data)
    try:
        response = requests.post(webhook_url, headers=headers, data=post_
data)
        logging.info('Send success')
    except requests.exceptions.HTTPError as exc:
        logging.error("Send Error,HTTP error: %d, reason: %s" % (exc.
response.status_code, exc.response.reason))
        raise
    except requests.exceptions.ConnectionError:
        logging.error("Send Error,HTTP connection error!")
        raise
    else:
        result = response.json()
        logging.info('Send Error:%s' % result)
        if result['errcode']:
            error_data = {"msgtype": "text", "text": {"content": "Send Error,
reason:%s" % result['errmsg']}, "at": {"isAtAll": True}}
            logging.error("Send Error:%s" % error_data)
            requests.post(webhook_url, headers=headers, data=json.
dumps(error_data))
        return result

# 发送钉钉 markdown 消息
def post_markdown(title,text):
    data = {
        "msgtype": "markdown",
        "markdown": {
            "title": title,
            "text": text
        },
```

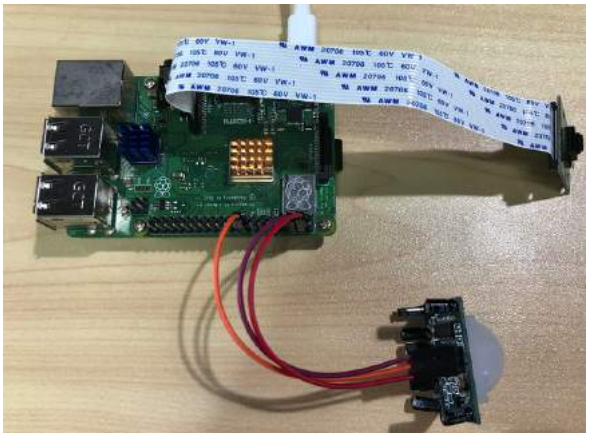
```
        "at": {
            "atMobiles": [],
            "isAtAll": False
        }
    }
    post(data)

# 函数计算入口
def handler(event, context):
    logger = logging.getLogger()
    evt = json.loads(event)
    #OSS endpoint url
    post_markdown('告警','![screenshot](https://${bucket}.oss-cn-hangzhou.
aliyuncs.com/photo/%s)' % evt.get('photo',''))
    logger.info('photo name is %s', evt.get('photo',''))
    return 'OK'
```

2. 设备端开发

HC-SR501 模块感应到有人移动时，会输出高电平，则触发摄像头拍照，并将照片文件存储到 OSS，同时发送消息到 IOT 平台的 **/ProductName/${deviceName}/update** 消息队列中

2.1 硬件安装

1. 连接好摄像头
2. 将 HC-SR501 人体红外感应器的 vcc 引脚接 5v，也就是 pin4，I/O 引脚接 pin18，GND 引脚接地 pin6



2.2　环境准备

我们在树莓派上使用 python2.7 作为开发语言。

```
1. pip install aliyun-python-sdk-iot-client
2. pip install oss2
3. mkdir py-demo （项目程序文件夹）
4. cd py-demo
5. mkdir photo （本地照片临时目录）
6. vim monitor.py
```

2.3　代码开发

monitor.py 内容如下：

```
# -*- coding: utf-8 -*-

import json
import uuid
import logging
from time import sleep
from picamera import PiCamera
import RPi.GPIO as GPIO
import oss2
import aliyunsdkiotclient.AliyunIotMqttClient as iot

# 参数定义
options = {
    'productKey': '${productKey}',  # 设备标识三元组
    'deviceName': '${deviceName}',  # 设备标识三元组
    'deviceSecret': '${deviceSecret}',  # 设备标识三元组
    'port': 1883,  # iot mqtt port
    'host': 'iot-as-mqtt.cn-shanghai.aliyuncs.com',  # iot mqtt endpoint
    'endpoint': 'http://oss-cn-hangzhou.aliyuncs.com',  # oss endpoint
    'ak': '${ak}',
    'sk': '${sk}',
    'bucket': '${bucket}',  # oss bucket
    'ir_pin': 24  # 人体红外感应器设置读取针脚标号
}

topic = '/' + options['productKey'] + '/' + options['deviceName'] + '/user/
test'

# 拍照存 oss，并发送通知
def takephoto2oss(client):

    # 拍照
```

```
    photo_filename = str(uuid.uuid1()) + ".jpg"
    print('take photo :' + photo_filename)
    camera.capture('photo/' + photo_filename)

    # 存 OSS
    print('save photo to oss :' + photo_filename)
    bucket.put_object_from_file(
        'photo/' + photo_filename, 'photo/' + photo_filename)

    # 消息上送
    payload_json = {
        'photo': photo_filename
    }
    print(‘send data to iot server: ‘ + str(payload_json))
    client.publish(topic, payload = str(payload_json))


def on_connect(client, userdata, flags_dict, rc):
    print(“Connected with result code “ + str(rc))


def on_disconnect(client, userdata, flags_dict, rc):
    print("Disconnected.")


if __name__ == '__main__':
    # GPIO 初始化
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(options['ir_pin'], GPIO.IN)

    # 摄像头 初始化
    camera = PiCamera()
    camera.resolution = (640, 480)
    camera.vflip = True
    camera.hflip = True

    # OSS 初始化
    auth = oss2.Auth(options['ak'], options['sk'])
    bucket = oss2.Bucket(auth, options['endpoint'], options['bucket'])

    # IOT Mqtt 初始化
    client = iot.getAliyunIotMqttClient(options['productKey'],
options['deviceName'], options['deviceSecret'], secure_mode = 3)
    client.on_connect = on_connect
    client.connect(host=options['productKey'] + '.' + options['host'],
port=options['port'], keepalive = 60)

    while True:
```

```
        # 当高电平信号输入时报警
        if GPIO.input(options['ir_pin']) == True:
            print " Someone is coming!"
            takephoto2oss(client)
        else:
            continue
        sleep(3)
```

3. 测试运行

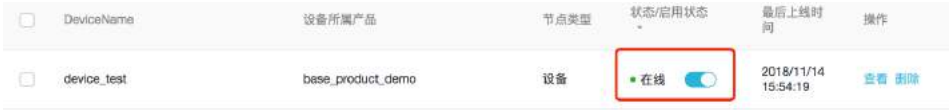
3.1　设备端运行

在 py-demo 文件夹下运行。

```
python monitor.py
```

3.2　云端查看上送消息

进入设备界面，观察设备状态



在设备的 Topic 列表中，也可以看到发布的消息个数

高级版的产品，还提供了消息日志，而本例中的产品是**基础版**，并无此功能。

3.3　钉钉群机器人结果

当有人出入门口的时候，钉钉群就可以收到机器人的消息推送。

3.4 后续完善

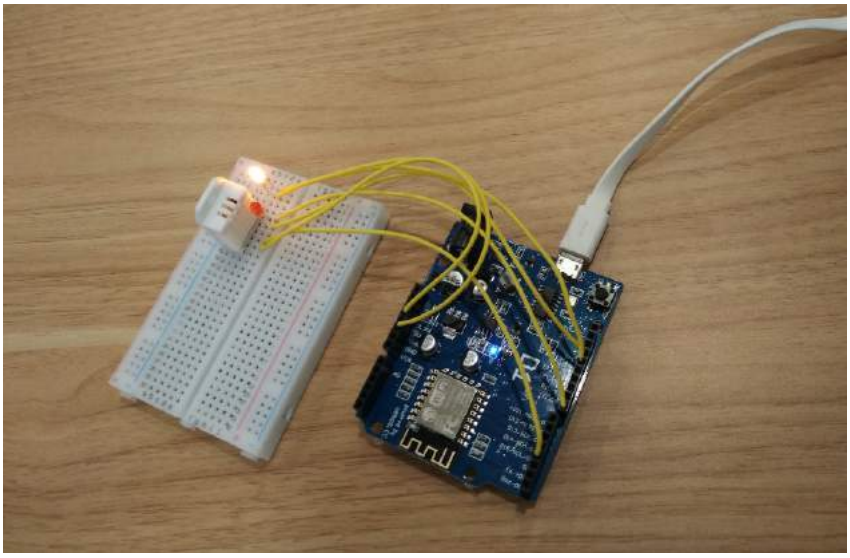
如果大家有兴趣，还可以进一步结合阿里云的人脸识别服务，再配合继电器，实现人员考勤和出入门禁的功能。

总结

通过阿里云物联网平台，结合阿里云提供的其他产品和服务，使用者可以快速地构建一套基于云边端一体的 IOT 产品，开发者只需关注业务层面的开发，而不用再花太多的精力在底层和通讯上，大大减少了开发周期，实现了产品的快速研发和迭代，节约了开发成本。

使用 IoT Studio 开发一个简单的温湿度监控器

作者：wusong119



概述

本文将介绍一个如何使用 IoT Studio 实现室内温湿度监控的简单案例。

室内温湿度监控需求如下：

1）业主可以随时查看室内的温度、热指数、湿度环境情况；

2）业主从 PC 浏览器上和手机 App 上可以实时查看室内的温度数据、热指数数据和湿度数据，温湿度正常不亮警示灯；

3）当室内温度高于 27 度或低于 10 度时，亮红灯，能够在钉钉群里报警或短信报警告知业主，提示应该开启空调的制冷或电加热模式；

4）当室内湿度高于 60 或低于 35 时，亮黄灯，能够在钉钉群里报警或短信报警告知业主，提示应该开启除湿机或加湿器；

室内温湿度监控实现方案如下：

1）室内的大气是均匀分布的，在室内安装一个温湿度传感器，根据温湿度传感器上报的数据和临界值进行室内温湿度是否正常的判断；

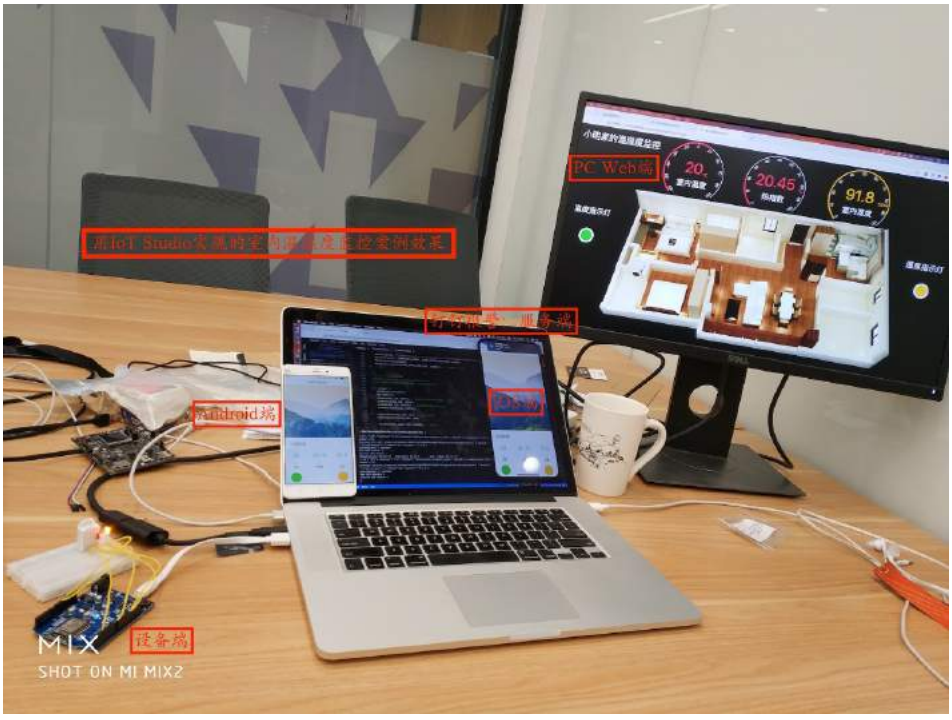
2）室内温湿度是否正常的统计需求：可以通过服务开发工作台开发一个云端 API 接口实现；

3）PC 浏览器上查看室内温湿度：可以通过 Web 可视化开发工作台开发一个网页应用实现；

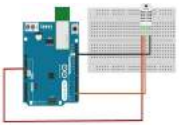
4）手机 App 上查看室内温湿度：可以通过移动可视化开发工作台开发一个移动 App 实现；

5）室内温湿度异常报警：可以通过服务开发工作台开发一个云端设备规则引擎实现；

最终的效果如下图所示：



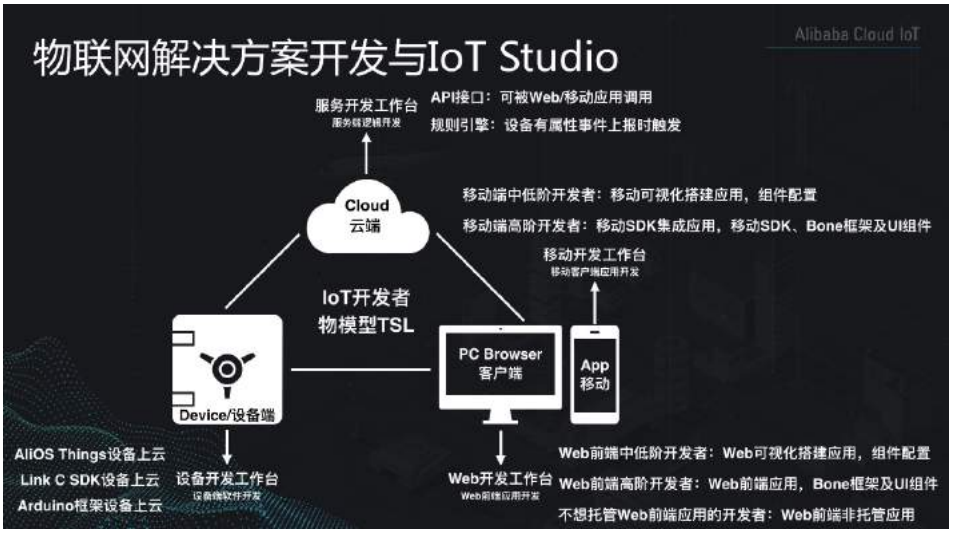
物料清单

硬件 (5)	软件 (5)	代码 (1)	图纸 (1)	相关资源 (2)
WeMos D1 开发板 WeMos D1 开发板，自带 ESP 8266 Wifi 模块，支持 Arduino 开发框架（×1） 	阿里云 IoT 物联网平台 阿里云 IoT 物联网平台，提供设备接入、设备管理功能。	WeMos 设备端代码 基于 Arduino 开发框架实现的 WeMos D1 开发板设备端代码逻辑	电路图 	Arduino 基础
DHT22 数字温湿度传感器 DHT22 数字温湿度传感器 AM2302 温湿度模块（×1） 	Hacklab WebIDE IoT Studio 的设备开发工作台，通过在线 WebIDE 降低设备上云门槛，提高设备上云效率。			IoT Studio 参考文档
LED 灯泡发光二极管 各种颜色的 LED 发光二极管（×2） 	服务开发工作台 通过可视化拖拽降低服务端逻辑的开发门槛，提高 API 接口和设备规则引擎的开发效率。			
面包板 无描述信息（×1） 	Web 可视化开发工作台 IoT Studio 的 Web 前端应用可视化开发工作台，通过可视化拖拽配置降低 Web 前端应用的开发门槛，提高 Web 前端应用的开发效率。			
跳线 / 杜邦线 / 连接线 无描述信息（×6） 	移动可视化开发工作台 IoT Studio 的移动 App 可视化开发工作台，通过可视化拖拽配置降低移动客户端应用的开发门槛，提高移动客户端应用的开发效率。			

方法 & 步骤

第 1 步　IoT Studio 简介

IoT Studio 提供了物联网解决方案开发时所需的工具。



1）**设备开发工作台**：通过在线 WebIDE 降低设备上云门槛，提高设备上云效率；开发的产物是设备端的嵌入式软件；

2）**服务开发工作台**：通过可视化拖拽降低服务端逻辑的开发门槛，提高 API 接口和设备规则引擎的开发效率；开发的产物是可供应用调用的 API 接口和云端自动运行的设备规则引擎；

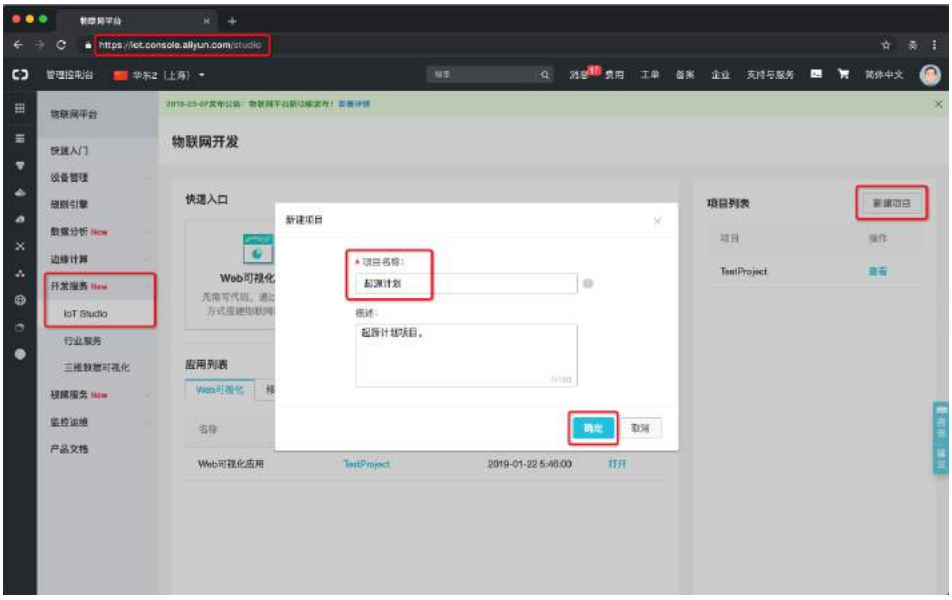
3）**Web 可视化开发工作台**：通过可视化拖拽配置降低 Web 前端应用的开发门槛，提高 Web 前端应用的开发效率；开发的产物是 Web 前端应用；

4）**移动可视化开发工作台**：通过可视化拖拽配置降低移动客户端应用的开发门槛，提高移动客户端应用的开发效率；开发的产物是 Android APK 安装包和 iOS 源码包；

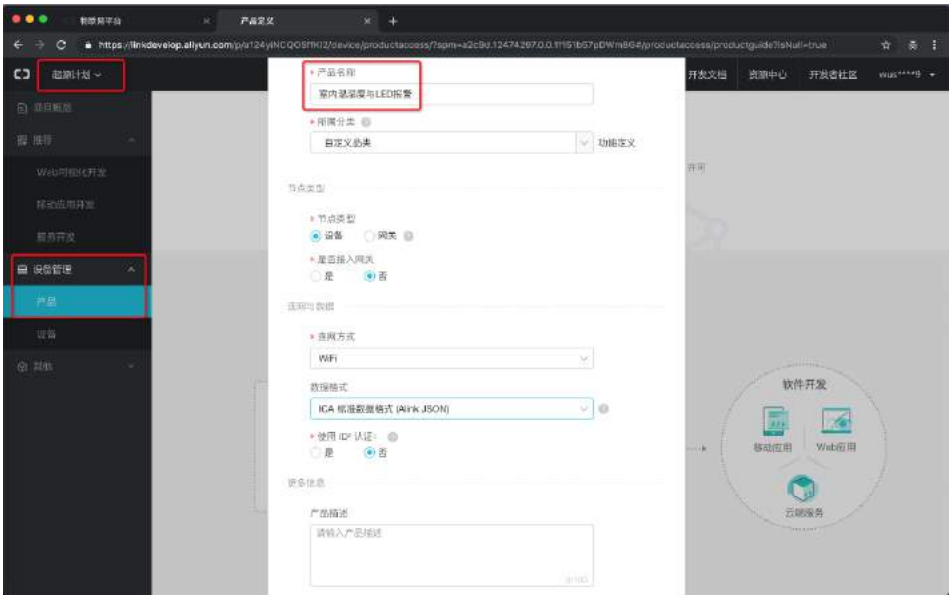
第 2 步　阿里云 IoT Link Platform：创建项目、产品和设备

1）登录自己的阿里云账号，进入阿里云 IoT 物联网平台的开发服务页面：

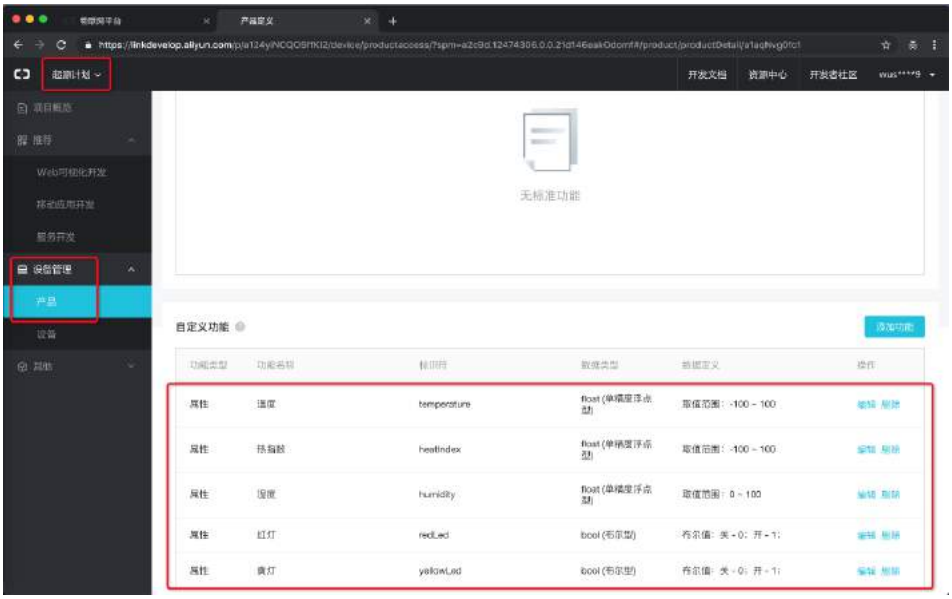
https://iot.console.aliyun.com/studio，创建一个“起源计划”项目；



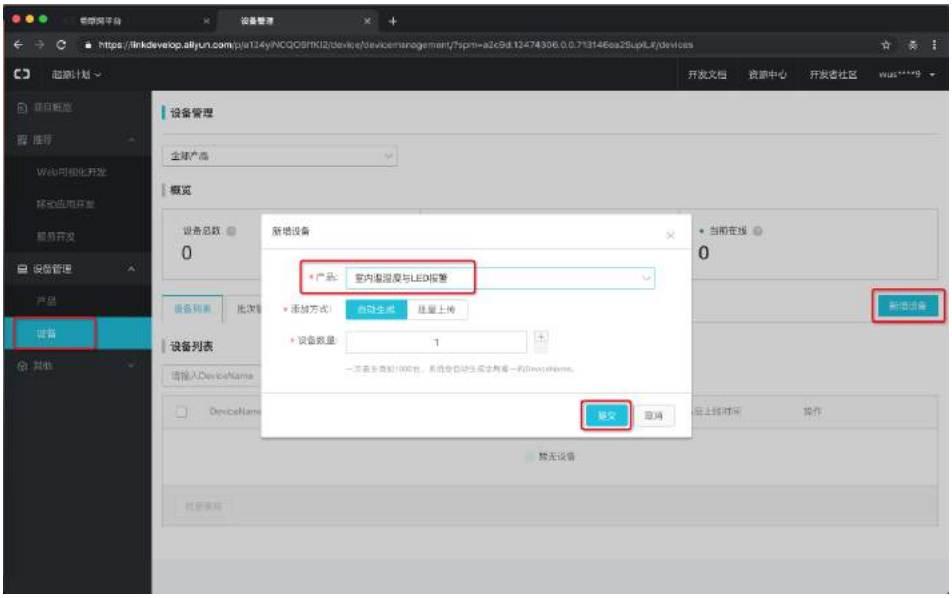
2）进入起源计划项目的产品页面，创建一个“室内温湿度与 LED 报警”产品，并完成物的功能属性定义；

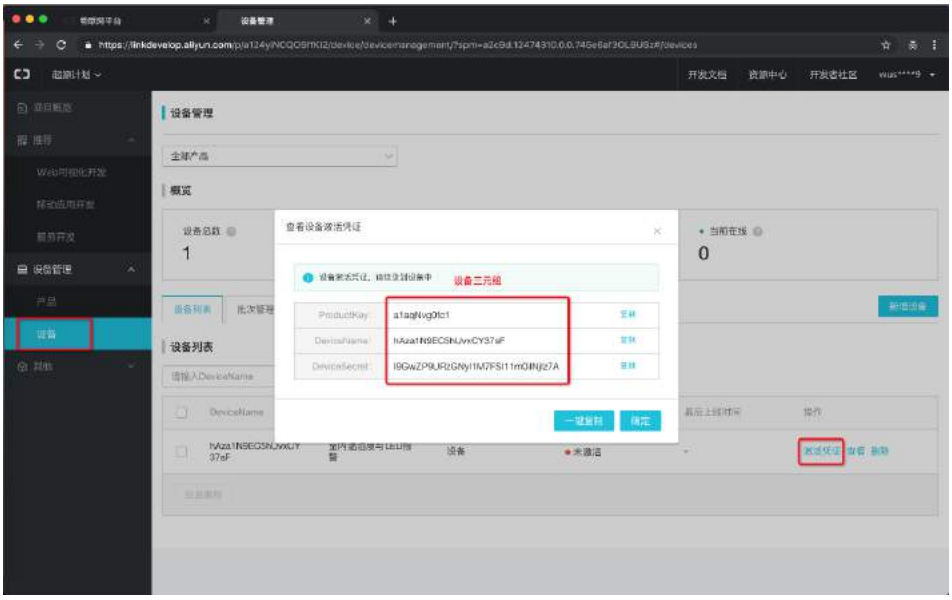



产品定义
超级计划
开发文档
资源中心
开发者社区
项目概览
推荐
Web可视化开发
移动应用开发
服务开发
设备管理
产品
设备
其他
室内温湿度与LE
ProductKey: a1aqNvg
产品信息
Top
其他
标准功能
导入物模型
查看物模型
添加功能
功能类型
功能名称
数据定义
操作
功能类型:
属性
服务
事件
功能名称:
温度
标识符:
temperature
数据类型:
float (单精度浮点型)
取值范围:
-50
~
50
步长:
0.01
单位:
摄氏度 / °C
读写类型:
读写
只读
定义湿度、体指数、温度、红灯、黄灯5个自定义属性
描述
请输入描述
0/128
自定义功能
添加功能
确认
取消


产品定义
超级计划
开发文档
资源中心
开发者社区
项目概览
推荐
Web可视化开发
移动应用开发
服务开发
设备管理
产品
设备
其他
无标准功能
自定义功能
添加功能
功能类型
功能名称
标识符
数据类型
数据定义
操作
属性
温度
temperature
float (单精度浮点型)
取值范围：-100 ~ 100
编辑 删除
属性
体指数
heatIndex
float (单精度浮点型)
取值范围：-100 ~ 100
编辑 删除
属性
湿度
humidity
float (单精度浮点型)
取值范围：0 ~ 100
编辑 删除
属性
红灯
redLed
bool (布尔型)
布尔值：关 - 0；开 - 1；
编辑 删除
属性
黄灯
yellowLed
bool (布尔型)
布尔值：关 - 0；开 - 1；
编辑 删除

3）进入起源计划项目的设备页面，创建一个室内温湿度与 LED 报警产品的设备；

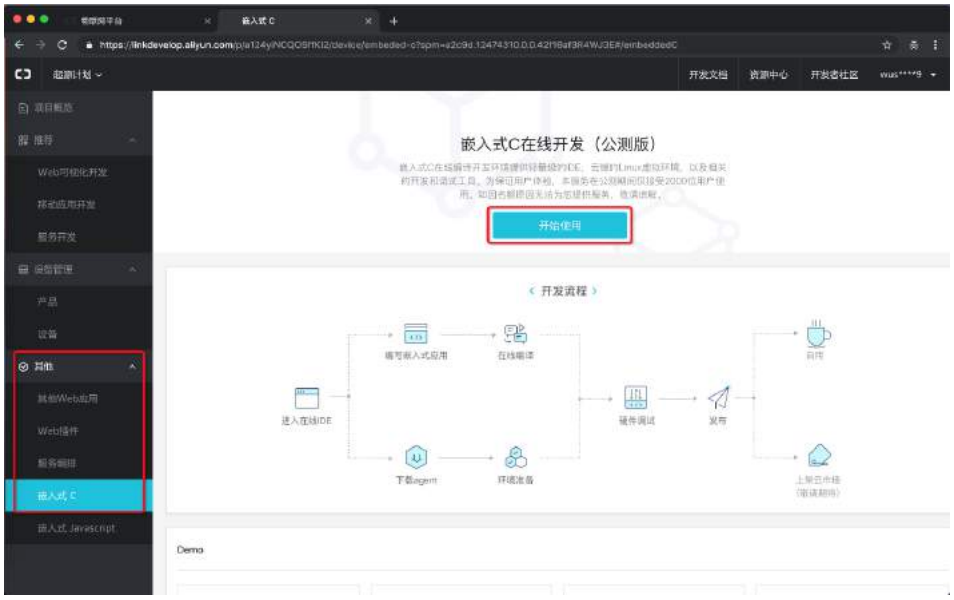


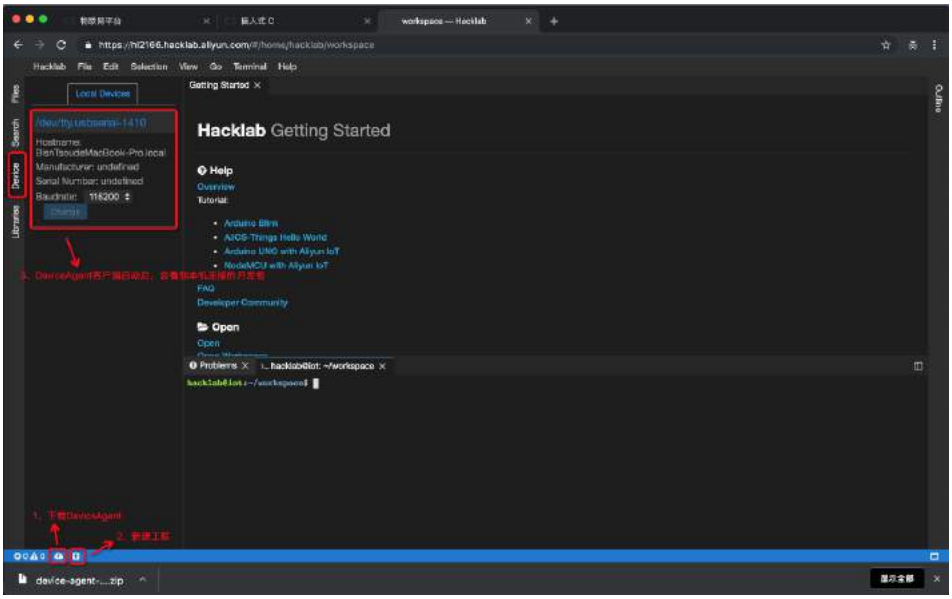


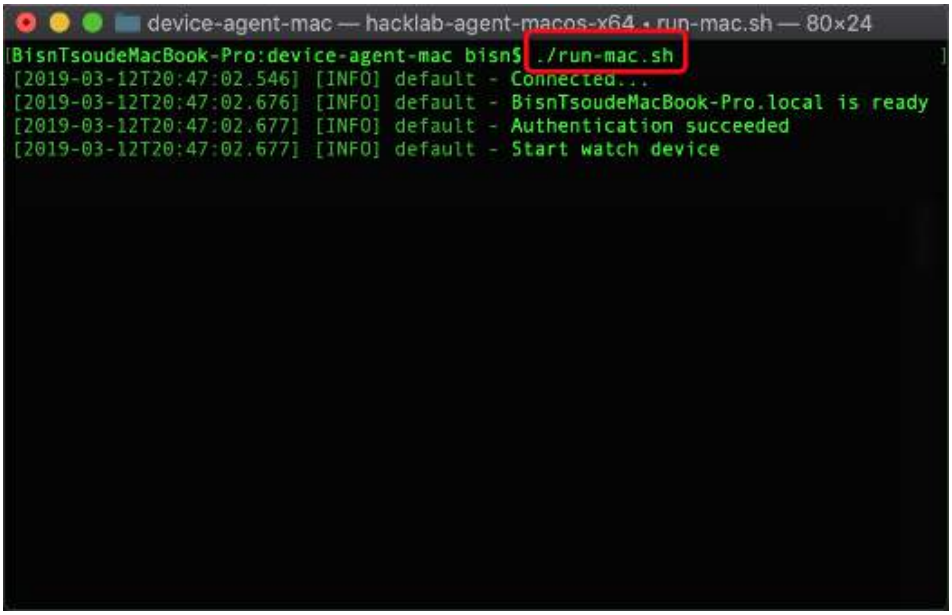
第 3 步　Hacklab WebIDE：实现设备数据上云

1）根据物料清单的硬件列表，采买并准备好响应的开发板、传感器、LED 灯、面包板设备，用导线连接好各个硬件（可参考 https://www.w3cschool.cn/arduino/arduino_humidity_sensor.html），用 USB 数据线将开发板与 PC 机连接；

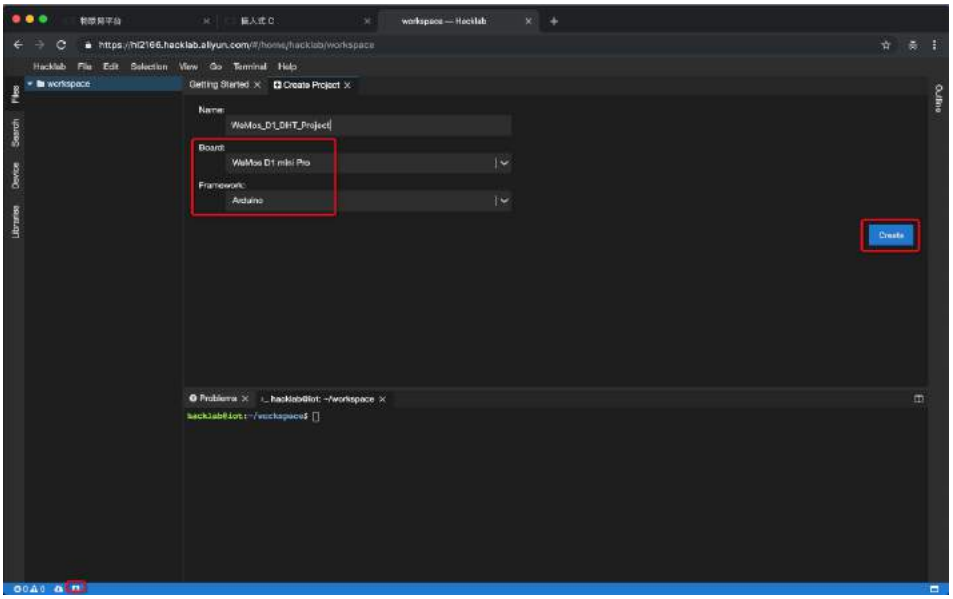
2）进入起源计划项目的其他—> 嵌入式 C 页面，点击“开始使用”，进入 Hacklab WebIDE 页面：https://hacklab.aliyun.com，从左下角下载 DeviceAgent 到本地，并启动运行，会看到开发板已经和 WebIDE 建立了连接；



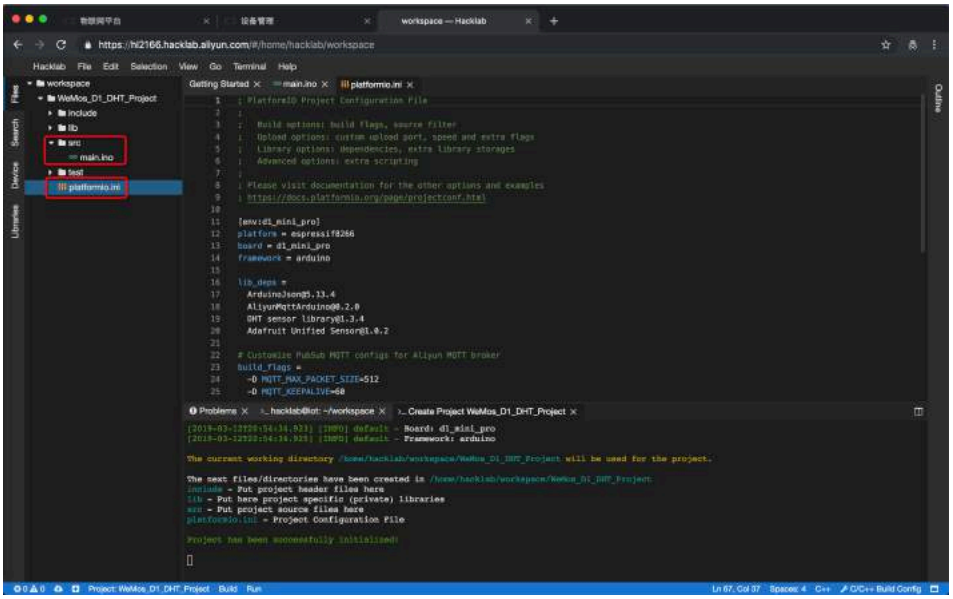

Hacklab Getting Started
Local Devices
Getting Started
Help
Overview
Tutorial:
Arduino Blink
AIOS Things Hello World
Arduino UNO with Aliyun IoT
NodeMCU with Aliyun IoT
FAQ
Developer Community
Open
Open
Problems
hacklabBot: ~/workspace
1、下载DeviceAgent
2、新建工程
device-agent-....zip


device-agent-mac — hacklab-agent-macos-x64 • run-mac.sh — 80×24
BisnTsoudeMacBook-Pro:device-agent-mac bisn$ ./run-mac.sh
[2019-03-12T20:47:02.546] [INFO] default - Connected...
[2019-03-12T20:47:02.676] [INFO] default - BisnTsoudeMacBook-Pro.local is ready
[2019-03-12T20:47:02.677] [INFO] default - Authentication succeeded
[2019-03-12T20:47:02.677] [INFO] default - Start watch device

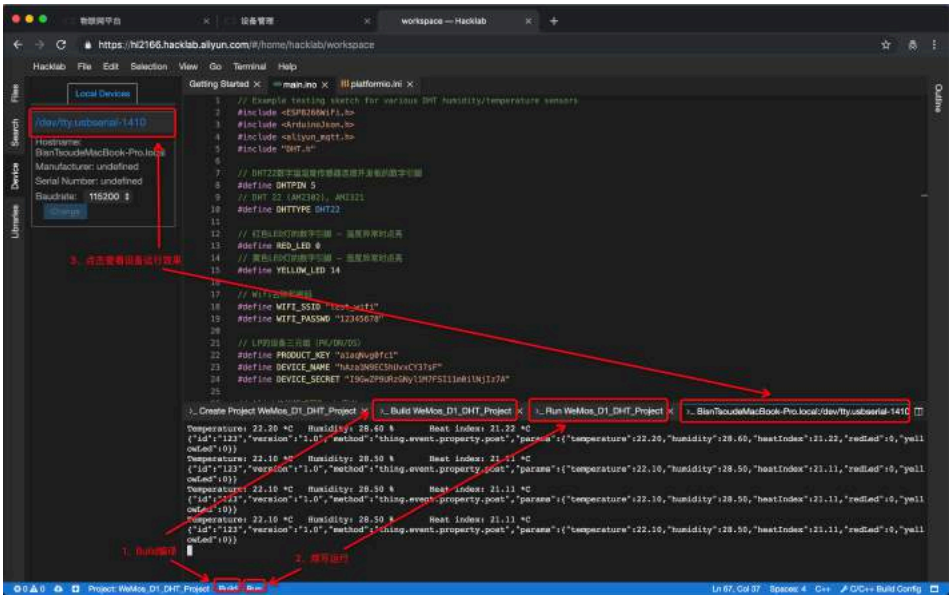
3）在 Hacklab WebIDE 里创建一个 WeMos D1 开发板的工程；



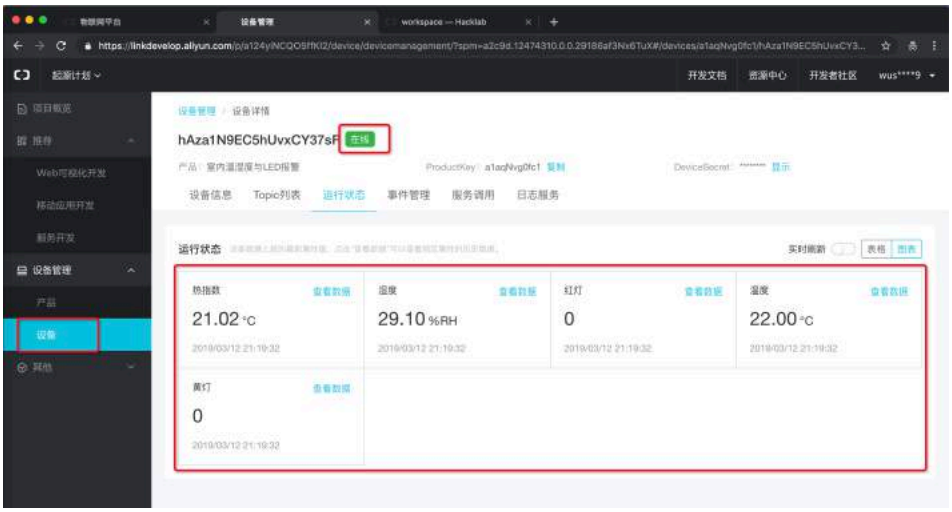
4）在 src 目录下新建一个 main.ino 文件，修改 platformio.ini 配置文件（管理工程依赖的三方库），代码可参考 https://code.aliyun.com/wusong119/arduino_dht_project；



5）点击左下角的 Build 按钮，完成设备端软件的编译，再点击左下角的 Run 按钮，完成设备端软件的烧写运行，并点击设备节点查看实际的运行效果；



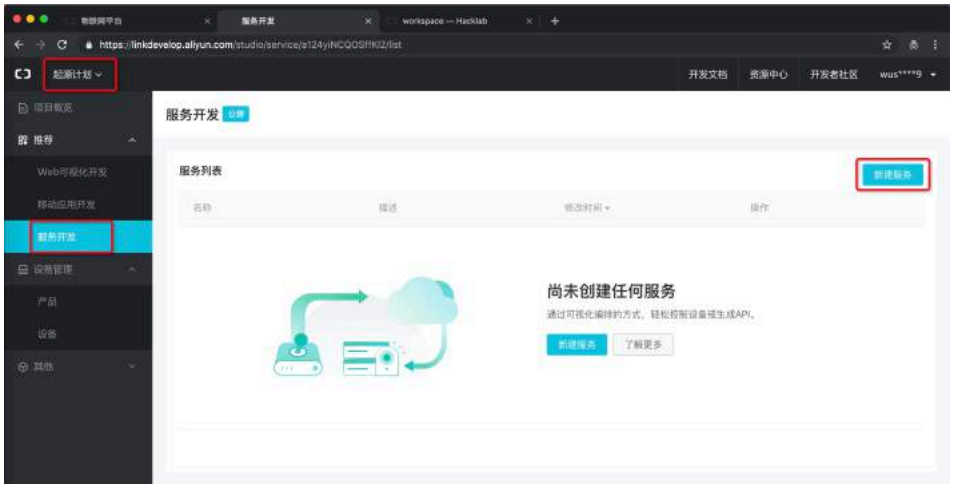
6）手机开启设备端软件需要的 Wifi 热点，此时 WeMos D1 开发板会连到这个 Wifi，同时会根据刚刚烧录的设备端软件逻辑连接到阿里云 IoT 物联网平台，并定时将传感器采集的数据上报到云端，这点从起源计划项目的设备页面可以看到；

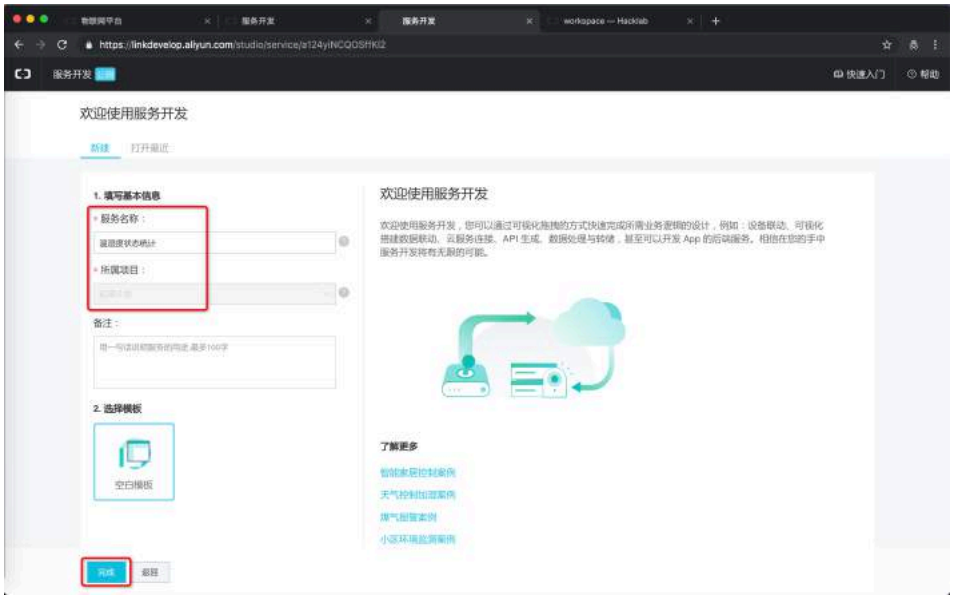


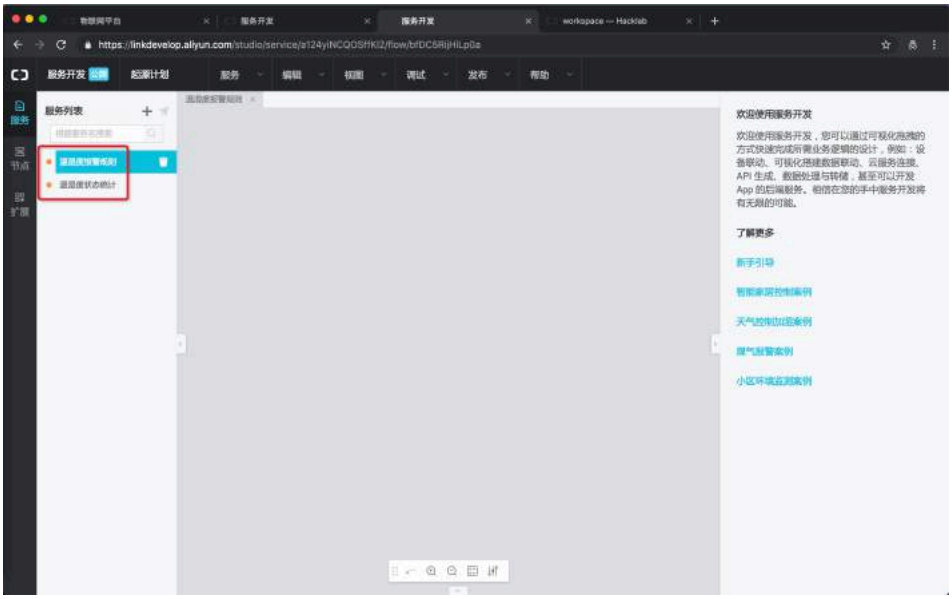
综上，我们完成了设备数据上云的实践。

第 4 步　CloudService Workbench：可视化开发 API 接口和规则引擎

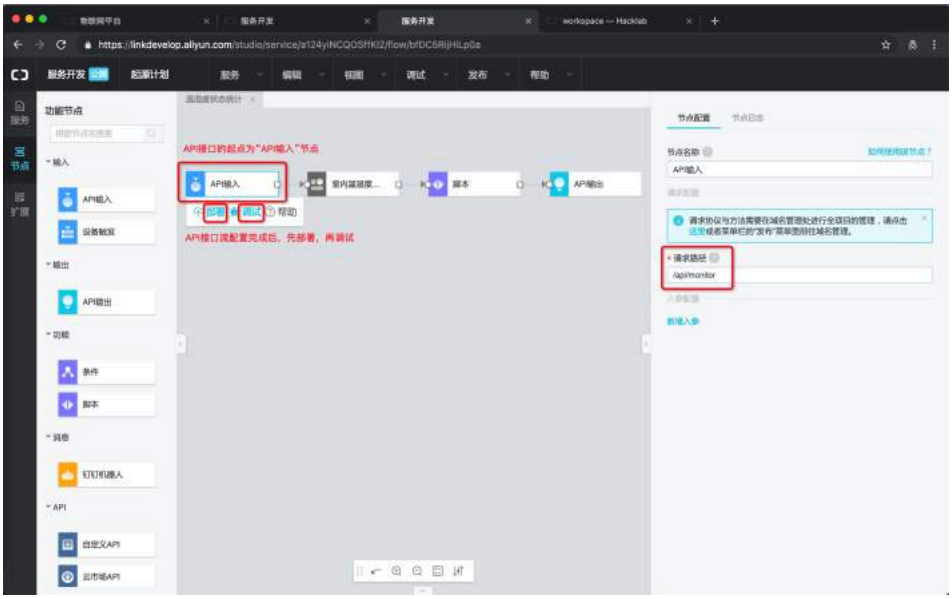
1）进入起源计划项目的服务开发页面，创建一个“温湿度状态统计”的 API 接口和一个“温湿度报警规则”的设备规则引擎；

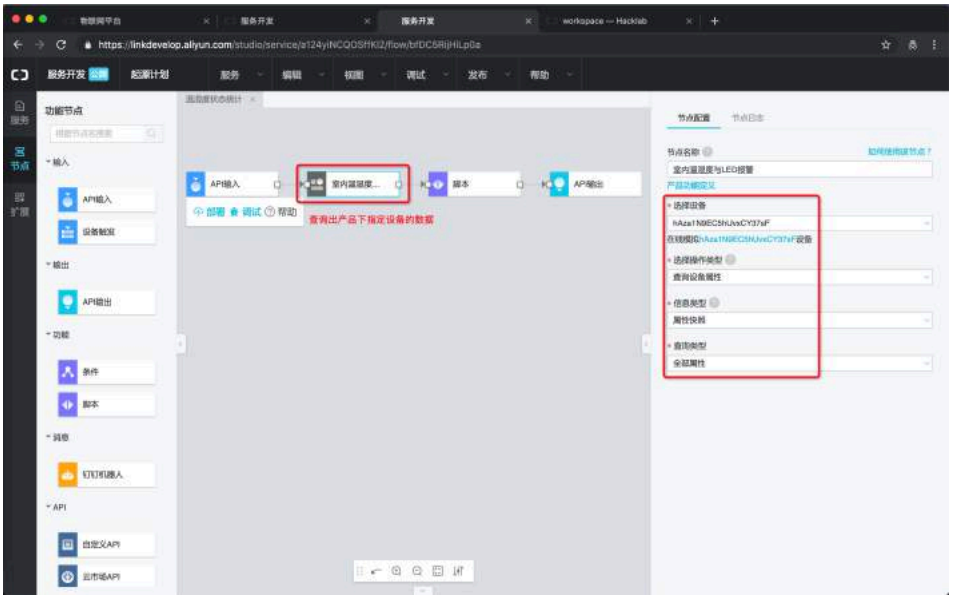


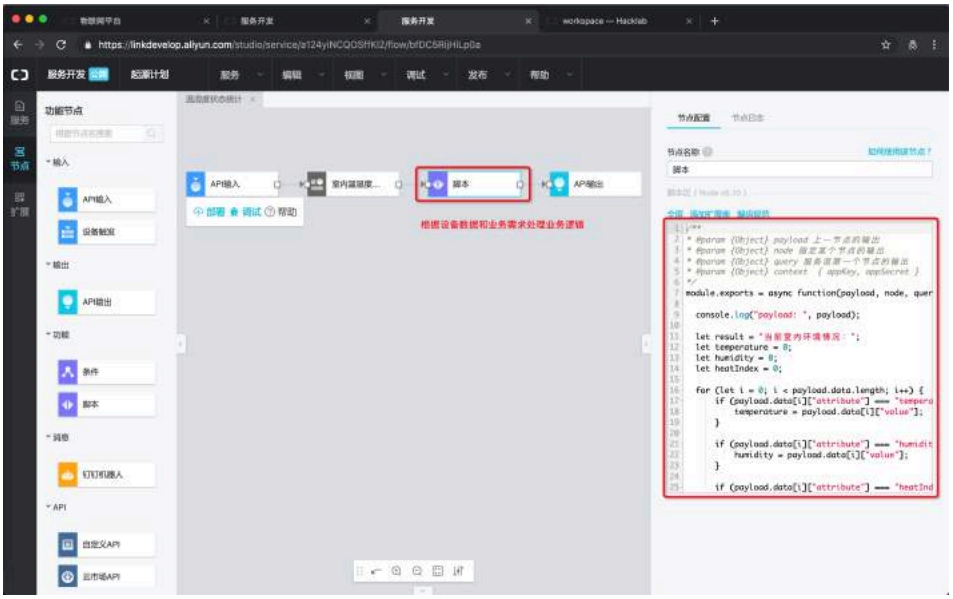


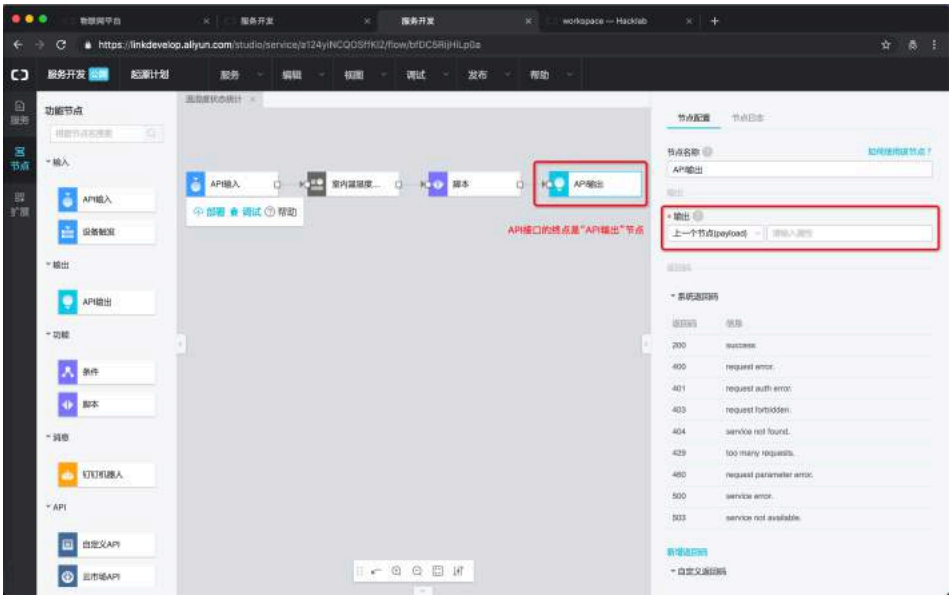


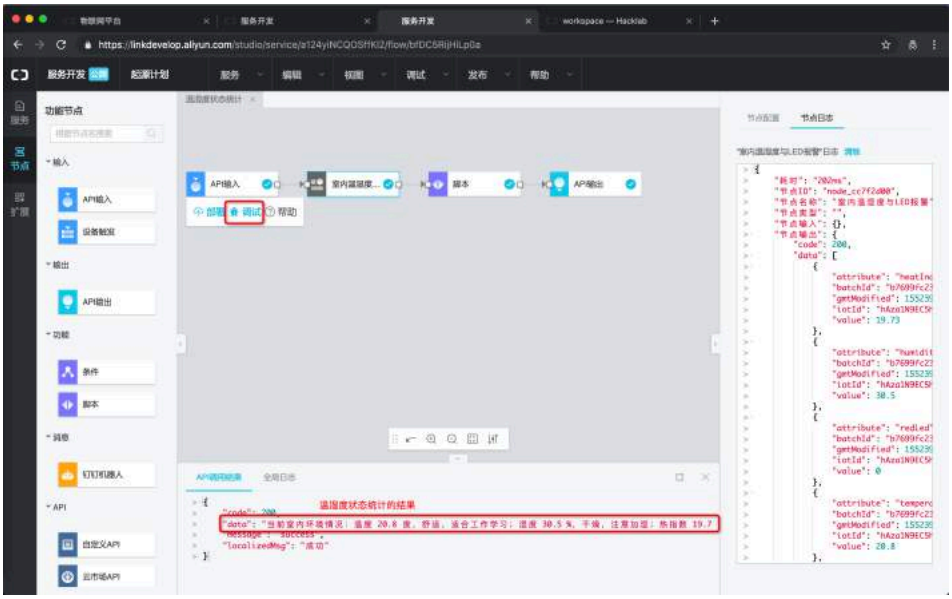
2）“温湿度状态统计”的 API 接口如下图所示，可以点击 API 输入节点下方的部署和调试运行试一试，这个 API 接口将会被 Web 前端应用调用；




服务开发
API输入
室内温湿度...
脚本
API输出
查询出产品下指定设备的数据


服务开发
API输入
室内温湿度...
脚本
API输出
根据设备数据和业务需求处理业务逻辑



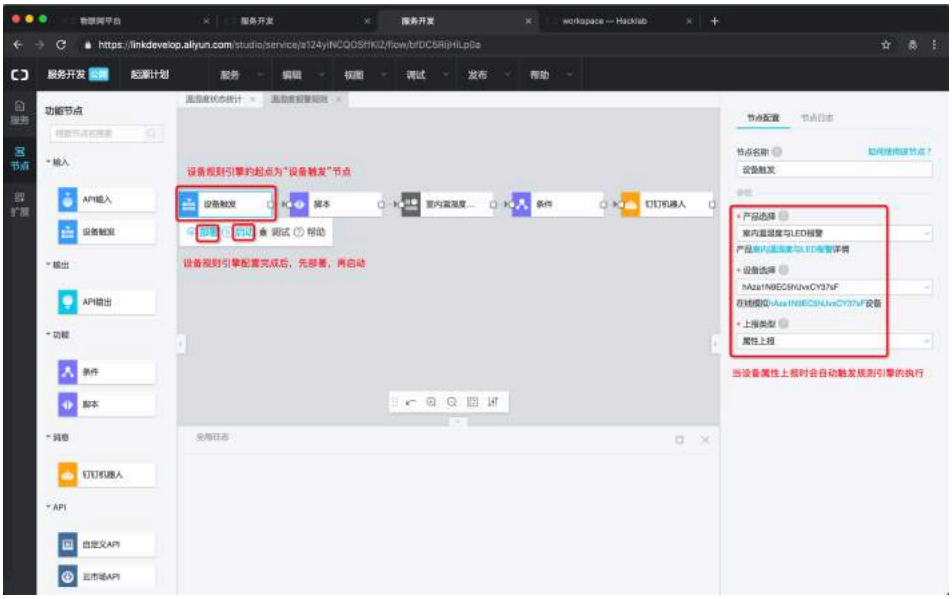


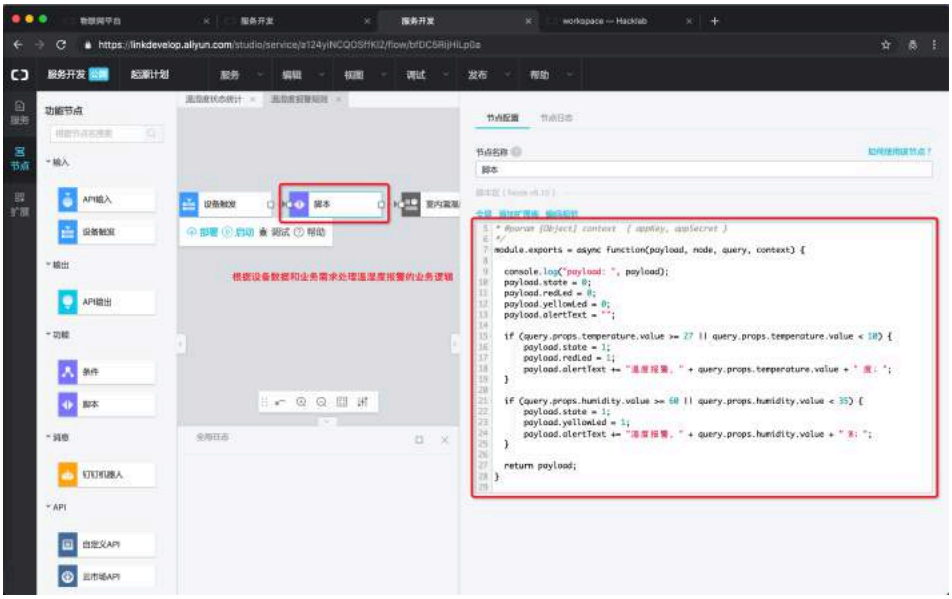
附脚本节点代码：

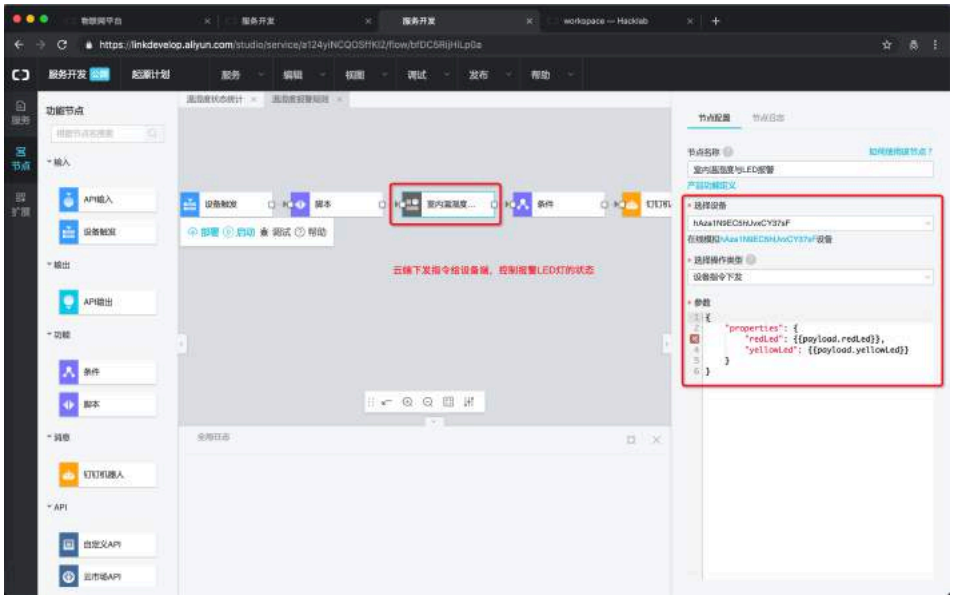
```
/**
* @param {Object} payload 上一节点的输出
```

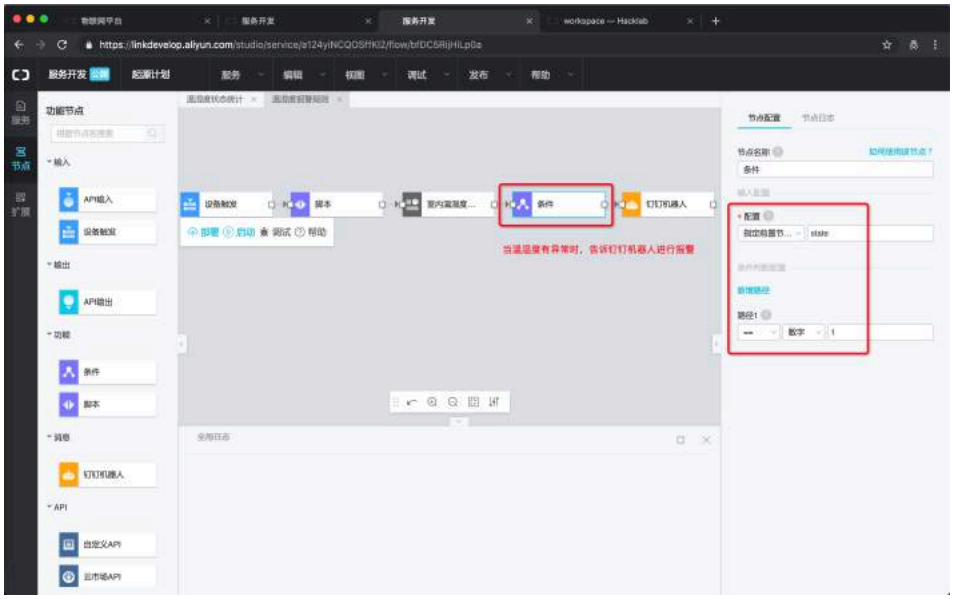
```
* @param {Object} node 指定某个节点的输出
* @param {Object} query 服务流第一个节点的输出
* @param {Object} context  { appKey, appSecret }
*/
module.exports = async function(payload, node, query, context) {

  console.log("payload: ", payload);

  let result = "当前室内环境情况：";
  let temperature = 0;
  let humidity = 0;
  let heatIndex = 0;

  for (let i = 0; i < payload.data.length; i++) {
      if (payload.data[i]["attribute"] === "temperature") {
          temperature = payload.data[i]["value"];
      }

      if (payload.data[i]["attribute"] === "humidity") {
          humidity = payload.data[i]["value"];
      }

      if (payload.data[i]["attribute"] === "heatIndex") {
          heatIndex = payload.data[i]["value"];
      }
  }

  if (temperature >= 27) {
      result += "温度 " + temperature + " 度，太热，小心中暑；"
  } else if (temperature >= 10 && temperature < 27) {
      result += "温度 " + temperature + " 度，舒适，适合工作学习；"
  } else {
      result += "温度 " + temperature + " 度，太冷，注意防寒保暖；"
  }

  if (humidity >= 60) {
      result += "湿度 " + humidity + " %，潮湿，注意除湿；"
  } else if (humidity >= 35 && humidity < 60) {
      result += "湿度 " + humidity + " %，适宜，适合锻炼；"
  } else {
      result += "湿度 " + humidity + " %，干燥，注意加湿；"
  }

  result += "热指数 " + heatIndex + " 度。"

  return result;
}
```

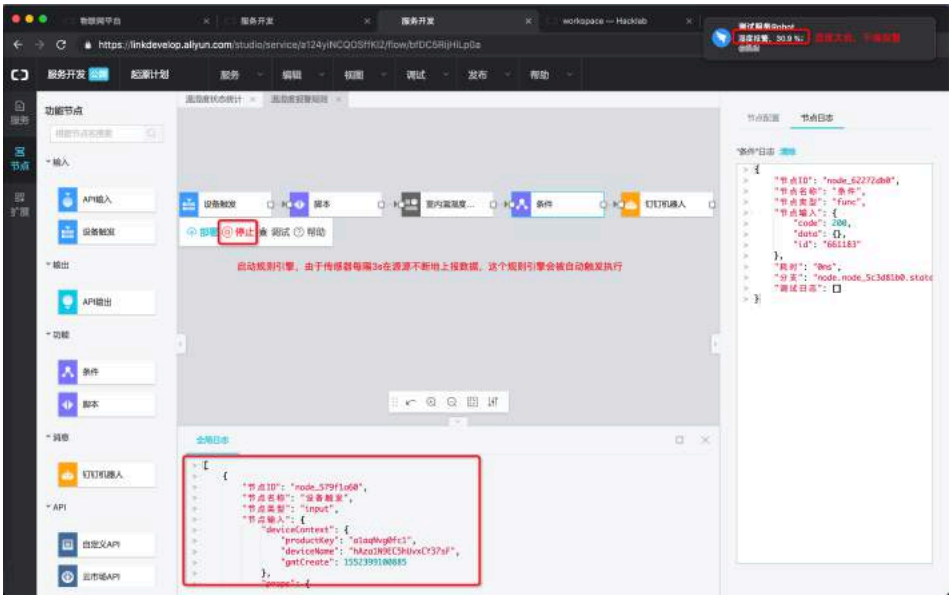
3）“温湿度报警规则“的设备规则引擎如下图所示，可以点击设备触发节点下方的部署和启动试一试，这个规则引擎将会在设备有属性上报时自动执行；






服务开发
编辑
视图
调试
发布
帮助
功能节点
设备触发
脚本
室内温湿度...
条件
钉钉机
云端下发指令给设备端，控制报警LED灯的状态
节点配置
节点名称
选择设备
选择操作类型
设备指令下发
参数
{
"properties": {
"redLed": {{payload.redLed}},
"yellowLed": {{payload.yellowLed}}
}
}


服务开发
编辑
视图
调试
发布
帮助
功能节点
设备触发
脚本
室内温湿度...
条件
钉钉机器人
当温湿度有异常时，告诉钉钉机器人进行报警
节点配置
节点名称
条件
配置

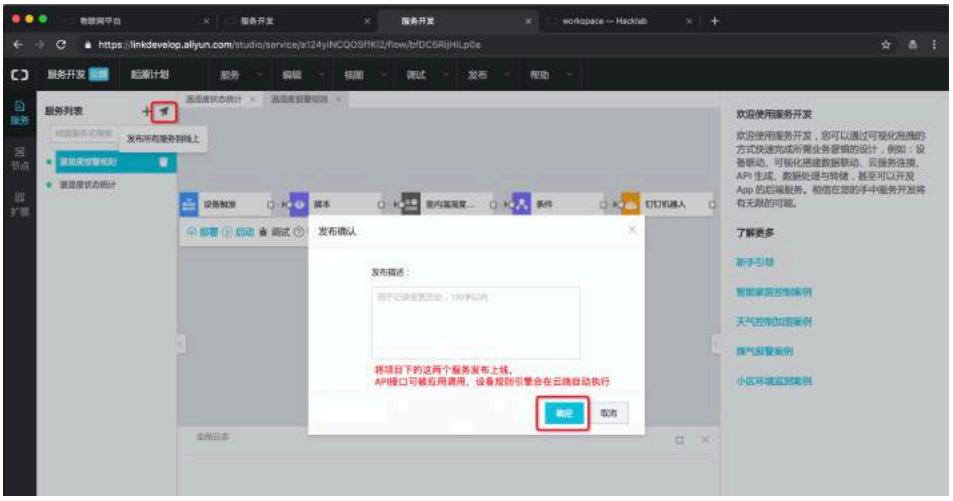


附脚本节点代码：

```
/**
* @param {Object} payload 上一节点的输出
* @param {Object} node 指定某个节点的输出
* @param {Object} query 服务流第一个节点的输出
* @param {Object} context  { appKey, appSecret }
*/
module.exports = async function(payload, node, query, context) {

  console.log("payload:", payload);
  payload.state = 0;
  payload.redLed = 0;
  payload.yellowLed = 0;
  payload.alertText = "";

  if (query.props.temperature.value >= 27 || query.props.temperature.value < 10) {
      payload.state = 1;
      payload.redLed = 1;
      payload.alertText += "温度报警，" + query.props.temperature.value + "度；";
  }

  if (query.props.humidity.value >= 60 || query.props.humidity.value < 35) {
      payload.state = 1;
      payload.yellowLed = 1;
      payload.alertText += "湿度报警，" + query.props.humidity.value + "%；";
  }
```

```
    return payload;
}
```

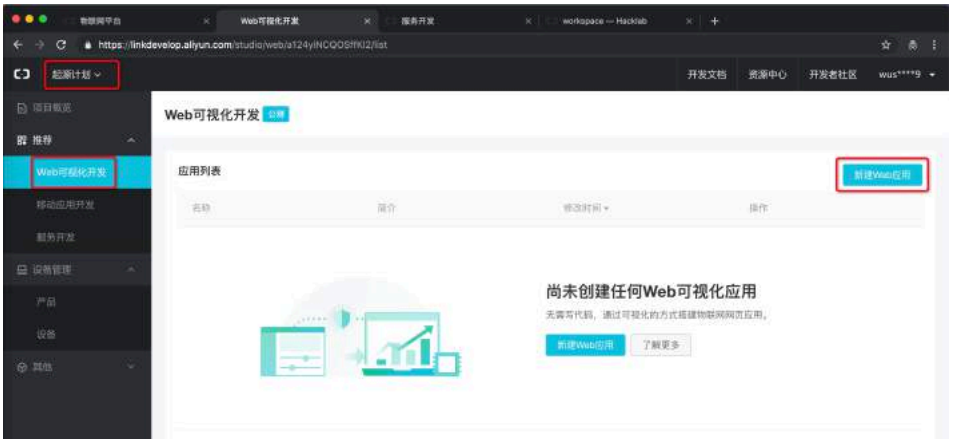
4）API 接口和规则引擎调试完成后，可以点击发布按钮将其正式发布上线；

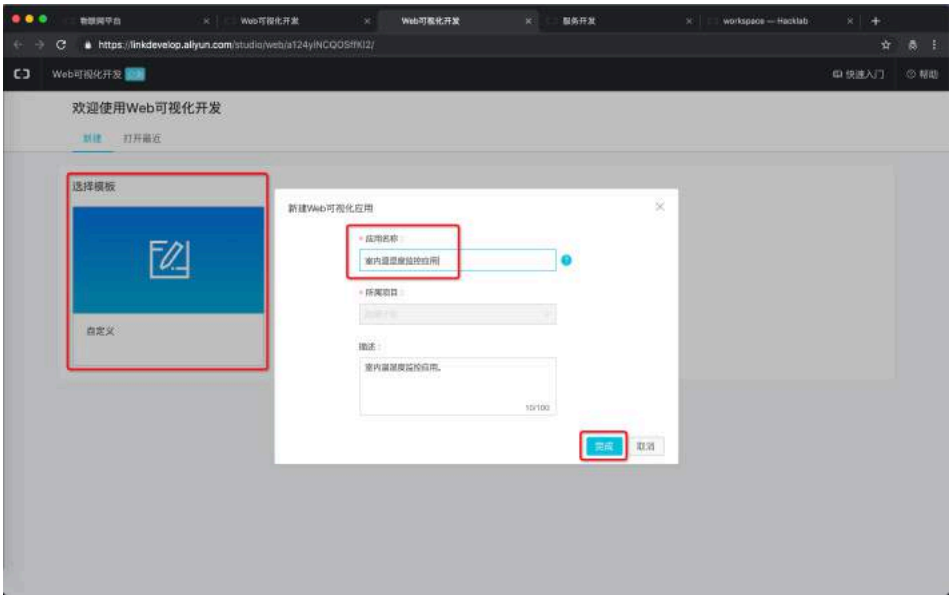


综上，我们完成了 API 接口和规则引擎的开发和部署上线。

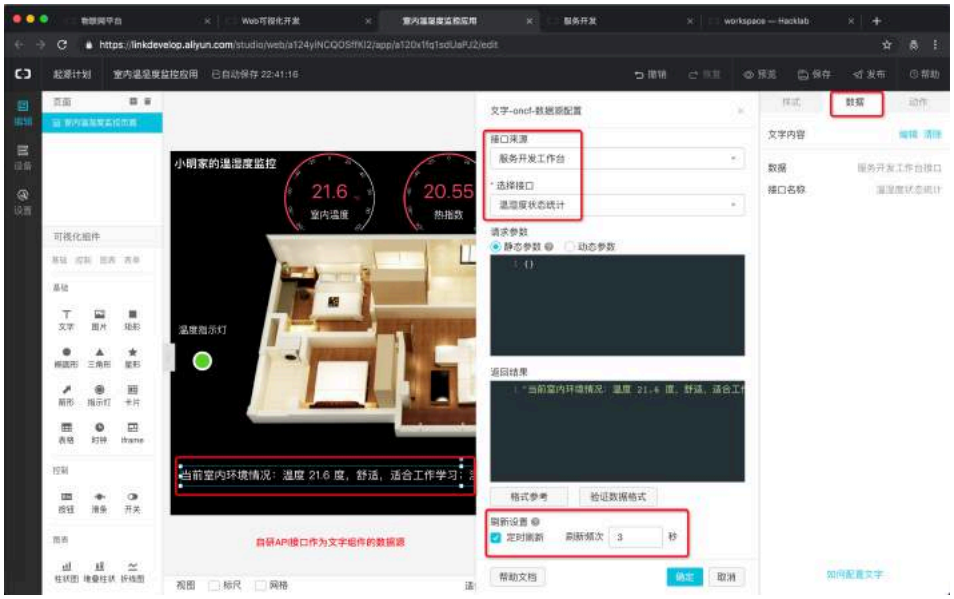
第 5 步　WebPage Workbench：可视化开发 Web 前端页面

1）进入起源计划项目的 Web 可视化开发页面，创建一个“室内温湿度监控应用”的 Web 前端应用；

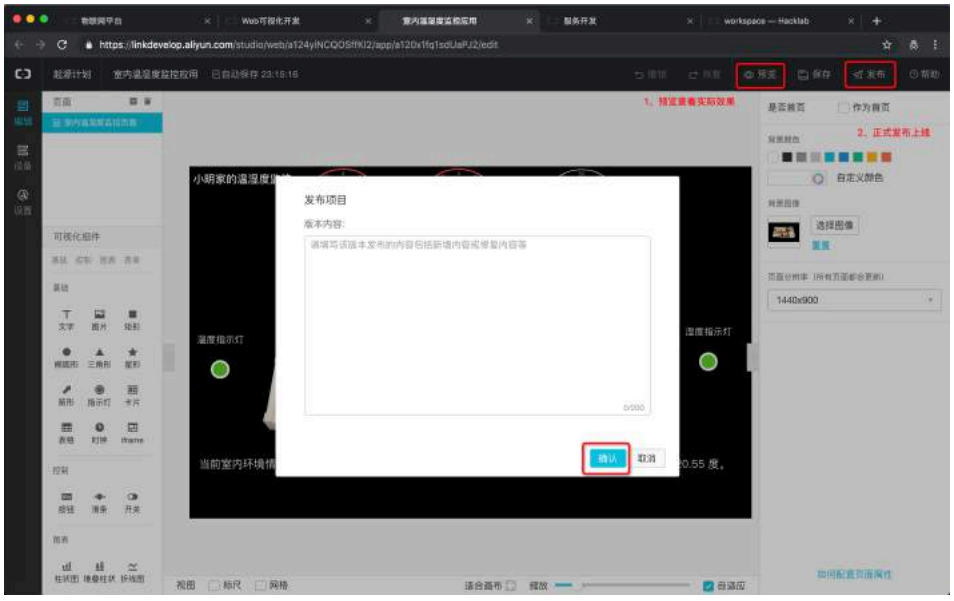




2）对室内温湿度监控应用进行可视化配置，包括修改背景颜色、添加并配置组件等；



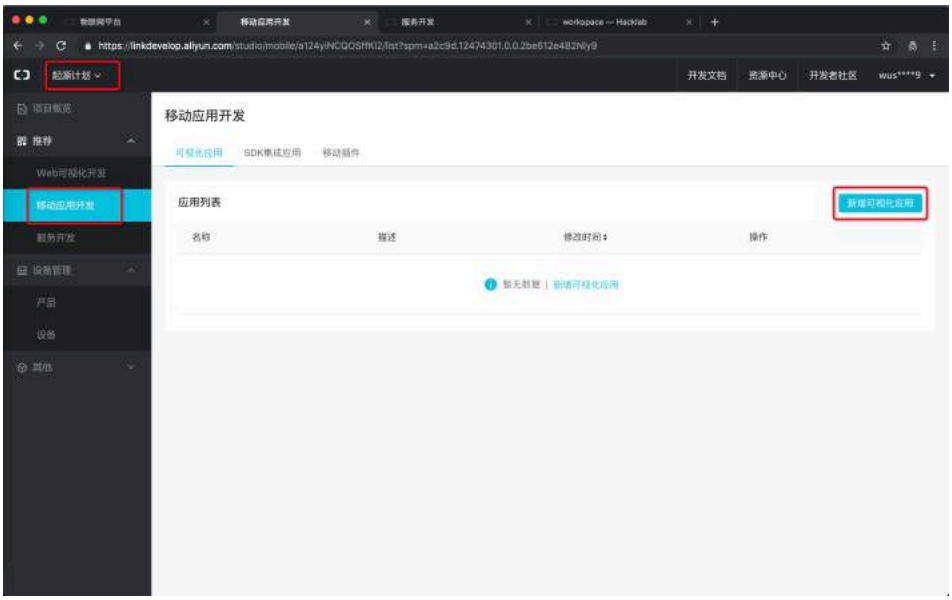
3）配置完成后，可以点击右上方的“预览”来查看实际效果，符合预期则可以正式发布上线；

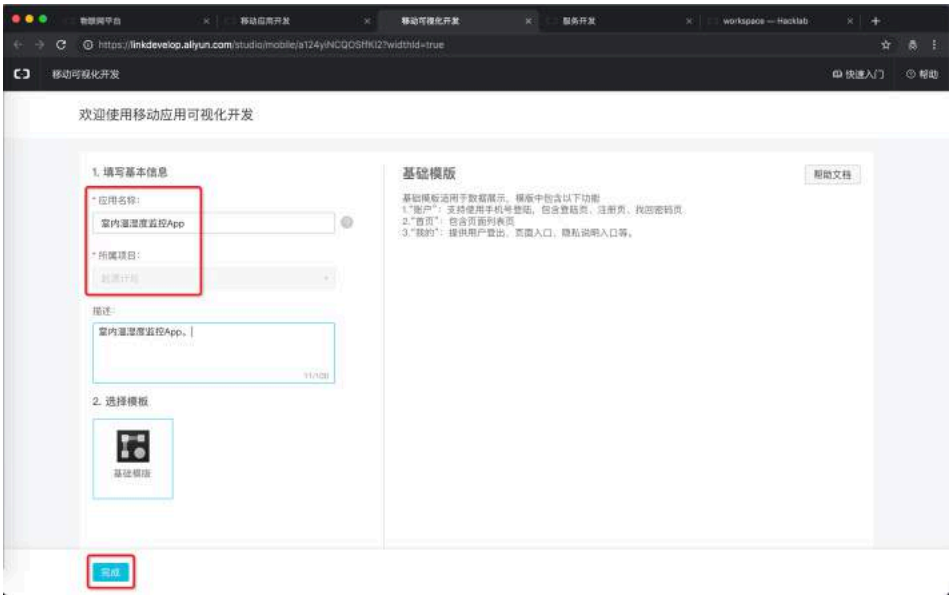


综上，我们完成了 Web 前端页面应用的开发和部署上线。

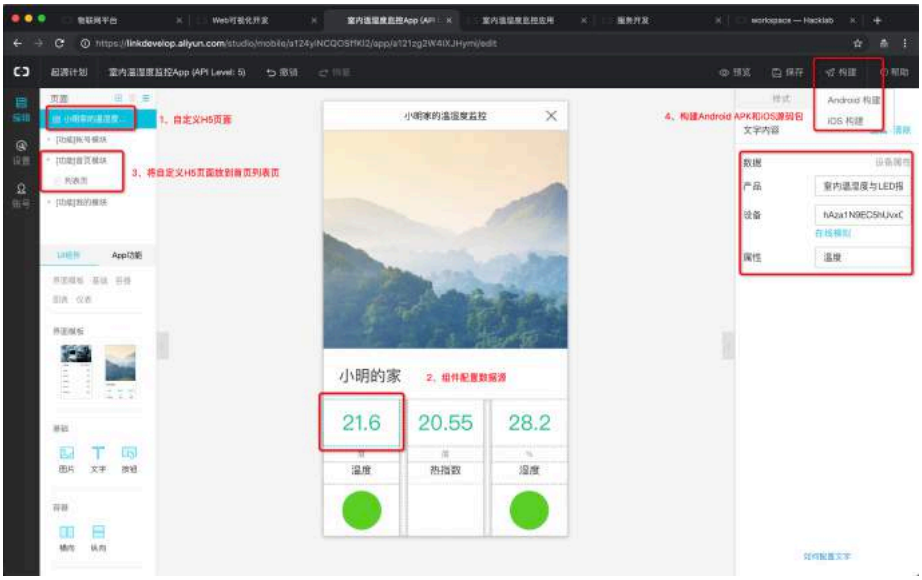
第 6 步　MobileApp Workbench：可视化开发移动 App

1）进入起源计划项目的移动应用开发页面，创建一个“室内温湿度监控 App”的移动应用；

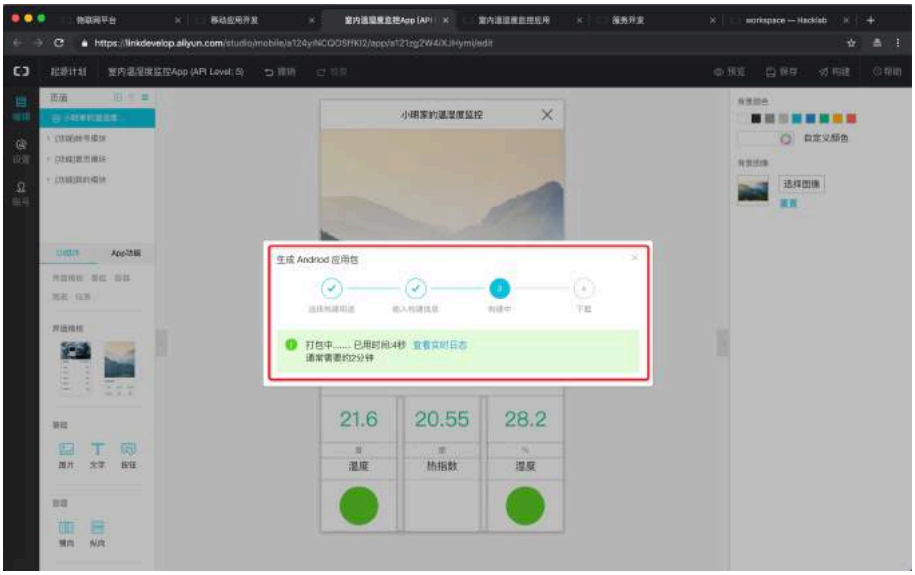




2）进入室内温湿度监控 App 的编辑页面，对应用进行可视化配置，包括页面、首页内容等；



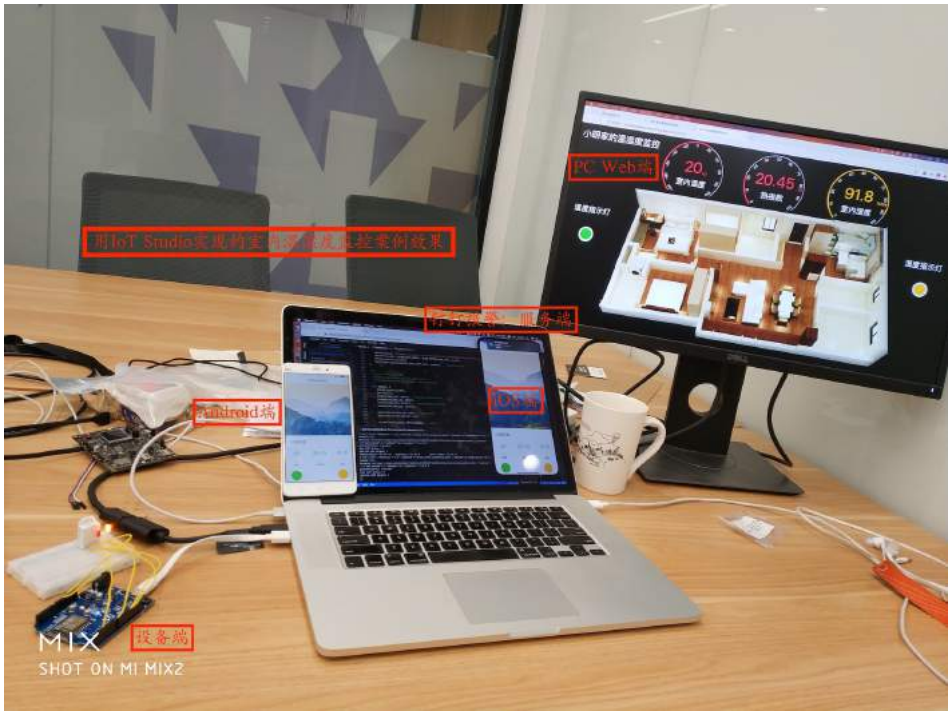
3）配置完成后，可以点击右上方的“预览”来查看页面的实际效果，符合预期则可以构建 Android APK 安装包（直接扫码下载安装即可使用）和 iOS 源码包（在 XCode 打包编译即可使用）；



综上，我们完成了移动客户端应用的开发和打包构建。

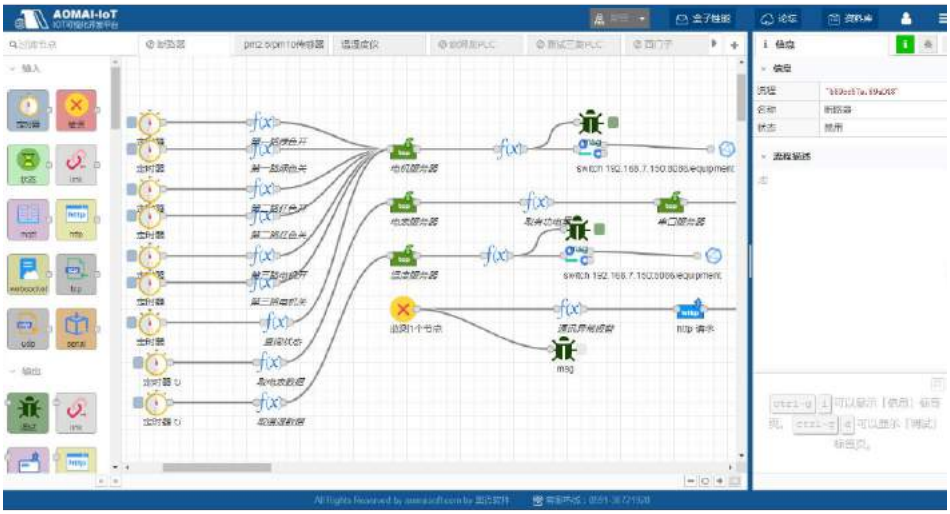
第 7 步　整体效果

综上，我们使用 IoT Studio 提供的工具能力完成了一个简单的“温湿度监控”端到端物联网应用解决方案。大家也可以结合自己的实际场景使用 IoT Studio 开发一个自己的物联网应用解决方案。



5 分钟完成硬核工业 PM2.5 监控

作者：SL



概述

为什么要做这个平台呢？起初我们想的是在这个时间就是金钱的时代，在这个数据可以产生价值的时代，有没有一款软件可以非常快速并且不需要写复杂代码就可以帮助我们实现数据上面的采集，这就是我们想做这款软件的初衷。

物料清单

硬件 (1)	软件 (1)
pm2.5/pm10 激光传感器 本产品使用激光散射原理，能够得到空气中 0.3 ~ 10 微米悬浮颗粒物浓度，使用进口激光器与感光部件，数据稳定可靠；内置风扇，数字化输出，集成度高。(×1)	AM-IoT 可视化开发平台 可视化的流编程开发，快速、简单、直观、易上手、方便调试。 预装了兼容市面上 90% 的 PLC 以及其他的数据采集通讯协议。 支持自定义节点，自由创作自定义协议。

方法 & 步骤

第 1 步　通讯协议

首先先把 pm2.5/pm10 激光传感器插在电脑上

然后通过厂家给的说明书里的通讯协议我们可得知设备默认参数

- **串口通讯协议：9600 8N1（速率 9600，数据位 8，校验位无，停止位 1）**
- **串口自动上报通讯周期：1+0.5 秒**
- **数据帧（10 字节）：报文头 + 指令号 + 数据（6 字节）+ 校验和 + 报文尾**

示例报文为

```
AA C0 71 01 CA 01 B9 93 89 AB
```

具体含义：

- AA---- 报文头
- C0---- 指令号，客户开发产品时，看到接收到有 CO，即表示是由 PM2.5 传感器输出的信号
- 71----PM2.5 低字节
- 01----PM2.5 高字节
- CA----PM10 低字
- 01----PM10 高字节
- B9---- 传感器的 ID
- 93---- 传感器的 ID
- 89---- 校验和，即 71+01+CA+01+B9+93=289 即 0x0289 ，这里我们舍弃了高字节 02，只保留了低字节 89
- AB---- 报文尾

因为输出的是，16 进制数据，请转换成 10 进制数进行计算。

PM2.5 值的计算：**71 01**

- 低字节 71: 7*16+1=113
- 高字节 01: 0*16+1=1 ((PM2.5 高字节*256) + PM2.5 低字节)/10 (1*256+113)/10=36.9ug/m3

PM10 值的计算：CA 01

- 低字节 CA: C*16+A=202
- 高字节 01: 0*16+1=1 ((PM10 高字节*256) + PM10 低字节)/10 (1*256+202)/10=45.8ug/m3

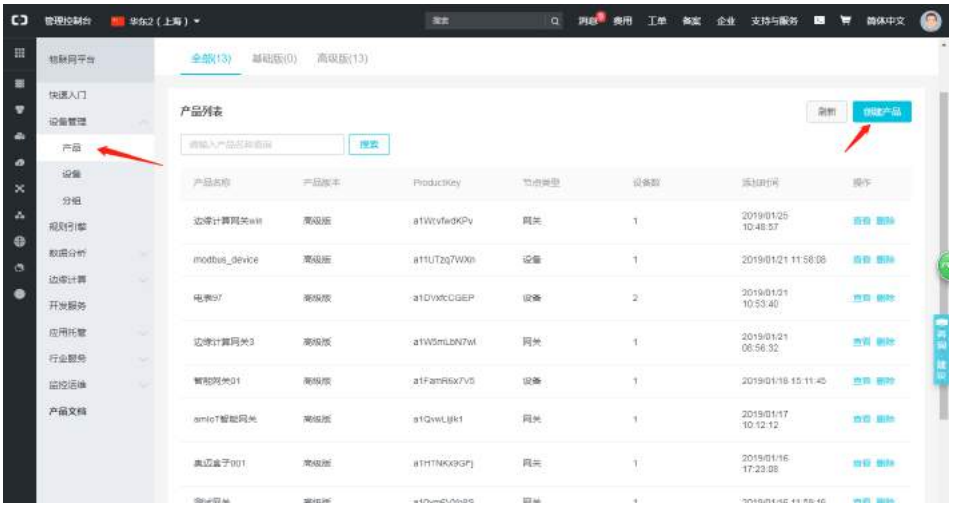
校验和：

- 89 71+01+CA+01+B9+93=289，舍弃高字节 02，留低字节 89

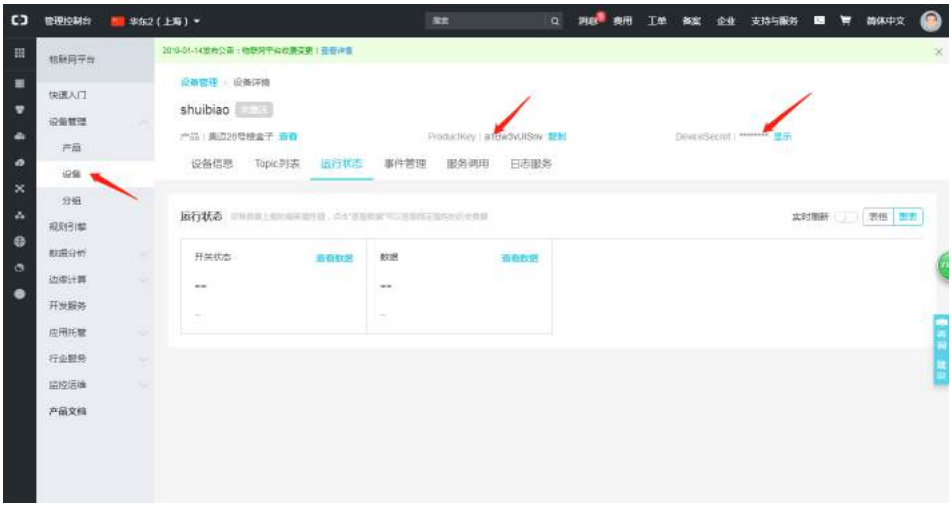
第 2 步　配置阿里云 IoT

注册阿里云 IoT 账号 在阿里云 IoT 注册账号

创建产品　登录账号后在阿里云 IoT 产品页创建产品，选择高级版，填写相应信息，这边我们选择上传的是温湿度信息，节点类型选择网关，点击完成即可快速创建。



创建设备　创建产品完毕后创建相应设备，查看设备详情 点击刚才创建的设备查看 ProductKey(产品 key)、DeviceSecret(产品密钥)、DeviceName(设备名称)。



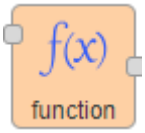
第 3 步　配置节点流

打开 AM-IoT 可视化开发平台软件，进入可视化编辑器。本次教程需要用到如下节点，在左侧节点栏中拖拽出使用。

1. **serial** 在输入栏目，用于读取串口二进制流。

2. **延迟** 在功能栏目，用于延迟消息速率。

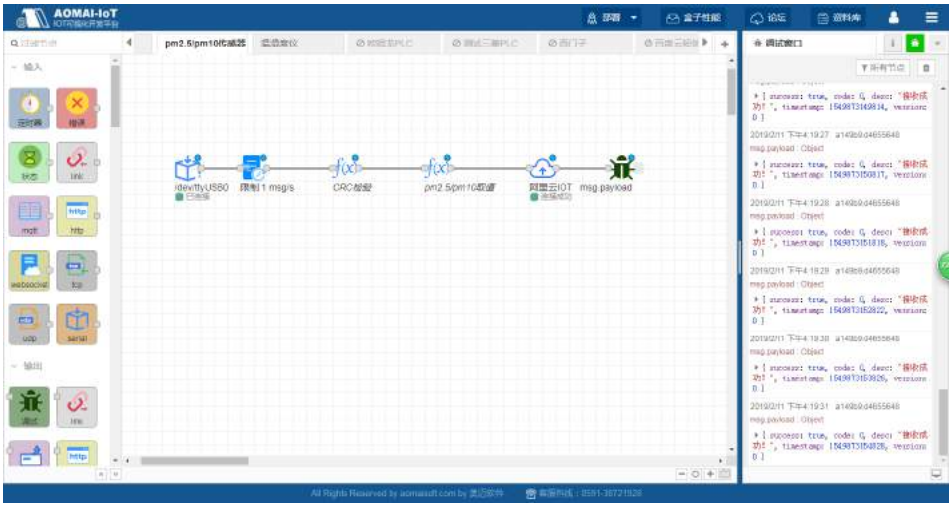
3. **function** 在功能栏目，用于配置逻辑代码。



4. **阿里云 IOT** 数据上传到阿里云。

5. **调试** 在输出栏目，用于调试输出。

配置界面详情



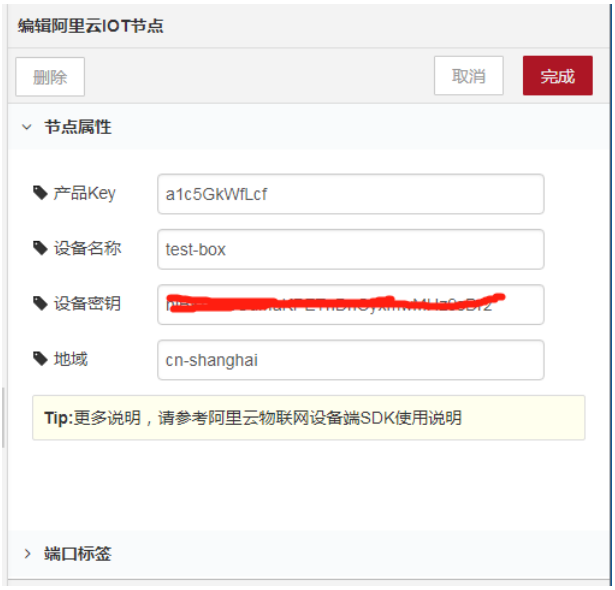
接下来我们来配置如上图的节点流，首先将左侧节点栏的 **serial、延迟、function、调试**节点分别拖拽到工作区，再点击相应的流节点的端口依次按配置界面所示连接起来，再双击相应流节点进入配置界面配置相应属性。

- **serial** 是用于读取串口输出的节点工具。该工具需要配置串口名称 **(Serial Port)**、波特率 **(Baud Rate) 9600**、数据位 **(Data Bits) 8**、校验位 **(Parity) Node**、停止位 **(Stop Bits) 1**。该 USB 转串口程序在奥迈智能网关中使用不需要下载任何驱动程序，即插即用。在插入智能网关的 USB 口后，会在系统的驱动目录下 **/dev** 生成一个串口文件 **ttyUSB0**（没有其他 USB 转串口的工具插入下），我们需要在配置串口名称 **(Serial Port)** 配置 **/dev/ttyUSB0**。Windows 下需要安装官方驱动。由于 **serial in** 读取串口数据是连续的，所以我们需要将 Split input（拆分输入）配置成 after a silence of（在没有新的二进制流输出后），默认时间填写 50ms。详情配置如下图。

- **延迟** 用于延迟 serial 节点输出的信息速率。由于该设备自定义速率为 0.5 秒 / 信息，我们实际采集的时候用不到这么频繁的数据，所以我们选择限制消息速率，如下配置限制消息 1 分钟 / 条。

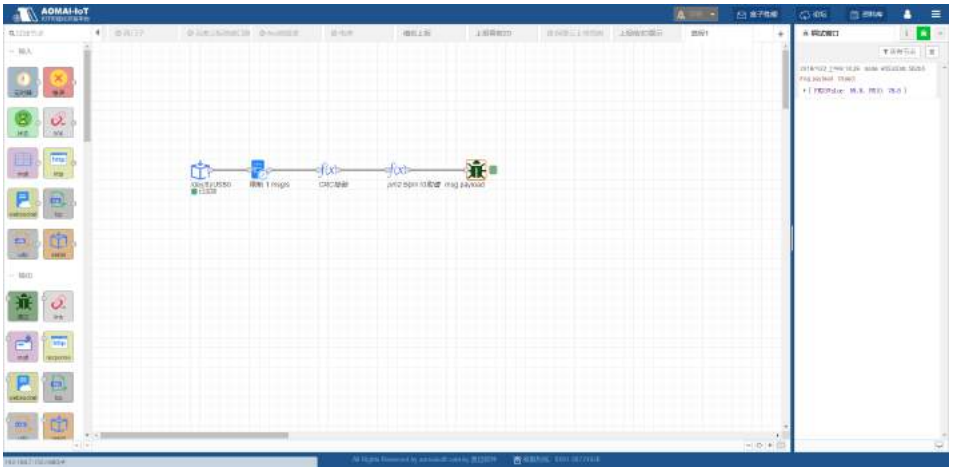
- **function**　是用于编写自定义代码的节点工具，该控件支持 nodejs 语法，可以实现您所有的业务逻辑。根据该设备的通讯协议，我们编写出如下的逻辑代码，需要注意的是在我们的框架内定义一般流的数据向下流动时都将数据存入 msg.payload 这个对象中。关于更多 Buffer 类的更多使用请参考 Buffer。详情配置如下图。

- **调试** 用于界面调试输出结果。我们需要将上面的程序输出结果打印在界面右侧的调试窗口，按配置界面图链接即可
- **阿里云 IOT** 用于将数据上传到阿里云 IOT



第 4 步 部署调试

经过上面所有步骤后，即可部署程序，部署后可以在右侧的调试窗口看到输出，如下图。



树莓派实现人脸识别

作者：林楚昂

摘要

【新兵日记】例程通过树莓派和阿里云物联网平台实现了人脸识别的应用。本文详细地记录了操作过程的细节与遇到的问题，希望帮助像我们一样的小白快速熟悉平台。上篇为逻辑解释和准备工作介绍。

【例程实现】金锐 ZJUAIoT、林楚昂 ZJUAIoT

原始文档链接

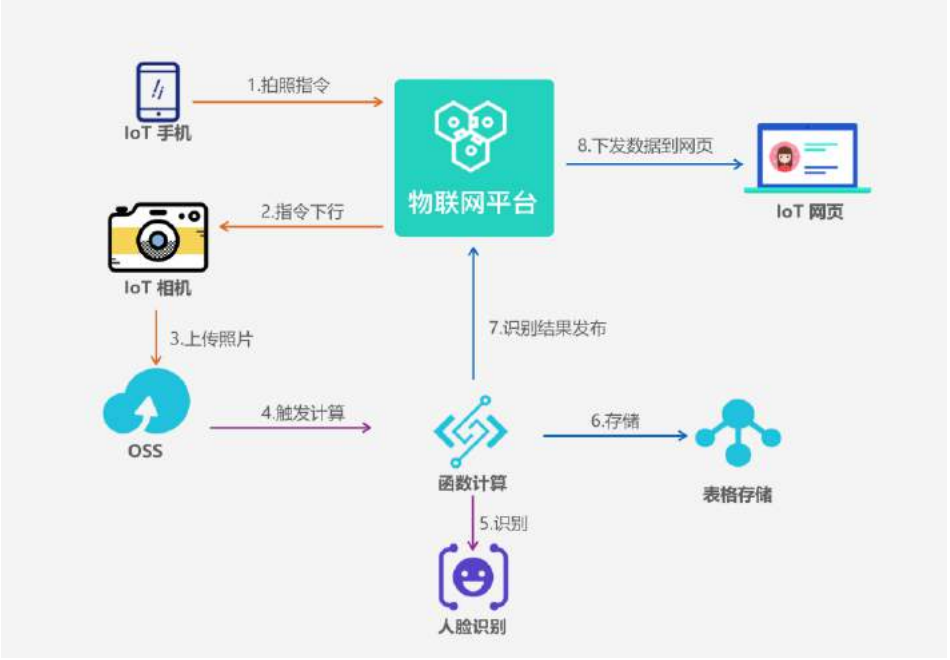
原始阿里云专家文档链接：https://www.yuque.com/cloud-dev/iot-tech/potgof

本文基于该专家文档进行学习与复现。

一、流程实现的理论知识

1. 流转逻辑

在开始操作之前，我们有必要先了解一下整个业务逻辑的流转过程。（小编开始时没有准找流程无脑开始，中间遇到的问题比较多。）这是一套基于阿里云的 Serverless 架构，主要由下图中的 8 个步骤实现。



1.1 拍照指令

这一步很容易理解，通过在 IoT 手机端或其他设备向**物联网平台**发送请求拍照的指令。平台接受消息后编写 SQL 对 Topic 中的数据进行处理，通过规则引擎配置转发动作将数据流转到控制拍照的 topic。可用于会场门禁等场所实现拍照指令的下达。

1.2 指令下达

连接好的 IoT 相机订阅了控制拍照的 topic，物联网平台在接受命令之后，开始对 IoT 相机下达命令，通过向设备的 /control Topic 上发送消息，即可触发，由相机完成拍摄的动作。例程实现使用的是树莓派连接的摄像头，通过申请的设备三元组连接平台。

1.3 上传照片

拍摄好的照片上传到**对象存储 OSS** 的 bucket，通过 Python3 oss2 的 SDK 包将图片上传至云平台图像 bucket，等待下一步命令的执行。

1.4　触发计算

函数计算关联着 OSS 触发事件，编写过函数计算的内容后，函数计算可以被 OSS 的一些事件触发，即 OSS 一旦有图片上传过来会触发函数计算，就自动执行函数计算中的函数。

1.5　识别

人脸识别的主要部分。函数计算可以通过 OSS 触发的参数查询到图片在 OSS 的 bucket 中的存储地址，函数计算使用该图片的 uri 链接作为图片资源调用**阿里云人脸识别服务**的 API（需在人脸识别平台开通服务）。

1.6　存储

函数计算表格存储 OSS 发送的消息以及获取到的参数的一种方式。

1.7　识别结果发布

函数计算会将阿里云人脸识别服务 API 返回的结果发布到物联网平台。

1.8　下发数据到网页

物联网平台在获取函数计算的结果后推送到 IoT 网页，结合上传的静态页面，展示在网页上。

2. 阿里云产品

IoT 平台：https://www.aliyun.com/product/iot

函数计算：https://www.aliyun.com/product/fc

表格存储：https://www.aliyun.com/product/ots

OSS 存储：https://www.aliyun.com/product/oss

人脸识别：https://data.aliyun.com/product/face

3. 官方原始文链接

阿里云文档链接：https://www.yuque.com/cloud-dev/iot-tech/potgof

二、实验设备准备工作

1. 硬件设备

- 树莓派

- 摄像头

- PC 端电脑（演示使用的是 Windows 系统，Mac 和 Linux 系统操作类似）
- 网线
- 树莓派显示屏（可不需要）

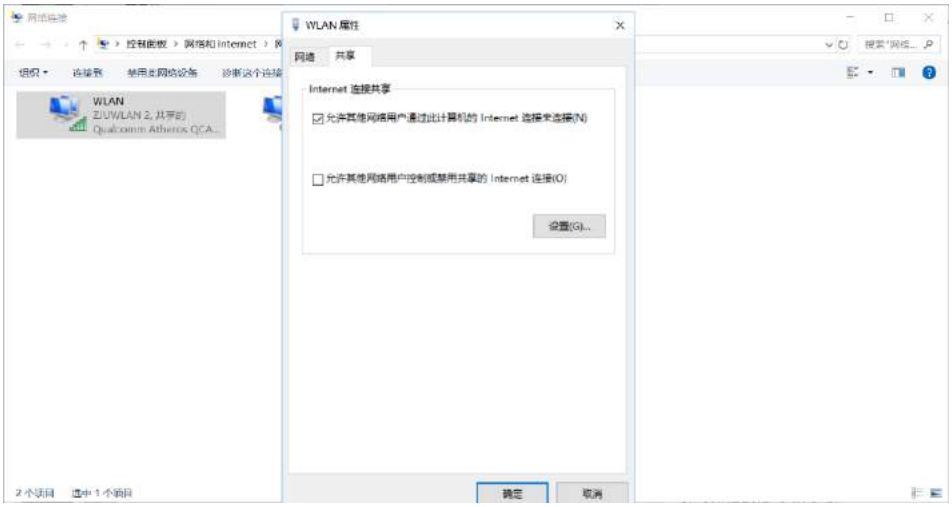
2. 硬件配置

树莓派配置

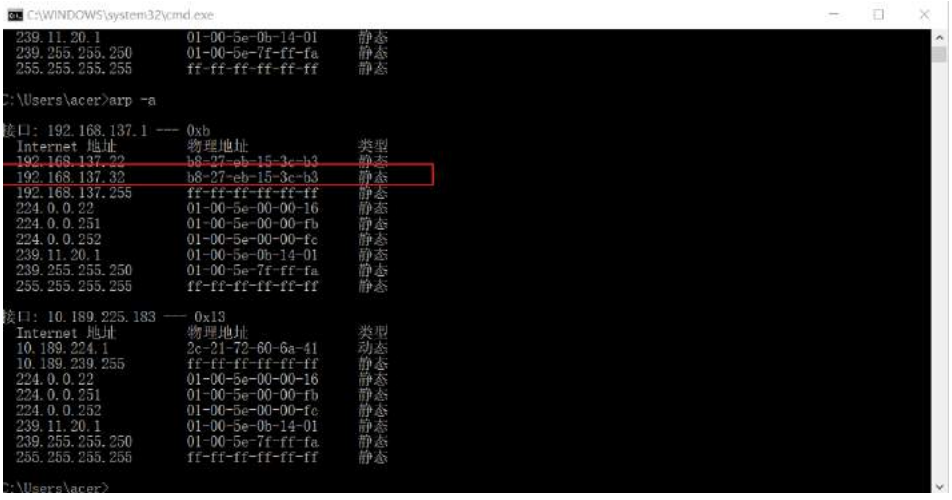
- 摄像头接入树莓派；

- 用网线连接树莓派与电脑；
- 要获取树莓派的 IP 地址有多种方法，演示不使用显示器的方式：

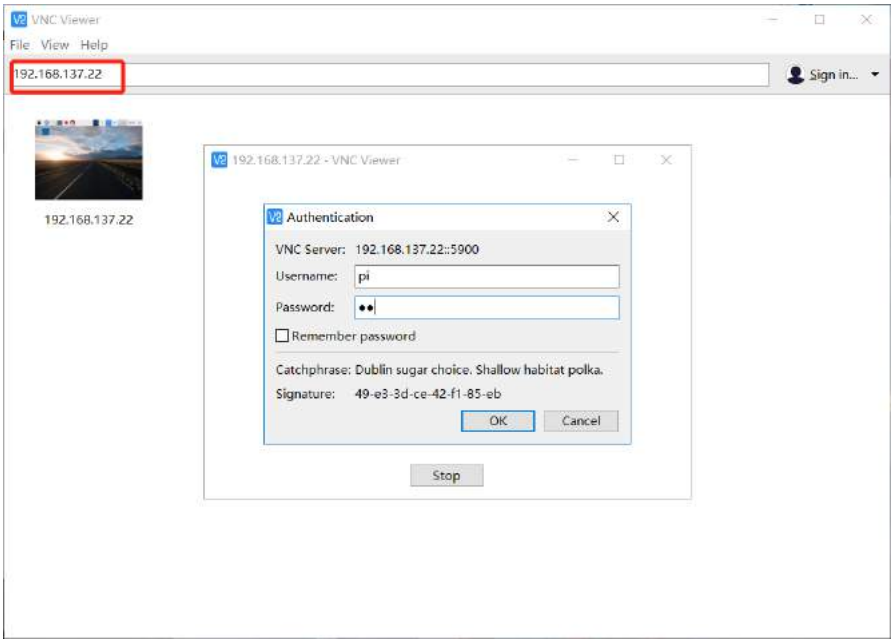
（1）开机（确保树莓派烧录系统，推荐官方 raspbain 系统），连接网络，并设置 WLAN 属性，选择“更改适配器选项”，勾选上“允许其他网络用户通过此计算机的 Internet 连接来连接”。



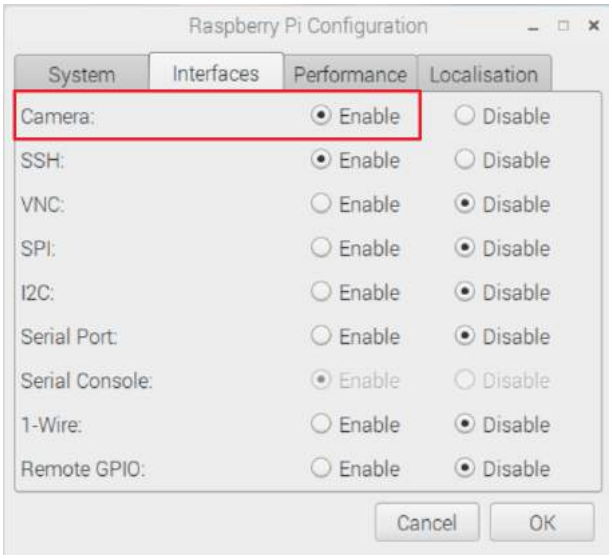
（2）打开命令行，输入“arp -a”，查询并记住树莓派的 IP 地址。



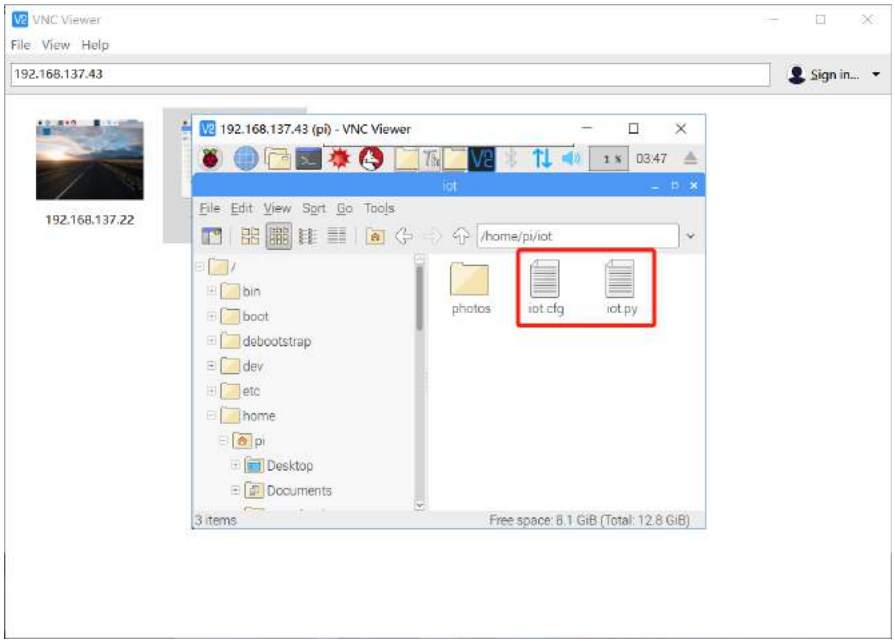
（3）可以使用 putty（ssh）或 VNC 远程控制树莓派，演示使用 VNC（有清晰的图形界面）。在方框中输入 IP 地址，输入用户名和密码。



（4）配置树莓派使摄像头处于工作状态：**详细操作**。



（5）文件夹创建：在树莓派相应目录（/home/pi/iot）下，创建两个新的文件：iot.cfg 和 iot.py。



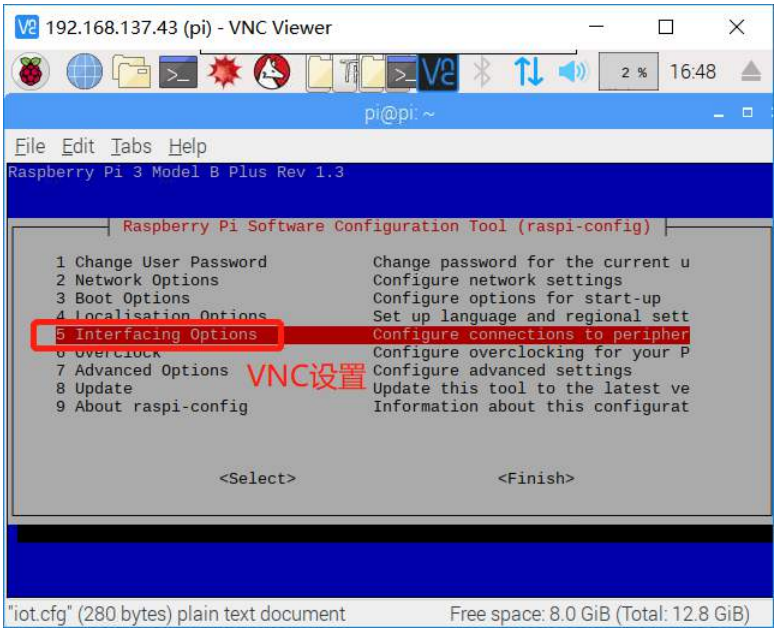
三、开启树莓派摄像头

1. 确保摄像头可用

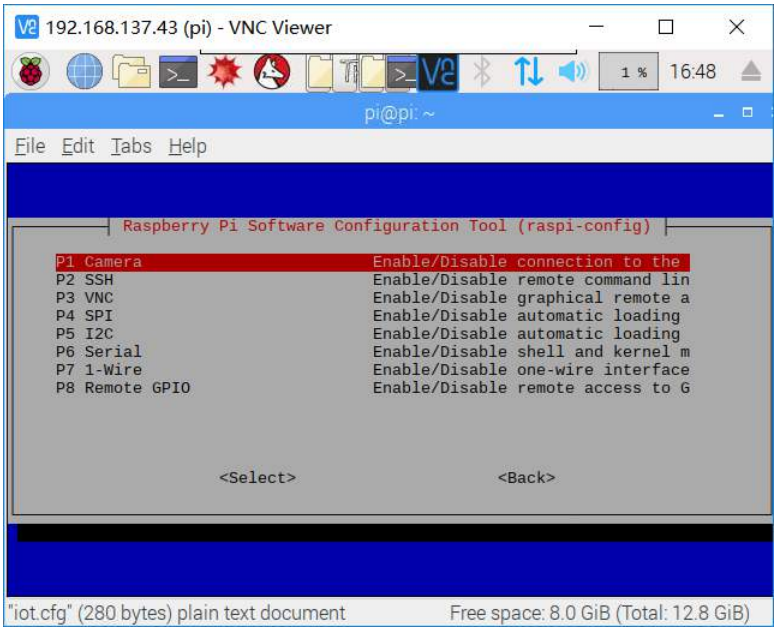
调用树莓派摄像头之前需要先确保树莓派连接的摄像头处于打开的状态

（1）通过 VNC 进入树莓派的终端。

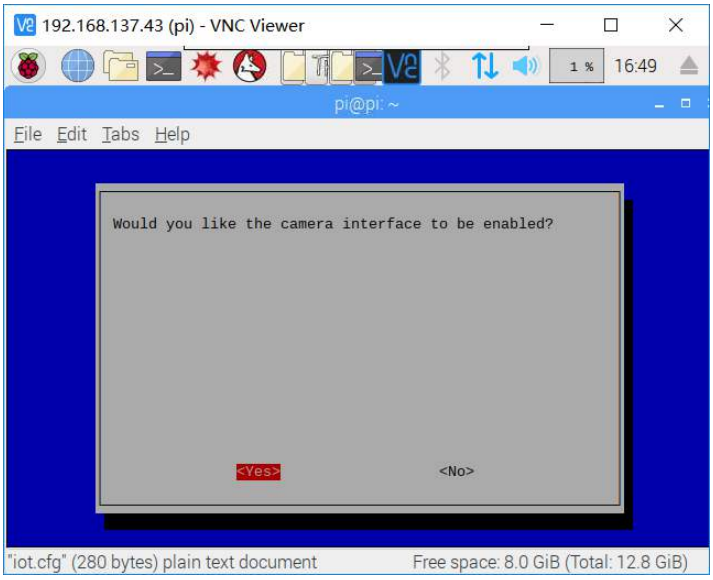
（2）输入 `sudo raspi-config` 进入到树莓派的系统配置界面。上下箭头移动光标，选择“Interfacing Options”为 VNC 设置，点击 OK 确认后，系统返回命令行安装，输入 y 确认等待安装完成即可。

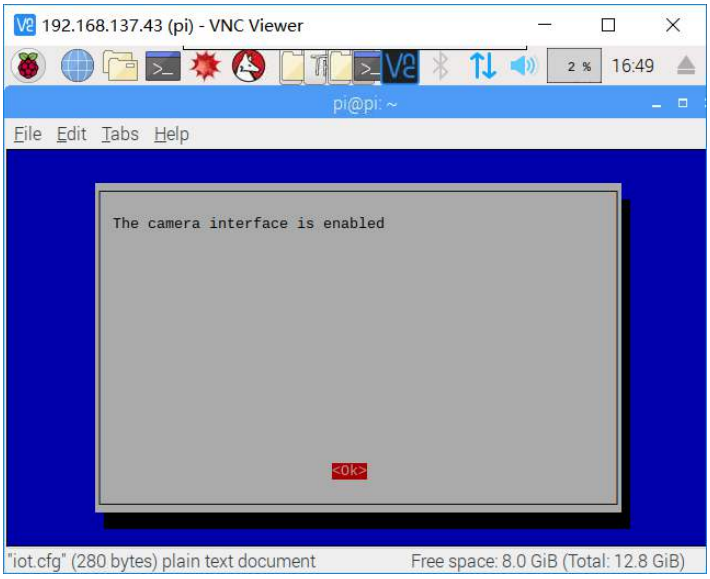


（3）“p1 camera”开始设置摄像头。



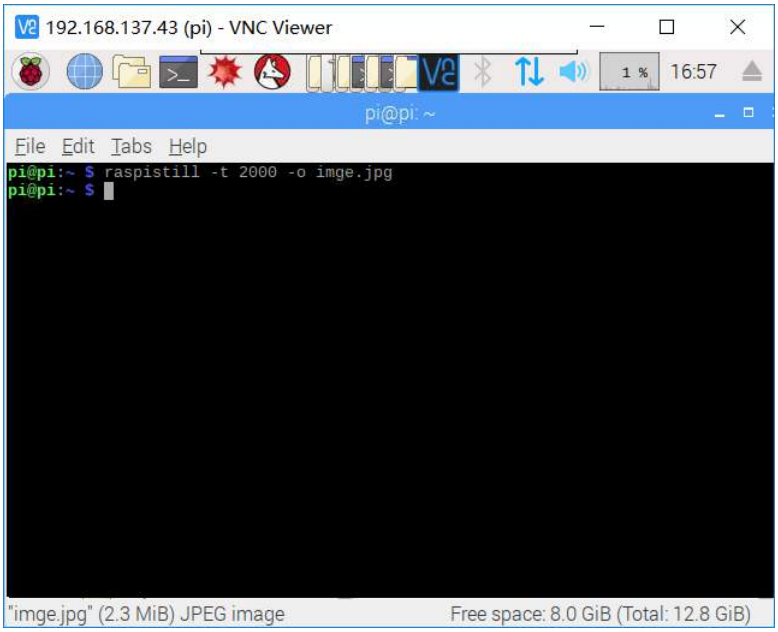
（4）选择 yes（也可能是 enable）打开摄像头。在退出 raspi-config 时会要求您重新启动。启用选项是为了确保重启后 GPU 固件能够正确运行（包括摄像头驱动和调节电路），并且 GPU 从主内存划分到了足够的内存使摄像头能够正确运行。





2. 拍摄测试

- 测试系统已经完成安装，并且可正常工作。可以尝试以下命令：`raspistill -v -o test.jpg` 这将显示来自摄像头 5 秒钟的预览图像，并且拍摄一张照片，然后保存为文件 test.jpg，同时显示出需要相关信息。 或者尝试 `raspitill -t 2000 -o imge.jpg` 这会在两秒钟（时间单位为毫秒）延迟后拍摄一张照片，并保存为 imge.jpg。



四、阿里云平台有关产品信息的获取方式

1. 设备接入方式

阿里云 IoT 支持设备两种接入方式：

1.1　先认证再连接

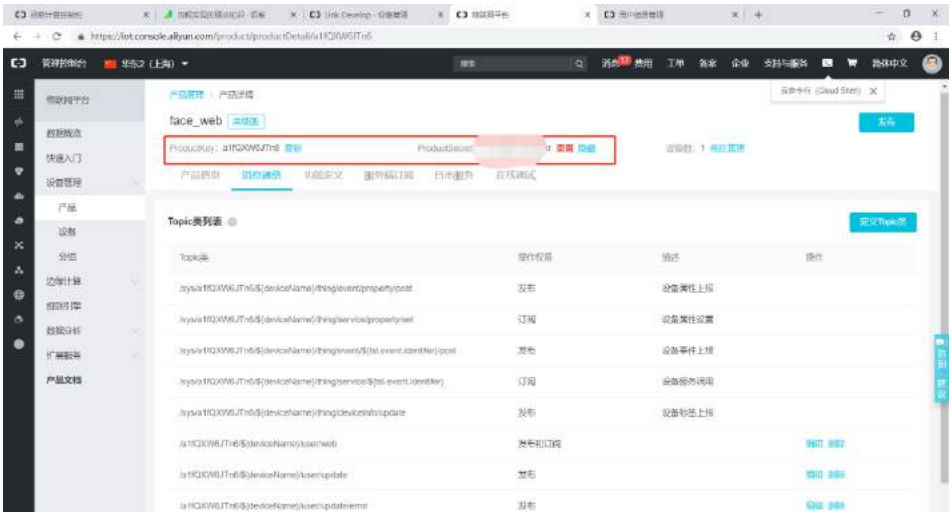
设备先使用 HTTPS 到 __iot-auth.${regionId}.aliyuncs.com:443__ 获取认证 __iotId 和 iotToken__ 后，再使用 MQTT 连接到 __ 指定的 endpoint 和 port__

1.2　使用域名直连

设备直连 endpoint 地址：${productKey}.iot-as-mqtt.cn-shanghai.aliyuncs.com:1883

2. 设备三元组的获取

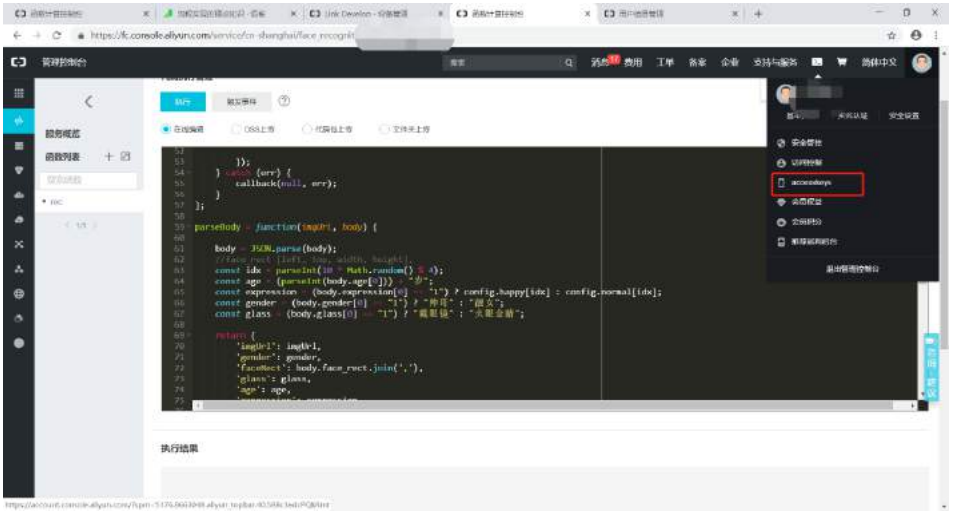
物联网平台上每一个新建的设备都会有对应的设备三元组（ProductKey、ProductSecret 和 DeviceName），打开任意一个设备详细界面即可看到三元组的详细信息。复制后可以调用，实现互相连接。



Web 设备的三元组获取

3. AccessKey 的获取

AccessKey ID 与 AccessKeySecret 对应的是每个账号的接入接口，不随设备而改变。他们是访问阿里云 API 的密钥，具有该账户完全的权限，每一次使用都需要获取许可，需妥善保管。具体可以在任意界面进入个人信息 accesskeys，通过许可后即可查看。



4. iotid 的获取

4.1 认证设备

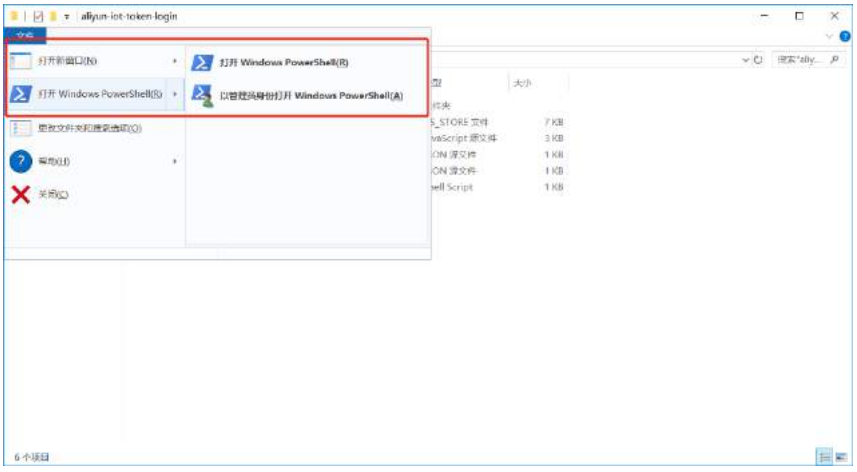
使用 HTTPS 进行设备认证，认证域名为：https://iot-auth.${YourRegionId}.aliyuncs.com/auth/devicename。其中，${YourRegionId} 替换为 Region ID。参考：**地域和可用区**。

4.2　项目代码包

download: aliyun-iot-token-login.zip

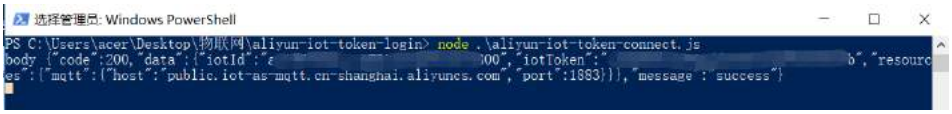
4.3　具体操作

解压压缩包，用管理员身份打开 powersell 界面。



(1) 替换 js 文件当中的 DeviceName 和 ProductKey 为待查找设备的三元组信息。

(2) 输入 `node .\aliyun-iot-token-connect.js` 运行。



4.4 参考

参考链接：https://www.yuque.com/cloud-dev/iot-tech/emv110

注：iotid 的获取是带有时间戳的，当运行设备过久时可能需要重新获取 iotid

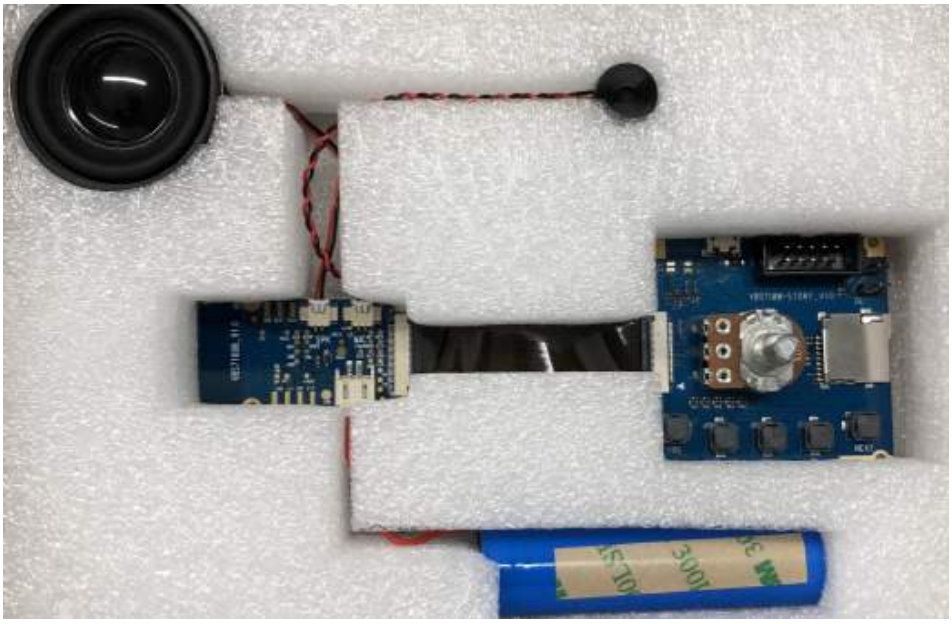
5. 消息通信 Topic

- 消息通信的 topic 一般是自己申请的地址，只需要到相应的界面就可复制使用。
- 应注意申请的是室友发布或订阅功能的还是兼有“发布和订阅”的 topic。

第三章　技术进阶——打造你的智能家居

基于 VBS7100B 的智能语音 LED 灯的开发案例

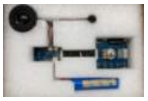
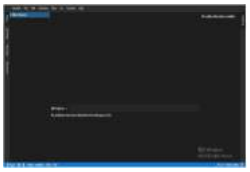
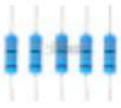
作者：mxchiplqh



概述

基于庆科信息 VBS7100B，利用 arduino 实现对 LED 灯的语音控制。当人说“开灯”时，LED 灯变亮；当人说“关灯”时，LED 灯熄灭。

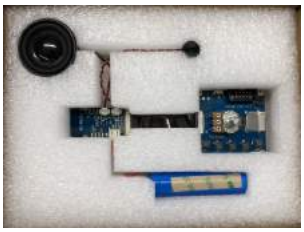
物料清单

硬件 (8)	软件 (2)	代码 (1)
庆科信息 VBS7100B 目前淘宝未上市 需要购买的请联系下方：联系人：刘涛 微信号：liutaoiot 电话号：15821321816 (× 1) 	ComTool 串口软件 	mxchip-VBS-liu
arduino (UNO) 开发板 (× 1) 	Hacklab 编程软件 	
USB 数据线 (× 1) 		
USB 转 TTL、USB 转串口 (× 1) 		
杜邦线 (× 9) 		
面包板 (× 1) 		
LED 二极管灯 (× 1) 		
电阻 (× 1) 		

方法 & 步骤

第 1 步　硬件准备

- 庆科信息 VBS7100B 智能语音套件：包括核心板、喇叭、麦克风、电池、扩展板



具体详解见：

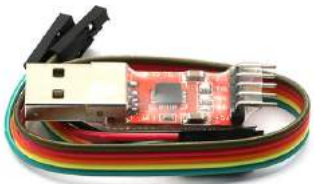
数据手册：http://developer.mxchip.com/at4/14

用户手册：http://developer.mxchip.com/at4/15

串口协议：http://developer.mxchip.com/at4/16

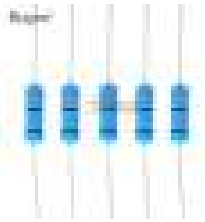
- arduino（UNO）开发板及其数据线

- USB 转 TTL、USB 转串口和三个杜邦线



- 杜邦线，两条公对公，三条公对母

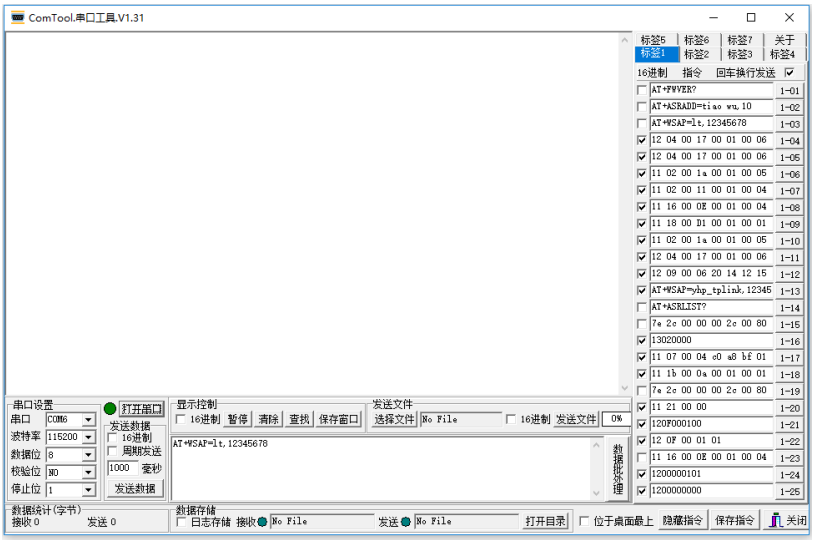
- 一个面包板，一个 LED 灯，一个电阻（100 欧到 1000 欧之间）



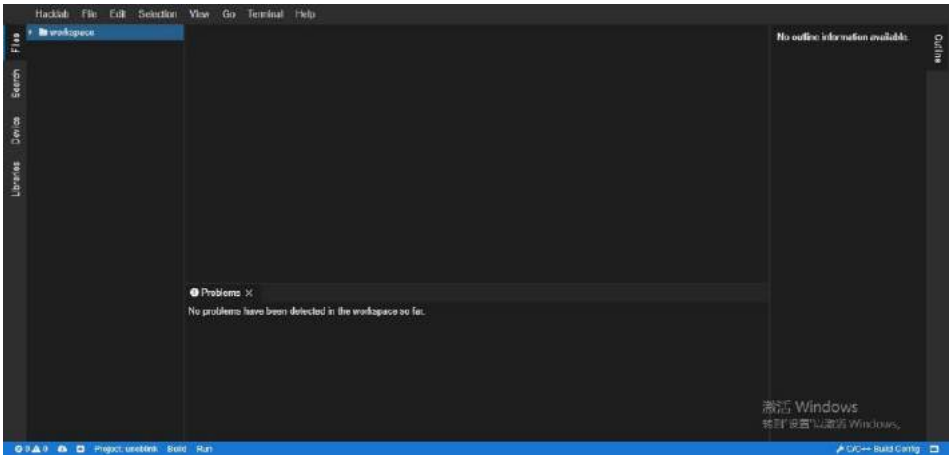
- 电脑一台

第 2 步　软件准备

- ComTool 串口软件一个（可以在网上下载）



- Hacklab 编程工具



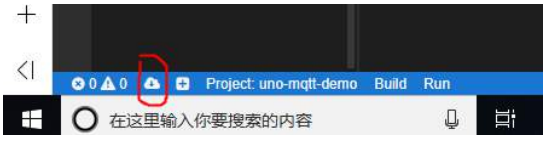
(1) 需要使用 Chrome 浏览器

(2) 登陆：https://hacklab.aliyun.com

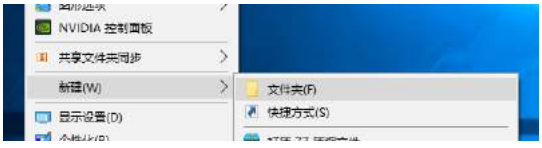
(3) 下载 Device Agent

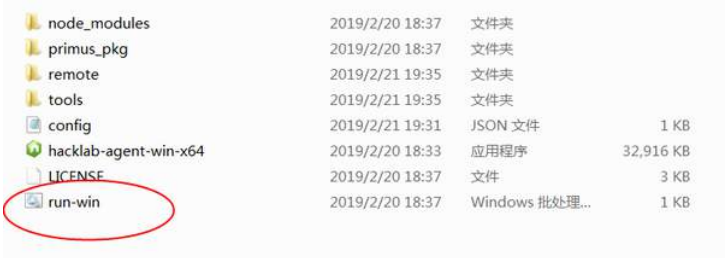
(4) 下面步骤有不懂的参考以下链接：

https://gaic.alicdn.com/doc/hacklab/duaucu.html

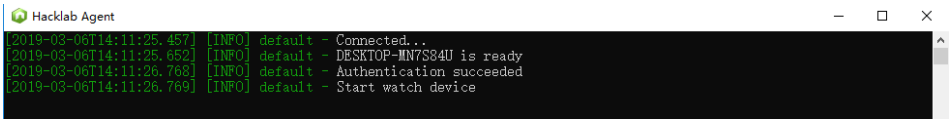


- 将 device-agent-win.zip 移至新建的文件夹（名字自取）中进行解压操作，解压后 Device-agent 目录如下（以 Windows 为例）

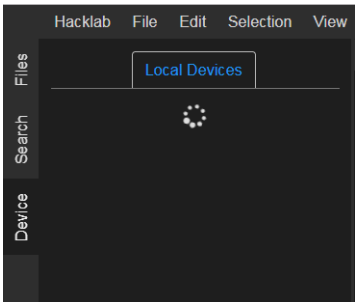




- 再次点击 run-win（自动运行 Device Agent，打开串口如下图）



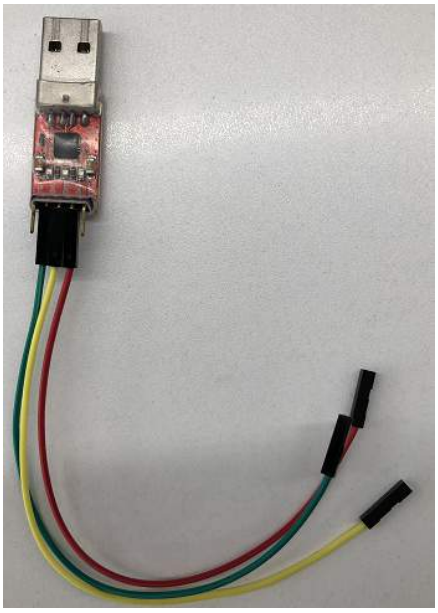
- IDE 中的 Device 设备管理 Tab 中的 Local Devices 本地设备管理的状态由搜索状态变为连接状态



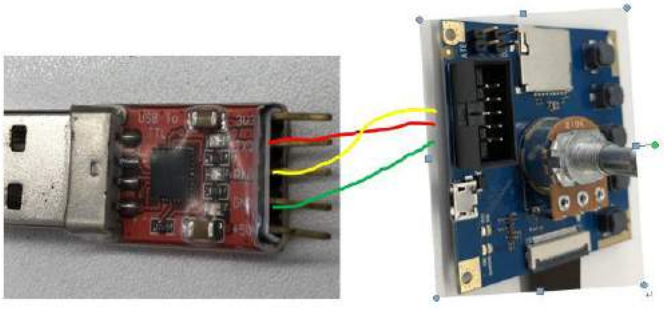
- 使用 USB 连接 UNO 和本地机，Device 设备管理显示 UNO 的信息，连接成功

第 3 步　配网操作

- 准备好 USB 转串口工具以及 3 条母对母杜邦线，如下图所示



- 用杜邦线连接 USB 转串口工具和语音开发板（如过 comtool 未出现数据，那么请看线的连接是否出错或者连接反了）
  - 绿色 GND 连接最右边的针
  - 黄色 RXD 连接中间的针
  - 红色 TXD 连接右边第二个针

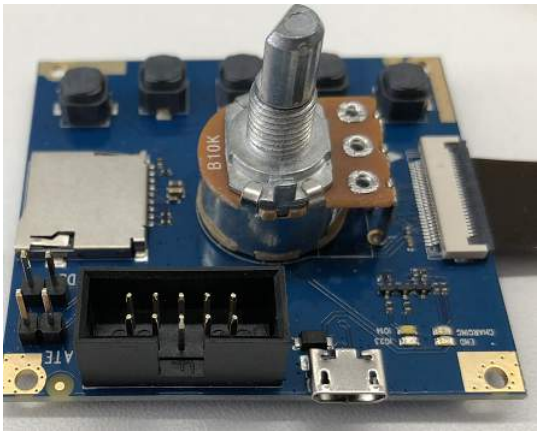


- 连接好的实物图如下所示

- 将串口调试工具插入电脑，在设备管理器中的端口确认端口

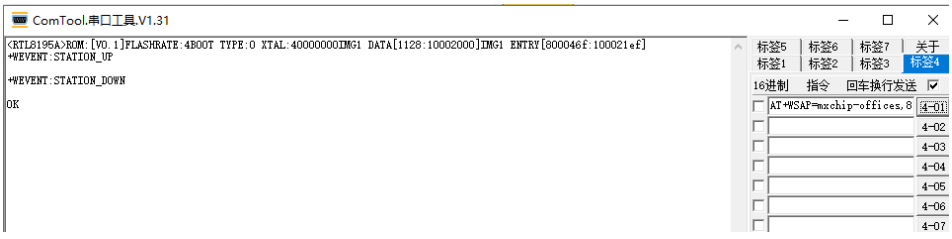
- 以管理员身份运行 comtool 串口调试软件，并进行串口设置，串口选择跟设备管理器中端口一致的端口数，波特率调为“115200”，数据位为“8”，校验位选“NO”，停止位为“1”，具体如下图所示

- 串口设置完毕后顺时针旋转语音开发板的电源开关按钮，如下图所示



- 在调试软件右侧，任意选择一个标签，并进行设置，配网时不要勾选 16 进制复选框，勾选回车换行发送右边复选框，然后编辑 AT 指令：AT+WSAP=wifi 用户名，密码（如：AT+WSAP=shanghai,123456），并点击右侧指令发送按钮即可完成配网

- 配网成功后软件界面显示如下图



- 定义命令：在开发相关产品前，要定义用语音控制哪些动作，比如本实验需要定义两个动作指令：开灯（指令为 14），关灯（指令为 15），具体定义如下

如上图所示，在红色圈里输入两个指令，分别是：AT+ASRADD=kai deng,14；AT+ASRADD=guan deng,15（kai deng;guan deng, 中间的空格很重要不能省略）；然后，分别点击 1-02 和 1-04 两个按钮，如果设置成功，左边会显示出 OK

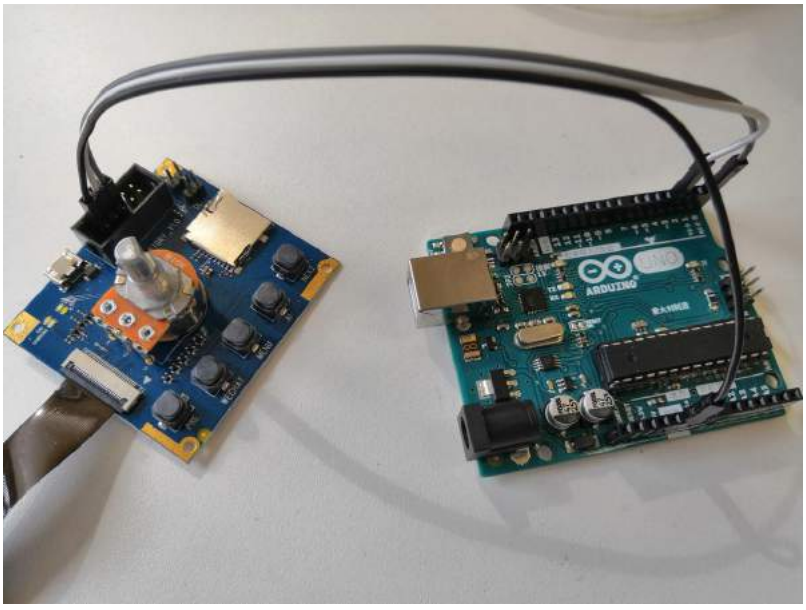
- 如果要检查是否配置好了命令，那么请输入命令 AT+ASRLIST?，返回图如下证明已经配置完毕

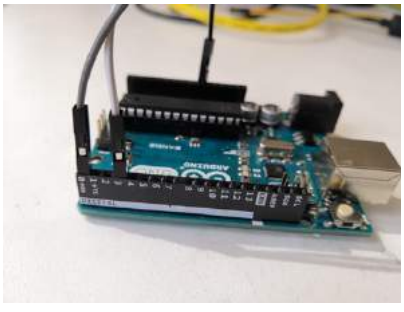
```
+WEVENT:STATION_UP
+ASRLIST:kai deng,guan deng,
OK
```

第 4 步　连线操作

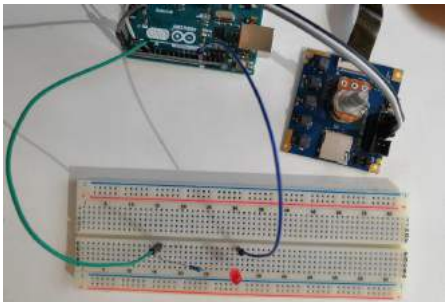
- VBS7100B 与 Arduino 连线

VBS7100B		Arduino	
GND	杜邦线连接	GND	
RX	杜邦线连接	RX	
TX	杜邦线连接	3	用软串口（暂定为引脚 3，也可以用其他引脚）从 Arduino 向 VBS7100B 写指令。





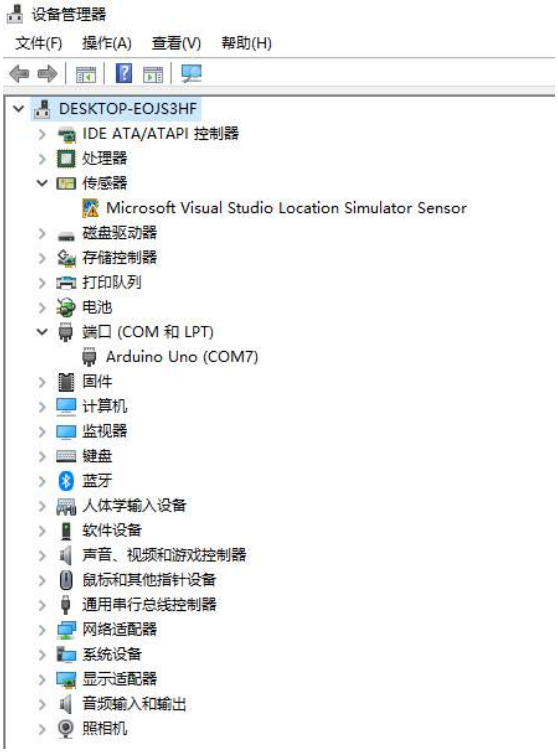
- Arduino 与 LED 灯连线



LED 灯长角的正极，短角的为负极。正极通过一个电阻和绿线进行连接，绿线连接到 Arduino 管角 5 上（其他角也可以，这里暂定角 5）；负极通过蓝色线连接到 Arduino 的管角上。

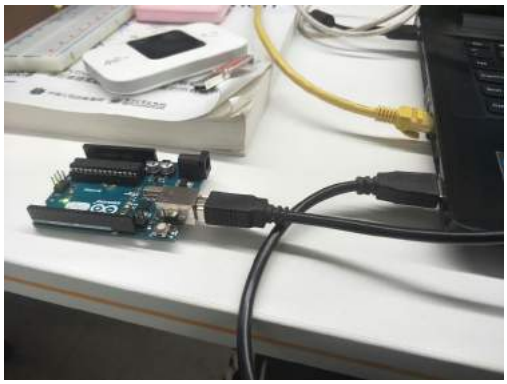
- Arduino 板与电脑连接

  Arduino 通过数据线与电脑进行连接

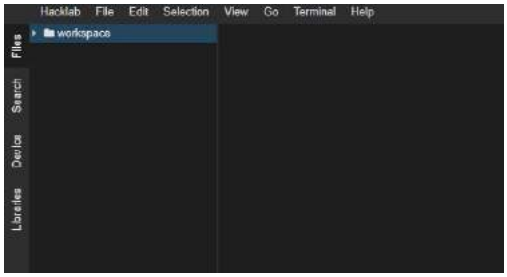


第 5 步　编译程序

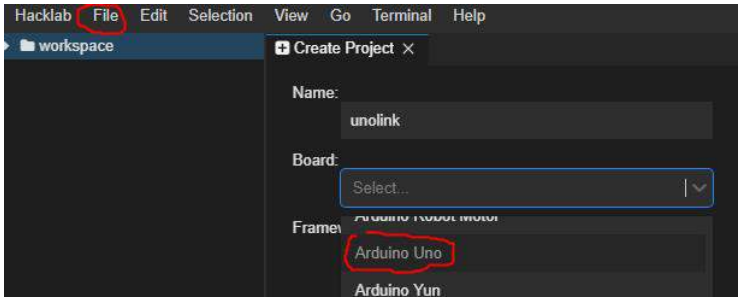
- USB 连接 Arduino Uno



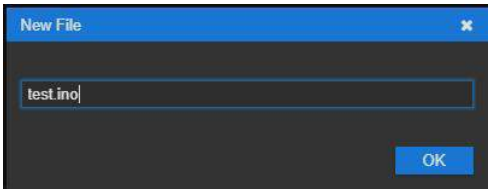
- 点击 File 再点 Create Project 创建新的项目（名字自定义）

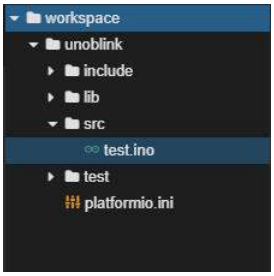


- 在 Board 选项中选择 Arduino Uno，点击 create 创建完成

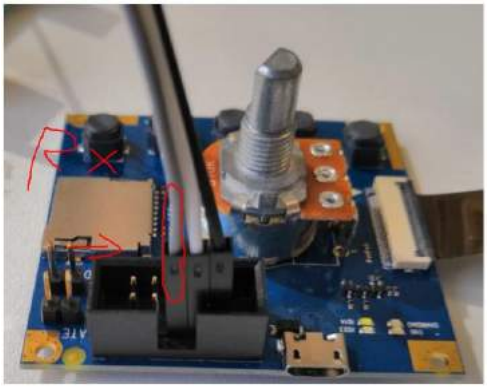


- 在 src 选项中选择新建文件（文件尾椎不一定是 .ino 文件，.cpp 文件也可以）





- 上传数据的时候记得把连接在庆科信息 VBS7100B（RX）上的杜邦线拔掉以后，再进行上传



- 点击设备开发工作台底部的 Build 按钮启动编译

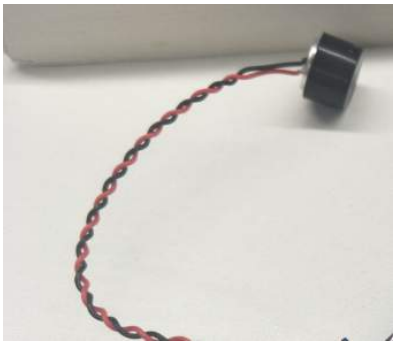

```
Compiling .pioenvs/uno/FrameworkArduino/wiring_pulse.c.o
Compiling .pioenvs/uno/FrameworkArduino/wiring_shift.c.o
Archiving .pioenvs/uno/libFrameworkArduino.a
Indexing .pioenvs/uno/libFrameworkArduino.a
Linking .pioenvs/uno/firmware.elf
Building .pioenvs/uno/firmware.hex
Checking size .pioenvs/uno/firmware.elf
Memory Usage -> http://bit.ly/pio-memory-usage
DATA:    [=         ]  22.9% (used 469 bytes from 2048 bytes)
PROGRAM: [=         ]  20.7% (used 6662 bytes from 32256 bytes)
========================================= [SUCCESS] Took 12.12 seconds =========================================
```

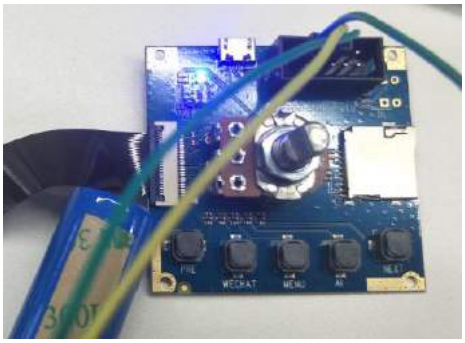
- 编译成功以后，点击 Run 按钮启动烧写

```
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.87s

avrdude: verifying ...
avrdude: 6662 bytes of flash verified

avrdude done.  Thank you.

[2019-03-11T16:26:40.037] [INFO] default - Remote process exited
========================================= [SUCCESS] Took 9.73 seconds =========================================
```

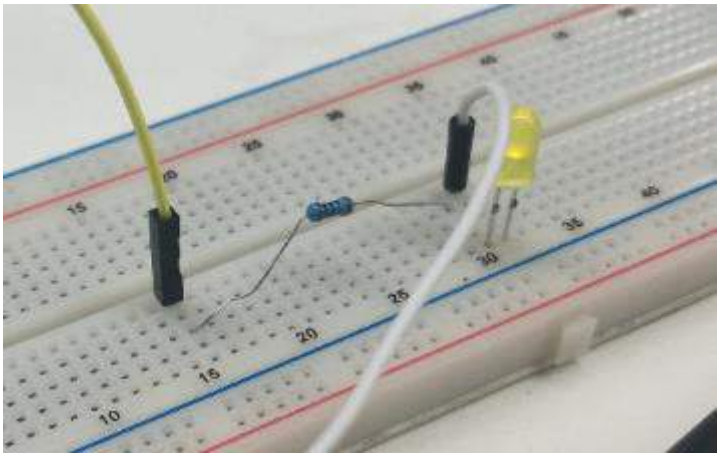
第 6 步　运行程序

- 说出命令：芝麻开门 指示灯如图所示

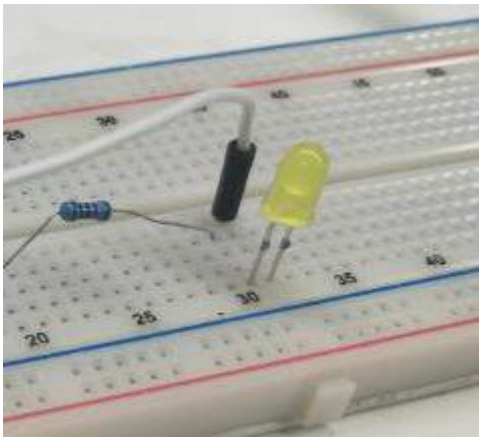




- 说出命令：开灯 LED 灯如图所示（系统提示音：正在为您打开灯）



- 说出命令：关灯 LED 灯如图所示（系统提示音：正在为您关灯）



使用 IoT Studio 开发你的智能家居控制台

作者：祥木

概述

这个教程里，我们将学习如何用 Web 可视化开发搭建一个 H5 家居应用控制面板。在这个文档里，我们将学习以下几个点：

1. 画布分辨率自定义；
2. 开关组件的图片样式运用；
3. 组件的复制粘贴功能；
4. 开关的数据源配置。

物料清单

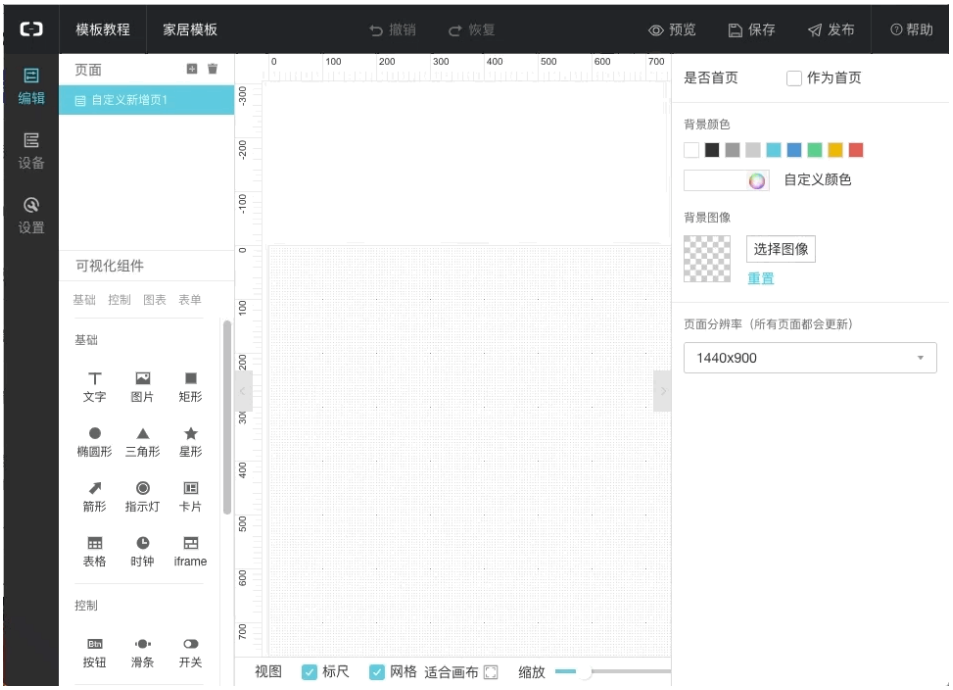
硬件 (1)
智能家电 (× 1)

最终效果预览→

方法 & 步骤

第 1 步　新建应用

在页面设置中，点击页面分辨率下拉框，选择自定义，在出来的选项中改画布分辨率为：375*667（iPhone8 尺寸）。

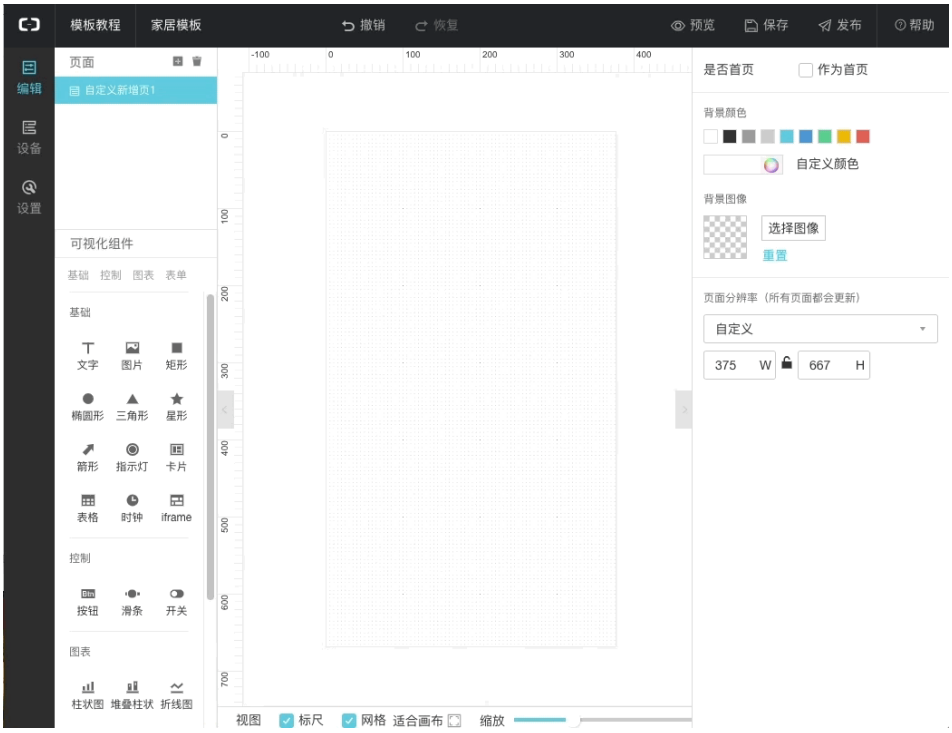


tips：

1. 鼠标点击画布任意非组件区域，右边操控面板都会变成当前页面的配置项。
2. 页面分辨率一旦调整，所有新建的页面画布都遵循该分辨率。
3. H5 分辨率参考：iPhone8 375*667*；*iPhone 8 Plus* 736414；iPhone XS 812*375*；*iPhone XR & iPhone XS Max* 896414；Android 640*360；
4. Web 可视化编辑器暂时不支持自动保存，切记随时 Ctrl+S 保存一下。

第 2 步　添加页面背景

找到页面设置项的背景图片，选择上传图片，在弹框中选择背景图片（需要自己上传）。



tips：

图片分辨率建议为画布分辨率的 **2 倍**，保证实际在手机上看的效果不会模糊。

第 3 步　布局标题和时间

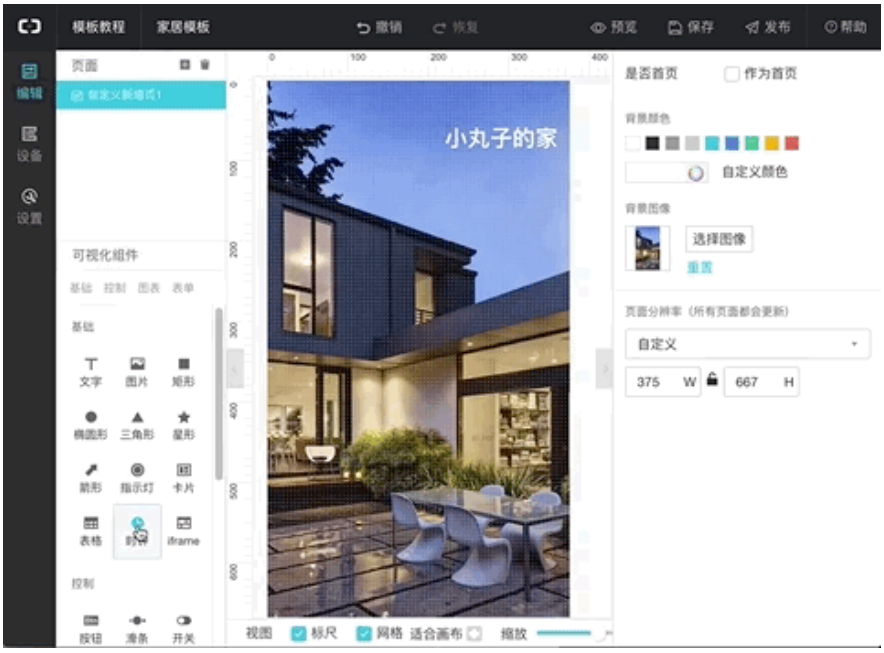
步骤 3–1　标题

拖拽左侧组件的文字到画布，在右侧操作栏中设置文字内容以及文字样式。最终调整到合理的位置。

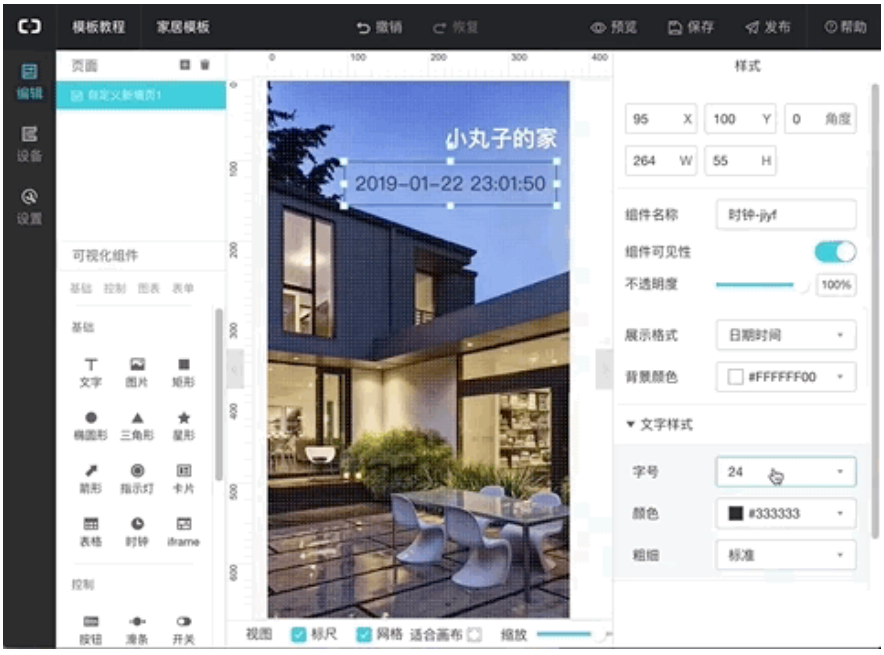
步骤 3–2　时间

拖拽时钟组件到画布，将展示格式设为日期时间，调整背景颜色不透明度为 0，调节文字字号和颜色，调整边框宽度为 0，最终拖拽组件到合适的位置。

时钟调整背景透明：



时钟调整文字及边框透明：



Tips：**

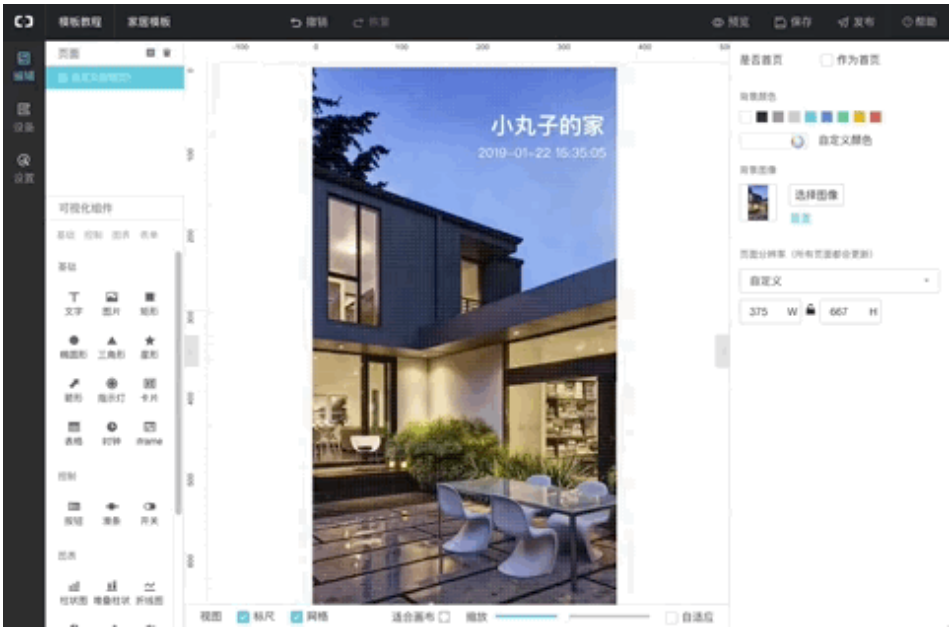
1. 时钟组件默认带背景和边框，如果想要去掉，可设置背景颜色不透明度为 0，边框粗细设置为 0。
2. 在当前版本中，组件尺寸及位置的步长为 5px，即尺寸需要为 5 的倍数，x 及 y 轴位置也需要是 5 的倍数，如果你设置的值不满足 5 的倍数，则系统会自动调节到 5 的倍数，请悉知。

第 4 步　增加灯泡开关

目前支持设备开关控制的组件为：开关。我们将利用开关组件的图片样式功能制作符合预期的圆角开关按钮。

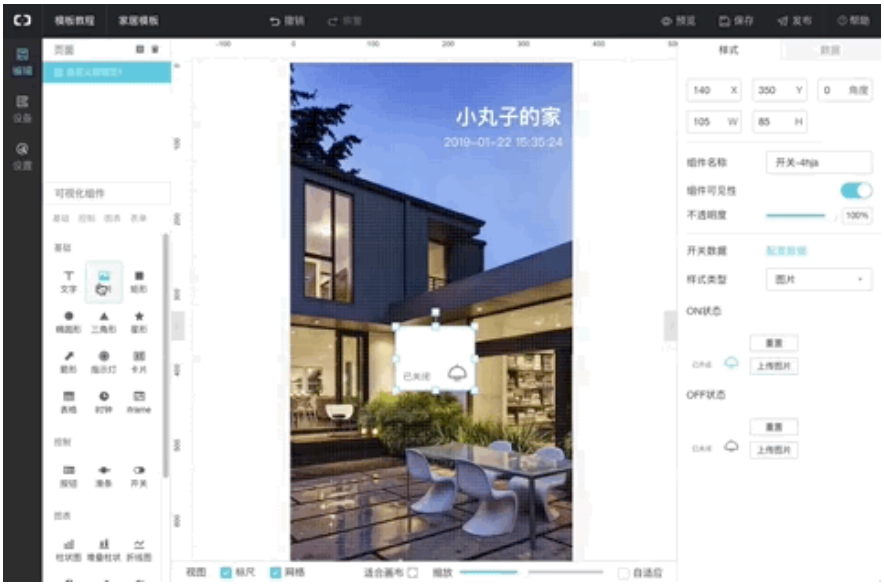
步骤 4–1

拖拽开关组件到画布上，将样式类型选择为图片，上传对应状态的图片。



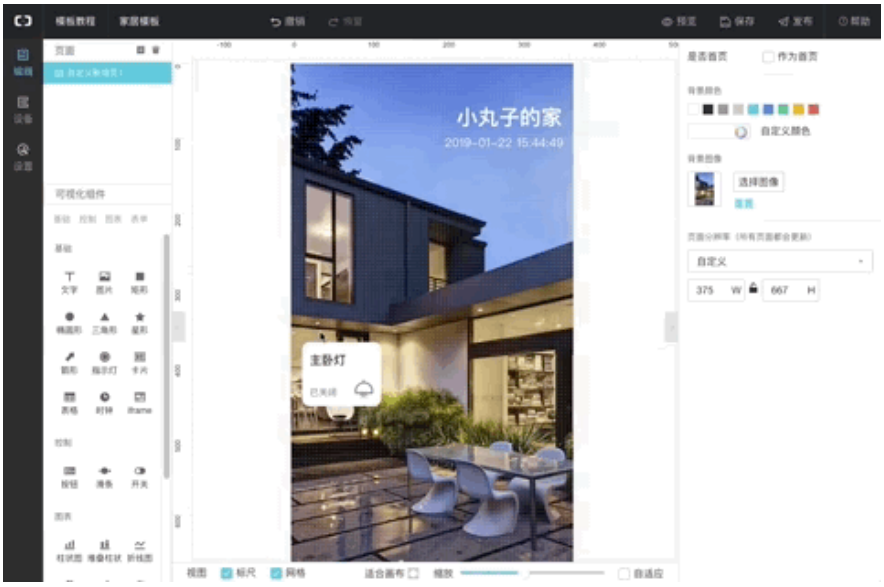
步骤 4–2

拖拽文字组件到开关图片上作为标题。



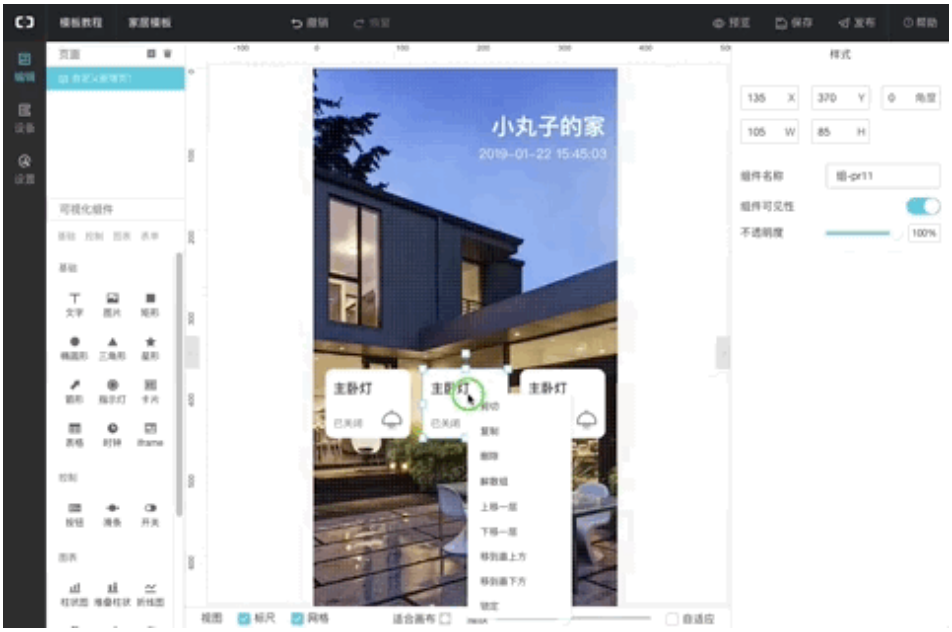
步骤 4–3

将鼠标拖拽选中标题和开关图片，右键，选择成组，然后通过快捷键 Ctrl + C，Ctrl + V 可复制多个相同的组件。



步骤 4–4

右键选择“解散组”，然后选中标题，可更改标题名称。

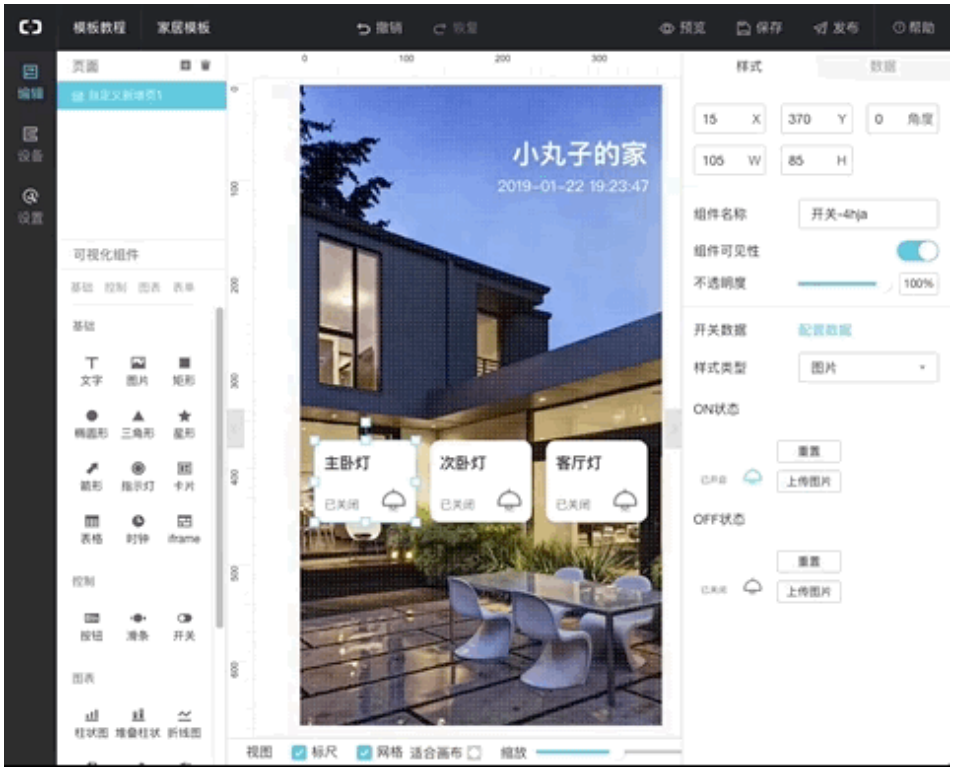


tips：

1. 开关组件支持默认样式和图片两种配置，选择图片可随心所欲配置想要的开关样式。
2. 开关组件默认为关闭状态，所以在编辑器里只能看到 off 状态的图，可以点击预览，在预览状态下点击开关组件，查看 on 状态是否符合预期。
3. 该案例中，标题部分单独出来加是考虑到灵活性，可通过更改标题来表达不同的房间控制。
4. 充分利用组件成组和复制功能，记住快捷键 Ctrl+C , Ctrl+V。
5. 成组状态下，是不能编辑组内内容的，所以要先解散组，再编辑。

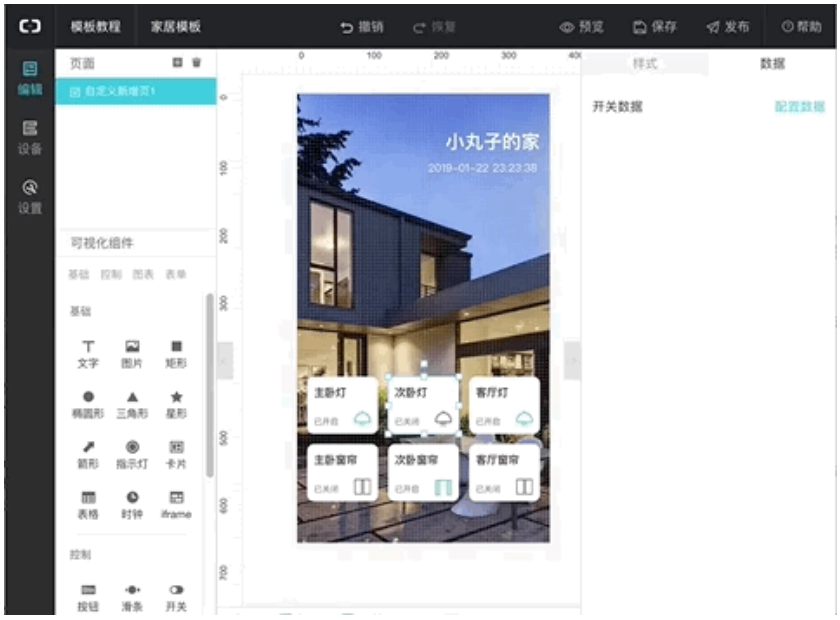
第 5 步　配置开关数据源

1. 点击已经配好的开关图片，选择右侧导航栏的数据面板，点击数据源配置。
2. 在数据源配置处选择已经建好的家居灯产品及设备（需要自己提前创建好含有布尔属性的灯产品及设备）。
3. 选择属性 – 主灯开关。
4. 点击右下角“确定”完成配置。

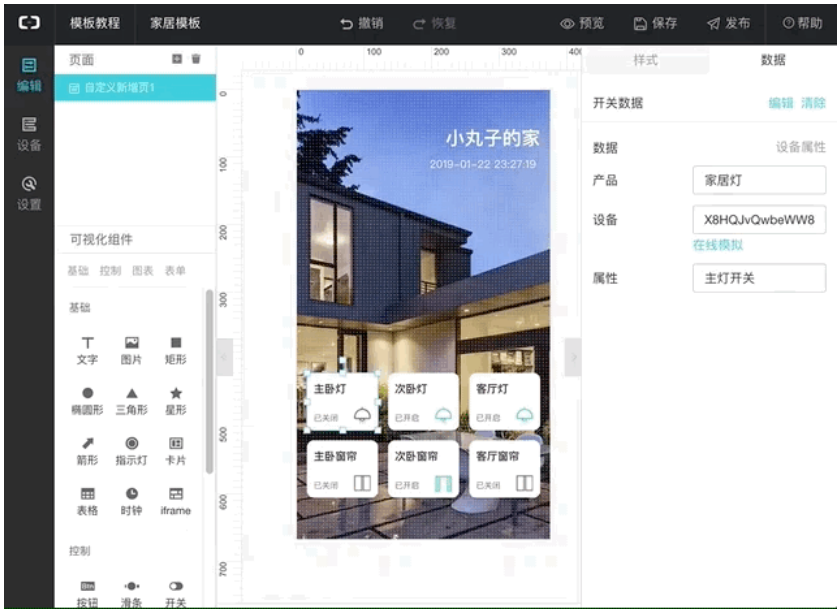


tips：

1. 设备选择可为“空”，即当前不配置任何具体设备，此时会出现 mock 数据的输入框选项，可通过 mock 数据方式来模拟设备运行。



2. 当选择设备后，如果没有真实设备，则需要通过“在线模拟”让设备上线。你可以通过“在线模拟”推送设备属性或事件消息，在编辑器中可直接看到设备状态引起的变化。



关于所有数据源的详细配置文档可点击此处查看。

第 6 步　增加窗帘开关

同 Step4-5 制作即可，最终效果：

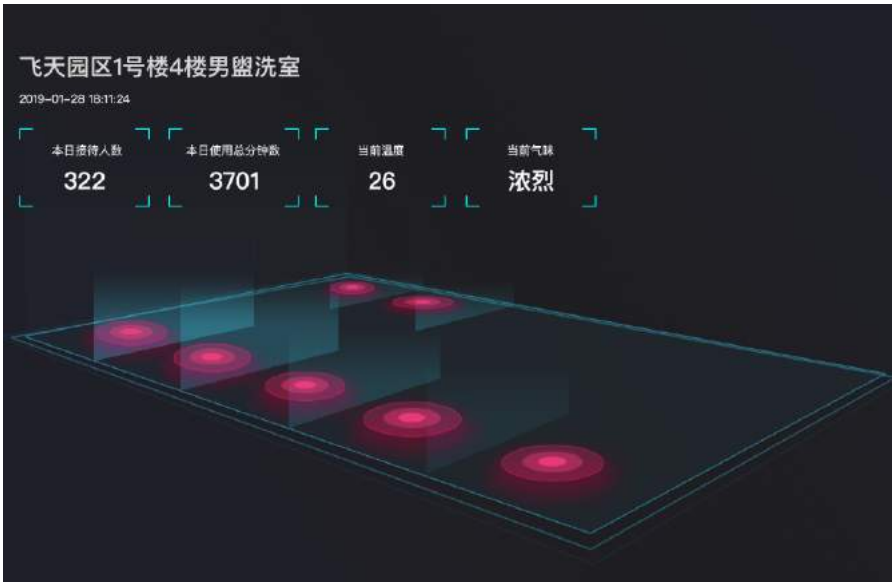
小结

IoT Studio 的详细使用方法可以参见帮助文档：

https://linkdevelop.aliyun.com/studioweb-doc#index.html

IoT Studio+LoRa 打造“又猛又持久”的智能厕所

作者：貔阁



概述

为了增加厕所使用效率，减少被味道“熏陶”的等待时间，同时也为了增加厕所的清洁效率，我们决定做一个非侵入式的智能厕所改造方案。它可以通过红外热释电检测每个坑位有没有人，在 web/app 上实时显示，方便如厕人员查询。并且可以检测厕所的臭味，达到阈值时通知清洁工进行清扫。

之前的文章里，我们使用了 Link Develop 平台 + arduino 搭建了一个基于 wifi 连接的智能厕所 demo。实际生产过程中，NodeMCU+PIR 有耗电比较高，结构不稳定，没有外壳防水等问题。因此如果需要商用化智能厕所的方案，需要一个更稳定的可靠的办法。

目前 Link Develop 已经升级成为 IoT Studio，我们将会用更强大，更方便的开发能力实现 LoRa 传感器数据上传 + 存储，以及对应应用的开发等。

物料清单

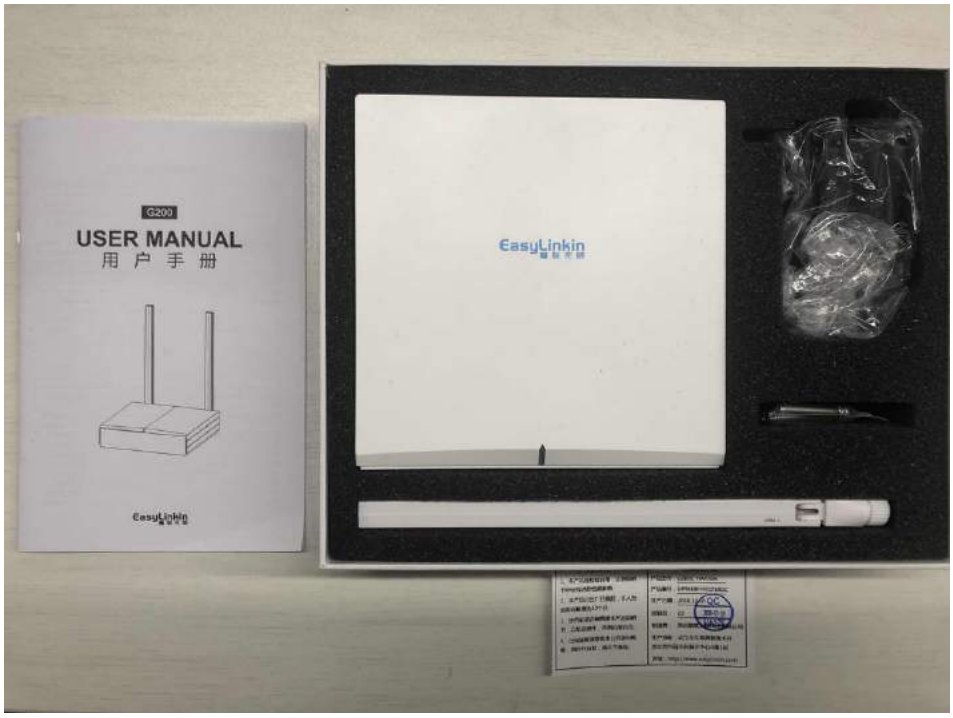
硬件 (3)
慧联无限 G200 LoRaWA (× 1)
慧联无限 LoRa 红外传感器 (× 1)
一台能联网的电脑 (× 1)

方法 & 步骤

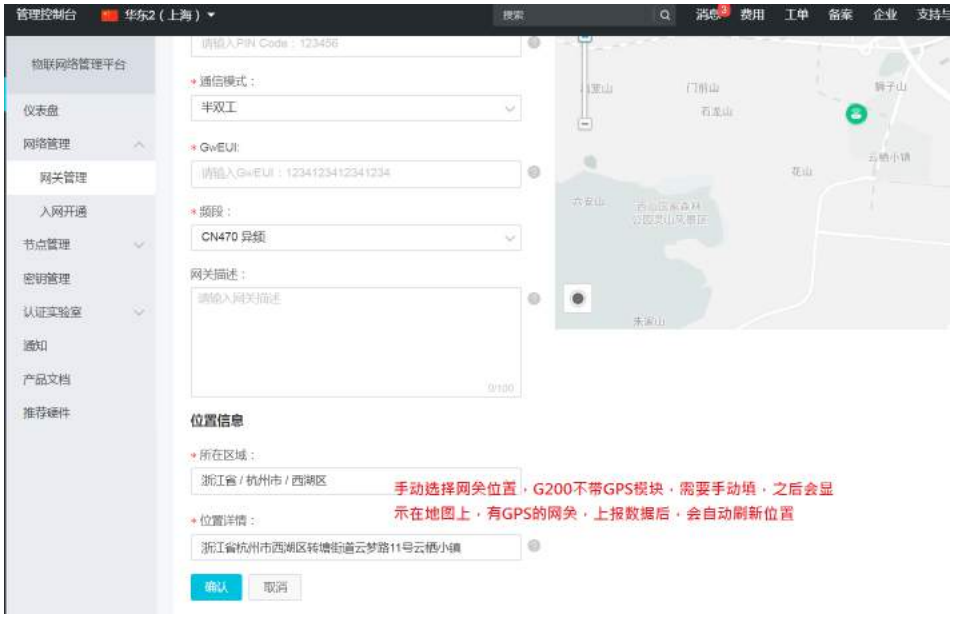
第 1 步　配置 LoRa 网关组网

前往 LinkWAN 进行网关配置。

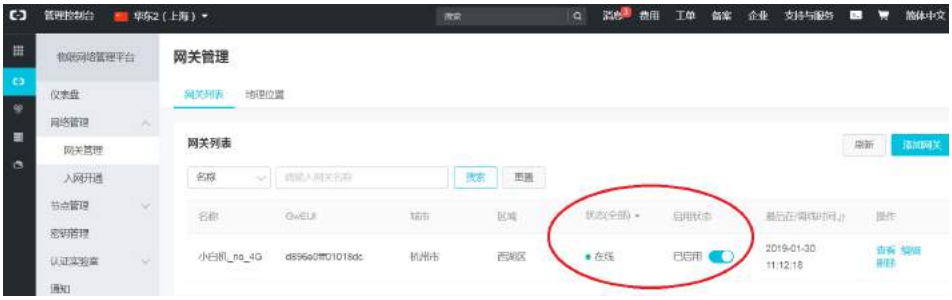
关于 LoRa 以及 LoRaWAN 的介绍可以查看这篇文档。如果有需要帮助请私信作者。我们可以选择从清单上 :https://linkmarket.aliyun.com/wan 购买网关自己组网络，或是也可以问问其他阿里云用户，透过分享入网凭证给我们，用上他的网络。这次我们购买慧联的室内小网关 G200，自己组建网络。



可以看的到，我们需要插上网线与电源。然后到 Link WAN 控制台注册网关



G200 回传网默认是 DHCP 上网配置，如果你是固定 IP 或 PPPoE 在依照实际环境调整，方法可以参考 G200 网关手册，如果一切顺利可以再 Link WAN 网关页面显示在线。



这样就是组好网络了。

第 2 步　分配网络给自己用

网络组好之后，我们要把网络分配给自己用，也可以分配其他人使用，当然被分配的人都必须有阿里云账号

PS. 阿里云园区都有 LoRaWAN 网络，你可以联系阿里云 IoT 的同学，分享给你用，这样就不用自己买网关组网了。分配的方式是透过入网凭证来取得网络使用权利。

- 首先我们来创建自己的网络凭证

- 我们自己要用网，所以就授权给自己

- 成功之后，就会在凭证清单看到了，当然如果是别人授权给你的，也会在此清单看到

第 3 步　拆封 LoRa 硬件

从网上购买来的认证过的 LoRa 传感器，拆出来的时候可以看到后面附了一

个 16 位码的贴纸。这个是节点的 DevEUI，在后面的新版硬件中还会加上 6 位的 PINCODE 字段。

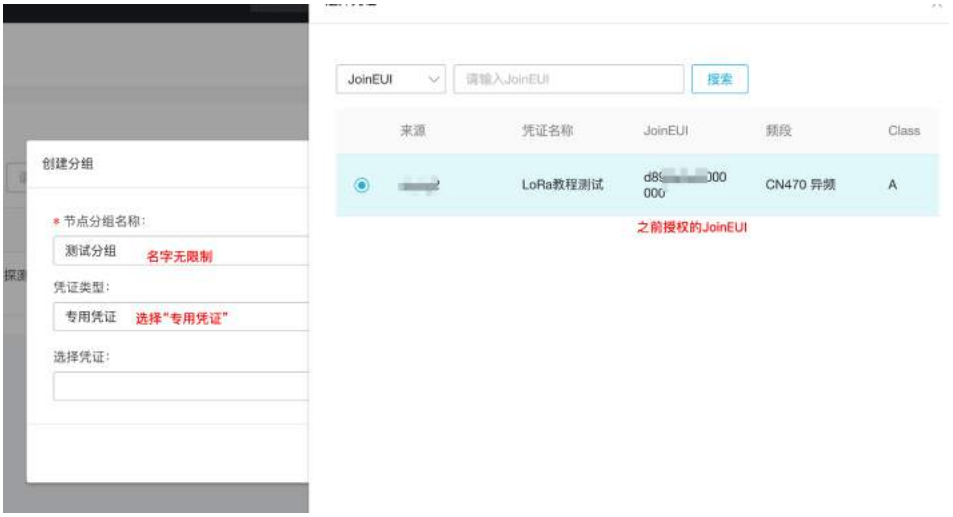
按照说明书，默认是 5 分钟模式——即收到一次警报之后 5 分钟之内不会再上传。这跟厕所的使用场景不一致，我们需要调成测试模式——即每次收到警报都会上传。把背部的壳子掰开（不需要螺丝刀）之后，调整跳针针帽位置到 ON。

然后合上盖子，先不拔掉塑料片。我们需要先在 LinkWAN 配置 LoRa 设备信息。

第 4 步　LoRa 节点上云

在步骤一中我们已经把 LoRaWAN 网关配置完成，获得了入网凭证，接下来我们要用这个凭证来使用网络进入节点分组管理，点击创建分组。

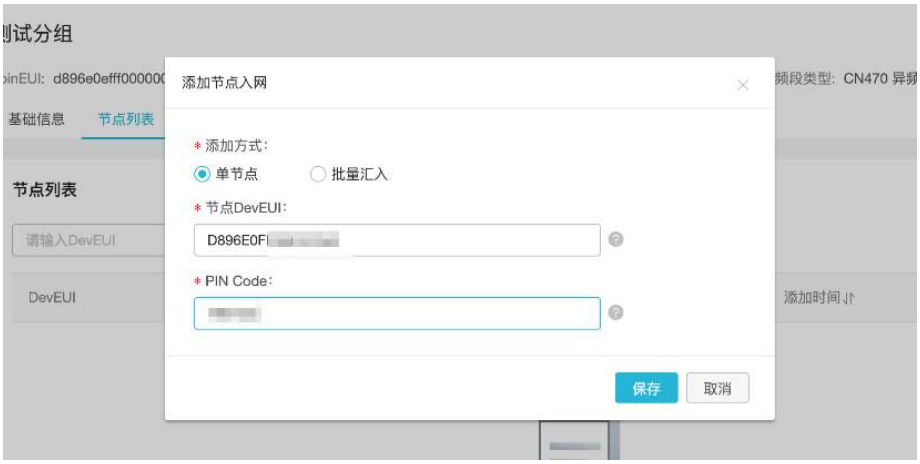
需要点击“专用凭证”，然后选择之前“入网开通”申请的凭证。



点击完成，即可看到新建的分组，然后点击节点进入管理。

选择添加节点。

然后在弹窗内输入传感器节点上贴纸的信息。（注：作者的传感器为早期版本，只有 DevEUI，PINCODE 是自己拿的。目前市面的传感器都有贴上 DevEUI 和 PINCODE 信息，如果没有请联系供应商。）



添加成功。这样就成功把一个传感器 ID 授权加入 LinkWAN 上了。接下来需要前往物联网平台进行节点上云。

第 5 步　在物联网平台配置

首先进入物联网平台 Link Platform 的产品页面。选择新建产品。产品相当于一个类（class），是某一类设备的集合，该类设备具有相同的功能，您可以根据产品批量管理对应设备。

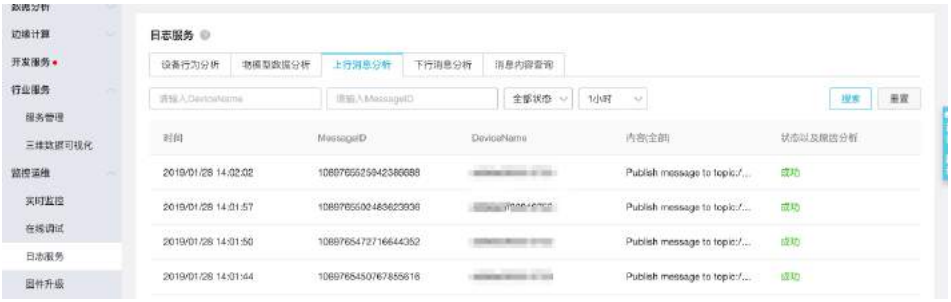
新建产品时，要选择透传。

建立了产品之后，点击“设备管理”前往设备管理页面。然后选择“添加设备”。输入 16 位 DevEUI 作为 DeviceName（注意要小写）。然后确定。

接下来返回到 LinkWAN 平台。为刚才的节点分组添加“数据流转”，选择刚才添加的产品。

这时候可以拔掉塑料片让传感器上电了。可以看到 LinkWAN 有收到上行日志。

同时物联网平台的设备也收到了日志。



接下来就是解析设备上传的信息了。从传感器的说明文件可以得知，传感器上报的是二进制数据。我们如何把二进制数据转化为可以理解的属性名称呢？下图是厂家提供的传感器的二进制配置文件。

下图是原始的上行日志。

对这个传感器，“020100”中的第一个 BYTE“02”表示协议，“01”对红外传感器表示“有人”，“00”表示传感器状态正常。我们首先需要在产品里定义“有人经过”和“传感器状态”两个功能，用于记录这两个属性。前往产品详情页的“功能定义”tab，点击右下方“自定义功能”的“添加功能”。

然后在弹窗内配置属性，“室内人体探测开关”配置为布尔型，“传感器属性”配置为枚举型。注意要是“读写型”。点击确定添加属性。

最后结果如图。

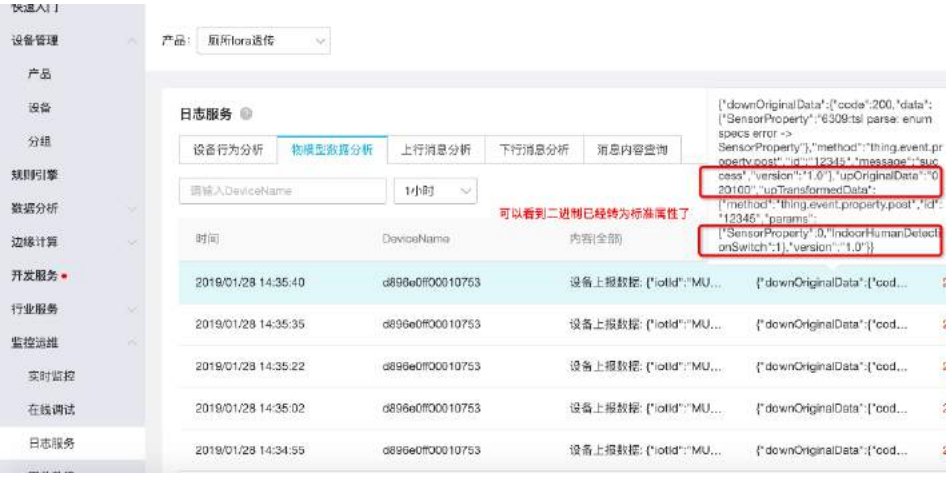
然后我们需要使用产品定义里的数据解析，把二进制数据自动转化为 Alink-JSON 格式，以对应上刚才设置的“室内人体探测开关”与“传感器属性”。转化规则可以参考文档。我们这里只放出最后的代码。注意：数据解析需要产品为“未发布”状态。如果已经发布请点击右上角“撤回发布”。

输入如下的解析脚本。

```
var COMMAND_REPORT = 02;
var COMMAND_SET = 01;
var ALINK_PROP_REPORT_METHOD = 'thing.event.property.post'; // 标准 ALink JSON 格式 topic，设备 上传属性数据到 云端
var ALINK_PROP_SET_METHOD = 'thing.service.property.set';
function rawDataToProtocol(bytes) {
    var uint8Array = new Uint8Array(bytes.length);
    for (var i = 0; i < bytes.length; i++) {
        uint8Array[i] = bytes[i] & 0xff;
    }
    var dataView = new DataView(uint8Array.buffer, 0);
    var jsonMap = new Object();
    var fHead = uint8Array[0]; // 第 0 个 BYTE 为上报协议
    if (fHead == COMMAND_REPORT) {
        jsonMap['method'] = ALINK_PROP_REPORT_METHOD; //ALink JSON 格式 - 属性上报 topic
        jsonMap['version'] = '1.0'; //ALink JSON 格式 - 协议版本号固定字段
        jsonMap['id'] = '' + 12345; //ALink JSON 格式 - 标示该次请求 id 值
        var params = {};
        params['IndoorHumanDetectionSwitch'] = uint8Array[1]; // 第 1 个 BYTE 为传感器读数判断有没有人
        params['SensorProperty'] = uint8Array[2]; // 第 2 个 BYTE 为传感器本身的状态，对应产品属性中 prop_float
        jsonMap['params'] = params; //ALink JSON 格式 - params 标准字段
    }
    return jsonMap;
}
```

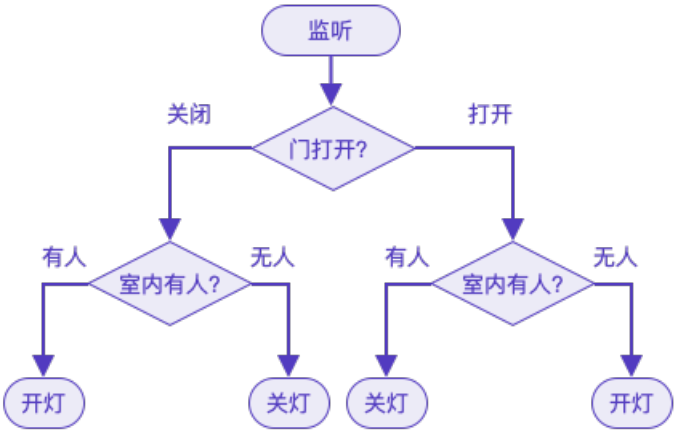
然后点击“保存草稿”。之后输入我们的原始数据“020100”进行调试。可以看到右边解析成功了。这时候点击“运行”即可让脚本生效。

在日志里可以看到二进制的 020100 已经转为 {“SensorProperty”:0,”IndoorHumanDetectionSwitch”:1} 了。这样就完成了设备接入。



用阿里云物联网服务开创你的智能家居联动

作者：GXIC



概述

有一个智能门，可以根据门磁检测门是打开还是关闭，并且屋内一侧有 PIR 传感器可以检测室内是否有人。我们需要实现以下的逻辑：持续监听智能门的上报。如果监听到智能门打开，判断室内是否有人——如果此时室内有人，判定用户要出门离开，发出指令关闭智能灯；如果此时室内没人，判定用户回家，发出指令打开智能灯。如果此时门关闭的时候保持原有的灯开关状态。

物料清单

硬件 (3)
PIR 热释电红外传感器 (× 1)
门磁传感器 (× 1)
智能灯泡 (× 1)

方法 & 步骤

第 1 步　开通阿里云物联网服务

开通 阿里云物联网平台服务 ，进入开发服务：

第 2 步　创建智能门产品

1. 创建一个智能门产品并添加 2 个布尔型功能：

2. 为智能门产品创建一个设备：

第 3 步　创建智能灯产品

创建一个智能灯产品并添加 1 个布尔型功能：

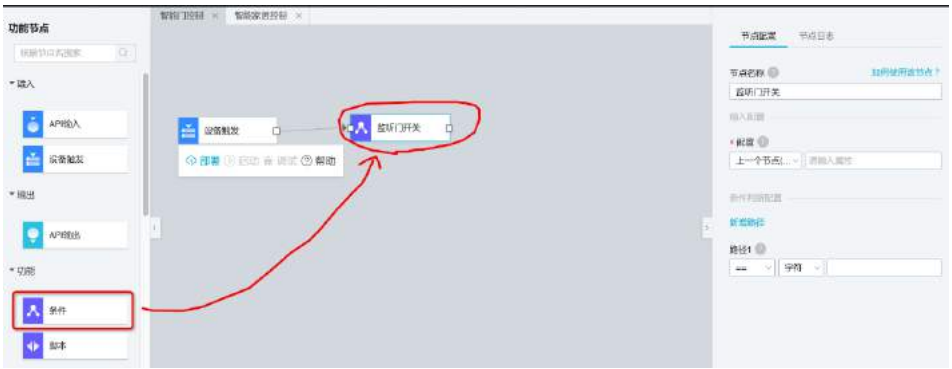
第 4 步　创建服务

由于本服务配置项较多，因此请在合适的时候点击顶部“服务”下拉菜单选择“保存”或点击快捷键 Ctrl/Cmd + S 保存您的编辑操作。平台也会每分钟为您自动保存一次。

1. 新建一个服务，命名为“智能家居控制”。

2. 在左侧栏节点列表中“输入”类选择“设备触发”节点，右侧栏配置项分别选择“智能门”产品，创建的设备名以及选择监听“属性上报”。

3. 添加一个“条件”节点，条件节点相当于一个 if-else 判断。并且把设备触发节点与条件节点连接起来。



4. 修改右侧栏配置项——把这个条件节点改名字为“监听门开关”；把输入设置为“上一个节点”+“props.DoorOpen.value”；增加一条路径，把路径1的第二个选择框设置为“数字”，第三个框输入1（表示没人），把路径2的第二个选择框设置为“数字”，第三个框输入0（表示有人）。下面将介绍这一步的配置项的原理：

5. 同上原理所述，把业务逻辑分支也用条件节点搭建出来。

6. 最后加上设备节点。设备节点是以产品 / 设备维度去查询设备信息 / 下发指令的功能节点。每个项目内创建 / 导入的产品对应一个设备节点。我们之前添加了智能灯，将其拖到画布内。然后选择在步骤三中创建的智能灯设备，第二个框选择设备指令下发的操作类型，最后参数框内把“properties”默认项添加智能灯的开关属性标识符“LightOn”并在后面附数值 1（代表开灯）。这样就完成了一个路径的设备节点配置了。

7. 最终成果如图：

小结

传统的智能家居设备联动需要编写复杂的服务端代码甚至涉及到设备端代码的改动，现在利用 IoT Studio 的开发服务，只需鼠标拖拽即可轻松实现复杂的联动逻辑。

第四章　高阶实战——不止停留在 Demo

使用 IoT Studio 搭建农业监控大屏

作者：祥木

概述

在这个教程里，我们将学习如何用 Web 可视化开发搭建一个农业大屏。在这个面板里，我们将学习以下几个点：

1. 画布的自适应
2. 视频网页的嵌入

3. 设备数据的展示
4. 设备统计的展示
5. 使用形状进行页面布局

最终效果预览：

物料清单

硬件 (2)
农业大棚 (× 1)
农业环境监测传感器 (× 1)

方法 & 步骤

第 1 步　新建应用

页面设置中，点击页面分辨率下拉框，选择 1920*1080 (常见宽屏比例)。

在底部工具栏，选中自适应，这样在预览和发布的应用中，就可以自适应屏幕大小 (等比缩放，宽度撑满为止)。

tips：

1. 鼠标点击画布任意非组件区域，即可做页面配置
2. 页面分辨率一旦调整，所有新建的页面画布都遵循该分辨率
3. Web 可视化编辑器暂时不支持自动保存，切记随时 Ctrl+S 保存一下

第 2 步　添加页面背景色

找到页面设置项的背景颜色，使用自定义颜色功能，输入颜色值。

第 3 步　布局

步骤 3–1　使用矩形作为区块分隔

- 拖拽矩形组件到画布中。
- 设置背景色为 #FFF，边框粗细为 0。
- 拖动改变组件大小，满足一个区块要求。

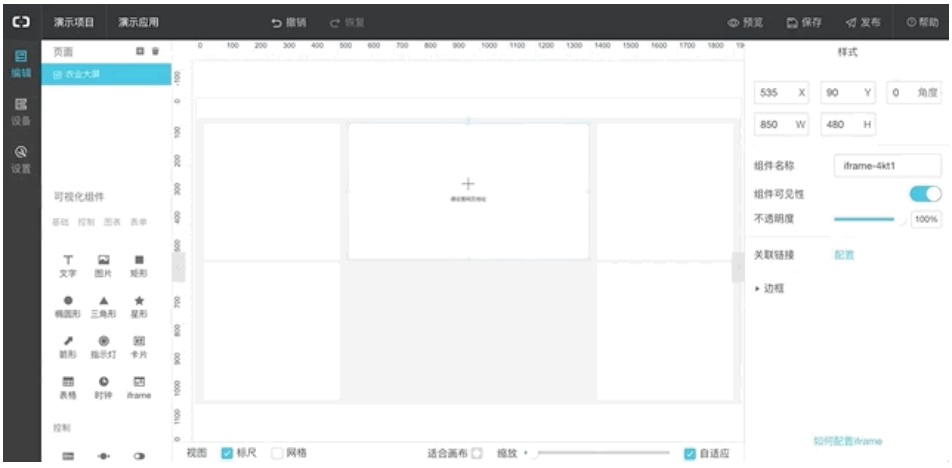
步骤 3–2　完成其他区块

复制、黏贴，并完成其余区块的大小、位置调整。

拖拽 iframe/ 图片组件到画布中，并调整大小和位置。

步骤 3–3　精确调整位置和大小

可以用右侧面板的输入框来完成最后的微调。



第 4 步　配置各组件

步骤 4–1　添加文本

拖拽文字组件到画布，在右侧操作栏中设置文字内容以及文字样式。最终调整到合理的位置。

步骤 4–2　添加仪表盘

- 拖拽文字组件到画布，在右侧操作栏中点击配置数据。
- 选择设备数据源，逐项选择产品、设备、属性即可。
- 若实体设备未到位：
  - 可以不选择设备，使用模拟数据来预览调试，应用发布以后再绑定真实设备。
  - 也可以通过“在线模拟”入口，去产品管理页面创建模拟设备。

步骤 4–3　设置 iframe 组件

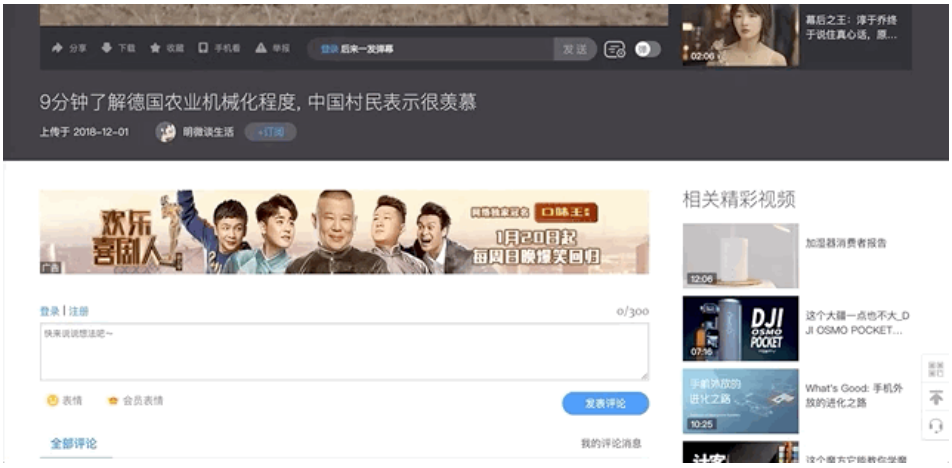
选中 iframe 组件，在关联链接处，点击配置，填入 URL 即可，推荐使用 HTTPS 的 URL。

tips：

由于 HTTPS 网站内是不允许内嵌 HTTP 的地址，因此在编辑器中无法实时预览，可以通过预览页面（需要手动改为 HTTP 协议）来查看界面。

tips：

如果像嵌入视频网站的视频，必须使用分享地址中的 URL，否则视频网站会禁止被嵌入。



步骤 4–4　添加时钟组件

拖拽时钟组件，配置时钟的样式。

步骤 4–5　配置图片

选中图片组件，在右侧配置图片。

步骤 4–6　为文字组件绑定设备数据

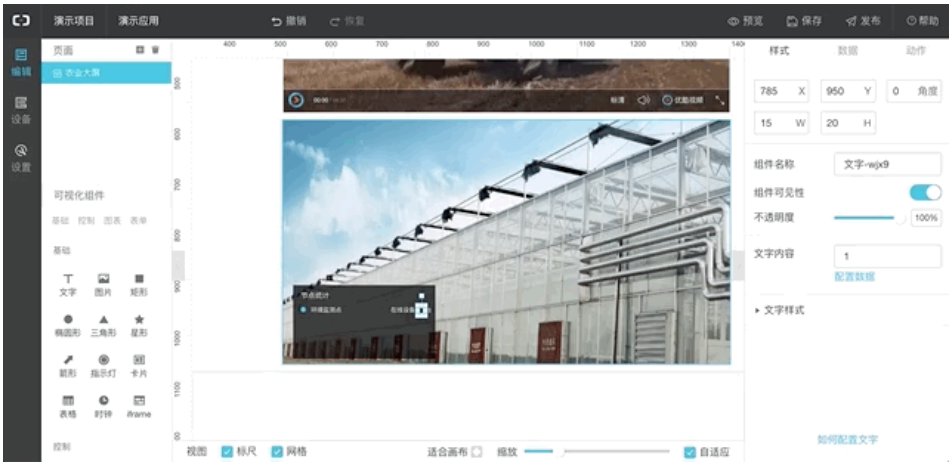
拖拽一个文字组件，选中组件，配置数据项，选择设备数属性作为数据源。

步骤 4–7　为文字组件绑定产品统计数据

拖拽一个文字组件，选中组件，配置数据项，选择接口 –》产品与设备信息 –》获取物的数量，即可动态显示设备数量

tips：

- 如动图所示，描述文字可以放在动态文本的左右，通过左右对齐完成样式调整。
- 获取物的数量接口，可以支持多种筛选参数，如总数、激活数、在线数等等，详见接口文档。



步骤 4–8　添加实时曲线图

拖拽曲线图组件，配置设备数据，选产品 –》设备 –》设备历史属性，完成配置后，即可看到设备属性的实时曲线。

tips：

若使用的是虚拟设备，则可能因为历史数据太少，曲线图为空的情况，只需要多做几次模拟设备数据下发。

小结

IoT Studio 的详细使用方法可以参见帮助文档：

https://linkdevelop.aliyun.com/studioweb-doc#index.html

IoT SaaS 加速器——助力阿尔茨海默病人护理

作者：貔阁

简介：用技术解决阿尔茨海默病护理的问题，让老人和其护理者有更好的生活质量，是技术可以解决的。基于物联网技术，已经有一些设备实现了阿尔茨海默病老人走失定位。但是我们要做更高一层，除了单独分发的硬件之外，我们要使用开发工具 IoT Studio 帮助医疗机构做一个硬件 SaaS 管理系统，让他们可以随时监控旗下所有阿尔兹海默护理设备的数据以及定位，对老人的情况实现实时监控。

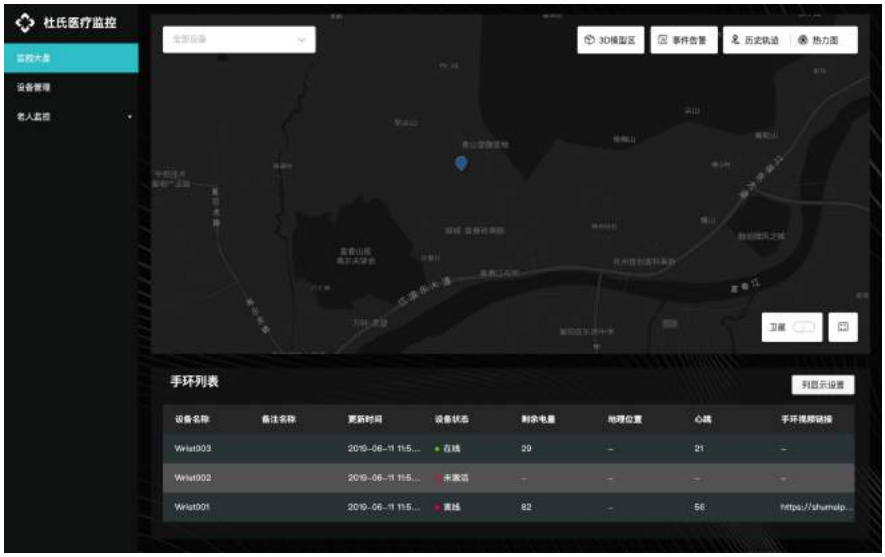
场景介绍

阿尔茨海默病，是导致中老年人认知功能障碍的最常见疾病之一，是发生在老年期及老年前期的一种原发性退行性脑病。据估计，全世界痴呆症患者数量为 4700 万，到 2030 年将达到 7500 万人。痴呆症患者数量到 2050 年预计将是现在的近三倍。疾病的高昂费用给卫生系统应对未来预计不断增加的病例构成挑战。据估计，目前每年的支出为 8180 亿美元，而支出的增长速度预计会比疾病流行率上升还要快。照料痴呆症患者给照护者带来巨大压力，包括身体上、情感上和经济上的压力。（by 世界卫生组织）

用技术解决阿尔茨海默病护理的问题，让老人和其护理者有更好的生活质量，是我们可以解决的方法。基于物联网技术，已经有一些设备实现了阿尔茨海默病老人走失定位。但是我们要做更高一层，除了单独分发的硬件之外，我们要使用开发工具 IoT Studio 帮助医疗机构做一个硬件 SaaS 管理系统，让他们可以随时监控旗下所有阿尔兹海默护理设备的数据以及定位，对老人的情况实现实时监控。同时也有能力对掌控的设备进行增删改查，方便他们自己管理设备。通过 IoT Studio 赋能开发者，让他们帮助包括医疗在内的各个行业用上物联网技术，惠及百姓。

我们首先构建一个可以拍照，检测心跳的手环设备，然后基于这个设备帮助护理机构开发一个集合管理监控告警的 SaaS 系统。设备由一个可以检测心跳的光学模块，一个可以检测老人所在地场景的摄像头，一个 GPS 定位模块，一个物联网通讯模块（一般为 GPRS），MCU 和电源组成。云端由物联网平台为基础建立设备与云端通讯，配合 RDS 存储心跳 &GPS 数据，OSS 存储图片数据，最后用 IoT Studio 的服务开发与 Web 可视化开发功能完成功能页面搭建。整个云端开发过程只需要 2 小时以内即可。

最终效果如图。


杜氏医疗监控
监控大盘
设备管理
老人监控
3D模型区
事件告警
历史轨迹
热力图
手环列表
设备名称
备注名称
更新时间
设备状态
剩余电量
地理位置
心跳
手环视频链接
Wrist003
Wrist002
Wrist001


杜氏医疗监控
监控大盘
设备管理
老人监控
设备采买流程
联系供应商
审批预算
设备出货
应用添加设备
现有设备列表
全部状态
搜索
添加设备
设备名称
设备状态
创建时间
备注名称
操作
Wrist003
Wrist002
Wrist001


杜氏医疗监控
老人——桃阁
监控大盘
设备管理
老人监控
照片
82%
剩余电量
56BPM
当前心跳
告警
暂无事件告警
实况监控
24小时心跳曲线

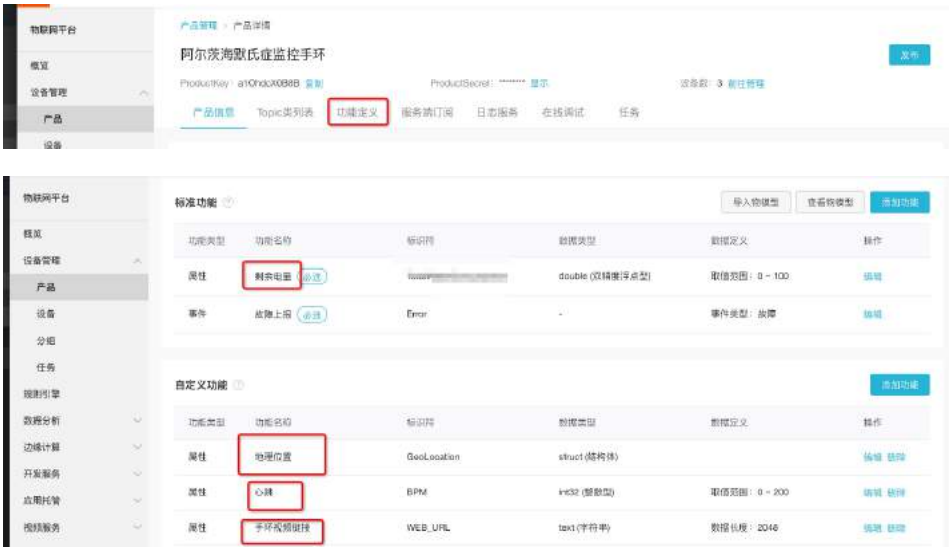
硬件部分

在 demo 阶段，我们采用树莓派 3B+ 摄像头 + 心跳模块 +GPS+ 电池的方法，验证不同数据的上报方法以及数据存储链路。考虑简单化，联网暂时采用 WIFI 方法。如果觉得使用电路比较麻烦，也可以使用服务开发 + 虚拟设备上报的方式，具体查看这篇文档。

虽然带手环的老人不一样，但是每个手环上报的属性类别是一样的，我们可以类似编程开发里把它们归结为同一个类 (class)。我们首先需要在物联网平台上为我们的 demo 手环建立一个设备类 (即产品)，这样我们才能在以后不断的往这个产品下实例化新的设备。

进入阿里云物联网平台，在产品页面新建一个产品，选择自定义品类即可，命名为“阿尔茨海默氏症老人监控手环”。

进入产品的功能定义页，定义 5 个自定义功能——剩余电量，地理位置，心跳，图片地址 (存放摄像机上传图片的 URL)。



地理位置只需要在“添加功能”里用标准的功能即可，如图，其他全部配置项默认即可。

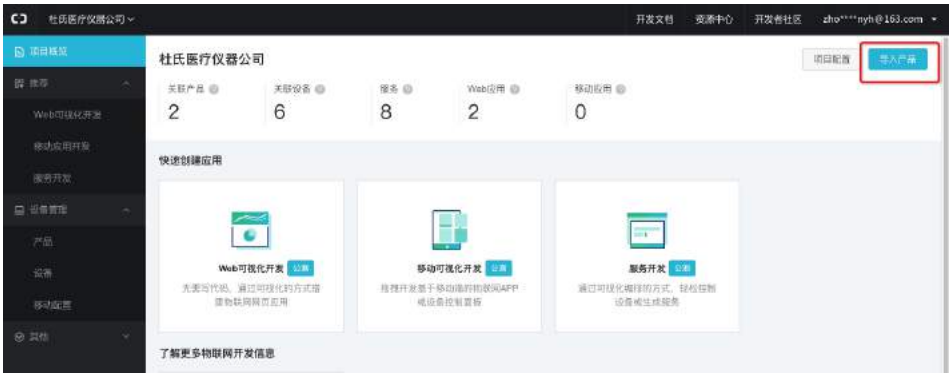
心跳为一个整型数据，剩余电量为浮点型数据，图片地址为字符型数据，如图。

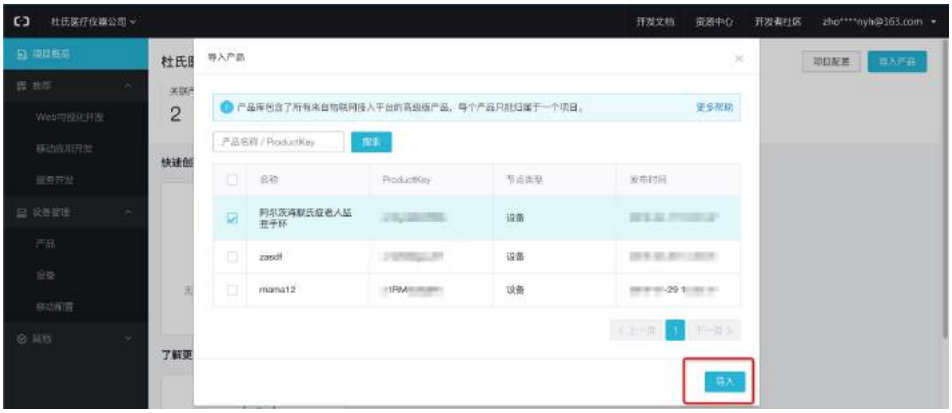
在设备面板点击“添加设备”，选择刚才创建的手环产品，然后输入随意的设备名称即可。

IoT Studio 为交付型业务做了项目维度的隔离，因此需要将用到的设备导入到对应的项目中。

首先打开物联网平台的“开发服务”选项进入 IoT Studio。点击某个项目名称的“查看”进入项目详情页。然后点击右上角的“导入产品”。选择刚才的手环产品，然后导入，可以在设备管理页看到产品以及下属的设备已经导入项目里。


物联网平台
概览
设备管理
产品
设备
分组
任务
规则引擎
数据分析
边缘计算
开发服务
IoT Studio
开放API
物联网开发
快速入口
Web可视化开发
移动可视化开发
服务开发
应用列表
Web可视化
移动应用
服务开发
项目列表
新建项目
查看


杜氏医疗仪器公司
开发文档
资源中心
开发者社区
项目概览
杜氏医疗仪器公司
项目配置
导入产品
关联产品
2
关联设备
6
服务
8
Web应用
2
移动应用
0
快速创建应用
Web可视化开发
移动可视化开发
服务开发
了解更多物联网开发信息


杜氏医疗仪器公司
开发文档
资源中心
开发者社区
导入产品
产品库包含了所有来自物联网接入平台的高级版产品，每个产品只能归属于一个项目。
更多帮助
产品名称 / ProductKey
搜索
名称
ProductKey
节点类型
发布时间
阿尔茨海默氏症老人监控手环
设备
zaxdf
设备
mama12
设备
导入


设备管理
产品
设备
移动配置
DeviceName
产品
节点类型
状态
最后上线时间
操作
Wrist003
阿尔茨海默氏症监控手环
设备
未激活
Wrist002
阿尔茨海默氏症监控手环
设备
未激活
Wrist001
阿尔茨海默氏症监控手环
设备
在线
2019-06-12 14:30:46

这样就完成了产品的定义，实例化与项目维度的隔离了。

上云部分

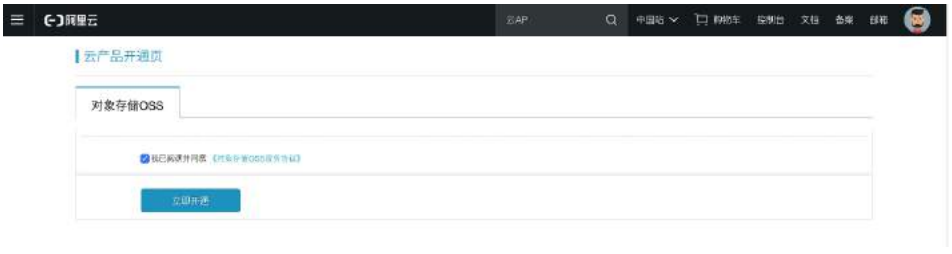
树莓派采用 python 编程，因此我们需要参考物联网平台的 python SDK，同时开发者社区也有很多相关文章。在这里我们直接跳过。

由于物联网平台的属性不支持直接存储图片，因此我们暂时使用 oss 进行存储。你也可以选择使用 HTTP/2 通道（已支持 python SDK）将图片上传至物联网平台每个设备单独的存储空间，不过从该存储空间调用图片的 URL 需要动态生成，可以参考这篇文档。

OSS 存储空间准备

阿里云对象存储服务（Object Storage Service，简称 OSS），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。您可以通过调用 API，在任何应用、任何时间、任何地点上传和下载数据，也可以通过 Web 控制台对数据进行简单的管理。OSS 适合存放任意类型的文件，适合各种网站、开发企业及开发者使用。

首先点击“立即开通”进入开通页面并点击同意协议。



然后进入控制台，新建一个 Bucket，一个 Bucket 相当于一个文件夹，可以通过 API 路径访问里面的文件。在这里我们选择公共读写。

然后可以看到 OSS 控制台新建了一个 bucket，里面是空的，我们可以尝试上传一些图片。

然后点击文件右方的操作项里的“复制文件 URL”，把复制的 URL 粘贴到浏览器，看看能否预览。

可以看到通过 URL 访问我们就能看到图片了，这样 OSS 的配置就完成了。

考虑到 Bucket 的公共读写特性，安全性会有一定的问题，可以考虑将 bucket 私有化，然后图片上传的时候设置图片为公共读写，并采用时间戳加盐等方式将图片文件名随机化的方式解决。当然安全性上 HTTP/2 通道为更优方案。

树莓派代码

树莓派的配置与连接在此不再赘述，可以在树莓派新建 py 文件，直接将此份代码复制过去，并且设置为开机执行，也可以参考这篇文档。

代码如下（基于 python 3。6），需要根据备注填入自己的账号信息，产品信息等：

注意，本 demo 代码忽略了电池电量检测模块

```
import aliyunsdkiotclient。AliyunIotMqttClient as iot ## 导入阿里云的设备 MQTT 库，
如果 import 失败需要先 pip3 install 一下
import json
import multiprocessing
import time
import random
import oss2 ## 导入阿里云的 OSS 库，如果 import 失败需要先 pip3 install oss2
from picamera import PiCamera ## 树莓派的摄像头，系统自带
import RPi。GPIO as GPIO ##GPIO 口，接红外 PIR 用
import serial
import pynmea2
from pulsesensor import Pulsesensor ## 导入树莓派的 pulsesensor 库，https://
github。com/tutRPi/Raspberry-Pi-Heartbeat-Pulse-Sensor/blob/master/example。py


auth = oss2。Auth('**AccessId*****','**AccessSecret*****') ##OSS 的授权需要阿里
云账号 AccessId 和 AccessSecret，具体查看 https://usercenter。console。aliyun。
```

```
com/#/manage/ak
bucket = oss2。Bucket(auth,'http://oss-cn-beijing。aliyuncs。com',’*** 你的
bucket 名称 ***’) ## 需要根据服务器区域修改节点路径，见文档
global picURLtoIoT
camera = PiCamera()
camera。resolution = (800,600)  ## 拍照分辨率，越高越容易分析，但是上次越慢
GlobalBpm = 0  ## 记录心跳数据
Latitude = 0    ## 记录 GPS 数据
Longtitude = 0


## 初始化树莓派
def init():
    GPIO. setwarnings(False)
    GPIO. setmode(GPIO. BOARD)
    GPIO. setup(3, GPIO. IN)
    pass

def take_photo():
    ticks = int(time。time())
    fileName = 'test%s。jpg' % ticks  ## 在文件名加入了时间戳作为简易加密手段
    filePath = '/home/pi/Pictures/%s' % fileName
    camera。capture(filePath)
    bucket。put_object_from_file('bucket_file_name/%s', fileName) ## 在这里改
bucket 名字
    global picURLtoIoT
    picURLtoIoT = ‘http://*** 你的 bucket 名称 **。oss-cn-beijing。aliyuncs。com/
bucket_file_name/%s’ % fileName
    ## 在这里改 bucket 名字和 bucket 内文件夹的名字
    print(str(picURLtoIoT))

def detect_Heartbeat():
    p = Pulsesensor()
    p。startAsyncBPM()
    try:
    while True:
        bpm = p。BPM
        if bpm > 0:
            print("BPM: %d" % bpm)
            GlobalBpm = bpm;
        else:
            print("No Heartbeat found")
        time。sleep(1)
    except:
        p。stopAsyncBPM()

def get_GPS():
    ser = serial。Serial("/dev/ttyAMA0",9600)
    while True:
```

```
        line = ser。readline()
        if line。startswith('$GNRMC'):
            rmc = pynmea2。parse(line)
            print "Latitude: ", float(rmc。lat)/100
            print "Longitude: ", float(rmc。lon)/100
            Latitude = float(rmc。lat)/100
            Longtitude = float(rmc。lon)/100
            break



options = {
    'productKey':'** 你的 ProductKey**',
    'deviceName':'** 你的 deviceName**',
    'deviceSecret':'** 你的 deviceSecret**',
    'port':1883,
    'host':'iot-as-mqtt。cn-shanghai。aliyuncs。com' ## 注意阿里云 IoT 国内都是华东
2，不一定跟 OSS 的节点一致
}
host = options['productKey'] + '。' + options['host']

def on_message(client, userdata, msg):
    topic = '/' + productKey + '/' + deviceName + '/update'
    print(msg。payload)

def on_connect(client, userdata, flags_dict, rc):
    print("Connected with result code " + str(rc))

def on_disconnect(client, userdata, flags_dict, rc):
    print("Disconnected。")

## 设备上报的定义
def upload_device(client):
    topic = '/sys/'+options['productKey']+'/'+options['deviceName']+'/thing/
event/property/post'
    while True:
        payload_json = {
            'id': int(time。time()),
            'params': {
                'BPM': GlobalBpm,
                'picURL': picURLtoIoT,
                'Geo':
                    {
                     'CoordinateSystem':1,
                     'Latitude':Latitdue,
                     'Longitude':Longtitude,
                     'Altitude':0
                    },
            },
```

```
            'method': "thing。event。property。post"
             }
        print('send data to iot server: ' + str(payload_json))
        client。publish(topic, payload=str(payload_json))

if __name__ == '__main__':
    client = iot。getAliyunIotMqttClient(options['productKey'],
options['deviceName'], options['deviceSecret'], secure_mode=3)
    client。on_connect = on_connect
    client。connect(host=host, port=options['port'], keepalive=60)
    p = multiprocessing。Process(target=upload_device, args=(client,))
    p。start()
    get_GPS()
    detect_Heartbeat()
    take_photo()
    GPIO. cleanup()
    client。loop_forever()
```

结束，把这个 python 文件设置为开机运行即可。

调试

可以看到数据已经上报到物联网平台了，同时 oss 的链接也可以用。

数据部分

接下来将演示如何通过 IoT Studio 服务开发工作台完成设备上报数据的转储以及根据规则进行告警（如心跳过低告警）。IoT Studio 服务开发是一个物联网业务逻辑的开发工具。通过编排服务节点的方式快速完成简单的物联网业务逻辑的设计。适用于以下场景：设备联动、设备数据处理、设备与服务联动、生成 API 、生成 App 的后端服务等。

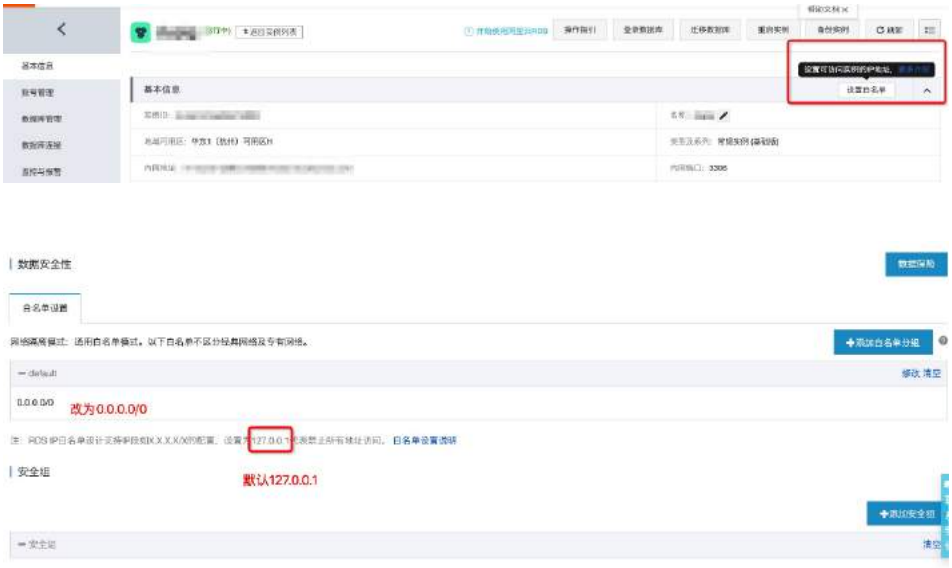
开通 RDS

阿里云关系型数据库 RDS（Relational Database Service）是一种稳定可靠、可弹性伸缩的在线数据库服务，提供容灾、备份、恢复、迁移等方面的全套解决方案，

彻底解决数据库运维的烦恼。目前六月份有 10 块钱试用半年的折扣，可以尝试购买。

开通以后进入管理控制台，会看到自己已经付费的实例出现，进入管理。

由于我们需要用外部的服务调用 RDS 数据录入，因此需要允许外网访问，需要首先设置白名单。然后把默认的 127。0。0。1 改为 0。0。0。0/0



然后返回管理页，可以看到外网地址出现了。

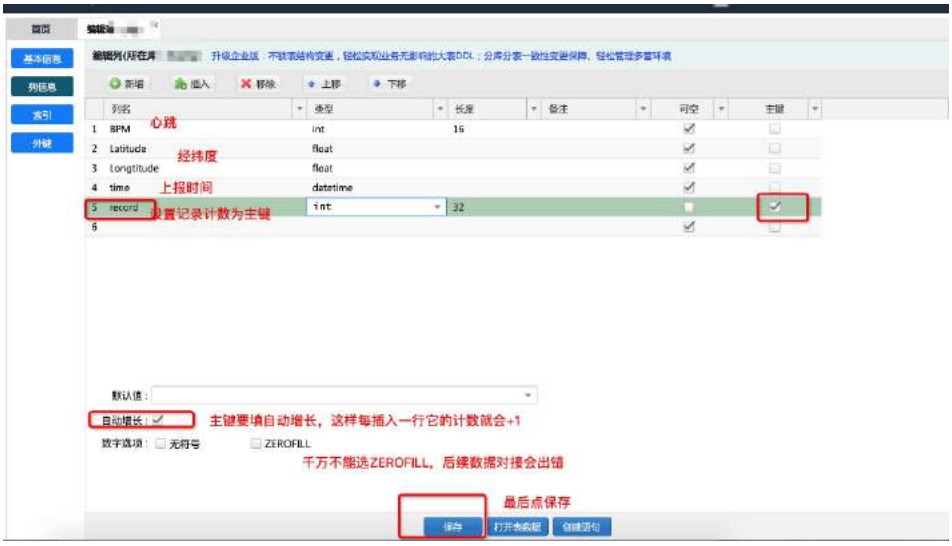
接下来需要设置登录数据库的账号，进入账号管理页点击创建账号，输入账号密码等信息，并且选择要授权的数据库。

完成后返回实例控制页，点击登录数据库，输入刚才设置的账号密码，即可登入 RDS 数据库。

3)

登录之后首先创建一个数据库，命名为 test，然后刷新一下，可以看到新建的 test 数据库，然后进入数据库建立一张表。

插入如下的几列，注意 time 可能需要改为 timestamp 类型，最后保存即可完成。这样就完成了表结构的配置。



数据对接 RDS

首先在物联网平台首页，开发服务下的 IoT Studio 的快速入口进入服务开发工作台。然后新建一个服务，命名为“手环心跳转储 RDS”。

然后在节点列表里选择“设备触发”节点，在右侧栏选择之前创建的产品“阿尔茨海默氏症监控手环”，监听所有设备的属性上报，如图。

接下来选择一个云数据库 mySQL 节点，将设备触发节点与云数据库节点连接起来。

参数如下，分别对应之前的列名，record 不填，因为 record 会自动增长：

```
{
    "table": "test",![_rds5](https://yqfile。alicdn。com/
d0979c17aa88ac03cd75ae1752c0b85e6b47fc23。png)

    "rows": [
        {
            "BPM": "{{query。props。BPM. value}}",
            "Latitude": "{{query。GeoLocation。value。Latitude}}",
            "Longtitude": "{{query。GeoLocation。value。Longtitude}}",
            "time": "{{query。deviceContext。gmtCreate}}"
        }
    ]
}
```

数据格式说明

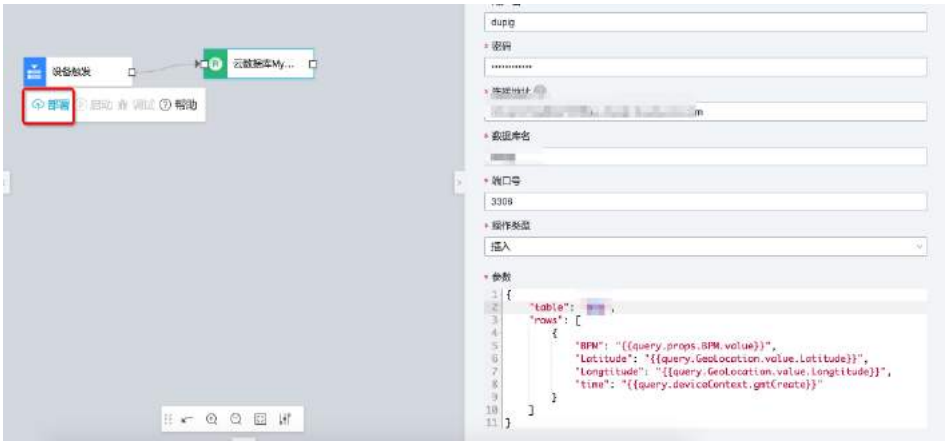
我们可以看到，设备上报的数据格式设这样的：


```
{
            "deviceContext": {
                "productKey": "a1OhdcX0B8B",
                "deviceName": "Wrist003",
```

```
            "gmtCreate": 1560497545957
        },
        "props": {
            "GeoLocation": {
                "time": 1560497545957,
                "value": {
                    "CoordinateSystem": 1,
                    "Latitude": 25。26,
                    "Longitude": 111。45,
                    "Altitude": 0
                }
            },
            "BPM": {
                "time": 1560497545957,
                "value": 21
            }
        }
    }
```

在服务开发中，如果需要在 json 参数里调用外部动态的参数，需要采用 {{xx。xx}} 的方式调用。如在”BPM”：“{{query。props。BPM. value}}” 里，第一个 query 表示参数来自于第一个节点，第二个 props 表示取设备上报上来的属性数据，第三个 BPM 表示取 props 下的 BPM 对象，最后的 value 表示取 BPM 对象的值。同理其他几个可以根据设备上报数据的结构进行填写。

完成后点击部署，调试（可以使用虚拟设备上报），回到 RDS 的数据库页面，可以看到数据更新了。这样就完成设备数据上报转 RDS 的操作。



数据即时告警规则

如果老人的心跳过低，我们会通过钉钉机器人把消息通知到护士群里，让他们即时知晓情况。

首先仍然是新建一个服务，命名为心跳过低告警。

然后同样选择一个设备触发节点，侦听所有手环设备上报的属性。

添加一个“条件判断”节点，条件节点相当于一个 if-else 判断。并且把设备触发节点与条件节点连接起来。

在条件判断节点中，第一个选择“同时满足所有条件”，在“条件 1”中第一个下拉框选择“设备触发”，在二级菜单选择“心跳”；判断条件选择“<=”，第二个框选择“静态值”“数值”。触发报警的条件我们设置为老人心跳值小于 50。

在之前的数据对接 RDS 里，我们使用代码化的 query。props。BPM. value 定义设备数值。而在条件判断等节点中，我们封装了数据源格式，可以让你直接选择数据源进行规定格式的告警信息配置，而无需输入 query/payload 等变量。

接下来在左侧节点列表的“功能”类拖入一个“钉钉机器人”节点。并与条件判断节点的上方出口（“满足条件”）进行连线。选择模板为“设备告警”模板，数据源选择“设备触发”“心跳”，可以选择 @ 所有人。

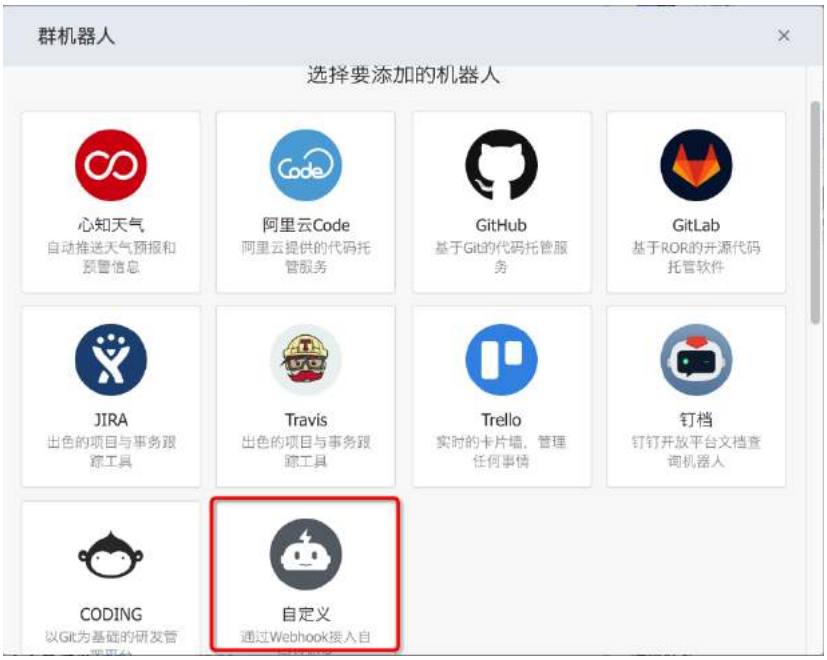
钉钉机器人的 Webhook 填入你要推送的钉钉群的钉钉机器人 Webhook。

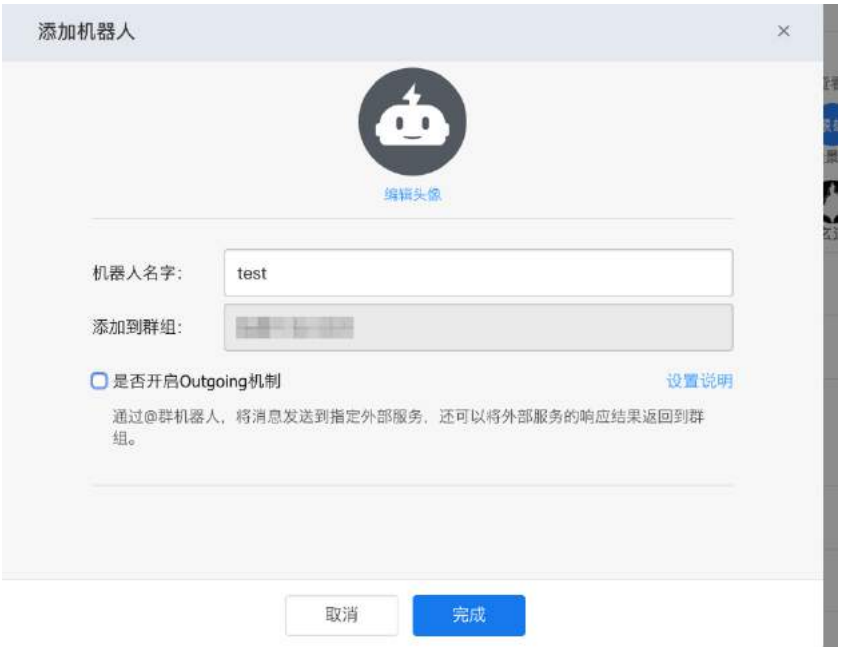
如何获取 Webhook？

钉钉机器人是钉钉群内一个自动化的消息发送工具。在一个钉钉群内打开右上角的“群设置”，可以发现以下弹窗。

点击钉钉机器人，进入机器人配置页面。

选择添加自定义机器人。



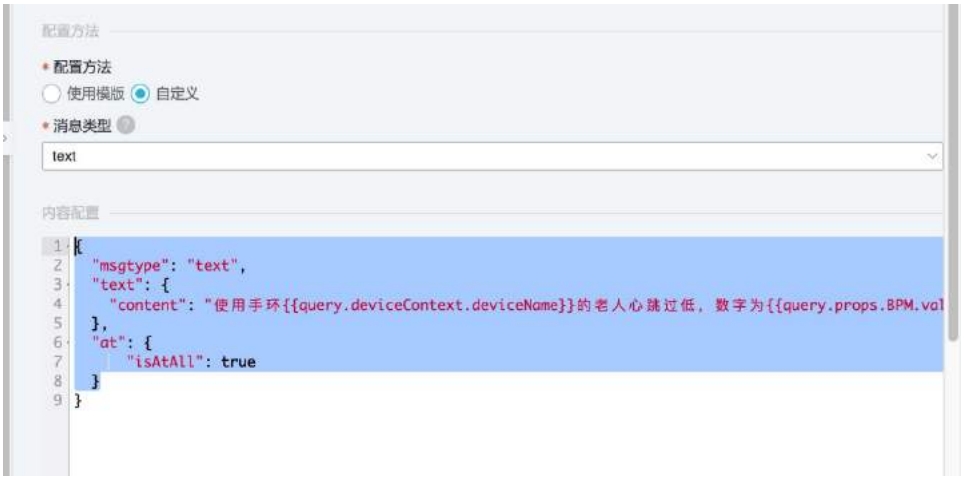


然后在完成页面可以看到 webhook。整个复制下来，粘贴到输入框内。

如果是自定义模板，如何动态配置文本 text？

由于监听了全部的煤气检测器，我们收到警告的时候需要知道是哪个煤气检测器报警了，因此需要接受上报的煤气检测器的 DeviceName 进行推送。

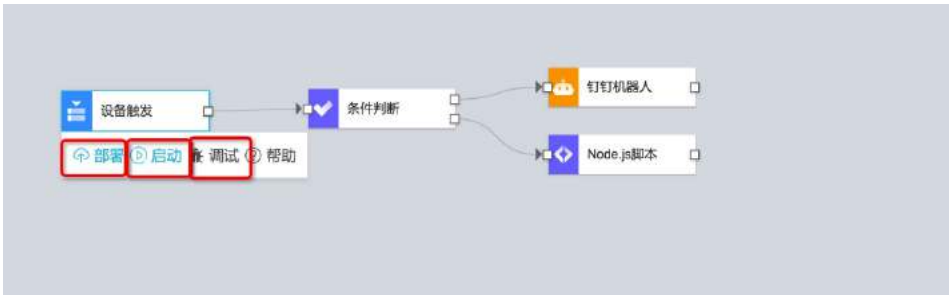
选择 text 推送类型，参数框内为一个 json 对象，因此调用方法要符合 json 的格式。我们采用了 {{value}} 的格式，如“{{query。deviceContext。deviceName}}”，可以查看下图的完整配置方法：




```
{
  "msgtype": "text",
  "text": {
    "content": "使用手环{{query。deviceContext。deviceName}}的老人心跳过低，数字为
{{query。props。BPM. value}}，大家快去看看吧！"
  },
  "at": {
      "isAtAll": true
  }
}
```

对条件判断节点的“不满足条件”，放置一个不做任何处理的 nodejs 脚本占位即可。

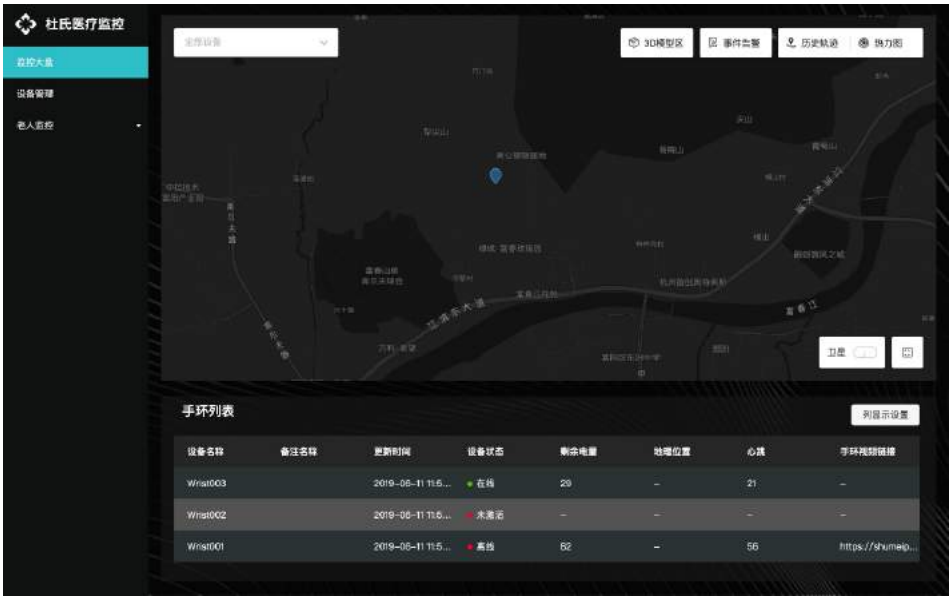
完成后，同样的进行部署启动调试。在虚拟设备那边上报一个小于 50 的 BPM，看看是否成功响应。这样就完成了一个即时响应的心跳告警功能。

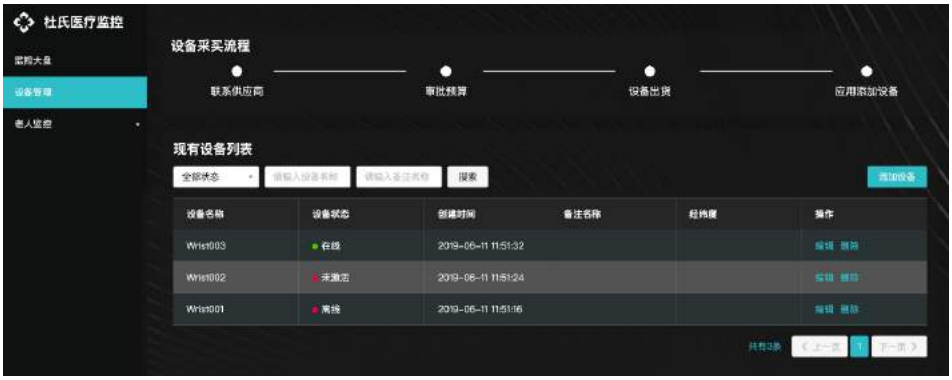


应用部分

本次的应用部分包括设备管理的页面，允许医院维护人员查看各个设备的状态，属性并根据需求添加新的设备。另外有当前所有设备在地图上的分布情况，方便监控老人动向。最后包含了一个监控手环上报图片的实时查看功能，可以查看老人有没有遇到危险，迷路等。

最终的实现效果如图：

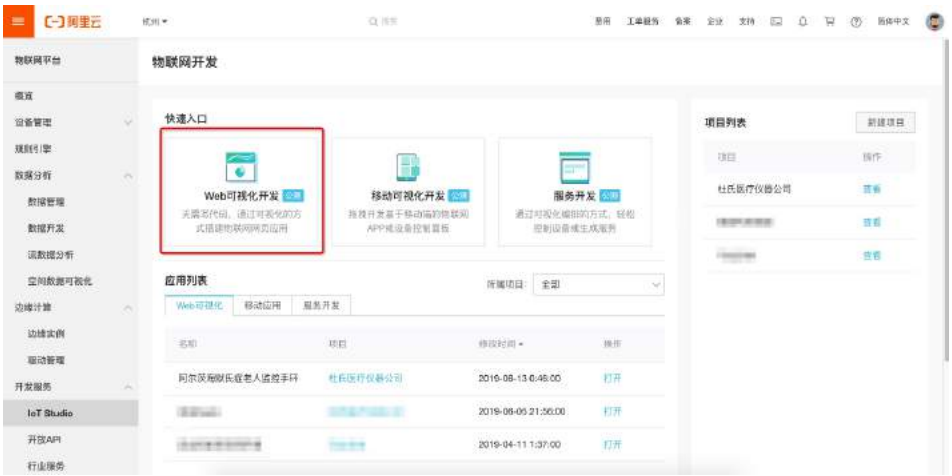

杜氏医疗监控
监控大盘
设备管理
老人监控
3D模型区
事件告警
历史轨迹
热力图
卫星
手环列表
列显示设置
设备名称
备注名称
更新时间
设备状态
剩余电量
地理位置
心跳
手环视频链接
Wrist003
2019-06-11 11:5...
在线
29
21
Wrist002
2019-06-11 11:5...
未激活
Wrist001
2019-06-11 11:5...
离线
82
56
https://shumeip...


杜氏医疗监控
监控大盘
设备管理
老人监控
设备采买流程
联系供应商
审批预算
设备出货
应用添加设备
现有设备列表
全部状态
搜索
添加设备
设备名称
设备状态
创建时间
备注名称
经纬度
操作
Wrist003
在线
2019-06-11 11:51:32
Wrist002
未激活
2019-06-11 11:51:24
Wrist001
离线
2019-06-11 11:51:16
编辑
删除

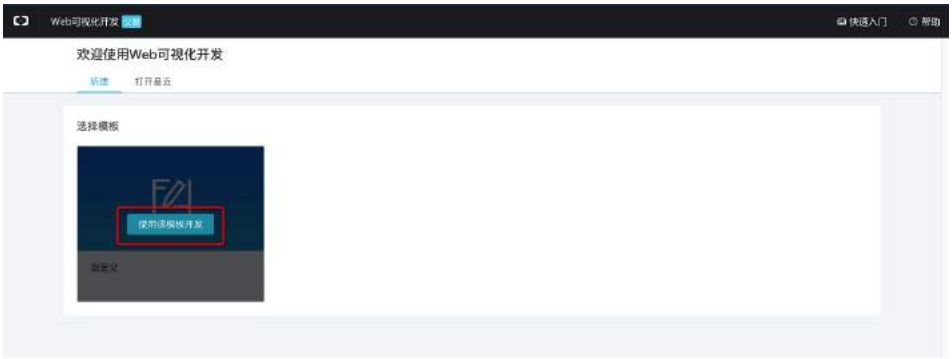

杜氏医疗监控
监控大盘
设备管理
老人监控
老人——姚阁
照片
82%
剩余电量
56BPM
当前心跳
告警
暂无事件告警
一键报警告警
实况监控
24小时心跳曲线
60
58
56
54
52
50
05/10 14:43
05/12 08:43
05/14 02:43
05/15 20:43
05/17 14:43

手环管理 SaaS 创建

首先进入 IoT Studio 页，点击 Web 可视化开发，进入 Web 可视化页面。



然后新建一个空白模板，输入应用的名字，如“阿尔兹海默症示例应用”。

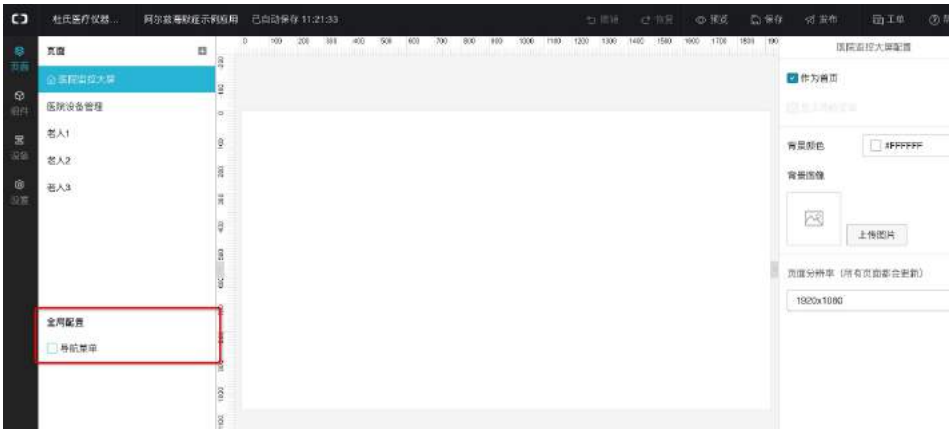


进入空白的 Web 页面，准备开始搭建我们的应用。

左侧栏的构建

首先把整个应用的架构搭建好，创建 5 个页面，分别为医院监控大屏，医院设备管理，老人 1 的详情页，老人 2 的详情页，老人 3 的详情页。

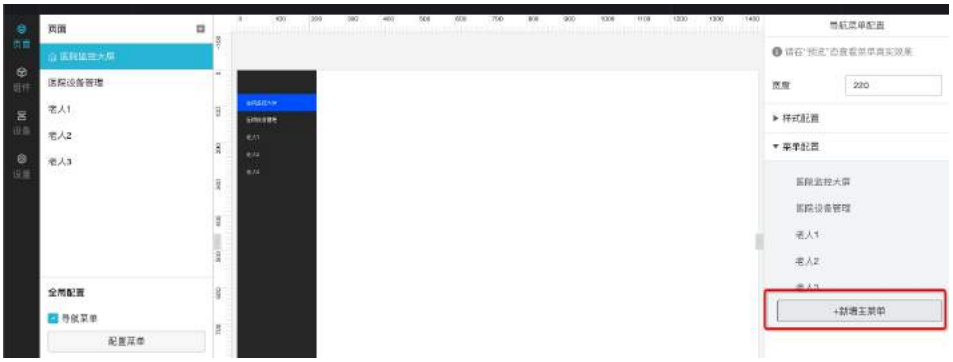
接下来我们为整个应用添加一个左侧栏，点击左下方的“导航菜单”。



由于我们已经建好了对应的空页面，可以选择自动生成。

可以看到生成的左侧菜单栏，现在想把老人 1，2，3 三个页面归属到一个“老人监控”的主分类下，因此需要修改配置菜单项。点击左下角的“配置菜单”。此外这里修改了分辨率为 1440x900。

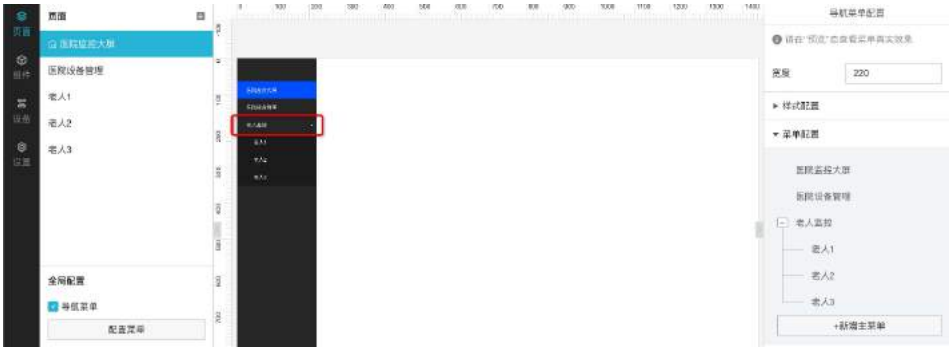
点击新增主菜单，输入“老人监控”这个主分类名。这个主类目不会对应任何实际的页面链接，只是一个分隔符。



出现二次确认弹窗确认即可。

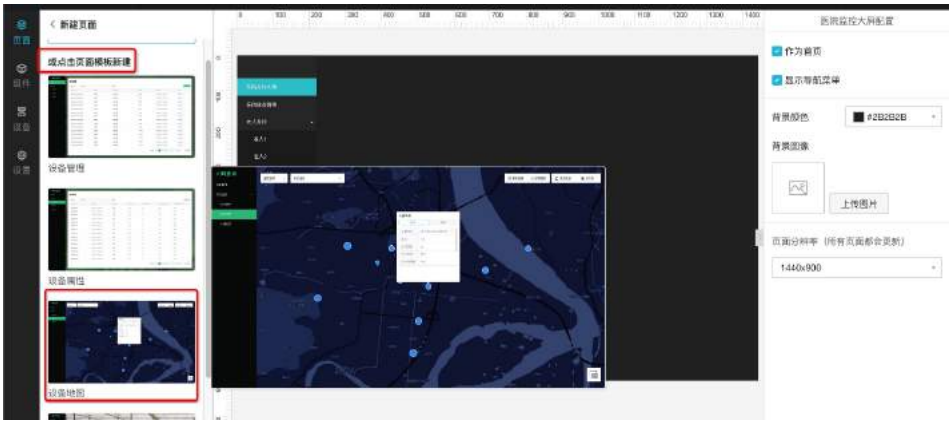
最后改成这样的结果。可以检查一下里面的链接是否正确的配置。

可以看到左侧菜单出现了“老人监管”这个分类。

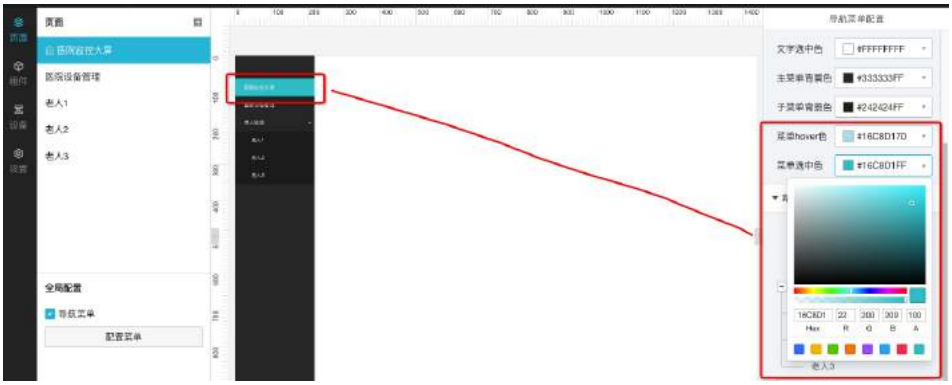


配置页面基本元素 & 样式

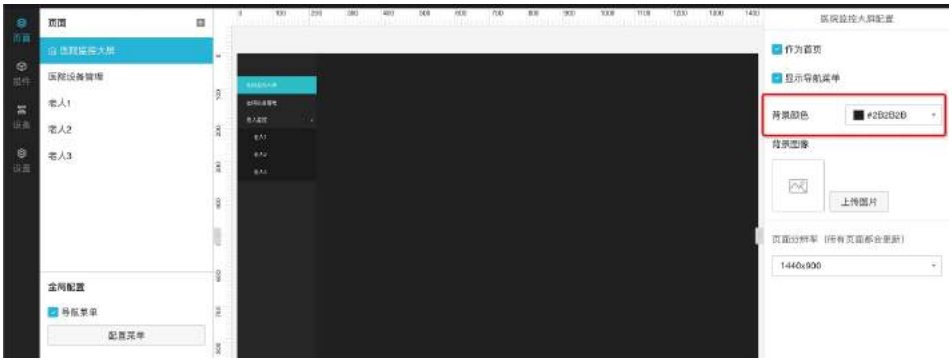
想要 SaaS 应用的样式与众不同？可以自定义各种样式，也可以使用标准模板。使用标准模板可以在新建页面的时候点击。



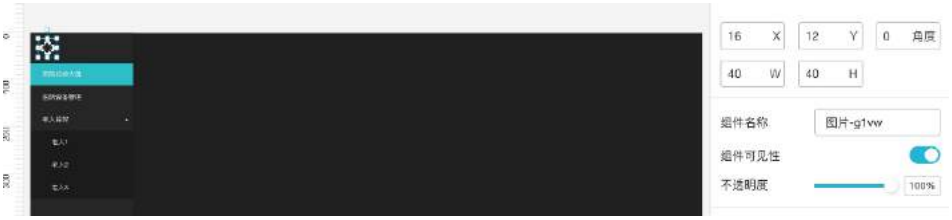
也可以根据自己的喜好，用空白模板，然后自定义样式，比如修改菜单栏颜色。



修改背景颜色。



放入一些图片。

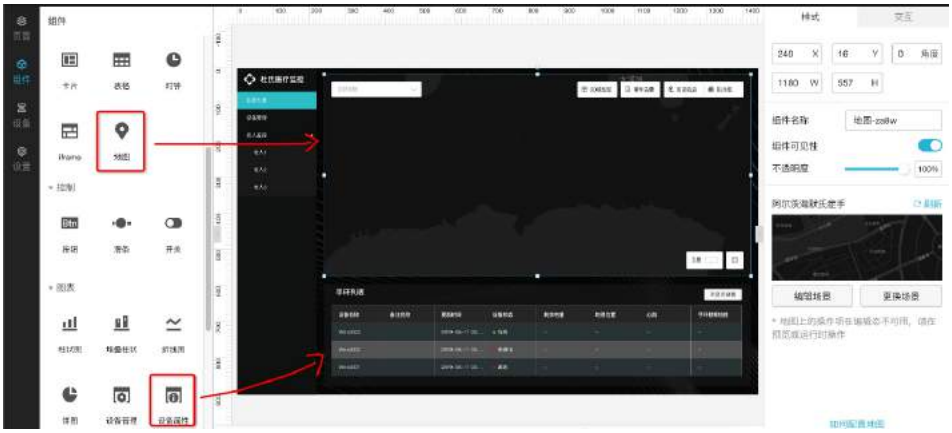


输入文字等装饰。最后变成这样的页面。

接下来我们需要添加一些有功能的组件，让整个 SaaS 应用运作起来。

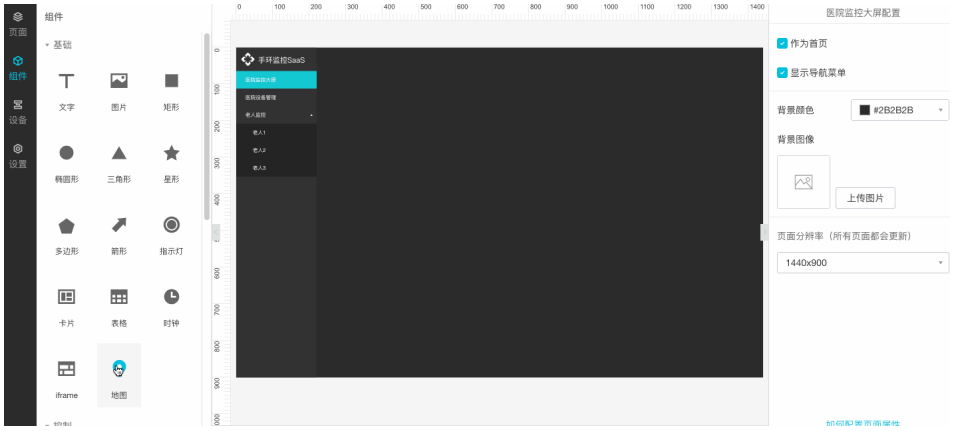
医院监控大屏页

如同示例，我们需要一个指示设备 GIS 的地图以及一个展示设备状况的列表。在组件栏对应的是这两个组件。



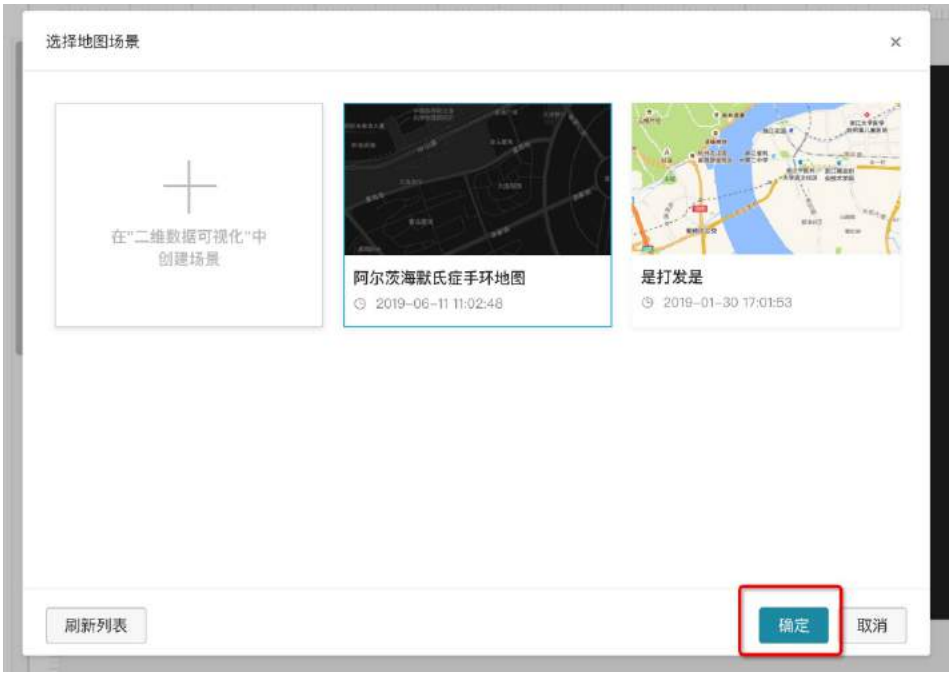
设备地图

设备地图需要用到物联网平台数据分析功能的空间数据可视化服务。过程如下图 gif 显示，首先在左侧栏拖入地图组件，然后点击添加场景，前往空间数据可视化页面，然后点击添加，选择“阿尔兹海默症手环”产品，点击确定。

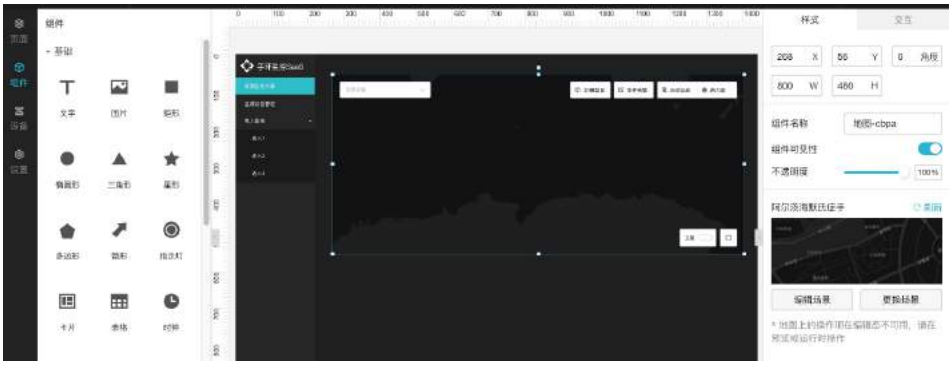


可以修改地图主题色等。

选择后返回 IoT Studio，重新点击地图组件选择刚才创建的场景，就可以看到在页面上出现了一份有设备位置的地图。



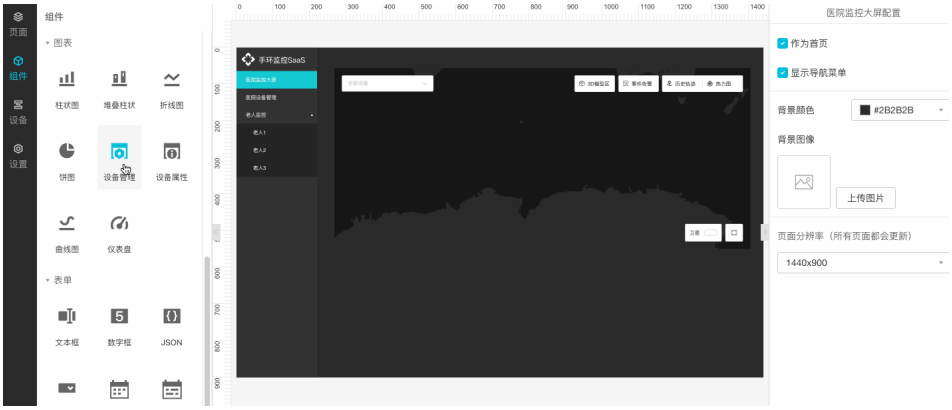
可以调节大小，放到合适的位置上，就完成了地图组件的配置了。



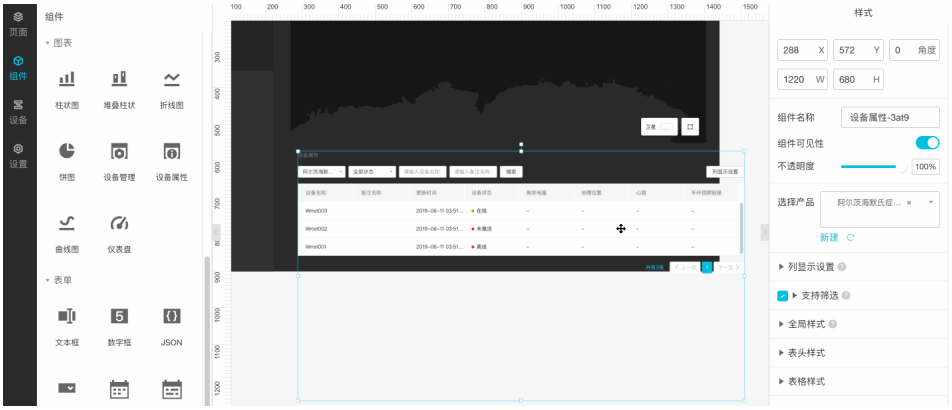
设备属性表格

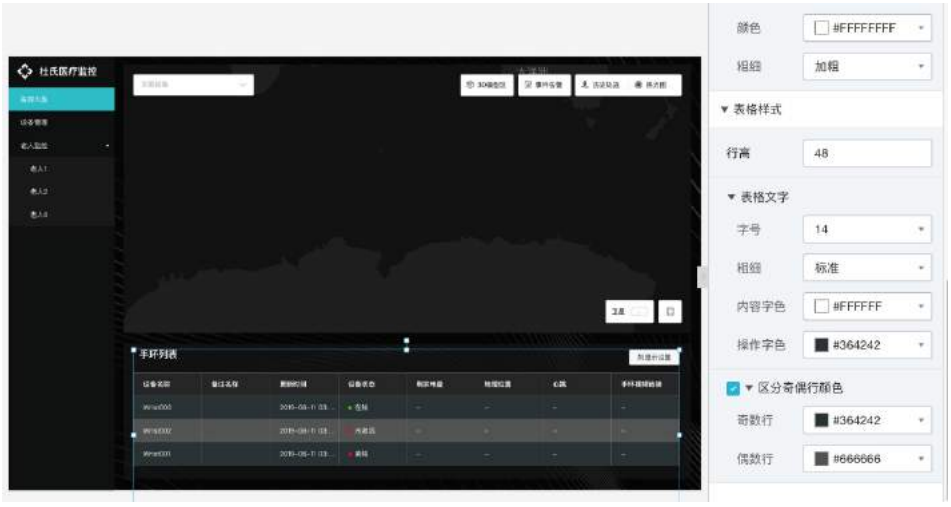
设备属性表单可以把当前产品下所有设备（手环）的在线状态，更新时间以及属性快照值的显示出来，并且在应用发布后可以调整显示的列内容。适合全局性的设备预览。

首先拖入设备属性组件，在右侧栏配置要关联的产品。

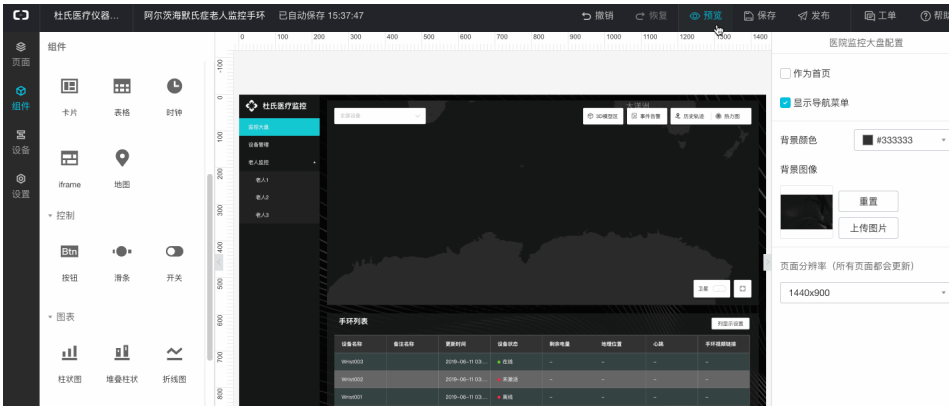


可以看到关联了产品后，表单自动显示产品下的全部设备。接下来我们可以修改一下右侧的配置项，让他的样式更符合黑色的背景。





然后我们可以点击右上角预览，查看一下页面效果。可以在预览时点击“列显示设置”按钮修改显示的列数，这个配置是本地存储的，不会同步到云端。

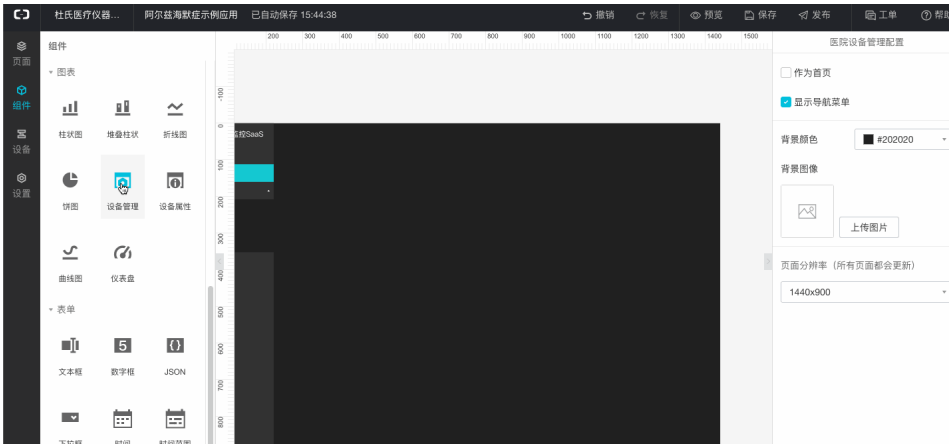


这样第一个医院监控大屏页面就完成了。

医院设备管理页

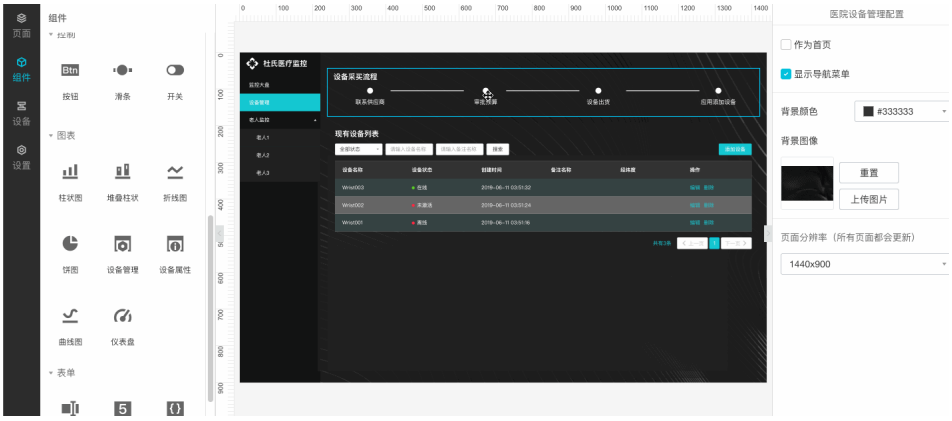
设备管理页允许运维人员（而不是开发者）直接添加新设备获取三元组，这样他们就可以不感知阿里云物联网平台而实现设备的添加。为了实现这个功能，需要设备管理这个表单组件，如图。

配置方法为：首先从左侧栏拖入设备属性组件，然后在右侧栏修改一些配置项——包括是否允许发布之后用户添加 / 编辑 / 删除设备，修改列显示排序，添加自定义标签，修改样式等等。具体功能说明可以查看文档。



然后可以通过圆形组件，矩形组件以及文字组件添加设备采买流程等，最后完成的效果如图。

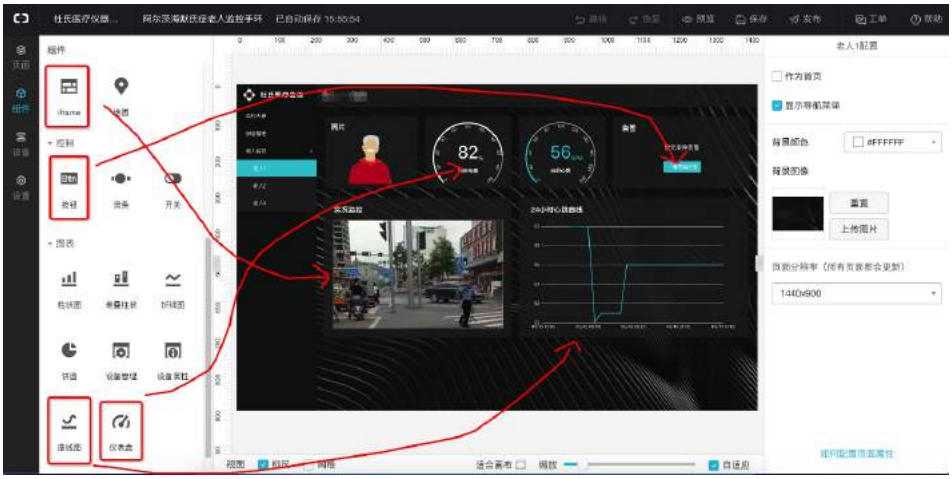
我们可以预览，看看是否可以添加 / 删除设备。注意这里的添加 / 删除是会直接影响物联网平台上的设备的。



这样设备管理页就完成了，最后就是每个设备的详情页，也就是每个监控手环的信息展览页。

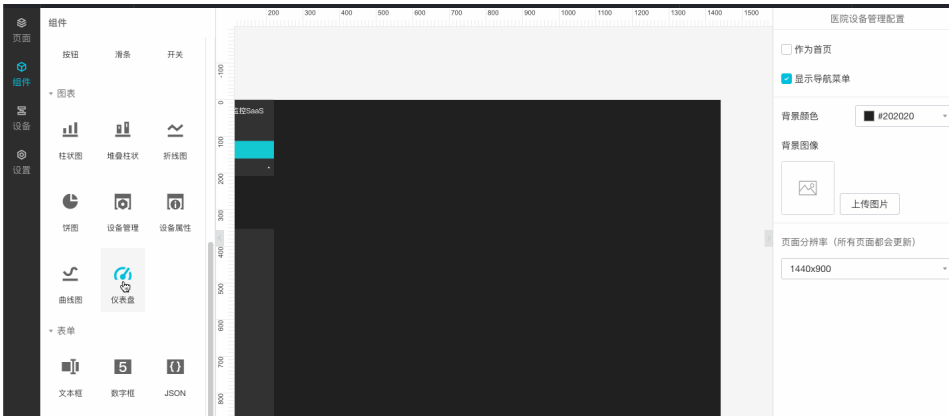
设备监控页

设备监控需要监控当前老人的心跳，剩余的电量，是否有告警信息，摄影设备的实时图像监控以及心跳的历史趋势图等。

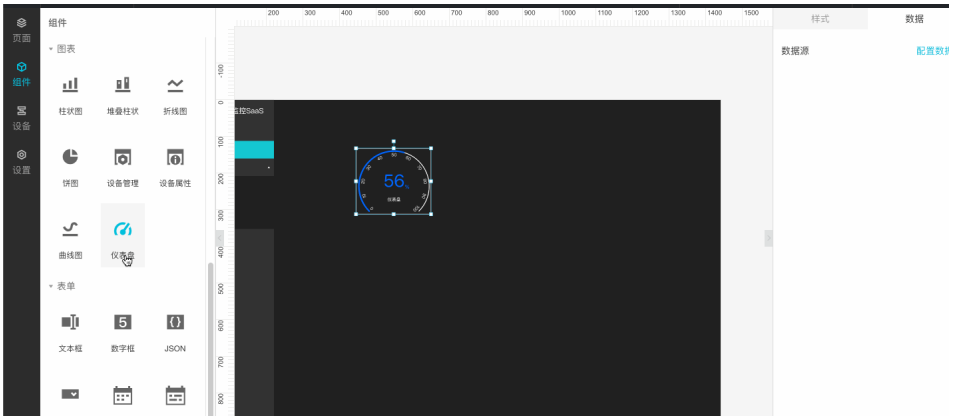


仪表盘的配置

首先从左侧栏拖入一个仪表盘，修改一下样式。



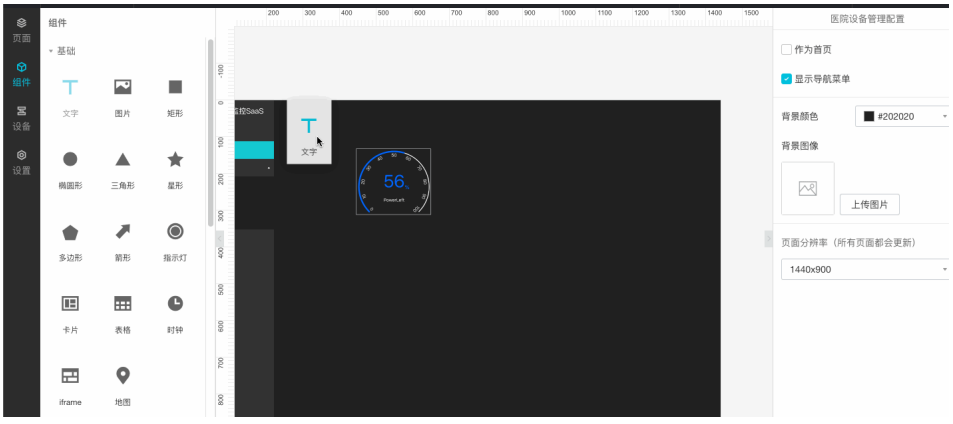
然后关联设备数据，这里关联的是老人 001 的 Wrist001 设备的剩余电量。选择产品，设备，属性，然后验证数据格式。注意如果设备没有上报过信息，数据格式验证是无法通过的，可以通过虚拟设备上报信息进行验证。

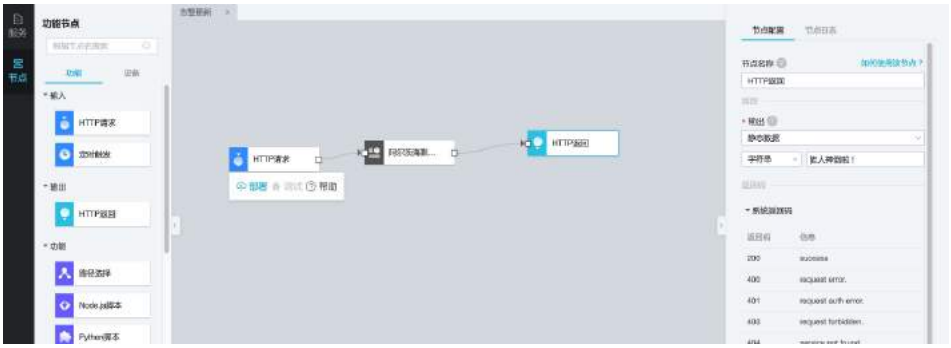


这样就完成了剩余电量的仪表盘了，同理心跳数据的仪表盘也一样操作。

告警信息系统

我们需要一段文字来接受来自设备的告警信息，同时需要一个按钮调用告警服务。首先拖入一段文字组件，然后关联对应设备的对应事件（如老人摔倒），验证格式之后文字会显示事件的快照值以及输出参数。当然可以通过服务开发工作台的 HTTP 请求接口修改文字内容，定制化等。



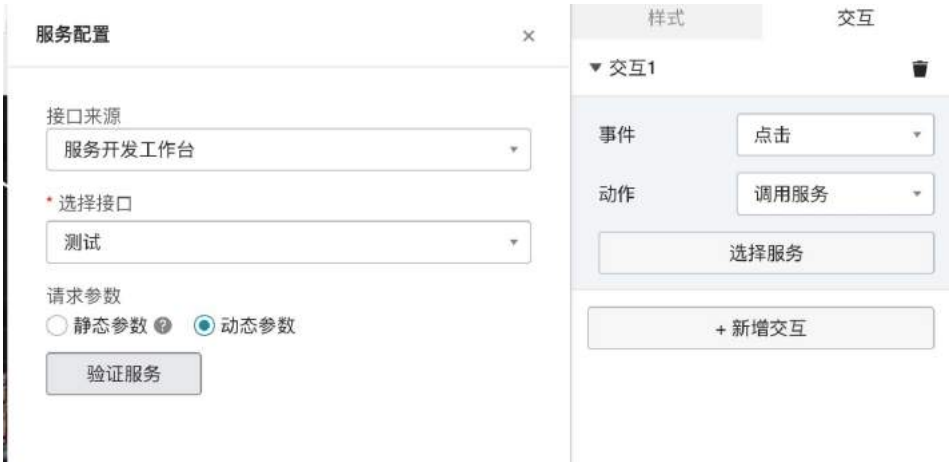


可以拖到最下方进行轮询，保证数据的实时性。

接下来是一个告警服务，比如短信通知家属，我们可以开通阿里云的短信服务或者钉钉机器人进行消息的输出，也可以用服务开发的“三方 API”节点进行微信公众平台等三方输出。这里我们以钉钉机器人为例。

新建一个服务，使用一个 HTTP 请求节点，中间接一个钉钉机器人节点，最后接一个 HTTP 返回节点即可。HTTP 请求不需要入参，HTTP 返回不需要额外配置，钉钉机器人节点的配置项如下。

在按钮那里选择交互 – 点击 – 调用服务，选择对应的服务，验证即可看到钉钉机器人推送。

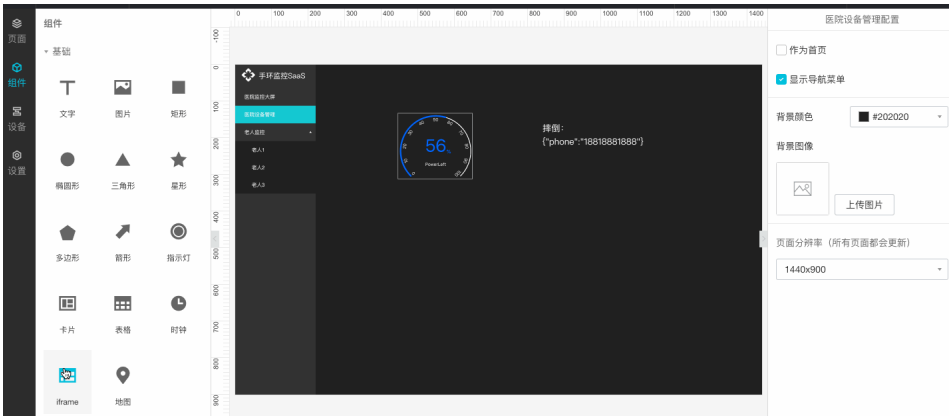


这就完成了告警模块的设置。

实况监控

图片的实况监控，如之前说的我们获取了设备图片的 url 作为属性上报，我们可以使用“变量”机制，帮助 iframe 组件获取设备上报的属性。

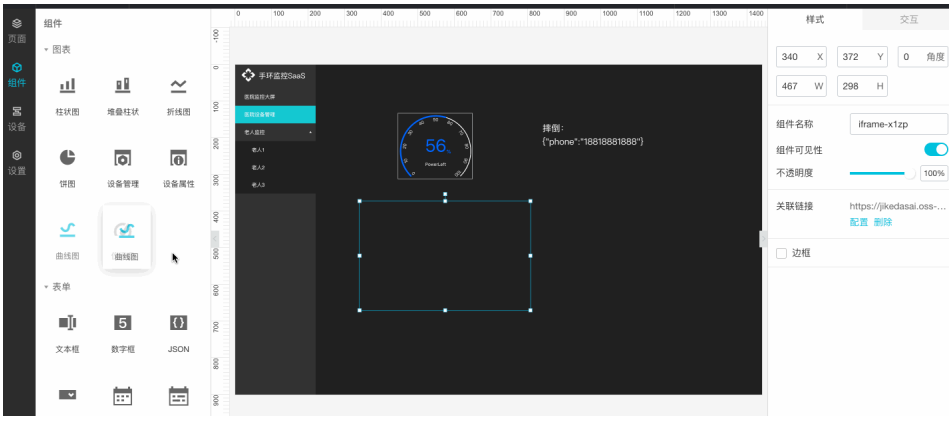
首先拖入一个 iframe 组件，然后直接把之前步骤里获得的 oss 图片链接粘贴上去即可。



心跳曲线

心跳曲线需要用到设备曲线图组件，可以直接关联具体的设备获取数据，无需额外的配置。

首先拖入一个曲线图组件，在数据源侧选择设备 Wrist001，点击验证数据，即可读取最近上报的数据。



然后调节样式即可，最后样式如下：

这样就完成了整个设备监控页。其他页面按此流程绑定不同设备即可，目前组件也已经支持跨页面复制。

发布应用

我们需要先在阿里云上申请一个域名，打开域名购买，购买一个域名。

然后前往 IoT Studio 的 Web 可视化工作台的设置菜单，点击域名管理。按照引导流程处理进行域名解析。

然后点击右上角的发布，点击确定即可。

然后就可以看到发布成功了。直接前往自己购买的域名查看即可。

这样就完成一套全链路的阿尔茨海默病人护理物联网解决方案的发布了。

Alibaba Group
阿里巴巴集团

阿里技术

扫一扫二维码图案，关注我吧

「阿里技术」微信公众号

AIoT 开发互动手册

阿里云开发者社区