

《AI机器视觉/语言教学平台》说明书

1 智云平台概述

一、实验目的

1. 了解智云平台及相关框架。
2. 掌握智云数据仿真工具的使用。
3. 掌握智云数据测试工具的使用。
4. 掌握智云平台的项目演示。

二、实验环境

硬件环境:	PC 机 Pentium 处理器双核 2GHz 以上, 内存 4GB 以上
操作系统:	Windows7 64 位及以上操作系统
开发软件:	智云硬件物元仿真平台 HSIMS、ZCloudWebTools、Chrome 浏览器

三、实验内容

1. 智云平台

1.1 平台介绍

智云物联是一个开放的公共物联网接入平台,目的是为服务所有的爱好者和开发者,使物联网传感器数据的接入、存储和展现变得轻松简单,让开发者能够快速开发出专业的物联网应用系统。



一个典型意义的物联网应用,一般要完成传感器数据的采集,存储,和数据的加工与处理这三项工作,举例来说,对于驾驶员,希望获取去目的地的路途上的路况,为了完成这个目标,就需要有大量的交通流量传感器对几个可能路线上的车流和天气状况进行实时的采集,并存储到集中的路况处理服务器,应用

在服务器上通过适当的算法，从而得出大概的到达时间，并将处理的结果展示给驾驶员。所以，我们能得出大概的系统架构设计可以分为如下三部分：

- 1) 传感器硬件和接入互联网的通信网关(负责将传感器数据采集起来，发送到互联网服务器)。
- 2) 高性能的数据接入服务器和海量存储。
- 3) 特定应用，处理结果展现服务。

要解决上述物联网系统架构的设计，需要有一个基于云计算与互联网的平台加以支撑，而这个平台的稳定性、可靠性、易用性，对该物联网项目的成功实施，有着非常关键的作用。智云物联公共服务平台就是这样的一个开放平台，实现了物联网服务平台的主要基础功能开发，提供开放程序接口，为用户提供基于互联网的物联网应用服务。同时满足针对高校的特殊应用需求。

智云物联提供了完整的物联网云应用实验室解决方案，目标是服务国内物联网应用技术教学，为高校师生提供一个共享的基于互联网的物联网云服务平台。

使用智云物联平台进行项目开发，具备以下优势：

- 1) 让无线传感网快速接入到互联网和电信网，支持手机和 Web 远程访问及控制。
- 2) 解决多用户对单一设备访问的互斥，数据对多用户的主动消息推送等技术难题。
- 3) 提供免费的物联网大数据存储服务，支持一年以上海量数据存储、查询、分析、获取等。
- 4) 开源稳定的底层工业级传感网络协议栈，轻量级的 ZXBee 数据通信格式(JSON 数据包) 易学易用。
- 5) 开源的海量传感器硬件驱动库，开源的海量应用项目资源。
- 6) 免应用编程的 BS 项目发布系统，虚拟仿真系统，LabView 数据接入系统。
- 7) 物联网分析工具，能够跟踪传感网络层、网关层、数据中心层、应用层的数据包信息，快速定位故障点。
- 8) 良好的社区服务与不断积累的开发者，享受分享和讨论的乐趣。

1.2 基本框架

物联网系统包括感知层、网络层、平台层、应用层四个部分组成，智云物联平台承担平台层的数据中枢智能，同时也为感知层、网络层、应用层提供软硬件平台和项目案例支撑。



● 感知层 - 全面感知

感知层以单片机、嵌入式技术为核心，赋予物品智能化、数字化，通过传感器采集物理世界中发生的物理事件和数据，包括各类物理量、标识、音频、视频数据。

物联网的数据采集涉及传感器、RFID、多媒体信息采集、二维码和实时定位等技术。

智云平台提供各种教学型、工业型控制节点及丰富的传感器外设，满足感知层应用需求。

● 网络层 - 网络传输

网络层包括传感网和互联网两部分。传感网通过无线技术实现物品之间组网、通信和数据传输，根据应用场合不同可选择各种不同类型协议，比如：ZigBee、BLE、Wi-Fi、NB-IoT、LoRa、LTE。互联网是现实社会信息交换的事实上标准，通过智能网关/路由设备实现传感网和互联网的数据交互。

智云平台提供各种教学型、工业型有线&无线节点和网关，支持异构网络融合技术，同时采用易懂易学的 JSON 数据通信格式的 ZXBee 轻量级通信协议，满足教学需求。

● 平台层 - 数据中枢

平台层是物联网数据的汇聚中心，通过云计算、大数据等技术，将感知到的信息无障碍、高可靠性、高安全性地进行处理和传输，负责对传感网络传递过来的数据进行中央处理，并为上层应用提供数据服务。

智云物联平台支持海量物联网数据的接入、分类存储、数据决策、数据分析及数据挖掘。采用分布式大数据技术，具备数据的即时消息推送处理、数据仓库存储与数据挖掘等功能。智云业务管理平台是基于 B/S 架构的后台分析管理系统，支持 Web 对数据中心进行管理和系统运营监控。主要功能模块：消息推送、数据存储、数据分析、触发逻辑、应用数据、位置服务、短信通知、视频传输等。

● 应用层 - 应用服务

应用层是产品展示给用户的交互视窗和接口，通过移动 App、Web、小程序、终端(手机、电视、电子屏、智能音箱)等方式进行操作互动。应用层根据 IoT 场景和业务不同，提供智能交通、智能医疗、智能家居、智能物流、智能电力等行业应用服务。

智云物联开放平台应用程序编程接口，提供 SensorHAL 层、Android 库、Web JavaScript 库等 API 二次开发编程接口，具有互联网/物联网应用所需的采集、控制、传输、显示、数据库访问、数据分析、自动辅助决策、手机/Web 应用等功能，可以基于该 API 上开发一整套完整的互联网/物联网应用系统；

提供实时数据(即时消息)、历史数据(表格/曲线)、视频监控(可操作云台转动、抓拍、录像等)、自动控制、短信/GPS 等编程接口；

基于 JSP 框架的 B/S 应用管理平台，支持用户注册及管理、后台登录管理等基本功能，支持项目属性和前端页面的修改，能够根据项目需求定制各个行业应用服务，比如：智能家居管理平台、智能农业管理平台、智能家庭用电管理平台、工业自动化专家系统等……

提供虚拟仿真软件，支持各种自定义设备，包括：传感器、执行器、摄像头等的动态添加、删除和管理，无需编程即可完成不同应用项目的构建；

支持与 LabView 仿真软件的数据接入，快速设计物联网组态项目原型。

1.3 开发入门

基于智云平台学习开发之前，要求用户预先学习以下基本知识和技能：

- 1) 掌握基于 CC2530 的单片机接口技术/传感器接口技术。
- 2) 掌握 ZigBee 无线传感网基础知识，以及基于 CC2530 的 ZigBee ZStack 组网原理。了解其他传感网知识和网络协议，比如：BLE、Wi-Fi、NB-IoT、LoRa、LTE。
- 3) 掌握 HTML、JavaScript、CSS、Ajax 开发，熟练使用 DIV+CSS 进行网页设计。了解 JAVA 编程。
- 4) 熟悉和了解 VUE 和 NodeJS 前后端开发框架和移动端开发框架。

用户可以通过基础实验教学平台进行上述内容的学习，包括以下课程：

《单片机与传感器》	《嵌入式与传感器》	《物联网识别技术》
		
《短距离通信技术》	《长距离通信技术》	



2. 仿真工具

虚拟仿真系统依托智云中间件平台，满足物联网传感网络与互联网的互联、互通和互动。虚拟仿真模块主要包括四大部分：三维场景模拟系统、物元数据仿真系统、图形组态应用系统和 VR 虚拟体验系统，同时能够与学校现有的物联网硬件平台形成虚实结合，另外可选物联网传感协议仿真和信息安全仿真平台对协议和安全的模拟仿真。



本实验重点讲述硬件物元仿真系统软件：

硬件物元仿真平台 HSIMS 主要满足物联网硬件的数据仿真，基于智云物联网中间件平台框架的硬件系统可以轻松接入，模拟物联网物元的属性，根据脚本及配置产生物元数据、执行状态等，为物联网应用提供硬件数据源支撑。

硬件物元仿真平台可以模拟各种传感器、执行器等常用硬件，同时也可以对稀有贵重设备进行数据/状态模拟，大大降低开发成本，对复杂场合运维的设备(比如对人体危害领域)也能够进行有效的模拟，降低开发风险。

四、实验步骤

1. 实验部署

本实验基于 xLab 实验平台来构建一套智能家居系统，需要准备以下设备(可通过硬件物元仿真平台软件创建虚拟的硬件，也可以使用 xLab 实验平台真实的硬件)：

- 1) 采集类节点(Sensor-A):采集类传感器包括温湿度、光强、空气质量、气压高度、三轴、距离。

- 2) 控制类节点(Sensor-B)：控制类传感器包括风扇、步进电机、蜂鸣器、LED、RGB、继电器。
- 3) 安防类节点(Sensor-C)：安防类传感器包括火焰、光栅、燃气、人体红外、触摸、振动、霍尔。

2. 原型搭建

2.1 创建项目

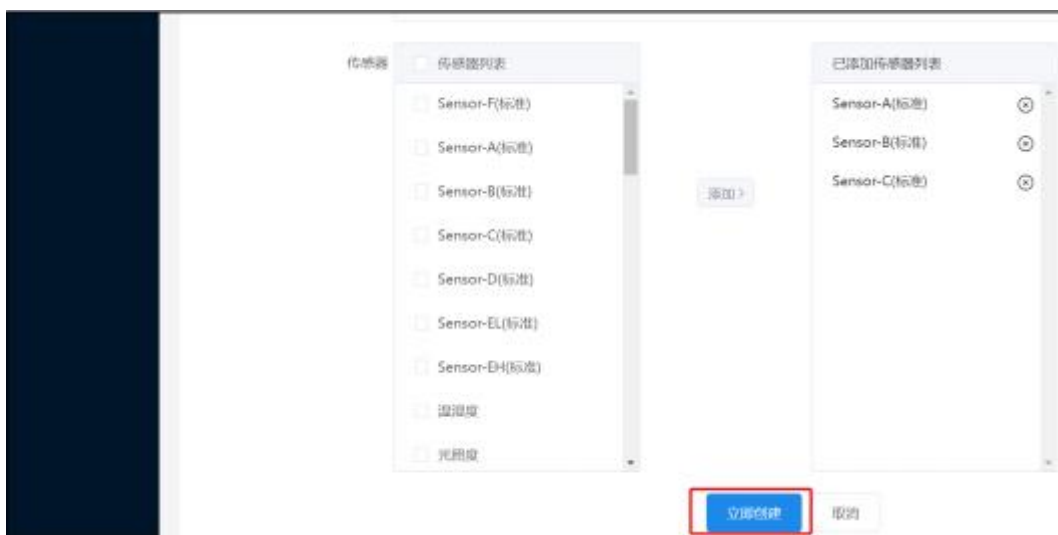
- 1) 用 Chrome 浏览器登录硬件物元仿真平台 HSIMS 软件平台(**说明：硬件物元仿真平台公有云版本地址：**<http://hsims.incloudlab.com/>), 并进行用户注册和登录。
- 2) 点击左侧导航栏的项目管理，点击项目列表的“新建项目”：



- 3) 名称：填写项目名称，如“智能家居项目”；用户 ID、用户密钥使用智云授权 ID 和密钥，服务器地址使用默认即可 (**智云帐号/密钥通过运行 windows 版本智云网关工具获取，DISK-Packages\50-智云物联工具智云网关。**)



- 4) 传感器：添加项目中需要使用的传感器 Sensor-A、Sensor-B、Sensor-C，添加到右边，即可完成传感器的添加。



5) 项目配置完成之后，点击“立即创建”按钮，即可创建“智能家居项目”虚拟仿真项目。

2.2 运行项目

1) 在“项目管理”菜单中找到新建的“智能家居项目”，单击进入项目。



2) 等待传感器列表加载完成，点击右上角的“启动”按钮，启动项目。

项目设置

返回

启动

项目信息

名称	ID	KEY	SERVER	二维码	网络拓扑图
智能家居项目			api.zhiiyun360.com:28083	点击扫描	点击进入

简介

3) 点击下方传感器的“开启”，大约 15S 后传感器数据开始上传并更新。

Sensor-A:

序号	名称	类别	类型	地址	编码	状态	操作
50682	Sensor-A(标准)	ZigBee	终端节点	01:12:4B:00:83:6B:0E:DD	12601	✕	开启 设置



距离	跌倒状态 (三轴)	大气压力	空气质量	光强	湿度	温度
0cm	0	0hPa	0	0Lux	0%	0°C

继电器K2	继电器K1					

序号	名称	类别	类型	地址	编码	状态	操作
50682	Sensor-A(标准)	ZigBee	终端节点	01:12:4B:00:83:6B:0E:DD	12601	✓	关闭



距离	跌倒状态 (三轴)	大气压力	空气质量	光强	湿度	温度
38cm	0	793.4hPa	9	14331Lux	42.97%	26.77°C

继电器K2	继电器K1					

Sensor-B:

序号	名称	类别	类型	地址	编码	状态	操作
50683	Sensor-B(标准)	ZigBee	终端节点	01:12:4B:00:F8:59:A1:98	12602	✕	开启 设置



继电器K2	继电器K1	LED2	LED1	风扇	窗帘 (步进电机)	三色灯



Sensor-C:



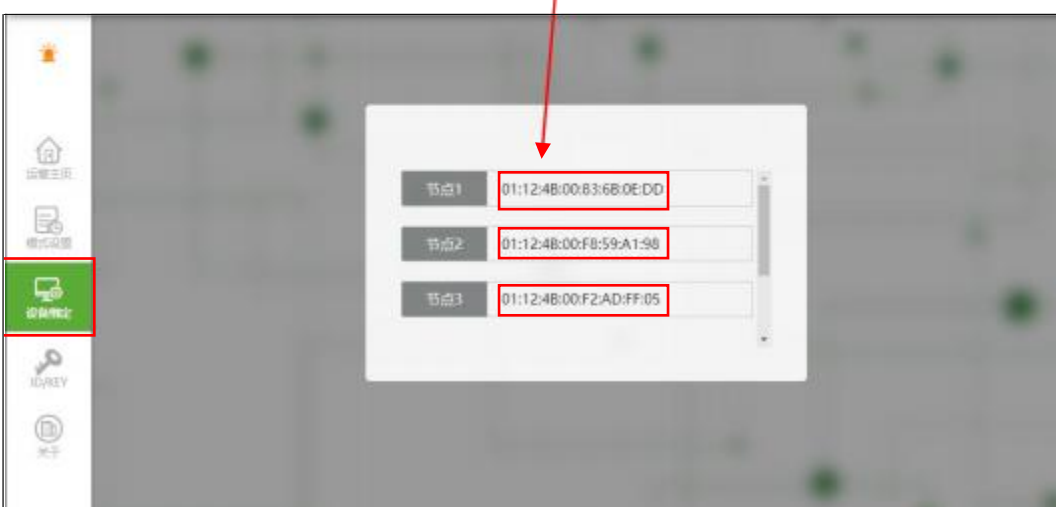
3. 应用体验

打开本实验目录下的工程 “xLabDemo-web”，用Chrome 浏览器打开文件下的 index.html。

1) 打开网页后点击 ID/KEY 页面，填写授权的 ID/KEY，服务器地址(Server) 为 api.zhiyun360.com，然后点击连接：



2) 连接成功后，点击设备绑定，在节点地址后的文本框中填入节点的 MAC 地址(地址要和智云硬件物元仿真平台的保持一致)：



3) 点击左侧导航栏的“运营首页”，进入主界面，主界面中从上至下分为 3 行，每行分别对应 Sensor-A、Sensor-B、Sensor-C 三个传感器，如下图所示(数据默认 30 秒上报一次，所以刚连接上时数据都为0，等待 30 秒后就可以看到各个传感器的数据，也可以通过点击色块来控制传感器和查询传感器数据)：



4) 主界面的第一行的 5 个色块用来显示 Sensor-A 的数据，分别对应 Sensor-A 传感器上的温湿度、光强、空气质量、气压、跌倒 5 种数据。第二行的 5 个色块用来显示 Sensor-B 传感器上的各个开关部件，分别是窗帘、风扇、LED 灯、空调、加湿器，对应 Sensor-B 传感器板上步进电机、风扇、LED灯、继电器(接口)。第三行的 5 个色块用来显示 Sensor-C 传感器上的各个安防部件，分别是燃气值、火焰、光栅、霍尔(门磁)、人体。

5) xLabDemo 有两种模式，手动模式和自动模式。

手动模式下，能够通过点击软件主界面上的控制类色块，控制 Sensor-B 传感器上的控制类元部件；自动模式下，设定光照、温度、空气质量、湿度四个阈值，来自动控制控制类传感器 Sensor-B 板上的窗帘(步进电机)、空调(继电器)、排风扇(风扇)、加湿器(继电器)。

点击左侧“模式设置”导航栏，进入模式设置界面。点击界面正中间的自动/手动圆形模式切换按钮，切换软件当前的模式。

当圆形模式切换按钮显示“手动”时，表明软件处于手动模式，通过点击“运营主页”第二行的控制类色块，可以手动控制 Sensor-B 传感器板上的控制类元部件；处于手动模式下，阈值设置滑条处于灰色不可操作状态。



当圆形模式切换按钮上显示“自动”时，表示软件处于自动模式，此时“运营主页”第二行的控制类色块不能控制，这时可以滑动滑条，来设置光照阈值、温度阈值、空气质量阈值、湿度阈值。其中光照度、温度、湿度是由最低阈值和最高阈值组成的阈值区间。



五、常见问题

1. 项目案例的运营主页传感器数据上传不了时，需要检查 MAC 地址是否与智云硬件物元仿真平台的保持一致。

2 智云通信协议

一、实验目的

1. 了解 ZXBee 智云通信协议。
2. 掌握 ZXBee 协议格式定义。
3. 掌握智云平台传感器节点协议设定。

二、实验环境

硬件环境:	PC 机 Pentium 处理器双核 2GHz 以上, 内存 4GB 以上
操作系统:	Windows7 64 位及以上操作系统
开发软件:	智云硬件物元仿真平台 HSIMS、ZCloudWebTools

三、实验内容

1. 通信协议

1.1 协议说明

智云平台支持物联网无线传感网数据的接入, 并定义了物联网数据通信的规范。智云平台采用轻量级的 ZXBee 通信协议, 它采用 json 数据格式, 更加清晰易懂。

ZXBee 数据通信协议对物联网整个项目从底层到上层的数据段作出了定义, 该协议有以下特点:

- 1) 数据格式的语法简单, 语义清晰, 参数少而精;
- 2) 参数命名合乎逻辑, 见名知义, 变量和命令的分工明确;
- 3) 参数读写权限分配合理, 可以有效抵抗不合理的操作, 能够在最大程度上确保数据安全;
- 4) 变量能对值进行查询, 可以方便应用程序调试;
- 5) 命令是对位进行操作, 能够避免内存资源浪费。

总之, ZXBee 数据通信协议在物联网无线传感网中值得应用和推广, 老师和学生也容易在其基础上根据需求进行定制、扩展和创新。

1.2 协议详解

● 通信协议数据格式

通信协议数据格式: {[参数]=[值],[参数]=[值],……}

- A. 每条数据以“{”作为起始字符;
- B. “{”内参数多个条目以“,”分隔;
- C. 示例: {CD0=1,D0=?}

注: 通信协议数据格式中的字符均为英文半角符号。

● 通信协议参数说明

通信协议参数说明如下:

- A. 参数名称定义为:
 - a) 变量: A0~A7、D0、D1、V0~V7;
 - b) 命令: CD0、OD0、CD1、OD1;

- c) 特殊参数: ECHO、TYPE、PN、PANID、CHANNEL。
- B. 变量可以对值进行查询, 示例: {A0=?}
- C. 变量 A0~A7 在物联网云数据中心可以存储保存为历史数据;
- D. 命令是对位进行操作。

● 通信协议参数详解

协议具体参数解释如下:

1) A0~A7: 用于传递传感器数值或者携带的信息量, 权限为只能通过赋值“?”来进行查询当前变量的数值, 支持上传到物联网云数据中心存储, 示例如下:

A. 温湿度传感器采用 A0 表示温度值, A1 表示湿度值, 数值类型为浮点型 0.1 精度;

B. 人体红外传感器采用 A0 表示警报状态, 数值类型为整型, 固定为 0(未检测到人体) 或者 1(检测到人体);

2) D0: D0 的 Bit0~Bit7 分别对应 A0~A7 的状态(是否主动上传状态), 权限为只能通过赋值“?”来进行查询当前变量的数值, 0 表示禁止上传, 1 表示允许主动上传, 示例如下:

A. 温湿度传感器 A0 表示温度值, A1 表示湿度值, D0=0 表示不上传温度和湿度信息, D0=1 表示主动上传温度值, D0=2 表示主动上传湿度值, D0=3 表示主动上传温度和湿度值;

B. 人体红外传感器采用 A0 表示警报状态, D0=0 表示不检测人体, D0=1 表示实时检测人体;

3) CD0/OD0: 对 D0 的位进行操作, CD0 表示位清零操作, OD0 表示位置一操作, 示例如下:

A. 温湿度传感器 A0 表示温度值, A1 表示湿度值, CD0=1 表示关闭 A0 温度值的主动上报;

B. 人体红外传感器采用 A0 表示警报状态, OD0=1 表示开启人体红外监测, 当检测到人体时, 会主动上报 A0 的数值。

4) D1: D1 表示控制编码, 权限为只能通过赋值“?”来进行查询当前变量的数值, 用户根据传感器属性来自定义功能, 示例如下:

A. RGB: D1 的 Bit0~Bit1 表示 RGB 三色灯的颜色状态, 例如: D1=0 表示 RGB 处于关闭状态, D1=1 表示红色, D1=2 表示绿色, D1=3 表示蓝色;

B. 继电器: D1 的 Bit6~Bit7 表示各路继电器状态, 例如: D1=0 关闭两路继电器 S1 和 S2, D1=64 开启继电器 S1, D1=128 开启继电器 S2, D1=192 开启两路继电器 S1 和 S2;

5) CD1/OD1: 对 D1 的位进行操作, CD1 表示位清零操作, OD1 表示位置一操作。

6) V0~V7: 用于表示传感器的参数, 用户根据传感器属性自定义功能, 权限为可读写, 示例如下:

A. V0 表示自动上传数据的时间间隔;

B. 语音合成: V1 表示需要合成的语音字符。

7) 特殊参数: ECHO、TYPE、PN。

ECHO: 用于检测节点在线的指令, 将发送的值进行回显, 比如发送: {ECHO=test}, 若节点在线则回复数据: {ECHO=test}。

TYPE: 表示节点类型, 该信息包含了节点类别、节点类型、节点名称, 权限为只能通过赋值“?”来进行查询当前值。TYPE 的值由 5 个 ascii 字节表示, 例如: 1 1 001, 第 1 字节表示节点类别 (1: ZigBee, 2: RF433, 3: Wi-Fi, 4: BLE, 5: IPv6, 6: LTE, 7: NB-IOT, 8: LoRa, 9: 其他); 第 2 字节表示节点类型(0: 汇集节点, 1: 路由/中继节点, 2: 终端节点); 第 3, 4, 5 字节合起来表示节点名称(编码用户自定义)。

PN: 表示节点的上行节点地址信息和所有邻居节点地址信息, 权限为只能通过赋值“?”来进行查询当前值。

PN 的值为上行节点地址和所有邻居节点地址的组合。其中每 4 个字节表示一个节点地址后 4 位, 第一个 4 字节表示该节点上行节点后 4 位, 第 2-n 个 4 字节表示其所有邻居节点地址后 4 位。

1.3 协议定义

xLab 实验平台的 ZXBee 通信协议参数定义如下表所示:

节点名称	TYPE	参数	含义	读写权限	说明
采集类 Sensor-A	601	A0	温度	R	温度值, 浮点型: 0.1 精度, -40.0~105.0, 单位: °C
		A1	湿度	R	湿度值, 浮点型: 0.1 精度, 0~100, 单位: %Rh
		A2	光强	R	光强值, 浮点型: 0.1 精度, 0~65535, 单位: Lux
		A3	空气质量	R	空气质量值, 表征空气污染程度, 整型, 0~20000, 单位: ppm
		A4	大气压力	R	大气压力值, 浮点型: 0.1 精度, 800.0~1200.0, 单位: hPa
		A5	跌倒状态	R	通过三轴传感器计算出跌倒状态, 0 表示未跌倒, 1 表示跌倒
		A6	距离	R	距离值, 浮点型: 0.1 精度, 10.0~80.0, 单位: cm
		A7	语音识别码	-	语音识别码, 整型: 1~49(主动上报, 不可查询)
		D0(OD0/CD0)	上报状态	RW	D0 的 Bit0~Bit7 分别代表 A0~A7的上报状态, 1 表示主动上报, 0为不上报
		D1(OD1/CD1)	继电器	RW	D1 的 Bit6~Bit7 分别代表继电器K1、K2 的开关状态, 0 表示断开, 1 表示吸合
		V0	上报时间间隔	RW	A0~A7、D1 主动上报时间间隔, 默认 30, 单位: S
控制类 Sensor-B	602	D1(OD1/CD1)	RGB	RW	D1 的 Bit0~Bit1 代表 RGB 三色灯的颜色状态 RGB: 00(关), 01(R), 10(G), 11(B)
		D1(OD1/CD1)	步进电机	RW	D1 的 Bit2 分别代表电机的正反转转动状态, 0 正转(5s 后停止), 1 反转(5s 后反转)
		D1(OD1/CD1)	风扇 / 蜂鸣器	RW	D1 的 Bit3 代表风扇/蜂鸣器的开关状态, 0 表示关闭, 1 表示打开
		D1(OD1/CD1)	LED	RW	D1 的 Bit4、Bit5 代表 LED1/LED2的开关状态, 0 表示关闭, 1 表示打开
		D1(OD1/CD1)	继电器	RW	D1 的 Bit6、Bit7 分别代表继电器K1、K2 的开关状态, 0 表示断开, 1 表示吸合
		V0	上报间隔	RW	A0~A7、D1 值的循环上报时间间隔
安防类 Sensor-C	603	A0	人体 / 触摸	R	人体红外/触摸传感器状态, 0 或 1 变化: 1 表示人体活动/触摸动作

	A1	震动	R	震动状态, 0 或 1 变化: 1 表示检测到震动
	A2	霍尔	R	霍尔状态, 0 或 1 变化: 1 表示检测到磁场
	A3	火焰	R	火焰状态, 0 或 1 变化: 1 表示检测到明火
	A4	燃气	R	燃气泄漏状态, 0 或 1 变化: 1 表示检测到燃气泄漏
	A5	光栅	R	光栅(红外对射) 状态值, 0 或 1 变化, 1 表示检测到阻挡
	D0(OD0/CD0)	上报状态	RW	D0 的 Bit0~Bit7 分别代表 A0~A7 的上报状态, 1 表示主动上报, 0 为不上报
	D1(OD1/CD1)	继电器	RW	D1 的 Bit6~Bit7 分别代表继电器 K1、K2 的开关状态, 0 表示断开, 1 表示吸合
	V0	上报间隔	RW	A0~A7、D1 值的循环上报时间间隔
	V1	语音合成数据	W	文字的 Unicode 编码

1.4 协议分析

● Sensor-A:

Sensor-A 节点能够准确测量室内温湿度、光强值、控制质量、大气压力等数据。

根据前面的智云平台 ZXBee 协议定义, Sensor-A 节点支持以下操作:

发送命令	接收结果	含义
{A0=?}	{A0=XX}	温度值, 浮点型, 0.1 精度, 单位: °C
{A1=?}	{A1=XX}	湿度值, 浮点型: 0.1 精度, 单位: %Rh
{A2=?}	{A2=XX}	光强值, 浮点型: 0.1 精度, 单位: Lux
{A3=?}	{A3=XX}	空气质量值, 整型, 单位: ppm
{A4=?}	{A4=XX}	大气压力值, 浮点型: 0.1 精度, 单位: hPa
{A5=?}	{A5=XX}	跌倒状态, 0 表示未跌倒, 1 表示跌倒
{A6=?}	{A6=XX}	距离值, 浮点型: 0.1 精度, 单位: cm
{D0=?}	{D0=XX}	D0 的 bit0~bit7 对应 A0~A7 的主动上报使能, 0 表示允许主动上报, 1 表示不允许主动上报
{D1=?}	{D1=XX}	D1 的 bit6~bit7 表示继电器 k1、k2 的开关状态, 0 表示断开, 1 表示吸合
{A0=?,A1=?,A2=?,A3=?,A4=?,A5=?,A6=?}	回复 A0 ~ A6、	查询所有数据

D0=?,D1=?}	D0~D1 所有数据	
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: Bit6: 继电器 K1 开关, 0 表示关, 1 表示开 Bit7: 继电器 K2 开关, 0 表示关, 1 表示开
{V0=?} {V0=XXX,V0=?}	{V0=XXX}	查询/设置 A0~A7、D1 主动上报时间间隔, 默认为 30, 单位: S

● **Sensor-B:**

根据前面的智云平台 ZXBee 协议定义, Sensor-B 节点支持以下操作:

发送命令	接收结果	含义
{D1=?}	{D1=XX}	查询所有数据
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: bit0~bit1 代表 RGB 三色灯的颜色状态: 00(关), 01(R), 10(G), 11(B)
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: bit2: 电机正反转状态, 0 表示正转(5S 后停止), 1 表示反转(5S 后反转)
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: bit3: 风扇 /蜂鸣器的开关, 0 表示关, 1 表示开
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: bit4: LED1 的开关, 0 表示关, 1 表示开 bit5: LED2 的开关, 0 表示关, 1 表示开
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: bit6: 继电器 K1 的开关, 0 表示断开, 1 表示吸合 bit7: 继电器 K2 的开关, 0 表示断开, 1 表示吸合
{V0=?} {V0=XXX,V0=?}	{V0=XXX}	查询/设置 D1 主动上报时间间隔, 默认为 30, 单位: S

● **Sensor-C:**

根据前面的智云平台 ZXBee 协议定义, Sensor-C 节点支持以下操作:

发送命令	接收结果	含义
{A0=?}	{A0=XX}	人体红外/触摸传感器状态, 1 表示人体活动/触摸动作, 0 表示未检测到
{A1=?}	{A1=XX}	震动状态, 1 表示检测到震动, 0 表示未检测到震动

{A2=?}	{A2=XX}	霍尔状态, 1 表示检测到磁场, 0 表示未检测到磁场
{A3=?}	{A3=XX}	火焰状态, 1 表示检测到明火, 0 表示未检测到明火
{A4=?}	{A4=XX}	燃气泄漏状态, 1 表示检测到燃气泄漏, 0 表示未检测到燃气泄漏
{A5=?}	{A5=XX}	光栅(红外对射) 状态值, 1 表示检测到阻挡, 0 表示未检测到阻挡
{D0=?}	{D0=XX}	D0 的 bit0~bit7 对应 A0~A7 的主动上报使能, 0 表示允许主动上报, 1 表示不允许主动上报
{D1=?}	{D1=XX}	D1 的 bit6~bit7 表示继电器 k1、k2 的开关状态, 0 表示断开, 1 表示吸合
{A0=?,A1=?,A2=?,A3=?,A4=?,A5=?, D0=?,D1=?}	回复 A0 ~ A5、 D0~D1 所有数据	查询所有数据
{CD1=XXX,D1=?} {OD1=XXX,D1=?}	{D1=XX}	CD1 表示位清零, OD1 表示位置一; D1 的 bit 位控制设备: Bit6: 继电器 K1 开关, 0 表示关, 1 表示开 Bit7: 继电器 K2 开关, 0 表示关, 1 表示开
{V0=?} {V0=XXX,V0=?}	{V0=XXX}	查询/设置 A0~A7、D1 主动上报时间间隔, 默认为 30, 单位: S

2. 调试工具

2.1 工具介绍

智云数据调试工具 ZCloudWebTools (DISK-Packages\50-智云物联工具\调试工具\ZCloudWebTools\index.html) 是一款基于智云物联云平台的综合性调试工具, 能够进行网络拓扑、实时数据、历史数据、视频监控、自动控制、用户数据等功能进行测试。

1) 实时数据: 实时数据推送与采集测试工具。通过消息推送接口, 能够实时抓取项目上下行所有节点数据包, 支持通过命令对节点进行操作, 获取节点实时信息、控制节点状态等操作。

2) 历史数据: 数值/图片历史数据获取测试工具。能够接入到数据中心数据库, 对项目任意时间段历史数据进行获取, 支持数值型数据曲线图展示、JSON 数据格式展示, 同时支持摄像头抓拍的照片在曲线时间轴展示。

3) 网络拓扑: 网络拓扑图分析工具。能够实时接收并解析传感网络数据包, 将接收到的网络信息通过拓扑图的形式展示, 通过颜色对不同节点类型进行区分, 显示节点的 IEEE 地址。

4) 视频监控: 视频监控测试工具。支持对项目摄像头进行管理, 能够实时获取摄像头采集的画面, 并支持对摄像头云台进行控制, 支持上、下、左、右、水平巡航、垂直巡航等, 同时支持截屏操作。

5) 用户数据: 用户数据存储与查询测试工具。通过用户数据库接口, 支持在该项目下存取用户数据, 以 key-value 键值对的形式保存到数据中心服务器, 同时支持通过 key 获取到其对应的 value 数值。

6) 自动控制: 自动控制模块测试工具。通过内置的逻辑编辑器实现复杂的自动控制逻辑, 包括触发器(传感器类型、定时器类型)、执行器(传感器类型、短信类型、摄像头类型、任务类型)、执行任务、执行记录四大模块, 每个模块都具有查询、创建、删除功能。



2.2 工具使用

ZCloudWebTools 工具提供实时数据功能，能够通过智云平台向传感器节点发送命令，也能够接收传感器节点主动上报的数据，可用于数据分析和调试。

1) 在实际使用时，首先将 ZCloudWebTools 工具切换到实时数据选项，然后将网关的用户账号和用户密钥填写到相应的账号和密钥位置，点击连接即可。



2) 连接成功之后，在地址处填写传感器节点的 MAC 地址，在数据处填写 ZXBee协议定义的命令(以 Sensor-B 为例)，然后点击发送，下方会显示传感器节点返回数据，如下图所示：



四、实验步骤

1. 实验部署

本实验基于 xLab 实验平台来构建一套智能家居系统来理解智云 ZXBee 通信协议，需要准备以下设备(可通过硬件物元仿真平台软件创建虚拟的硬件，也可以使用 xLab 实验平台真实的硬件)：

- 1) 采集类节点(Sensor-A):采集类传感器包括温湿度、光强、空气质量、气压高度、三轴、距离。
- 2) 控制类节点(Sensor-B):控制类传感器包括风扇、步进电机、蜂鸣器、LED、RGB、继电器。
- 3) 安防类节点(Sensor-C):安防类传感器包括火焰、光栅、燃气、人体红外、触摸、振动、霍尔。

2. 原型搭建

2.1 创建项目

可参考实验 1 的原型仿真步骤来创建项目。

2.2 运行项目

可参考实验 1 的原型仿真步骤来运行项目。

3. 数据调试

在硬件物元仿真平台 HSIMS 软件平台运行项目之后，可以使用 ZCloudWebTools 工具进行数据调试:

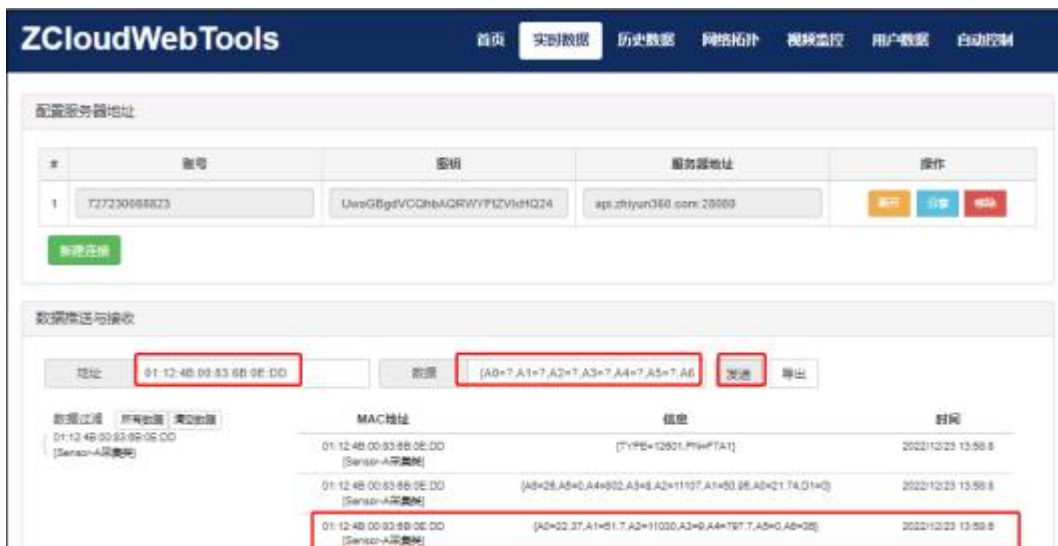
3.1 采集类 Sensor-A 传感器

1) 将 ZCloudWebTools 工具切换到实时数据选项，然后将网关的用户账号和用户密钥填写到相应的账号和密钥位置，点击连接，如下图所示：



2) 连接成功后, 在 ZCloudWebTools 工具的地址处填写 Sensor-A 的 MAC 地址, 在数据处填写 Sensor-A 节点调试命令, 输入{A0=?,A1=?,A2=?,A3=?,A4=?,A5=?,A6=?}命令可以查询 Sensor-A 的温度、湿度、光强、空气质量、大气压力、跌倒状态、距离数据, 点击发送, 如下图所示:

注意: 填写的MAC 地址和调试命令不能出现空格。



3) 在 ZCloudWebTools 工具的数据处, 可以单个查询任意一个传感器数据信息, 例如输入{A0=?}命令查询温度值, 点击发送, 如下图所示。



4) 根据 Sensor-A 节点的协议表, 通过命令调试理解 ZXBee 协议。(完成对 Sensor-A 节点的操作之

后，可以选择关闭 Sensor-A 传感器，方便后边对其他节点操作的观察)

3.2 控制类 Sensor-B 传感器

1) 在 ZCloudWebTools 工具的地址处填写 Sensor-B 的 MAC 地址, 在数据处填写 Sensor-B 节点调试命令, 输入{OD1=8,D1=?}命令, 点击发送, 可以打开风扇, 如下图所示:



此时观察智云硬件物元仿真平台 HSIMS 的 Sensor-B 栏, 风扇图标在一直转动, 表明风扇处于开启状态:



2) 输入{CD1=8,D1=?}命令, 点击发送, 关闭风扇, 如下图所示:



此时观察智云硬件物元仿真平台 HSIMS 的 Sensor-B 栏, 风扇图标停止转动, 表明风扇处于关闭状态。

3) 在 ZCloudWebTools 工具的数据处输入{OD1=48,D1=?}命令, 点击发送, 打开 LED1 和 LED2, 如下图所示。



此时观察智云硬件物元仿真平台 HSIMS 的 Sensor-B 栏, LED1 和 LED2 图标点亮, 表明 LED 灯处于二级开启状态。



4) 输入{CD1=48,D1=?}命令, 点击发送, 关闭 LED1 和 LED2, 如下图所示:



此时观察智云硬件物元仿真平台 HSIMS 的 Sensor-B 栏, LED1 和 LED2 图标熄灭, 表明 LED 灯处于关闭状态。

5) 根据 Sensor-B 节点的协议表, 通过命令调试理解 ZXBee 协议。(完成对 Sensor-B 节点的操作之后, 可以选择关闭 Sensor-B 传感器, 方便后边对其他节点操作的观察)

3.3 安防类 Sensor-C 传感器

检测到报警后, 节点会立刻收到报警数据, 不需要主动去查询。

例如: 当检测到火焰时, 观察智云硬件物元仿真平台 HSIMS 的 Sensor-C 栏, 火焰状态为 1, 表示检测到明火。



根据 Sensor-C 节点的协议表，通过收到的数据信息去理解 ZXBee 协议。

五、常见问题

1. ZXBee 协议采用 JSON 格式通信，注意中英文大小写的格式。

3 智云硬件接口

一、实验目的

1. 掌握智云硬件驱动开发。
2. 掌握智云无线传感网协议。
3. 掌握 ZXBee协议在硬件层的应用。

二、实验环境

硬件环境:	PC 机 Pentium 处理器双核 2GHz 以上, 内存 4GB 以上
操作系统:	Windows7 64 位及以上操作系统
开发软件:	SmartRF Flash Programmer 、 IAR for 8051 、 xLabTools 、 ZXBeeGW4 、 ZCloudWebTools
实验器材:	ZigBee 节点、ZigBee 协调器
实验配件:	SmartRF04EB 下载器、Mini USB 线、xLab 实验基板、12V 电源

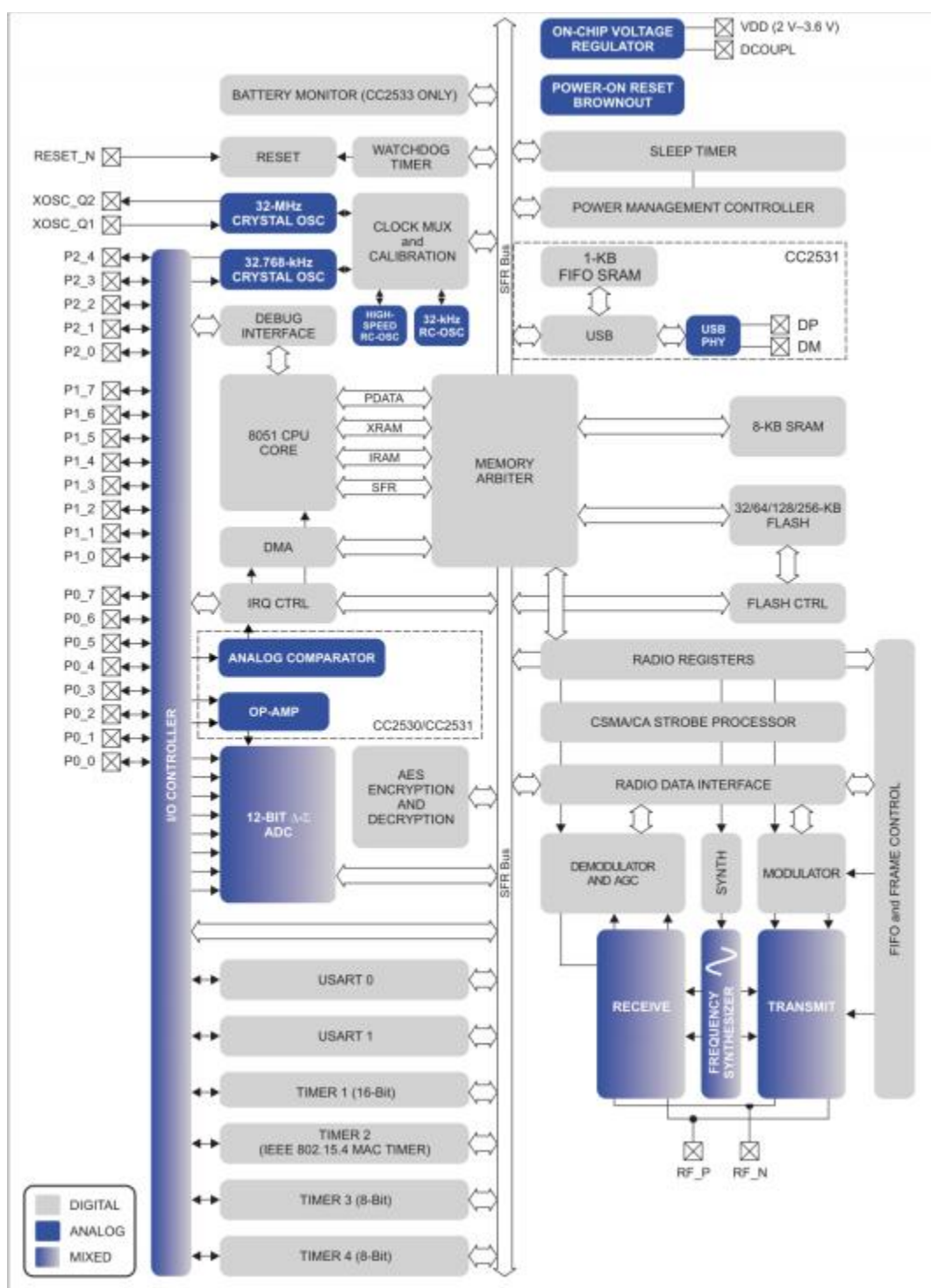
三、实验内容

1. 硬件框架

CC2530 是 TI 公司用于 2.4-GHz IEEE 802.15.4、ZigBee 和 RF4CE 应用的一个真正的片上系统(SoC) 解决方案, 是 ZigBee 无线传感网学习的依托平台。CC2530 采用集成 MCU+射频收发模块的SoC 设计方式, 而且这种设计方式能实现节点的更微小化和极低的功耗。开发者可以采用 TI 公司的CC2530 为核心来设计传感器节点。

CC2530 包括了极好性能的一流的 RF 收发器, 工业标准增强型 8051MUC, 系统中可编程的闪存, 8KB RAM 以及许多其它功能强大的特性, 具有不同的运行模式, 使得它尤其适合超低功耗要求的系统, 结合 TI 的业界领先的黄金单元 ZigBee 协议栈(Z-Stack), 提供了一个强大和完整的 ZigBee 解决方案。

CC2530 微处理器有着丰富的片上外设和内存资源, 除了使用增强型 8051 内核外还有众多的总线结构上的优化。CC2530 微处理器结构框图如下图所示。



由上图可知，CC2530 硬件结构大致可以分为三个部分：CPU 和内存相关的模块；外设、时钟和电源管理相关的模块；无线电相关的模块。

1) CPU 与内存

CC2530 设备系列使用的 8051CPU 内核是一个单周期的 8051 兼容内核。它有三个不同的存储器访问总线(SFR、DATA 和CODE/XDATA),以单周期访问 SFR、DATA 和主 SRAM。它还包括一个调试接口和一个 18 位输入的扩展中断单元。

2) 时钟与电源管理

数字内核和外设由一个 1.8-V 低差稳压器供电。另外 CC2530 包括一个电源管理功能，可以实现使用不同供电模式的长电池寿命的低功耗应用运行。系统兼有五种不同的复位源来复位设备。

3) 片上外设

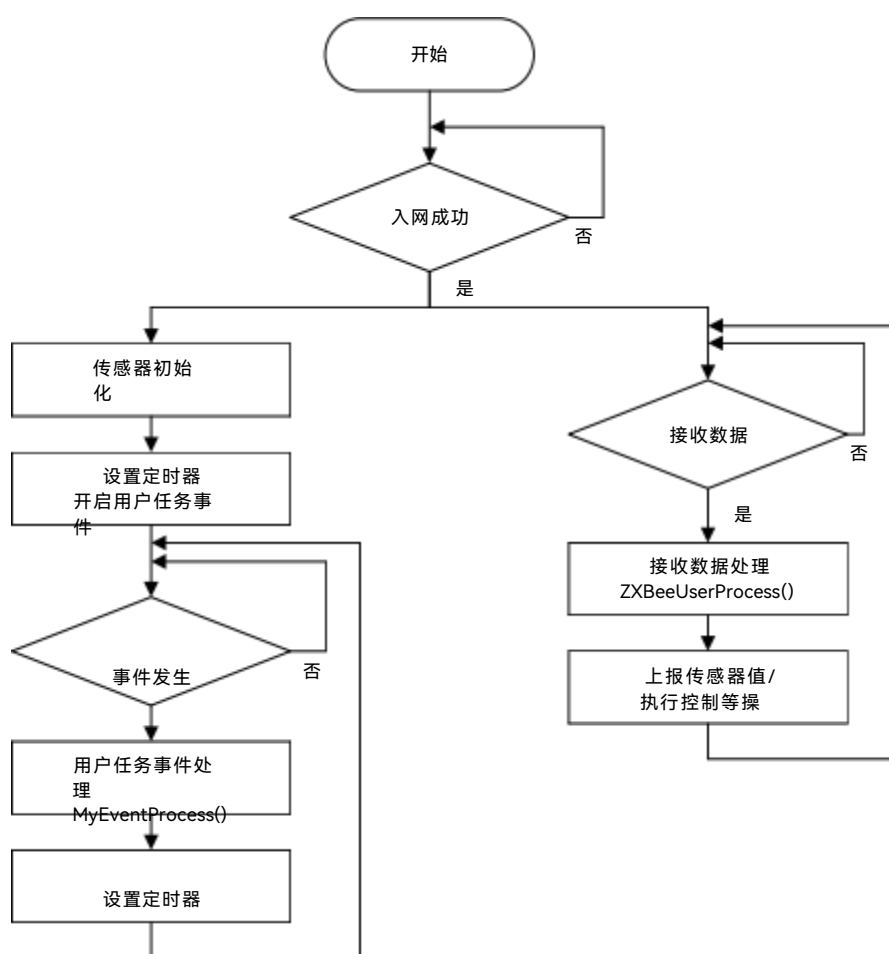
CC2530 包括许多不同的外设，如：I/O 控制器、DMA 控制器、定时器、ADC 外设、随机数发生器、AES 协处理 SDG、看门狗、串口(USART) 、

4) 无线射频收发器

CC2530 设备系列提供了一个 IEEE802.15.4 兼容无线收发器。RF 内核控制模拟无线模块。另外它提供了 MCU 和无线设备之间的一个接口，这使得可以发出命令、读取状态、自动操作和确定无线设备事件的顺序。无线设备还包括一个数据包过滤和地址识别模块。

2. 软件框架

无线节点采用 TI 公司的 CC2530 ZigBee 处理器，运行 ZStack 协议栈，它为 CC2530 节点提供基于OSAL 操作系统的无线自组网功能，并提供 sapi 框架进行应用的开发，sapi 接口实现了对应用的简单封装，开发者只需要实现部分接口函数即可完成整个节点程序的开发。详细的程序流程图如下：



其中 sapi 应用接口在 AppCommon.c 文件中实现，其中主要的几个函数如下：



智云平台为 ZigBee ZStack 协议栈上层应用提供分层的软件设计结构，将传感器的私有操作部分封装到 sensor.c 文件中，详细函数解释如下：

函数名称	函数说明
sensorInit()	传感器硬件初始化
sensorUpdate()	传感器数据定时上报
sensorCheck()	传感器报警状态实时监测
sensorControl()	传感器/执行器控制函数
ZXBeeUserProcess ()	解析接收到的传感器控制命令函数
MyEventProcess()	自定义事件处理函数，启动定时器触发事件 MY_REPORT_EVT

3. 示例场景

示例节点用于理解和掌握智云硬件程序的开发，该节点集成了采集、控制和安防三种类型传感器，具体内容如下。

1) 采集类传感器：集成光强、温度、湿度采集类传感器，根据设定的时间间隔定时上报实时的传感器数值，本示例程序通过随机数生成传感器数值。

2) 控制类传感器：集成继电器，通过控制编码来进行控制。

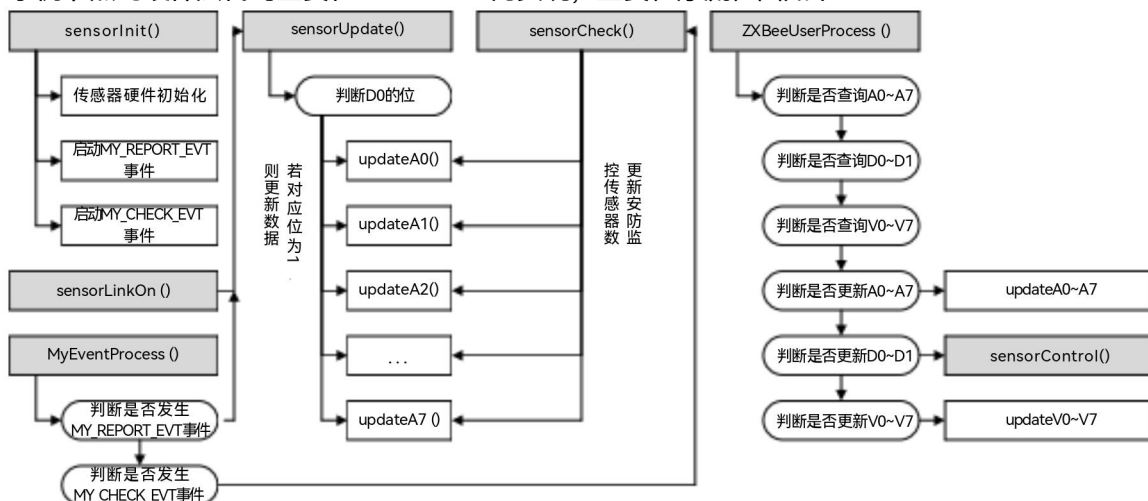
3) 安防类传感器：集成人体红外传感器，本示例程序通过随机数生成传感器数值。

设定的硬件协议如下：

节点名称	TYPE	参数	含义	读写权限	说明
示例节点	500	A0	光强值	R	光强值，整型：0~18000，单位：Lux
		A1	温度值	R	温度值，浮点型：0.1 精度，-40.0~70.0，单位：°C
		A2	湿度值	R	湿度值，浮点型：0.1 精度，0~100.0，单位：%RH
		A3	人体红外	R	监测是否周边有人体活动：1 表示检测到附近有人，0 表示未检测到有人体活动
		D0(OD0/CD0)	主动上报使能	RW	D0 的 bit0~bit7 对应A0~A7 主动上报使能，0 表示不允许主动上报，1 表示允许主动上报
		D1(OD1/CD1)	开关控制	RW	D1 的 bit 位控制设备： bit0~bit1：LED 照明控制，LED 关(00)、LED 一级光(01)、LED 二级光(10)、LED 三级光(11) bit2：电机开关控制，关闭电机(0)、打开电机(1)
		V0	主动上报时间间隔	RW	A0~A7 主动上报时间间隔，默认 30，单位：S

4. 关键代码

示例节点的硬件层代码主要在 sensor.c 内实现，主要程序流程图如下：



函数如下：

函数名称	函数说明
sensorInit()	传感器任务初始化：创建定时器、创建信号量、传感器硬件初始化
sensorLinkOn()	传感网入网成功后调用：更新传感器数据
sensorUpdate()	传感器A0~A7数据实时采集及数据上报
sensorCheck()	安防类传感器数据实时采集及数据上报

sensorControl()	根据D1的值，对控制器进行实际操作控制
ZXBeeUserProcess()	处理上层应用发过来的指令
updateA0~A7()	实时采集A0~A7传感器数据

程序实现过程如下：

1) 在 sensor.c 文件中，修改全局变量中 A0~A3 的数据类型：

```

/*****
* 全局变量
*****/

static uint8 D0 = 0xFF;           // 默认打开主动上报功能
static uint8 D1 = 0;             // 继电器初始状态为全关
static uint16 A0 = 0;            // A0 存储光强值
static float A1 = 0.0;           // A1 存储温度值
static float A2 = 0.0;           // A2 存储湿度值
static uint8 A3 = 0;             // A3 表示人体红外状态值
static uint16 V0 = 30;           // V0 设置为上报时间间隔，默认为 30s

```

2) 在 sensor.c 文件中实现传感器初始化 sensor_init()。首先初始化传感器，然后启动MY_REPORT_EVT 事件定时上报数据,启动 MY_CHECK_EVT 事件定时监测:此 Demo 中不使用实体传感器，通过随机数模拟数据变化，所以此处只初始化随机数发生器即可。代码实现如下：

```

/*****
* 名称: sensor_init()
* 功能: 传感器初始化
* 参数: 无
* 返回: 无
* 修改:
* 注释:
*****/

void sensorInit(void)
{
    printf("sensor_init...\r\n");
    // 初始化燃气传感器代码
    APCFG |= 0x20;
    // 模拟 I/O 使能
    POSEL |= 0x20;
    // 端口 0_5 功能选择外设功能
    P0DIR &= ~0x20;
    // 设置输入模式

    // 初始化继电器代码
    POSEL &= ~0xC0;
    // 配置管脚为通用 IO 模式
    P0DIR |= 0xC0;
    // 配置控制管脚为输入模式

    // 启动定时器，触发传感器上报数据事件: MY_REPORT_EVT
    osal_start_timerEx(sapi_TaskID, MY_REPORT_EVT, (uint16)((osal_rand()%10) * 1000));
}

```

据上报。代码实现如下：

```

/*****
* 名称: sensorLinkOn()
* 功能: 传感器节点入网成功调用函数
* 参数: 无
* 返回: 无
* 修改:
* 注释:
*****/

void sensorLinkOn(void)
{
    printf("sensorLinkOn...\r\n");
    sensorUpdate();
}

```

4) 在 sensor.c 文件中实现定时器超时回调函数 MyEventProcess(), 每隔 V0 秒触发一次数据上报, 每隔 1 秒监测一次安防事件。代码实现如下:

```

/*****
* 名称: void MyEventProcess()
* 功能: 触发定时器
* 参数: event 表示事件名称
* 返回: 无
* 修改:
* 注释:
*****/

void MyEventProcess( uint16 event )
{
    if (event & MY_REPORT_EVT) {
        sensorUpdate();
        //启动定时器, 触发事件: MY_REPORT_EVT
        osal_start_timerEx(sapi_TaskID, MY_REPORT_EVT, V0*1000);
    }
    if (event & MY_CHECK_EVT) {
        sensorCheck();
        // 启动定时器, 触发事件: MY_CHECK_EVT
        osal_start_timerEx(sapi_TaskID, MY_CHECK_EVT, 1000);
    }
}

```

5) 在 sensor.c 文件中实现传感器查询、上报函数 sensorUpdate()。根据 D0 的 bit0-bit3 的值, 决定是否将 A0-A3 的值打包进发送字符串; 打包完成后, 通过 ZXBeelnfSend 函数, 将数据发送到智云网关。代码实现如下:

```

/*****
* 名称: sensorUpdate()
* 功能: 查询各传感器值, 达到 V0 设定时间时上报
* 参数: 无
* 返回: 无
* 修改:

```

```

* 注释:
*****/

void sensorUpdate(void)
{
    char pData[16];
    char *p = pData;

    ZXBeeBegin();
    // 智云数据帧格式包头

    // 根据 D0 的位状态判定需要主动上报的数
    值if ((D0 & 0x01) == 0x01){
        // 若光强上报允许, 则 pData 的数据包中
        添加温度数据
        updateA0();
        sprintf(p, "%u", A0);
        ZXBeeAdd("A0", p);
    }
    if ((D0 & 0x02) == 0x02){
        // 若温度上报允许, 则 pData 的数据包中
        添加湿度数据
        updateA1();
        sprintf(p, "%.1f", A1);
        ZXBeeAdd("A1", p);
    }
    if ((D0 & 0x04) == 0x04){
        // 若湿度上报允许, 则 pData 的数据包中
        添加温度数据
        updateA2();
        sprintf(p, "%.1f", A2);
        ZXBeeAdd("A2", p);
    }
    if ((D0 & 0x08) == 0x08){
        // 若人体红外上报允许, 则 pData 的数据
        包中添加湿度数据
        updateA3();
        sprintf(p, "%u", A3);
        ZXBeeAdd("A3", p);
    }
    sprintf(p, "%u", D1);
    ZXBeeAdd("D1", p);
    // 上报控制编码

```

```

    p = ZXBeeEnd();
    // 智云数据帧格式包尾
    if (p != NULL) {
        ZXBeeInfSend(p, strlen(p));          // 将需要上传的数据进行打包操作, 并通过 zb_SendDataRequest()发送到
        协
    }
    调器
    printf("sensorUpdate... %s\r\n", p);
}

```

sensorUpdate 函数中, 用到了 updateA0、updateA1、updateA2、updateA3 函数, 来获取最新的传感器值。此 Demo 中没有使用实体传感器, 各传感器的值均通过随机数来生成。rand()函数会生成一个 0~0x3FFFFFFF 之间的随机数, 通过取余运算, 获取指定范围内的随机数。代码实现如下:

```

/*****
* 名称: updateA0()
* 功能: 更新 A0 值
* 参数:
* 返回:
* 修改:
* 注释:

*****/

void updateA0(void)
{
    // 光强值, 整型: 0~18000 , 单位: Lux A0
    = (uint16)rand() % 300;
}

/*****
* 名称: updateA1()
* 功能: 更新 A1 值
* 参数:
* 返回:
* 修改:
* 注释:

*****/

void updateA1(void)
{
    // 温度值, 浮点型: 0.1 精度, -40.0~70.0, 单位: ° C
    A1 = (245 + rand()%40) / 10.f;          // 温度值在
    24.5 和 28.4 之间取随机数
}

/*****
* 名称: updateA2()
* 功能: 更新 A2 值
* 参数:
* 返回:

```

```

* 修改:
* 注释:
*****/

void updateA2(void)
{
    // 湿度值, 浮点型: 0.1 精度, 0~100.0, 单位: %RH
    A2 = (456 + rand()%40) / 10.f; // 湿度值在
    45.6 和 55.5 之间取随机数
}

/*****
* 名称: updateA3()
功能: 更新 A3 值
* 参数:
* 返回:
* 修改:
* 注释:
*****/

void updateA3(void)
{
    // 人体红外(0/1)
    uint8 val[10]={0,0,0,0,0,0,0,0,0,1};
    A3 = val[rand()%10]; //
    取 0-1 之间的随机数
}

```

6) 在 sensor.c 文件中实现安防类传感器检测函数 sensorCheck(), 每隔 3 秒判断 A3 的值, 为 1 时上报一次{A3=1}。代码实现如下:

```

/*****
* 名称: sensorCheck()
* 功能: 传感器监测, 安防类传感器异常时快速发送消息
* 参数: 无
* 返回: 无
* 修改:
* 注释:
*****/

void sensorCheck(void)
{
    static char lastA3=0;
    static uint32 ct3=0;
    char pData[16];
    char *p = pData;

    updateA3();
    ZXBeeBegin();

    if (lastA3 != A3 || (ct3 != 0 && osal_GetSystemClock() > (ct3+3000))) {
        if (D0 & 0x08) {
            sprintf(p, "%u", A3);
            ZXBeeAdd("A3", p);
        }
    }
}

```

```

        ct3 = osal_GetSystemClock();
    }
    if (A3 == 0) {
        ct3 = 0;
    }
    lastA3 = A3;
}

p = ZXBeeEnd();
if (p != NULL) {

```

```

    int len = strlen(p);
    ZXBeeInfSend(p, strlen(p));
    printf("sensorCheck... %s\r\n", p);
}
}

```

7) 在 sensor.c 文件中实现接收指令处理函数 ZXBeeUserProcess()。判断到指令标签为 A 且通道编号为 0~3(A0~A3) 时, 如果指令值为 i ?'(如: A0=?), 则上报对应通道的传感器值; 判断到指令标签为 D 且通道编号为 0~1(D0~D1) 时, 如果指令值为 i ?'(如: D0=?), 则上报对应参数的值; 判断到指令标签为 V 且通道编号为 0(V0) 时, 如果指令值为 i ?'(V0=?), 则上报 V0 值, 否则通过updateV0函数, 设置 V0 的值; 判断到指令标签为 OD 且通道编号为 0~1(OD0~OD1) 时, 通过 sensorControl 函数进行相关控制, 并更新 D0/D1 对应的 bit 位; 判断到指令标签为 CD 且通道编号为 0~1(CD0~CD1) 时, 通过 sensorControl 函数进行相关控制, 将 D0/D1 对应的 bit 位清零。代码实现如下:

```

/*****
* 名称: ZXBeeUserProcess()
* 功能: 处理上层应用发过来的指令
* 参数: ptag: 指令标识 D0, D1, A0 ...
*       pval: 指令值, “?”表示读取,
*       obuf: 指令处理结果存放地址
* 返回: >0 指令处理结果返回数据长度, 0: 没有返回数据, <0: 不支持指令。
* 修改:
* 注释:
*****/

int ZXBeeUserProcess(char *ptag, char *pval)
{
    int val;
    int ret = 0;
    char pData[16];
    char *p = pData;

    printf("ZXBeeUserProcess... %s=%s\r\n", ptag, pval);

    // 将字符串变量 pval 解析转换为整型变量赋值
    val = atoi(pval);
    // 控制命令解析
    if (0 == strcmp("CD0", ptag)){                                // 对 D0
        的位进行操作, CD0 表示位清零操作
        D0 &= ~val;
    }
    if (0 == strcmp("OD0", ptag)){                                // 对 D0 的
        位进行操作, OD0 表示位置一操作
        D0 |= val;
    }
    if (0 == strcmp("D0", ptag)){                                  // 查询上报使
        能编码
        if (0 == strcmp("?", pval)){

```

```

        ZXBeeAdd("D0", p);
    }
}
if (0 == strcmp("CD1", ptag)){
    D1 &= ~val;
    sensorControl(D1);
}
if (0 == strcmp("OD1", ptag)){
    D1 |= val;
    sensorControl(D1);
}
if (0 == strcmp("D1", ptag)){
    if (0 == strcmp("?", pval)){ r
        et = sprintf(p, "%u", D1);
        ZXBeeAdd("D1", p);
    }
}
if (0 == strcmp("A0", ptag)){
    if (0 == strcmp("?", pval)){
        updateA0();
        ret = sprintf(p, "%u", A0);
        ZXBeeAdd("A0", p);
    }
}
if (0 == strcmp("A1", ptag)){
    if (0 == strcmp("?", pval)){
        updateA1();
        ret = sprintf(p, "%.1f", A1);
        ZXBeeAdd("A1", p);
    }
}
if (0 == strcmp("A2", ptag)){
    if (0 == strcmp("?", pval)){
        updateA2();
        ret = sprintf(p, "%.1f", A2);
        ZXBeeAdd("A2", p);
    }
}
if (0 == strcmp("A3", ptag)){
    if (0 == strcmp("?", pval)){
        updateA3();
        ret = sprintf(p, "%u", A3);
        ZXBeeAdd("A3", p);
    }
}

```

作

作

// 对 D1 的位进行操作, CD1 表示位清零

// 处理执行命令

// 对 D1 的位进行操作, OD1 表示位置一

// 处理执行命令

// 查询执行器命令编

```

    }
}
if (0 == strcmp("V0", ptag)){
    if (0 == strcmp("?", pval)){
        ret = sprintf(p, "%u", V0);
        // 上报时间间隔ZXBeeAdd("V0", p);
    }else{
        updateV0(pval);
    }
}
}
return ret;
}

```

ZXBeeUserProcess 函数中，用到了 updateV0 函数，来设置 V0 的值，并更新定时器超时时间。首先通过 atoi 函数，获取要设置的 V0 值。代码实现如下：

```

/*****
* 名称: updateV0()
* 功能: 更新 V0 值
* 参数: val 表示
* 返回:
* 修改:
* 注释:
*****/

void updateV0(char *val)
{
    //将字符串变量 val 解析转换为整型变量赋值
    V0 = atoi(val);
    // 获取数据上报时间更改值
}

```

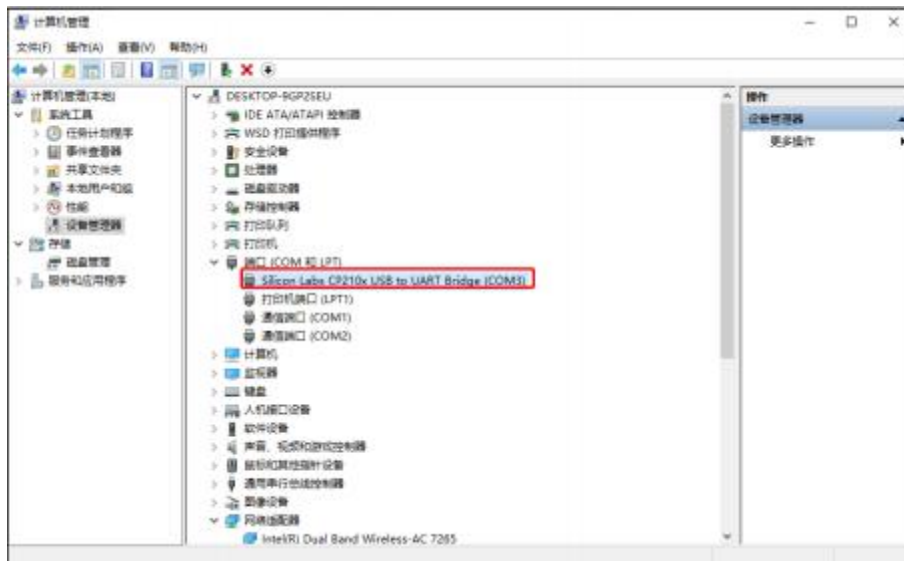
ZXBeeUserProcess 函数中，用到了 sensorControl 函数，进行修改 D1 对应的控制操作。D1 的 bit 位对应控制关系如下：bit0~bit1：LED 关(00)、LED 一级光(01)、LED 二级光(10)、LED 三级光(11)；bit2：电机开(1)关(0)。首先判断 D1 的 bit1-bit0(D1&0x03),为 0 时控制 PWM 占空比为 0，为 1 时控制 PWM 占空比为 50%，依此类推；然后判断 D1 的 bit2(D1&0x04)，非 0 时打开继电器从而打开电机，为 0 时关闭继电器从而关闭电机。此 Demo 中没有控制实体，所以对应的控制操作被注释了，仅作为参考。代码实现如下：

```

/*****
* 名称: sensorControl()
* 功能: 传感器控制
* 参数: cmd - 控制命令
* 返回: 无
* 修改:
* 注释:
*****/

void sensorControl(uint8 cmd)
{
    // 根据 cmd 参数处理对应的控制程序
    if(cmd & 0x01){

```

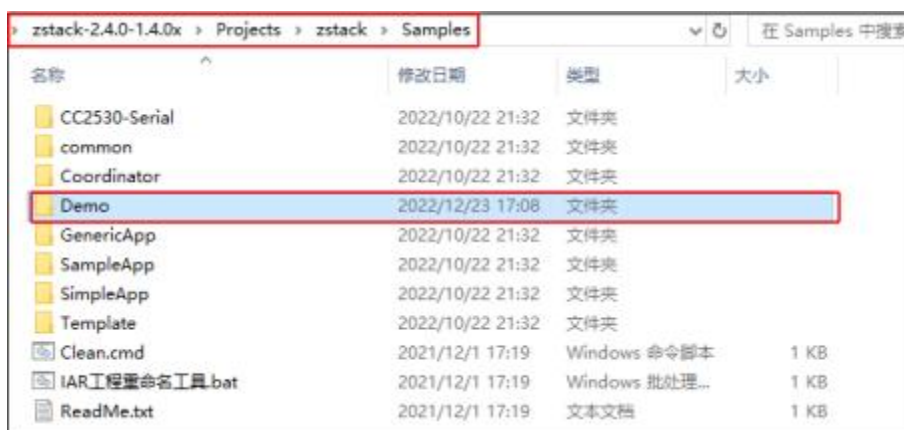



然后用 USB 线连接 ZigBee 无线节点到电脑，观察到当前串口为 COM5。

注意事项：第一次使用节点时，需要安装 USB 串口驱动(DISK-Packages\53-常用驱动程序\USB 串口驱动)，解压后根据电脑操作系统 32 位还是 64 位进行安装，安装完毕后，重新拔插 USB 线；(如果执行过此步骤，或者电脑已经有串口驱动，则可跳过此步骤)

2. 程序编译

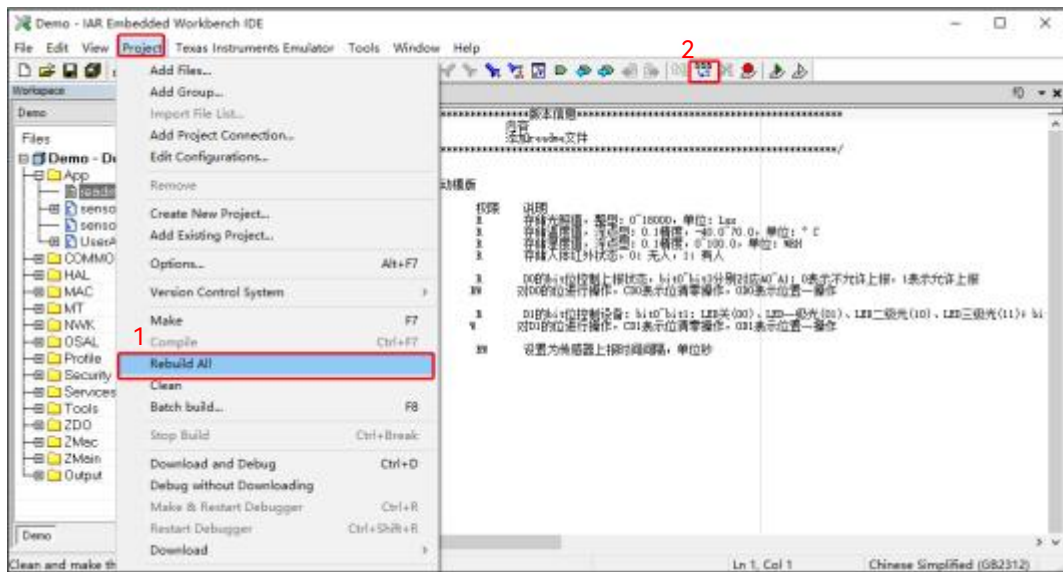
- 1) ZStack 协议栈安装文件路径为 DISK-xLabBase\02-软件资料\02-无线节点\zstack-2.4.0-1.4.0x.zip。
- 2) 将“zstack-2.4.0-1.4.0x.zip”压缩包以及 Demo.zip 压缩包(CH01-智云平台开发基础\03-硬件接口\Demo.zip) 都解压，注意 zstack-2.4.0-1.4.0x 存放的路径不能出现中文。
- 3) 将解压后的 Demo 文件夹复制到 zstack-2.4.0-1.4.0x 文件夹的“zstack-2.4.0-1.4.0x\Projects\zstack\Samples”目录下。



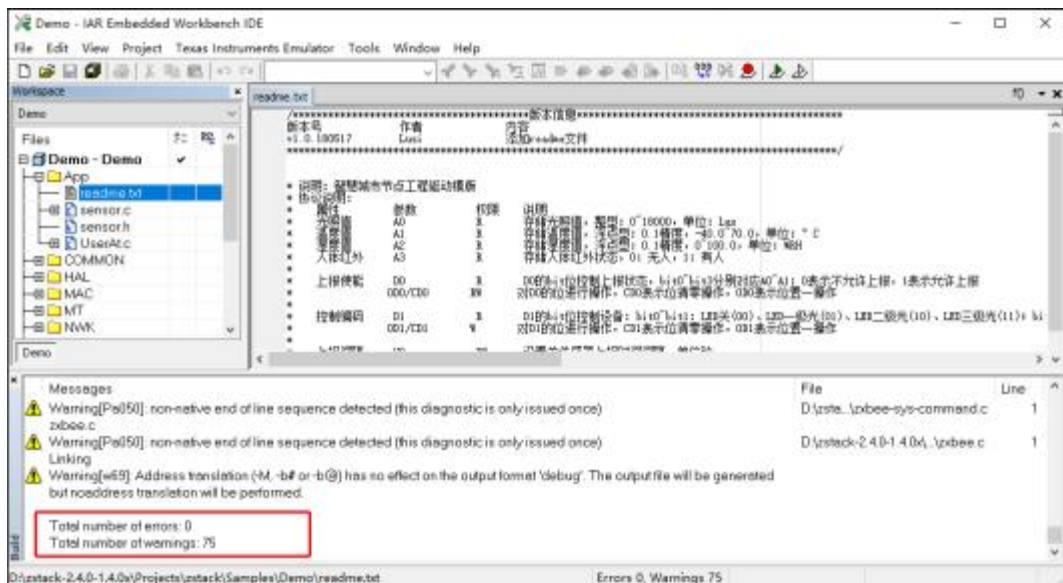
- 4) 使用 IAR for 8051 软件运行文件：双击 Demo\CC2530DB\Demo.eww 即可打开工程。

zstack-2.4.0-1.4.0x > Projects > zstack > Samples > Demo > CC2530DB >			
名称	修改日期	类型	大小
Demo	2022/12/24 9:00	文件夹	
settings	2022/12/24 9:00	文件夹	
Demo.dep	2022/12/24 9:00	DEP 文件	43 KB
Demo.ewd	2021/8/19 18:31	EWD 文件	18 KB
Demo.ewp	2021/8/19 18:31	EWP 文件	51 KB
Demo.ewt	2021/8/19 18:31	EWT 文件	20 KB
Demo.eww	2021/8/19 18:31	IAR IDE Worksp...	1 KB

5) 编译工程：编译代码查看程序是否有误。点击 Project-->Rebuild All。或者直接点击工具栏中的 make 按钮“”。



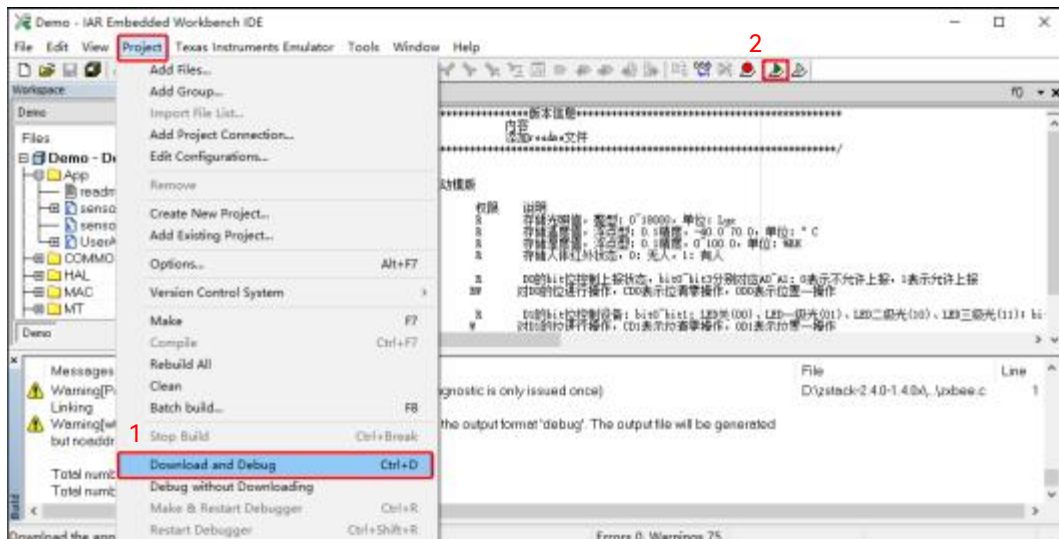
没有报错信息即为编译通过：



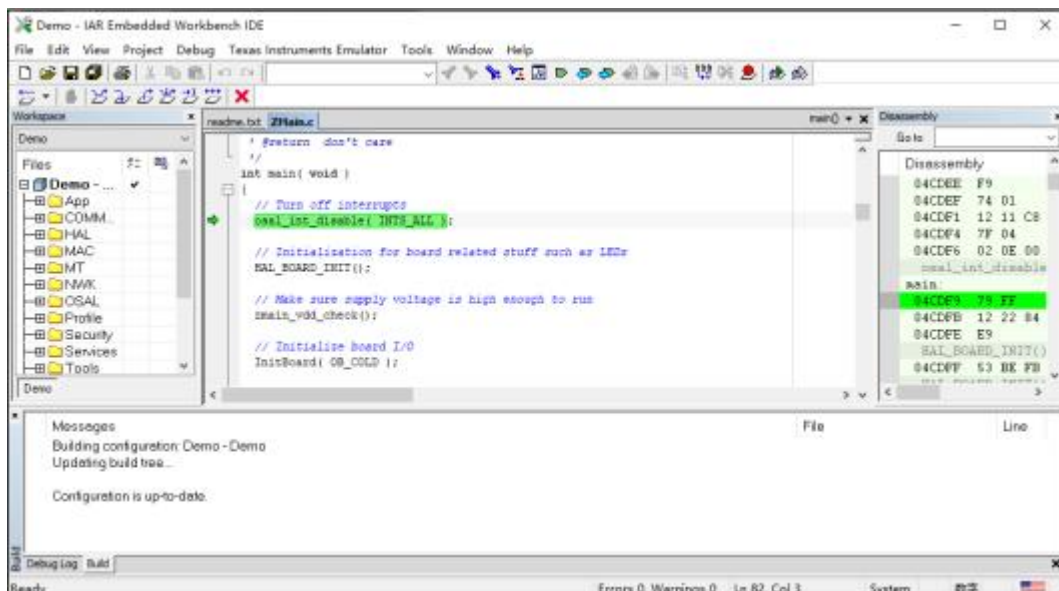
编译完成之后，会在当前目录下的 Demo\Exe 目录下生成 Demo.d51 和 Demo.hex 文件。

« Projects » zstack » Samples » Demo » CC2530DB » Demo » Exe			
名称	修改日期	类型	大小
Demo.d51	2022/12/24 9:00	D51 文件	2,096 KB
Demo.hex	2022/12/24 9:00	HEX 文件	434 KB

6) 调试/下载: 正确连接 SmartRF04 仿真器到PC 机和Zigbee 无线节点, 打开节点电源(上电), 按下 SmartRF04 仿真器上的复位按键, 点击 Project-->Download and Debug 或者直接点击工具栏的下载按钮  将程序下载到节点。



程序下载成功后 IAR 自动进入调试界面, 如下图所示界面。



7) 进入到调试界面后, 就可以对程序进行调试了, IAR 的调试按钮包括如下几个选项: 重置按钮 “Reset” 、终止按钮 “Break” 、跳过 “Step Over” 、跳入函数按钮 “Step Into” 、跳出函数按钮 “Step Out” 、下一条语句 “Next Statement” 、运行到光标的位置 “Run to Cursor” 

2、全速运行 “Go”  和停止调试按钮 “Stop Debugging” 。在调试的过程中, 也可以设置断点和通过其他调试窗口进行程序跟踪和数据的观察。

3. 固件更新

若需更新固件，参考《物联网短距离通信技术》实验步骤更新固件。

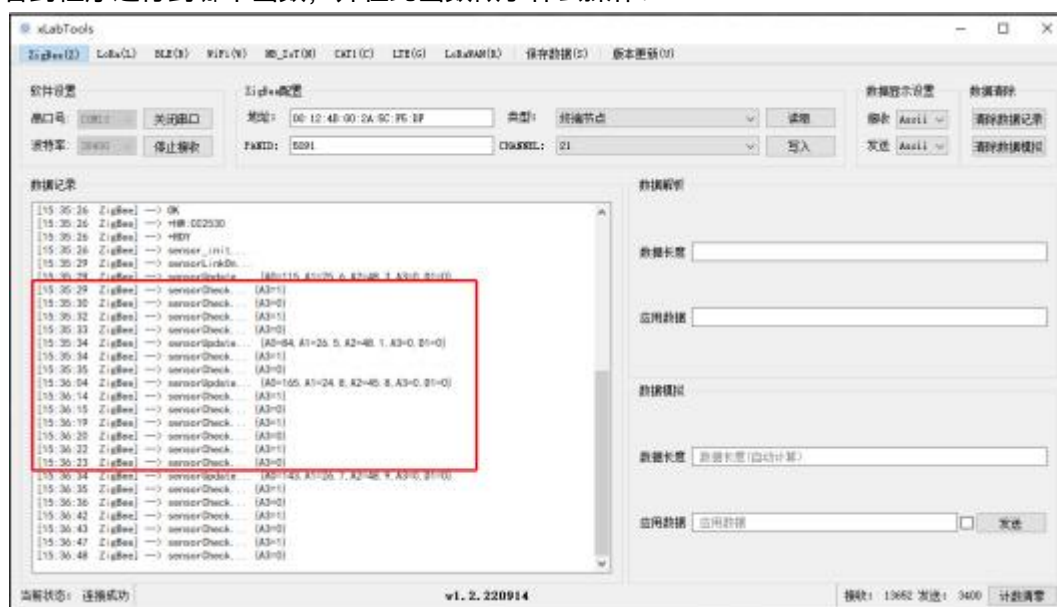
4. 数据分析

1) 参考《物联网短距离通信技术》实验步骤，让 ZigBee 无线节点与协调器组网(由于本示例程序是模拟传感器数据，所以此 ZigBee 无线节点不用外接传感器节点)。

2) USB 线连接协调器和电脑，运行 ZXBeeGW4 软件，启动智云服务，然后选择相应串口启用ZigBee。

3) 再使用另一条 USB 线连接 ZigBee 无线节点到电脑，运行 xLabTools 工具连入该节点的串口观察节点信息,可读取/修改节点的网络信息(地址、PANID、Channel、类型)。

4) 组网成功后，ZigBee 无线节点就开始发送数据到协调器，通过代码中的 printf 函数，我们可以清晰的看到程序运行到哪个函数，并在此函数做了什么操作：

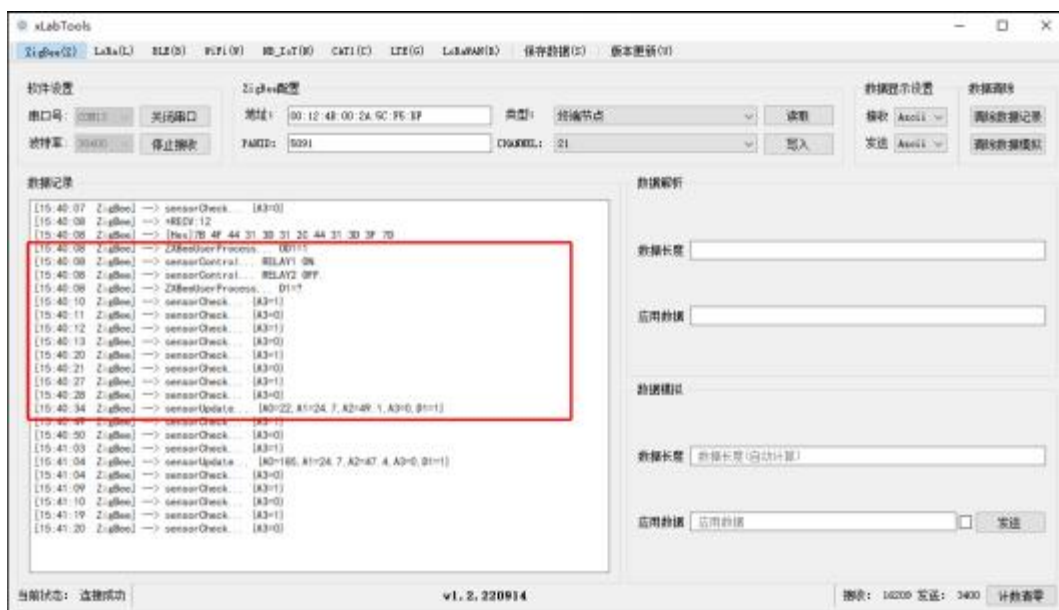


5) 参考示例节点的 ZXBee 协议，使用ZCloudWebTools 对节点进行调试。

下面为例说明：



同时使用 xLabTools 工具可以看到下发数据，首先会调用 ZXBeeUserProcess 函数解析“OD1=1”命令，然后通过 sensorControl 函数控制对应器件，最后再调用一次 ZXBeeUserProcess 函数查询 D1 的值：



五、常见问题

1. 调试指令使用英文字符，注意大小写，且不要包含空格，当调试指令包含条命令时使用逗号分隔开来。

4 智云 Web 应用接口

一、实验目的

1. 了解智云 Web 应用编程接口。
2. 掌握实时连接、历史数据、摄像监控、用户数据接口的编程。

二、实验环境

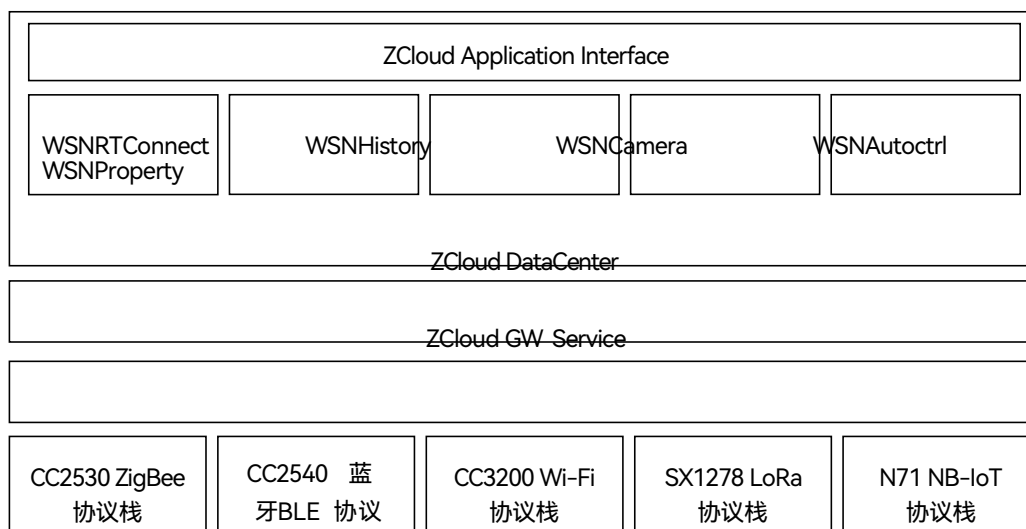
硬件环境:	PC 机 Pentium 处理器双核 2GHz 以上, 内存 4GB 以上
操作系统:	Windows7 64 位及以上操作系统
开发软件:	智云硬件物元仿真平台 HSIMS、ZCloudWebTools、ZXBeeGW4、Chrome 浏览器
实验配件:	USB 摄像头

三、实验内容

1. 智云框架

1.1 应用框架

智云平台提供五大应用接口供开发者使用, 包括: 实时连接(WSNRTConnect)、历史数据(WSNHistory)、摄像监控(WSNCamera)、自动控制(WSNAutoctrl)、用户数据(WSNProperty), 详细逻辑图如下:



1.2 功能设计

本实验通过 Web 的示例来介绍智云 API 接口的使用:

- 1) 实时连接示例: 实时采集温湿度传感器的温度和湿度数据; 远程控制灯的开关。
- 2) 历史数据示例: 实时采集光照度传感器的光照度数据; 获取光照度传感器的历史数据及曲线。
- 3) 摄像监控示例: 调用智云接口进行远程视频监控及控制。

4) 用户数据示例：调用智云接口对数据进行增加、修改和删除操作。
相关的设备采用示例节点，使用到的协议如下：

节点名称	TYPE	参数	含义	读写权限	说明
示例节点	500	A0	光强值	R	光强值，整型：0~18000，单位：Lux
		A1	温度值	R	温度值，浮点型：0.1 精度，-40.0~70.0，单位：°C
		A2	湿度值	R	湿度值，浮点型：0.1 精度，0~100.0，单位：%RH
		A3	人体红外	R	监测是否周边有人体活动：1 表示检测到附近有人，0 表示未检测到有人体活动
		D0(OD0/CD0)	主动上报使能	RW	D0 的 bit0~bit7 对应 A0~A7 主动上报使能，0 表示不允许主动上报，1 表示允许主动上报
		D1(OD1/CD1)	开关控制	RW	D1 的 bit 位控制设备： bit0~bit1：LED 照明控制，LED 关(00)、LED 一级光(01)、LED 二级光(10)、LED 三级光(11) bit2：电机开关控制，关闭电机(0)、打开电机(1)
		V0	主动上报时间间隔	RW	A0~A7 主动上报时间间隔，默认 30，单位：S

2. 应用接口

针对 Web 应用开发，智云平台提供 JavaScript 接口库，用户直接调用相应的接口即可完成简单 Web 应用的开发。以下重点介绍实时连接、历史数据、摄像监控、用户数据接口。

2.1 实时连接

基于 Web JavaScript 的接口如下：

函数	参数说明	功能
new WSNRTConnect(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	建立实时数据实例，并初始化智云 ID 及密钥
connect()	无	建立实时数据服务连接
disconnect()	无	断开实时数据服务连接
onConnect()	无	监听连接智云服务成功
onConnectLost()	无	监听连接智云服务失败
onMessageArrive(mac, dat)	mac:传感器的MAC地址 dat: 发送的消息内容	监听收到的数据
sendMessage(mac, dat)	mac:传感器的MAC地址 dat: 发送的消息内容	发送消息
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.2 历史数据

基于 Web JavaScript 的接口如下：

函数	参数说明	功能
new WSNHistory(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化历史数据对象，并初始化智云ID及密钥
queryLast1H(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1小时的历史数据
queryLast6H(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近6小时的历史数据
queryLast12H(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近12小时的历史数据
queryLast1D(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1天的历史数据
queryLast5D(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近5天的历史数据
queryLast14D(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近14天的历史数据
queryLast1M(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1月(30天)的历史数据
queryLast3M(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近3月(90天)的历史数据
queryLast6M(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近6月(180天)的历史数据
queryLast1Y(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1年(365天)的历史数据
query(cal);	cal:回调函数(处理历史数据)	获取所有通道最后一次数据
query(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	获取该通道下最后一次数据
query(channel, start, end, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据) start:起始时间 end:结束时间 时间为ISO 8601格式的日期，例如：2010-05-	通过起止时间查询指定时间段的历史数据(<u>根据时间范围默认选择采样间隔</u>)

	20T11:00:00Z	
query(channel, start, end, interval, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据) start:起始时间 end:结束时间 interval:采样点的时间间隔, 详细见后续说明 时间为ISO 8601格式的日期, 例如: 2010-05-20T11:00:00Z	通过起止时间查询指定时间段指定时间间隔的历史数据
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.3 摄像监控

基于 Web JavaScript 的接口如下:

函数	参数说明	功能
new WSNCamera(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化摄像头对象, 并初始化智云ID及密钥
initCamera(myCameralP, user, pwd, type);	myCameralP:摄像头外网域名/IP地址 user:摄像头用户名 pwd:摄像头密码 type:摄像头类型(F-Series、F3-Series、H3-Series、ZCamera) # 以上参数从摄像头手册获取	设置摄像头域名、用户名、密码、类型等参数
openVideo();	无	打开摄像头
closeVideo();	无	关闭摄像头
control(cmd);	cmd:云台控制命令, 参数如下: UP: 向上移动一次 DOWN: 向下移动一次 LEFT: 向左移动一次 RIGHT: 向右移动一次 HPATROL: 水平巡航转动 VPATROL: 垂直巡航转动 360PATROL: 360度巡航转动	发指令控制摄像头云台转动
checkOnline(cal);	cal:回调函数(处理检查结果)	检测摄像头是否在线
snapshot();	无	抓拍照片
setDiv(divID);	divID:网页标签	设置展示摄像头视频图像的标签
freeCamera(myCameralP);	myCameralP:摄像头外网域名/IP地址	释放摄像头资源
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.4 用户数据

基于 Web JavaScript 的接口如下:

函数	参数说明	功能
new WSNProperty(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化用户数据对象, 并初始化智云 ID 及密钥
put(key, value, cal);	key:名称 value:内容 cal:回调函数	创建用户应用数据
get(cal);	cal:回调函数	获取所有的键值对
get(key, cal);	key:名称 cal:回调函数	获取指定key的value值
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

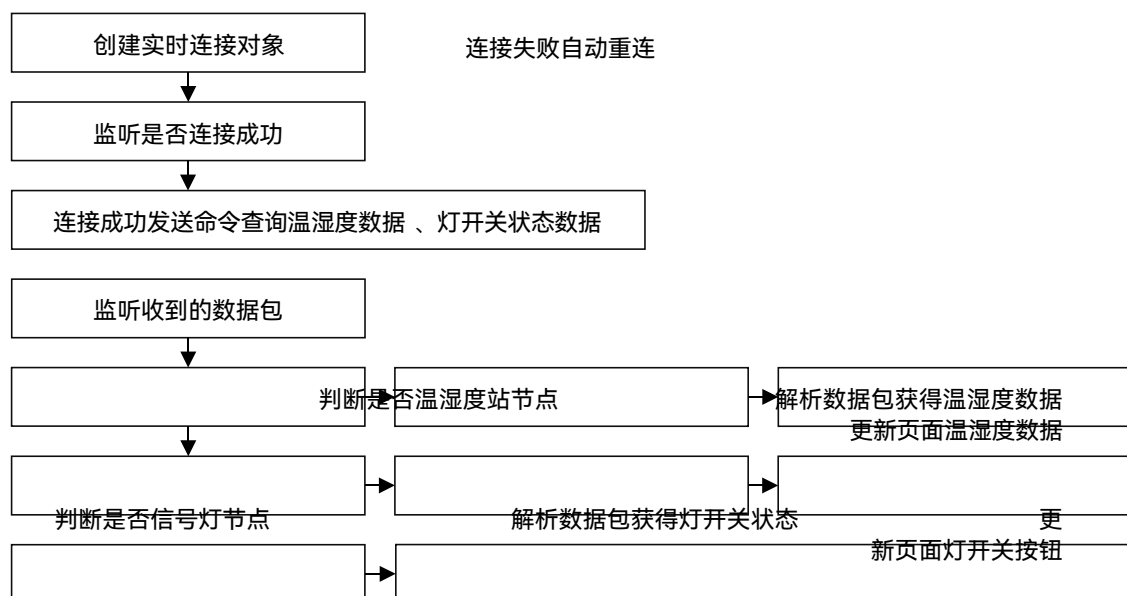
3. 程序分析

3.1 实时连接

RTConnectDemo-Web 是一个简单的实时连接应用接口示例, 实现了温湿度传感器数据的实时采集、灯光的开关控制等功能。

● 程序逻辑:

RTConnectDemo-Web 程序逻辑如下:



● 程序分析:

RTConnectDemo-Web 程序内容如下:

1) HTML 页面设计(index.html)

```

<!DOCTYPE html >

<html lang="zh-cmn-Hans">
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="css/bootstrap.min.css">
<link rel="stylesheet" href="css/index.css">
<title>实时连接示例</title>
</head>

```

```

<body>
  <!-- 页头 -->
  <div class="header">
    <div>
      <div>
        <h1>实时连接示例</h1>
      </div>
      <div id="ConnectState">数据服务连接中...</div>
    </div>
  </div>
  <!-- 页中 -->
  <div class="main">
    <div>
      <div>当前温度: </div>
      <div id="currentTem">--°C</div>
    </div>
    <div>
      <div>当前湿度: </div>
      <div id="currentHum">--%</div>
    </div>
    <div>
      <div>灯光开关: </div>
      
    </div>
  </div>
  <!-- 页尾 -->
  <div class="footer"></div>
  <!--引入 js-->
  <script src="js/jquery-1.11.0.min.js"></script>
  <script src="js/highcharts.js"></script>
  <script src="js/WSN/WSNRTConnect.js"></script>
  <script src="js/script.js"></script>
</body>
</html>

```

2) JavaScript 程序设计(js\script.js)

```

/*****
* 初始化变量
*****/

var myZCloudID = "123456789";           // 智云帐号
var myZCloudKey = "ABCDEFGHJKLMNOPQRSTUVWXYZ"; // 智云密钥
var DemoMac = "00:12:4B:00:15:D3:CE:29"; // 节点 MAC 地址
var rtc = new WSNRTConnect(myZCloudID, myZCloudKey); // 创建数据连接服务对象

/*****
* 与智云服务连接，并监听和解析实时数据并显示
*****/

```

```

*****/

$(function () {
    rtc.setServerAddr("api.zhiyun360.com"); // 设置服务器地址
    rtc.connect();
    // 数据推送服务连接
    rtc.onConnect = function () { // 连接成功回调函数
        rtc.sendMessage(DemoMac, "{A1=?,A2=?,D1=?}"); // 查询温湿度、灯光初始值$("#ConnectState").text("数据服务连接成功! ");
    };
    rtc.onConnectLost = function () { // 数据服务掉线回调函数
        $("#ConnectState").text("数据服务掉线! ");
    };
    rtc.onmessageArrive = function (mac, dat) { // 消息处理回调函数console.log(mac+">>> "+dat);

        if (mac == DemoMac) { // 判断是否是节点的数据

            if (dat[0] == '{' && dat[dat.length - 1] == '}') { // 判断字符串首尾是否为{}
                dat = dat.substr(1, dat.length - 2); // 截取{}内的字符串
                var its = dat.split(','); // 以','来分割字符串
                for (var x in its) { // 循环遍历
                    var t = its[x].split('='); // 以'='来分割字符串
                    if (t.length != 2) continue; // 满足条件时结束当前循环
                    if (t[0] == "A1") { // 判断参数 A1
                        var tem = parseFloat(t[1]); // 读取温度数据
                        $("#currentTem").text(tem + "°C"); // 在页面显示温度数据
                    }
                    if (t[0] == "A2") { // 判断参数 A2
                        var hum = parseFloat(t[1]); // 读取湿度数据
                        $("#currentHum").text(hum + "%"); // 在页面显示湿度数据
                    }
                    if (t[0] == "D1") { // 判断参数 D1
                        var LightStatus = parseInt(t[1]); // 根据 D1 的值来进行开关的切换
                        if ((LightStatus & 0x03) == 0x03) {
                            $('#btn_img').attr('src','images/an-on.png')
                        } else if ((LightStatus & 0x03) == 0) {
                            $('#btn_img').attr('src','images/an-off.png')
                        }
                    }
                }
            }
        }
    }
});

```

```

/*****
* 处理按键事件
*****/
$('#btn_img').click(function(){
    if($('#btn_img').attr('src') == 'images/an-on.png'){
        rtc.sendMessage(DemoMac, "{CD1=3,D1=?}");           // 发送关闭灯光指令
    }else{
        rtc.sendMessage(DemoMac, "{OD1=3,D1=?}");           // 发送打开灯光指令
    }
})

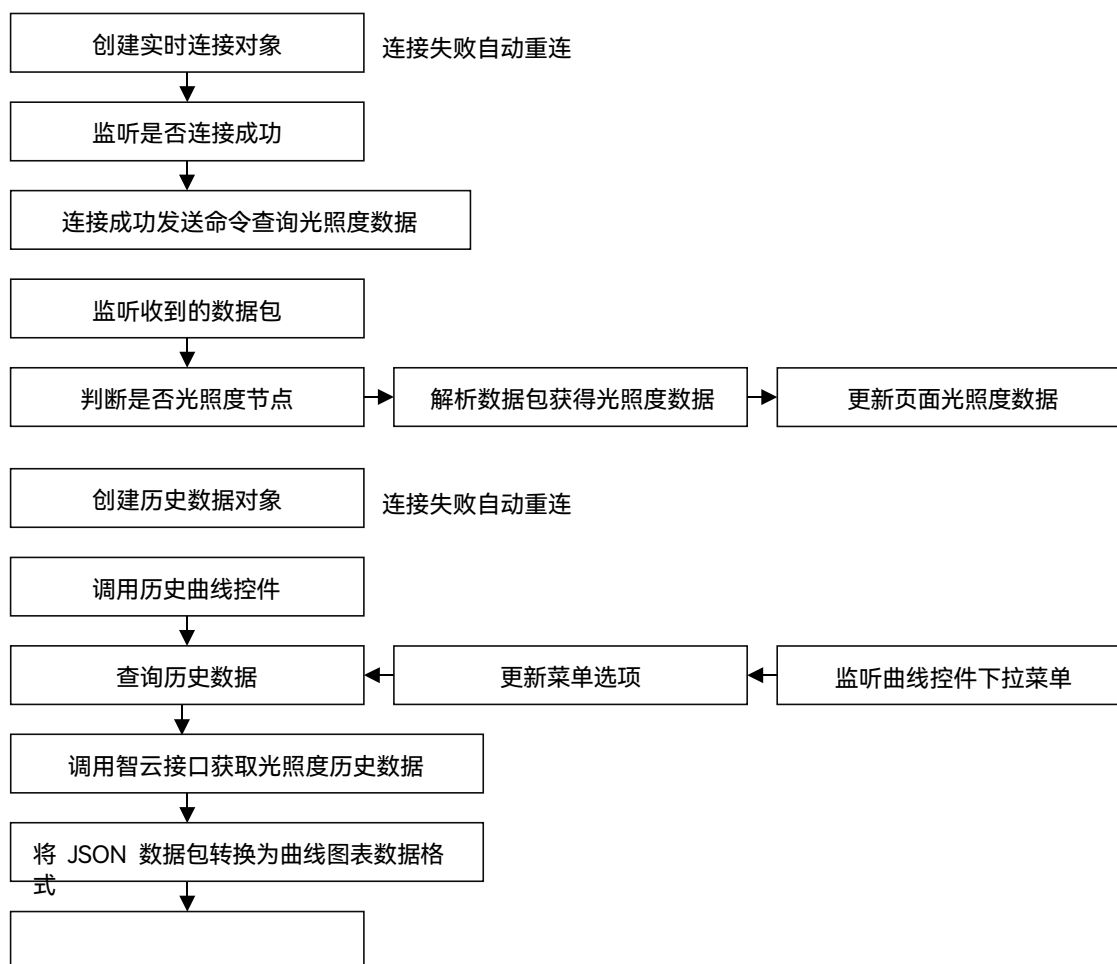
```

3.2 历史数据

HistoryDemo-Web 是一个简单的历史数据应用接口示例，实现了光照度传感器数据的实时采集、光照度历史数据曲线展示等功能。

● 程序逻辑:

HistoryDemo-Web 程序逻辑如下:



● 程序分析:

HistoryDemo-Web 程序内容如下:

1) HTML 页面设计(index.html)

```

<!DOCTYPE html >
<html lang="zh-cmn-Hans">
<head>

```

```

<meta charset="UTF-8">
<link rel="stylesheet" href="css/bootstrap.min.css">
<link rel="stylesheet" href="css/index.css">
<title>历史数据示例</title>
</head>
<body>
  <!-- 页头 -->
  <div class="header">
    <div>
      <div>
        <h1>历史数据示例</h1>
      </div>
      <div id="ConnectState">数据服务连接中...</div>
    </div>
  </div>
  <!-- 页中 -->
  <div class="main">
    <div>
      <div>光照强度: </div>
      <div id="currentTem">--Lux</div>
    </div>
    <div id="chart_box">
      <select class="form-control" id="MessSet" >
        <option value="queryLast1H">最近 1 小时</option>
        <option value="queryLast6H">最近 6 小时</option>
        <option value="queryLast12H">最近 12 小时</option>
        <option value="queryLast1D">最近 1 天</option>
        <option value="queryLast14D">最近 2 周</option>
        <option value="queryLast1M">最近 1 月</option>
        <option value="queryLast3M">最近 3 月</option>
        <option value="queryLast6M">最近半年</option>
        <option value="queryLast1Y">最近 1 年
      </option> </select>
      <div id="line_charts" style="width:70vw;height:40vh;margin:0 auto;"></div>
    </div>
  </div>
  <!-- 页尾 -->
  <div class="footer"></div>
  <!--引入 js-->
  <script src="js/jquery-1.11.0.min.js"></script>
  <script src="js/highcharts.js"></script>
  <script src="js/WSN/WSNRTConnect.js"></script>
  <script src="js/WSN/WSNHistory.js"></script>
  <script src="js/script.js"></script>
</body>

```

</html>

2) JavaScript 程序设计(js\script.js)

```
/******  
* 初始化变量  
*****/  
var myZCloudID = "123456789";  
智云帐号  
var myZCloudKey = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // 智云密钥  
var mySensorMac = "01:12:4B:00:69:F2:D3:54"; // 照明灯节点 MAC 地址  
var channel = `${mySensorMac}_A0`; // 传感器数据通道  
  
var rtc = new WSNRTConnect(myZCloudID, myZCloudKey); // 创建数据连接服务对象  
var myHisData = new WSNHistory(myZCloudID, myZCloudKey); // 创建历史数据  
服务对象var LightIntensity;  
  
/******  
* 与智云服务连接，并监听和解析实时数据并显示  
*****/  
$(function(){  
    rtc.setServerAddr("api.zhiyun360.com"); // 设置服务器地址  
    rtc.connect();  
    rtc.onConnect = function() { // 连接成功  
        回调函数  
        rtc.sendMessage(mySensorMac, "{A0=?}"); // 查询光强初始值  
        $("#ConnectState").text("数据服务连接成功! ");  
    };  
  
    rtc.onConnectLost = function() { // 数据服务掉线  
        回调函数  
        $("#ConnectState").text("数据服务掉线! ");  
    };  
  
    rtc.onmessageArrive = function(mac, dat) {  
        // 消息处理回调函数console.log(mac+">>> "+dat);
```

```

        if (mac == mySensorMac) {
            if (dat[0] == '{' && dat[dat.length - 1] == '}') { dat = dat.substr(1, dat.length - 2);
                var its = dat.split(',');
                for (var x in its) {
                    var t = its[x].split('=');
                    if (t.length != 2) continue;
                    if (t[0] == "A0") {
                        LightIntensity = parseInt(t[1]);
                        $("#currentTem").text(LightIntensity + "Lux"); }
                }
            }
        }

        // 判断传感器 Mac 地址
// 判断字符串首尾是否为{} // 截取{}内的字符串
// 以','来分割字符串

// 以'='来分割字符串

// 判断参数 A2

// 在页面显示光强数据

    }
};
})

/*****
* 默认调用历史数据图表，初参数为下拉选项初始值
*****/

checkHistory('MessSet', '#line_charts');

/*****
* 下拉选项切换历史数据时间范围
*****/

$('#MessSet').change(function () {
    checkHistory('MessSet', '#line_charts');
})

/*****
* 名称: checkHistory(set, tagIndex, hisDiv)
* 功能: 连接调用历史数据
* 参数: set: 获取选中的历史数据时间范围
*       tagindex: 判断后赋值给对应的历史查询对象
*       hisdiv: 显示图表的节点
* 返回: 无
* 修改:

```

```

* 注释:
*****/

function checkHistory(set, hisDiv) {
    var time = $('#'+ set).val(); // 设置时间
    myHisData.setServerAddr("api.zhiyun360.com"); // 设置服务器地址

    console.log('查询时间为: ' + time + ', 查询通道为: ' + channel);
    myHisData[time](channel, function (dat) {
        console.log(dat)
        // 输出查询到的历史数据
        if (dat.datapoints.length > 0) {
            var data = DataAnalysis(dat); // JSON 数据转化
            为图表数据
            showChart(hisDiv, 'spline', ", false, eval(data)); // 显示图表
            数据曲线 }
        });
    }

/*****
* 将 JSON 格式的数据转换成[x1,y1],[x2,y2],[x3,y3]...格式的数组(历史数据图表相关)
*****/

function DataAnalysis(data, timezone) {
    var str = "";

    var value;
    var len = data.datapoints.length;
    if (timezone == null) {
        timezone = "+8";
    }
    var zoneOp = timezone.substring(0, 1);
    var zoneVal = timezone.substring(1);
    var tzSecond = 0;

    $.each(data.datapoints, function (i, ele) {
        if (zoneOp == '+') {
            value = Date.parse(ele.at) + tzSecond;
        }
        if (zoneOp == '-') {
            value = Date.parse(ele.at) - tzSecond;
        }
        if (ele.value.indexOf("http") != -1) {
            str = str + '[' + value + ',' + ele.value + '];'
        } else {
            str = str + '[' + value + ',' + ele.value + '];'
        }
        if (i != len - 1)
            str = str + ',';
    });
}

```

```

    return "[" + str + "];"
}

/*****
* 画线图的方法(历史数据图表相关)
*****/

function showChart(sid, ctype, unit, step, data) {
    $(sid).highcharts({
        chart: {
            backgroundColor: 'transparent',
            type: ctype,
            animation: false,
            zoomType: 'x'
        },
        legend: {
            enabled: false
        },
        title: {
            text: ""
        },
        xAxis: {
            type: 'datetime',

            labels: {
                style: {
                    color: 'rgb(0, 0, 0)',
                }
            }
        },
        yAxis: {
            title: {
                text: ""
            },
            minorGridLineWidth: 0,
            gridLineWidth: 1,
            alternateGridColor: null,
            labels: {
                style: {
                    color: 'rgb(0, 0, 0)',
                }
            }
        },
        tooltip: {
            formatter: function () {
                return " +
                Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) + '<br><b>' + this.y + unit + '</b>';
            }
        },

```

```

plotOptions: {
  spline: {
    lineWidth: 2,
    states: {
      hover: {
        lineWidth: 3
      }
    },
  },
  marker: {
    enabled: false,
    states: {
      hover: {
        enabled: true,
        symbol: 'circle',
        radius: 3,
        lineWidth: 1
      }
    }
  }
},

```

```

line: {
  lineWidth: 1,
  states: {
    hover: {
      lineWidth: 1
    }
  },
  marker: {
    enabled: false,
    states: {
      hover: {
        enabled: true,
        symbol: 'circle',
        radius: 3,
        lineWidth: 1
      }
    }
  }
},
series: [{
  marker: {
    symbol: 'square'
  },
  data: data,

```

```

        step: step,
    }},
    navigation: {
        menuItemStyle: {
            fontSize: '10px'
        }
    }
});
}

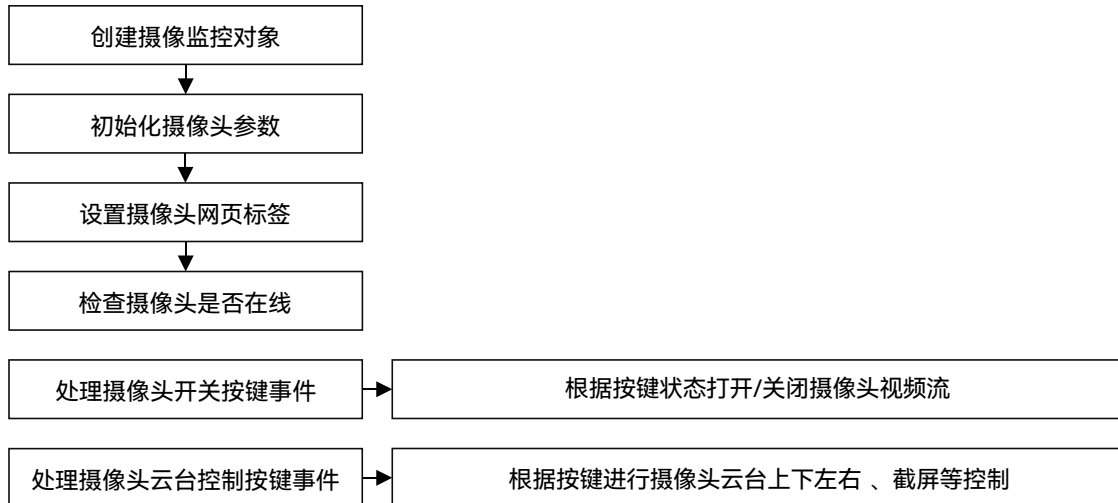
```

3.3 摄像监控

CameraDemo-Web 是一个简单的摄像监控应用接口示例，实现了网络实时摄像头视频流监控和摄像头云台转动控制等功能。

- **程序逻辑:**

CameraDemo-Web 程序逻辑如下:



● 程序分析:

CameraDemo-Web 程序内容如下:

1) HTML 页面设计(index.html)

```

<!DOCTYPE html >
<html lang="zh-cmn-Hans">
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="css/bootstrap.min.css">
<link rel="stylesheet" href="css/index.css">
<title>摄像监控示例</title>
</head>
<body>
  <!-- 页头 -->
  <div class="header">
    <div>
      <div>
        <h1>摄像监控示例</h1>
      </div>
      <div id="ConnectState">摄像头连接中...</div>
    </div>
  </div>
  <!-- 页中 -->
  <div class="content">
    <div class="container">
      <h3 class="text-center">
        视频监控
      </h3>
      <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
          <div class="panel panel-default">
            <div class="panel-heading">
              <h3 class="panel-title">

```

```

        摄像头控制
    </h3>
</div>
<div class="panel-body" style="height: 520px;">
    <div class="input-group controller col-xs-12 col-sm-12 col-md-4 col-lg-4">
        <button type="button" id="switch" class="btn btn-success"
style="margin-top:80px;margin-left:40%;">
            开
        </button>
        <button type="button" id="ct_up" class="btn btn-default" style="margin-top:140px;margin-
left:40%;">
            上
        </button>
        <button type="button" id="ct_left" class="btn btn-default" style="margin-top:180px;margin-
left:15%;">
            左
        </button>
        <button type="button" id="ct_right" class="btn btn-default"
style="margin-top:180px;margin-left:65%;">
            右
        </button>
        <button type="button" id="ct_down" class="btn btn-default"
style="margin-top:220px;margin-left:40%;">
            下
        </button>
        <button type="button" id="Snapshot" class="btn btn-default"
style="margin-top:280px;margin-left:40%;">
            截屏
        </button>
    </div>
    <div class="input-group monitor_video col-xs-12 col-sm-12 col-md-8 col-lg-8"
style="height: 482px;border:1px #ddd solid;width: 642px;padding-left: 0px;">
        <img id="img" width="640" height="480" border="0" src="" style="display: none;">
    </div>
</div>
</div>
</div>
</div>
<!-- 页尾 -->
<div class="footer"></div>

<!--引入 js-->
<script src="js/jquery-1.11.0.min.js"></script>

```

```

<script src="js/WSN/camera-1.1.js"></script>
<script src="js/WSN/WSNCamera.js"></script>
<script src="js/script.js"></script>
</body>
</html>

```

2) JavaScript 程序设计(js\script.js)

```

/*****
* 初始化变量
*****/

var myZCloudID = "12345678"; //智云帐号
var myZCloudKey = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //智云密钥
var mygwIP = "127.0.0.1:28083"; //网关地址(默认使用内网地址, 为空时使用外网地址)

var myCamera = "Camera:1"; //摄像头地址
var cam = new WSNCamera(myZCloudID, myZCloudKey); //创建摄像监控服务对象
var switch_cam = 0; //摄像头开关状态
var myImgId = "img"; //摄像头网页标签
/*****
* 初始化摄像头
*****/

$(function () {
    cam.setDiv(myImgId); //设置图像显示的位置
    cam.initCamera(mygwIP, myCamera); //摄像头初始化
})

/*****
* 拍照
*****/

$("#Snapshot").click(function () {
    if (switch_cam == 1) {
        cam.snapshot();
    }
})

/*****
* 打开/关闭摄像头
*****/

$("#switch").click(function () {
    if (switch_cam == 0) {
        cam.openVideo(); //打开摄像头并显示
        switch_cam = 1;
        $("#switch").text("关");
        $("#switch").siblings("button").attr("class", "btn btn-success");
    } else {
        $("#switch").text("开");
    }
})

```

```

        $("#switch").siblings("button").attr("class", "btn btn-default");
        cam.closeVideo();
        闭视频监控
        switch_cam = 0;
    }
})

/*****
*   控制摄像头向上移动一次
*****/
$("#ct_up").mousedown(function () {
    if (switch_cam == 1) {
        cam.control("UP");
        像头发送向上移动命令
    }
})

/*****
*   控制摄像头向下移动一次
*****/
$("#ct_down").mousedown(function () {
    if (switch_cam == 1) {
        cam.control("DOWN");
        像头发送向下移动命令
    }
})

/*****
*   控制摄像头向左移动一次
*****/
$("#ct_left").mousedown(function () {
    if (switch_cam == 1) {
        cam.control("LEFT");
        头发送向左移动命令
    }
})

/*****
*   控制摄像头向右移动一次
*****/
$("#ct_right").mousedown(function () {
    if (switch_cam == 1) {
        cam.control("RIGHT");
        头发送向右移动命令
    }
})

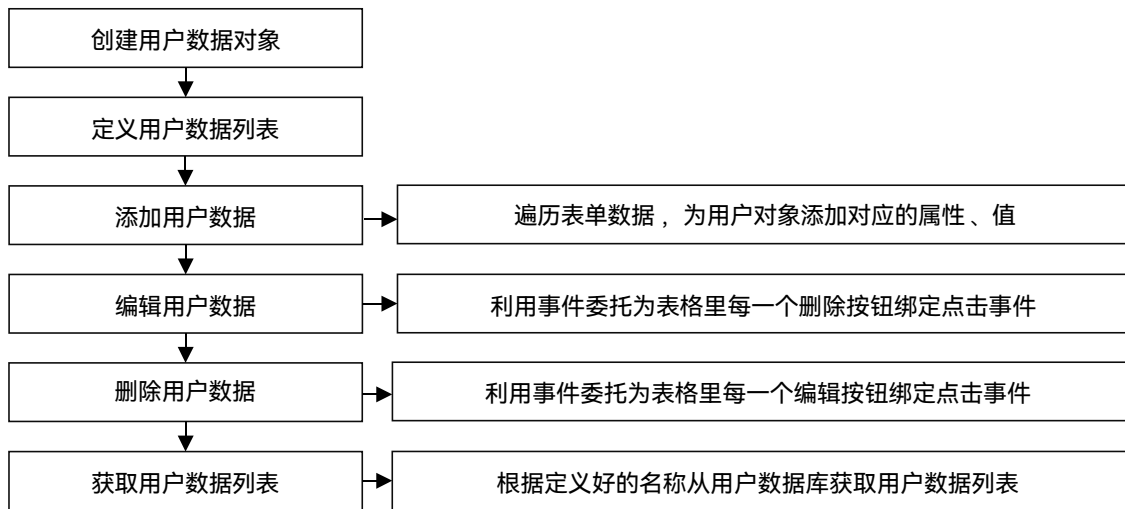
```

3.4 用户数据

PropertyDemo-Web 是一个简单的用户数据应用接口示例，实现了用户数据的“增删改”的功能。

● 程序逻辑:

PropertyDemo-Web 程序逻辑如下:



● 程序分析:

PropertyDemo-Web 程序内容如下:

1) HTML 页面设计(index.html)

```
<!DOCTYPE html>
<html lang="zh-cn-Hans">

<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/index.css">
  <title>用户数据示例</title>
</head>

<body>
  <!-- 页头 -->
  <div class="header">
    <div>
      <div>
        <h1>用户数据示例</h1>
      </div>
      <div id="ConnectState">用户数据连接成功! </div>
    </div>
  </div>
  <!-- 页中 -->
  <div class="main">
    <div id="box_1">
      <table class="table table-bordered">
        <thead>
          <th>卡号</th>
          <th>姓名</th>
          <th>性别</th>
          <th>操作</th>
        </thead>
      </table>
    </div>
  </div>
</body>
</html>
```

```

</thead>
<tbody>
  <tr>
    <td>123</td>
    <td>张三</td>
    <td>男</td>
    <td>
      <button type="button" class="btn btn-warning btn-sm edit">编辑</button>
      <button type="button" class="btn btn-danger btn-sm dlt">删除</button>
    </td>
  </tr>
</tbody>
</table>
<button type="button" class="btn btn-success" data-toggle="modal" data-target="#addModal">添加用户
</button>
</div>

```

```

<!-- 添加用户模态框 -->

```

```

<div class="modal fade" id="addModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel">
  <div class="modal-dialog modal-sm" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
          aria-hidden="true">&times;</span></button>
        <h4 class="modal-title" id="exampleModalLabel">用户信息</h4>
      </div>
      <div class="modal-body">
        <form id="addForm">
          <div class="input-group">
            <div class="input-group-addon">卡号: </div>
            <input type="text" class="form-control" name="id">
          </div>
          <div class="input-group">
            <div class="input-group-addon">姓名: </div>
            <input type="text" class="form-control" name="name">
          </div>
          <div class="form-group">
            <label for="message-text" class="control-label">性别:</label>
            <label class="radio-inline">
              <input type="radio" name="sex" value="男"> 男
            </label>
            <label class="radio-inline">
              <input type="radio" name="sex" value="女"> 女
            </label>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

```

```

        </div>
    </form>
</div>
<div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">取消</button>
    <button type="button" class="btn btn-primary" id="addUser">确认</button>
</div>
</div>
</div>
</div>

<!-- 修改用户模态框 -->
<div class="modal fade" id="editModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel">
    <div class="modal-dialog modal-sm" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
                    aria-hidden="true">&times;</span></button>
                <h4 class="modal-title" id="exampleModalLabel">用户信息</h4>
            </div>
            <div class="modal-body">
                <form id="editForm">
                    <div class="input-group">
                        <div class="input-group-addon">卡号: </div>
                        <input type="text" class="form-control" name="id">
                    </div>
                    <div class="input-group">
                        <div class="input-group-addon">姓名: </div>
                        <input type="text" class="form-control" name="name">
                    </div>
                    <div class="form-group">
                        <label for="message-text" class="control-label">性别:</label>
                        <label class="radio-inline">
                            <input type="radio" name="sex" value="男"> 男
                        </label>
                        <label class="radio-inline">
                            <input type="radio" name="sex" value="女"> 女
                        </label>
                    </div>
                </form>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">取消</button>
                <button type="button" class="btn btn-primary" id="editUser">确认</button>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</div>

<!-- 页尾 -->
<div class="footer"></div>
<!--引入 js-->
<script src="js/jquery-1.11.0.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/WSN/WSNProperty.js"></script>

<script src="js/script.js"></script>
</body>

</html>

```

2) JavaScript 程序设计(js\script.js)

```

/*****
*   初始化变量
*****/

let myZCloudID = "123456789";           //智云帐号
let myZCloudKey = "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //智云密钥
let myZCloudAddr = "api.zhiyun360.com"; //智云服务器地址
let myProperty = new WSNProperty(myZCloudID, myZCloudKey) //创建用户数据服务对象
    myProperty.setServerAddr(myZCloudAddr) //设置服务器地址
let userList = null //定义用户数据列表

//页面加载完后获取用户数据
$(function () {
    getUserList()
});

/*****
*   添加用户数据
*****/

$('#addUser').on('click', () => {
    let formData = $("#addFrom").serializeArray(); //获取序列化表单数据
    let obj = {}
    //创建当前添加的用户对象
    formData.forEach(item => {
        obj[item.name] = item.value //遍历表单数据,
        为用户对象添加对应的属
性、值
    });
    let { id, name, sex } = obj

```

```

        if (id && name && sex) { //判断用
            户对象的属性是否存在

            userList.push(obj) //将用
            户数据添加至列表

            myProperty.put('userList', JSON.stringify(userList), res => { //更新用户数据
                $('#addModal').modal('hide') //关闭会话框
                getUserList() //重新
                获取用户数据列表渲染页面
                alert('添加用户成功! ')
            });
        } else {
            alert('请输入完整信息! ')
        }
    })

    /*****
    * 删除用户数据
    *****/

    $('tbody').on('click', '.dlt', function () { //利用事件委托为表格里每一个删除按
        钮绑定点击事件
        const index = $(this).parents('tr').index() //获取当前点击行(用户数据) 索引
        userList.splice(index, 1) //从列表删除
        选中的数据
        myProperty.put('userList', JSON.stringify(userList), res => { //更新用户数据
            getUserList()
            //重新获取用户数据列表渲染页面
            alert('删除用户成功! ')
        });
    })

    /*****
    * 编辑用户数据
    *****/

    let editIndex = null
    //定义编辑中用户索引变量
    //第 1 步: 获取需要编辑的用户数据, 展示到会话框
    $('tbody').on('click', '.edit', function () { //利用事件委托为表格里每一个编辑按
        钮绑定点击事件
        editIndex = $(this).parents('tr').index() //获取编辑用户所在列表的索引
        let { id, name, sex } = userList[editIndex] //获取当前编辑用户的数据

        //将需要编辑的用户数据赋值到表单, 以方便用户编辑
        $('#editFrom input[name=id]').val(id)
        $('#editFrom input[name=name]').val(name)
        if (sex == '男') {
            $('#editFrom input[name=sex][value=男]').prop("checked", true);

```

```

    } else {
      $('#editFrom input[name='sex'][value='女']).prop("checked", true);
    }
    $('#editModal').modal('show') //显示编辑用户
    信息会话框
  })

//第 2 步：点击确认将编辑后的用户数据更新至数据库
$('#editUser').on('click', () => {
  let formData = $('#editFrom').serializeArray(); //获取序列化表单数据
  let obj = {} //创建当前编辑的用户对象
  formData.forEach(item => {
    obj[item.name] = item.value //遍历表单数据，为用户对象添加对应的属
    性、值
  });
  let { id, name, sex } = obj
  if (id && name && sex) { //判断用户对象的属性是否存在
    userList[editIndex] = obj //更新编辑后的数据列表
  }
})

```

```

myProperty.put('userList', JSON.stringify(userList), res => { //将编辑后的用户数据列表更新至用
    户
    数据库

    $('#editModal').modal('hide') //关闭会话框

    getUserList() //重新
    获取用户数据列表渲染页面

    alert('编辑用户成功! ')

    });
} else {
    alert('请输入完整信息! ')
}
})

/*****
* 获取用户数据列表
*****/

function getUserList() {
    myProperty.get('userList', res => { //根据定义好的名称从用户
        数据库获取用户数据

        列表

        userList = []
        if (res) {
            userList = JSON.parse(res) //将 JSON
            字符串转换为对象
            //判断数据是否存在，有则清除页面示例，有序的将用户数据渲染至页面

            if (userList.length > 0) {
                $("tbody").html("")
                for (let i = 0; i < userList.length; i++) {
                    let html = `<tr>

                        <td>${userList[i].id}</td>
                        <td>${userList[i].name}</td>
                        <td>${userList[i].sex}</td>
                        <td>

                            <button type="button" class="btn btn-warning btn-sm edit">
                                编 辑

                            <button type="button" class="btn btn-danger btn-sm dlt">
                                删 除

                        </td>
                    </tr>`

                    $("tbody").append(html);
                }
            }
        }
    });
}

```

四、实验步骤

1. 实验部署

本实验基于 xLab 实验平台来构建一个智云接口项目，可通过硬件物元仿真平台软件创建虚拟的硬件，也可以使用 xLab 实验平台真实的硬件(本实验默认通过硬件物元仿真平台软件创建)。

2. 原型搭建

2.1 创建传感器

1) 在传感设备栏，点击“新建传感设备”。



2) 基础设置：设置节点名称、编码和介绍，填写完成后点击下一步继续。

名称：填写传感器的名字，如“Demo”；

编码：填写控制类传感器 TYPE 值：500。



3) 传感器设置：A0~A7 代表节点所能携带的传感器及其数值，填写完成后点击下一步继续。

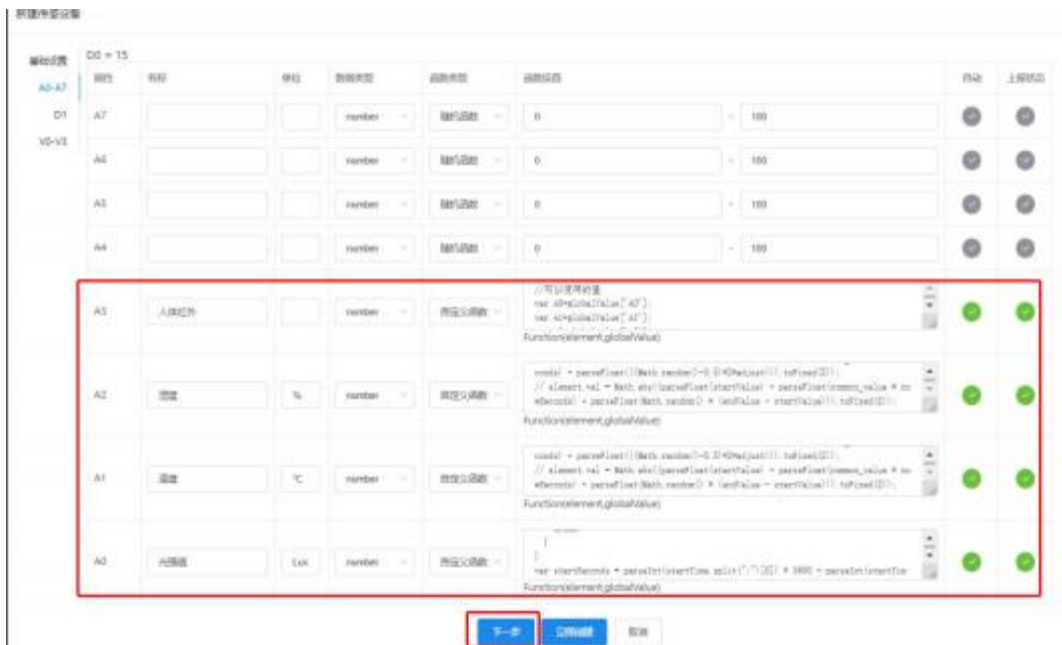
A0 表示光强单位为 Lux，数据类型为 number，函数类型选择自定义函数，将函数设置框内的代码

删除，打开“实验例程\04-应用接口(web)\虚拟仿真\”文件夹下的光强.txt，复制代码，然后粘贴到空白的函数设置框中，勾选后面的自动以及上报状态圆框。

A1 表示温度，单位为℃，数据类型为 number，函数类型选择自定义函数，将函数设置框内的代码删除，打开“实验例程\04-应用接口(web)\虚拟仿真\”文件夹下的温度.txt，复制代码，然后粘贴到空白的函数设置框中，勾选后面的自动以及上报状态圆框；

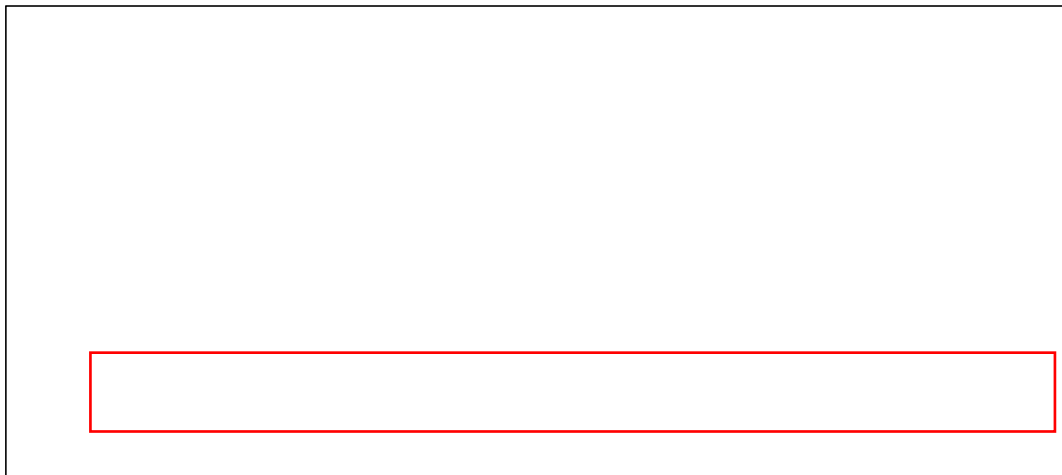
A2 表示湿度单位为%Rh，数据类型为 number，函数类型选择自定义函数，将函数设置框内的代码删除，打开“实验例程\04-应用接口(web)\虚拟仿真\”文件夹下的湿度.txt，复制代码，然后粘贴到空白的函数设置框中，勾选后面的自动以及上报状态圆框；

A3 表示检测人体状态，0 表示无人，1 表示有人，数据类型为 number，函数类型选择自定义函数，将函数设置框内的代码删除，打开“实验例程\04-应用接口(web)\虚拟仿真\”文件夹下的人体红外.txt，复制代码，然后粘贴到空白的函数设置框中，勾选后面的自动以及上报状态圆框；

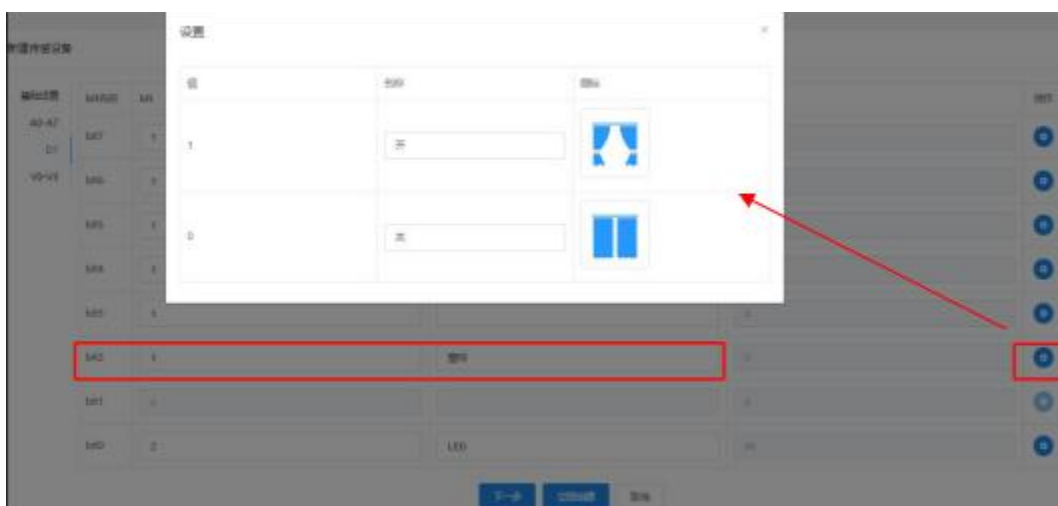


4) 执行器设置:

D1 的 bit0~bit1 控制 LED，点击操作栏的配置图标，在弹出的窗口填写名称和对应的动画图片(实验例程\04-应用接口(web) \虚拟仿真\led\)。



D1 的 bit2 控制电机，点击操作栏的配置图标，在弹出的窗口填写名称和对应的动画图片(实验例程\04-应用接口(web) \虚拟仿真\motor\)。

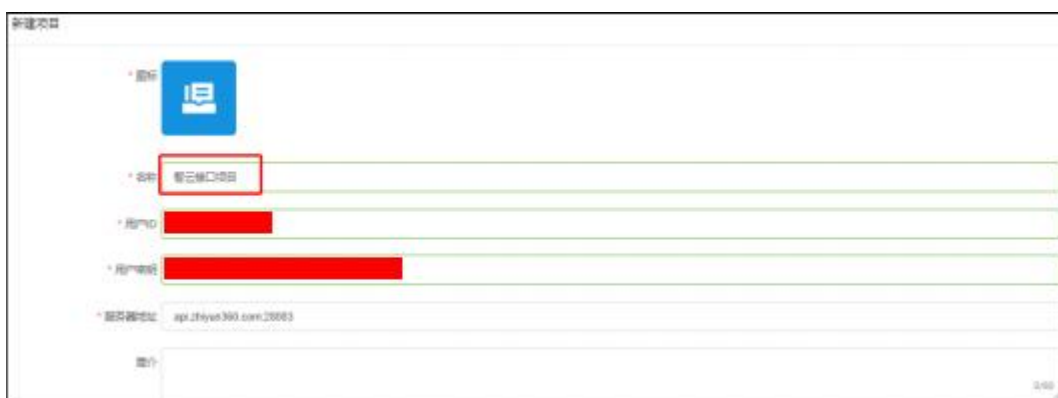


5) 节点参数: V0~V3 为节点参数, 填写完成后, 点击立即创建, 即可创建新的传感器。
V0: 传感器上报时间间隔, 默认为 30S;



2.2 创建项目

1) 参考实验 1 创建项目, 名称为“智云接口项目”。

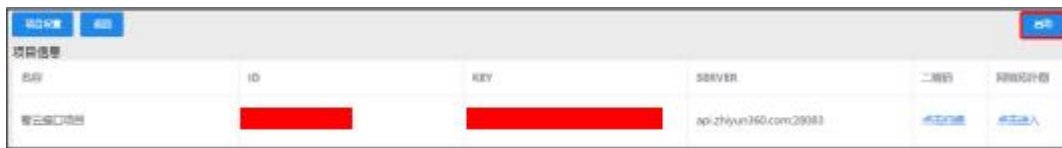


2) 传感器选用创建的“Demo”, 点击“立即创建”按钮, 完成项目创建。



2.3 运行项目

- 1) 在“项目管理”菜单中找到新建的“智云接口项目”，单击进入项目。
- 2) 等待传感器列表加载完成，点击右上角的“启动”按钮，启动项目，然后点击下方传感器的“开启”，大约 15S 后传感器数据开始上传并更新。



数据上传成功:



3. 应用体验

3.1 实时连接

- 1) 打开本实验目录下的工程 “RTConnectDemo-Web”。
- 2) 用记事本打开 script.js 文件(js\script.js), 填写授权的智云帐号、密钥和节点的 MAC 地址:

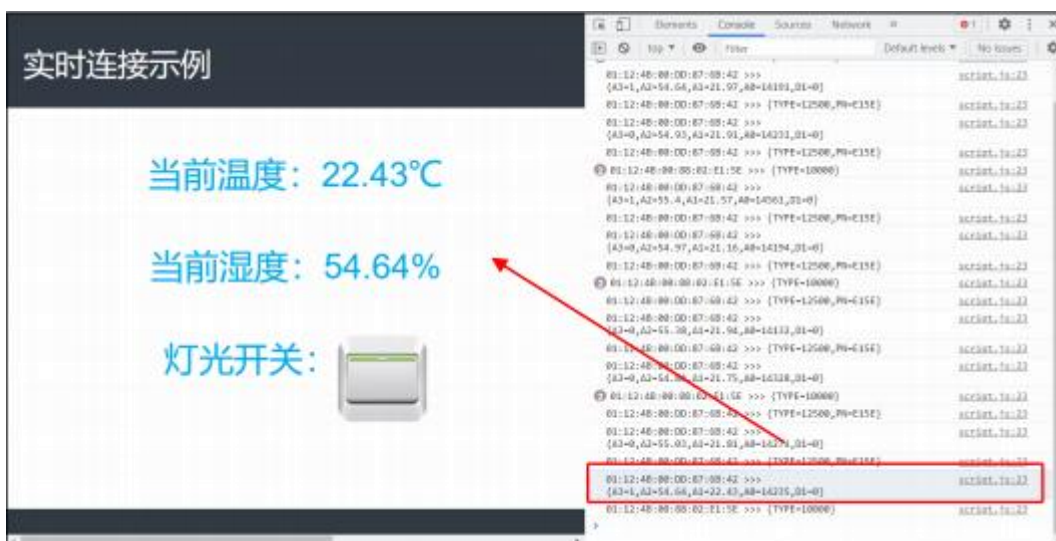
```
script.js - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
=====
* 初始化变量
=====
var myZCloudID = " "; // 智云帐号
var myZCloudKey = " "; // 智云密钥
var DemoMac = "01:12:48:00:DD:87:6B:42"; // 节点MAC地址
var rtc = new WSNRTConnect(myZCloudID, myZCloudKey); // 创建数据连接服务对象

//=====
* 与智云服务连接，并监听和解析实时数据并显示
//=====
$(function () {
    rtc.setServerAddr("api.zhiyun360.com"); // 设置服务器地址
    rtc.connect(); // 数据推送服务连接
    rtc.onConnect = function () { // 连接成功回调函数
        rtc.sendMessage(DemoMac, "A1=?,A2=?,D1=?"); // 查询温湿度、灯光初始值
        $("#ConnectState").text("数据服务连接成功!");
    };
    rtc.onConnectLost = function () { // 数据服务掉线回调函数
        $("#ConnectState").text("数据服务掉线!");
    };
    rtc.onmessageArrive = function (mac, dat) { // 消息处理回调函数
        console.log(mac + " >>> " + dat);
    };
});
```

3) 用 Chrome 浏览器打开实时连接示例程序 index.html，查看实时数据信息及 RGB 灯的开关操作。



4) 在 Chrome 浏览器页面单击右键弹出的菜单选择“检查”，可以进入到页面调试界面，在选项卡选择“Console”即可查看 Log 打印消息。



3.2 历史数据

- 1) 打开本实验目录下的工程 “HistoryDemo-Web”。
- 2) 用记事本打开 script.js 文件(js\script.js), 修改智云帐号、密钥和 ZigBee 无线节点的 MAC 地址:

```
script.js - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

/* 初始化变量 */
var myZCloudID = " "; // 智云帐号
var myZCloudKey = " "; // 智云密钥
var mySensorMac = "01:12:4B:00:DD:87:6B:42"; // 照明灯节点MAC地址
var channel = "${mySensorMac}_A0"; // 传感器数据通道

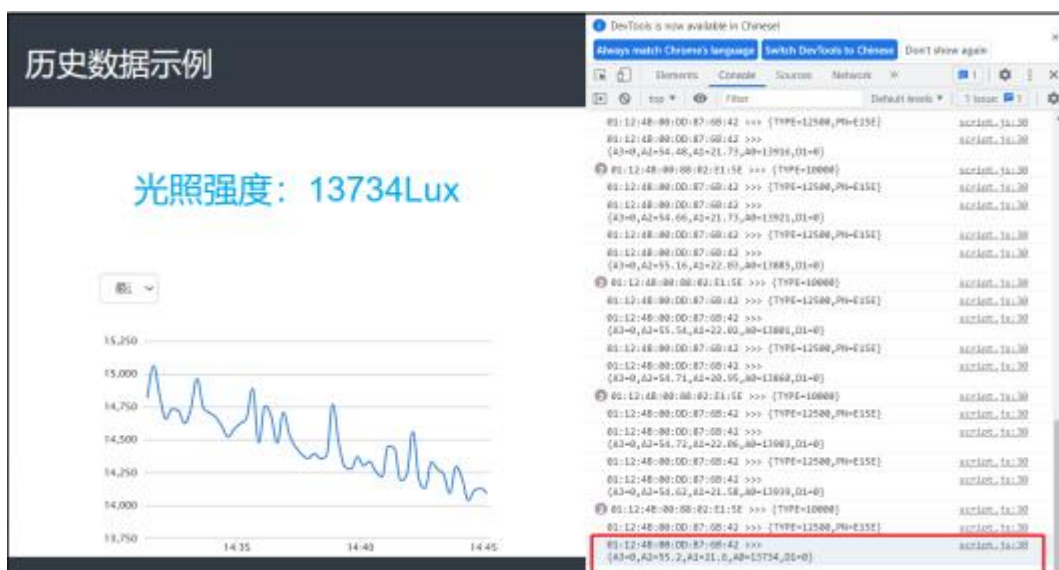
var rtc = new WSNRTCConnect(myZCloudID, myZCloudKey); // 创建数据连接服务对象
var myHisData = new WSNHistory(myZCloudID, myZCloudKey); // 创建历史数据服务对象
var LightIntensity;

/* 与智云服务连接，并监听和解析实时数据并显示 */
$(function(){
    rtc.setServerAddr("api.zhiyun360.com"); // 设置服务器地址
    rtc.connect();
    rtc.onConnect = function() { // 连接成功回调函数
        rtc.sendMessage(mySensorMac, "[A0-?]"); // 查询光强初始值
        $("#ConnectState").text("数据服务连接成功!");
    };
});
```

3) 用 Chrome 浏览器打开历史数据示例程序 index.html，查看光照强度实时数据信息及历史曲线信息，可以通过曲线的下拉选项选择查看其它时间的历史数据曲线。

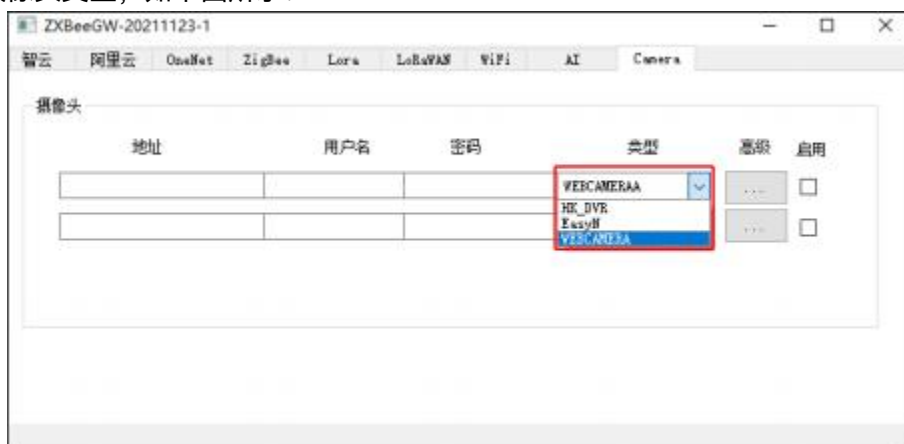


4) 在 Chrome 浏览器页面单击右键弹出的菜单选择“检查”，可以进入到页面调试界面，在选项卡选择“Console”即可查看 Log 打印消息。



3.3 摄像监控

- 1) 将 USB 摄像头插入电脑 USB 接口。
- 2) 打开软件 ZXBeeGW 软件，选择 Camera 选项卡，在类型下拉菜单下，可以查看当前网关服务软件所支持的摄像头类型，如下图所示：



上图中，表明该软件支持 WEBCAMERA(笔记本摄像头和 USB 摄像头)、HK-DVR(海康数字硬盘录像机)、EasyN(易视 IP 摄像头)。

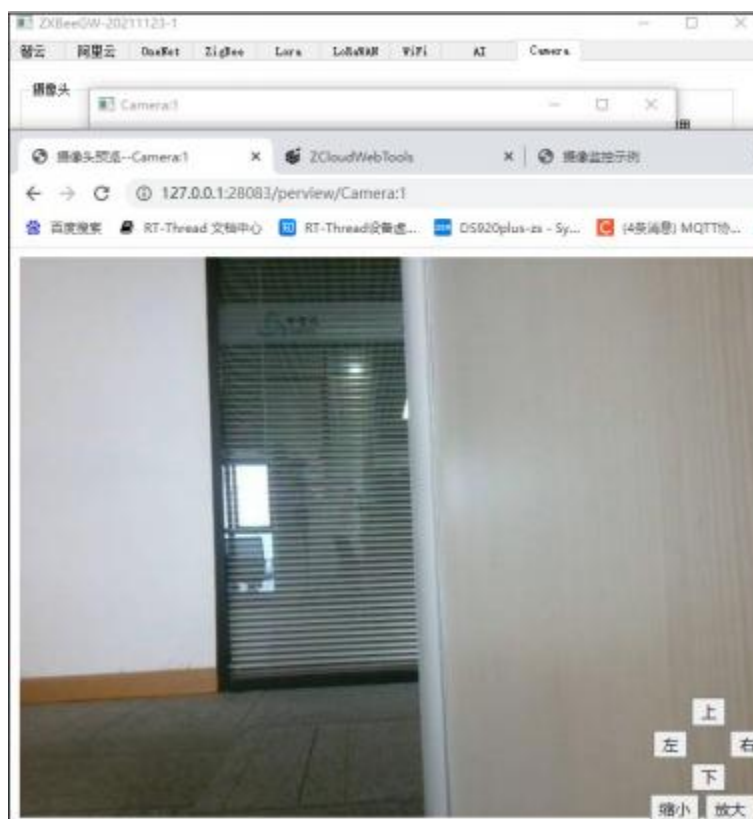
- 3) 类型选择：选择 WEBCAMERA(笔记本或 USB 摄像头)，不需填写地址、用户名和密码，勾选“启用”选项，如下图所示：



- 4) 点击“高级”选项，弹出对话框，如下图所示：



点击对话框上的“摄像头测试”按钮，软件会打开浏览器，并显示当前摄像头的画面，如下图所示：



5) 上述操作只是测试了摄像头和 PC 网关服务软件的连通性，还需要开启智云服务。选择“智云”选项卡，填入我司授权的用户账号和用户密钥(ID/KEY)，服务器地址填写“api.zhiyun360.com”，并启动远程服务、本地服务，即可将视频数据转发到云端数据中心和本地局域网内。



6) 用记事本打开 script.js 文件(js\script.js) , 修改智云帐号、密钥和摄像头的地址(这里默认使用内网地址) :

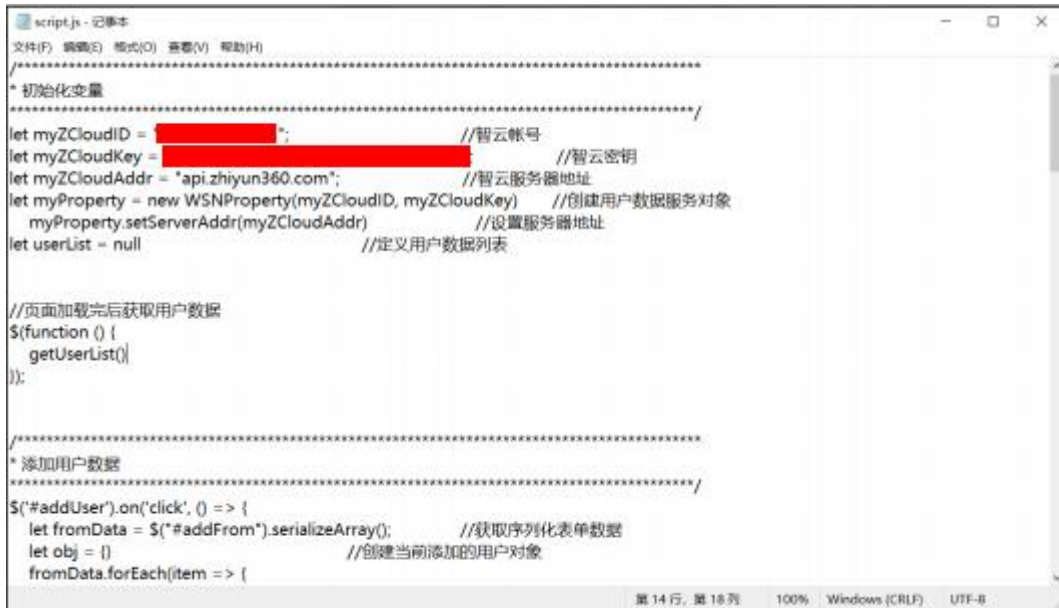


4) 用 Chrome 浏览器打开摄像监控示例程序 index.html, 查看摄像头视频流数据:



3.4 用户数据

1) 用记事本打开 script.js 文件(js\script.js), 修改智云帐号、密钥和智云服务器地址:



```
script.js - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

/* 初始化变量 */
let myZCloudID = " "; //智云帐号
let myZCloudKey = " "; //智云密钥
let myZCloudAddr = "api.zhiyun360.com"; //智云服务器地址
let myProperty = new WSNProperty(myZCloudID, myZCloudKey) //创建用户数据服务对象
myProperty.setServerAddr(myZCloudAddr) //设置服务器地址
let userList = null //定义用户数据列表

//页面加载完后获取用户数据
$(function () {
    getUserList();
});

/* 添加用户数据 */
$("#addUser").on('click', () => {
    let fromData = $("#addFrom").serializeArray(); //获取序列化表单数据
    let obj = {} //创建当前添加的用户对象
    fromData.forEach(item => {
```

2) 用户数据接口支持 key-value 字段数据的增删查改操作。用 Chrome 浏览器打开用户数据示例程序 index.html, 查看用户数据列表。



3) 添加用户数据: 点击“添加用户”, 输入卡号和姓名并选择性别, 然后点击“确认”进行数据的添加。

用户信息

卡号:

456

姓名:

小明

性别:

☒ 男

☐ 女

取消

确认

添加成功会有弹窗提示，点击“确定”：

此网页显示
添加用户成功!

确定

下图是添加后的列表：

用户数据示例

用户数据连接成功!

卡号	姓名	性别	操作
123	zongshikou	男	编辑 删除
456	小明	男	编辑 删除

添加用户

4) 编辑用户数据：下面来对第二条数据进行编辑：点击“编辑”按钮。

卡号	姓名	性别	操作
123	zoesion	男	编辑 删除
456	小明	男	编辑 删除

我们可以对卡号、姓名以及性别进行编辑(我这里改变了名字):

用户信息

卡号: 456

姓名: 小智

性别: ☒ 男 ☐ 女

取消

确认

点击“确认”就完成了编辑操作。同时会有弹窗提示: 编辑用户成功。随后点击“确定”:

此网页显示

编辑用户成功!

确定

下图是编辑好的列表:



5) 删除用户数据：下面来对第二条数据进行删除操作：点击“删除”按钮。

卡号	姓名	性别	操作
123	zonesion	男	编辑 删除
456	小智	男	编辑 删除

随后会有弹框提示：删除用户成功！点击“确定”按钮：



下图是删除后的列表：



五、常见问题

1. 智云帐号、智云密钥、设备地址全部填写正确，应用才能接收到数据。
2. 使用电脑自带摄像头不支持上下左右云台控制。

5 智云 Android 应用接口

一、实验目的

1. 掌握智云 Android 应用编程接口。
2. 掌握实时连接、历史数据、摄像监控、用户数据接口的编程。

二、实验环境

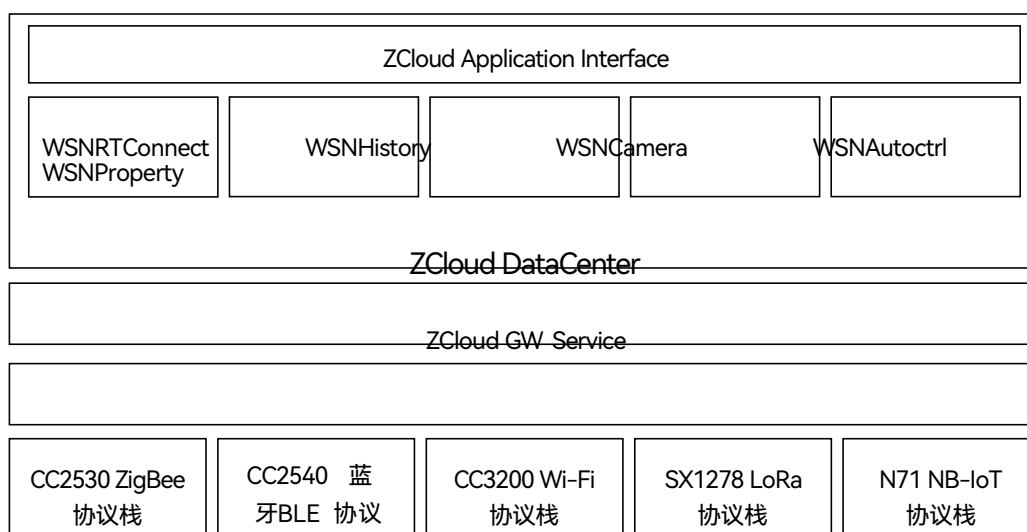
硬件环境:	PC 机 Pentium 处理器双核 2GHz 以上, 内存 4GB 以上
操作系统:	Windows7 64 位及以上操作系统
开发软件:	智云硬件物元仿真平台 HSIMS、ZCloudWebTools、ZXBeeGW4、Chrome 浏览器、Android Studio
实验配件:	USB 摄像头

三、实验内容

1. 智云框架

1.1 应用框架

智云平台提供五大应用接口供开发者使用, 包括: 实时连接(WSNRTConnect)、历史数据(WSNHistory)、摄像监控(WSNCamera)、自动控制(WSNAutoctrl)、用户数据(WSNProperty), 详细逻辑图如下:



1.2 功能设计

本实验通过 Android 的示例来介绍智云 API 接口的使用:

- 1) 实时连接示例: 实时采集温湿度传感器的温度和湿度数据; 远程控制灯的开关。
- 2) 历史数据示例: 实时采集光照度传感器的光照度数据; 获取光照度传感器的历史数据及曲线。
- 3) 摄像监控示例: 调用智云接口进行远程视频监控及控制。

4) 用户数据示例：调用智云接口对数据进行增加、修改和删除操作。相关的设备采用示例节点，使用到的协议如下：

节点名称	TYPE	参数	含义	读写权限	说明
示例节点	500	A0	光强值	R	光强值，整型：0~18000，单位：Lux
		A1	温度值	R	温度值，浮点型：0.1 精度，-40.0~70.0，单位：°C
		A2	湿度值	R	湿度值，浮点型：0.1 精度，0~100.0，单位：%RH
		A3	人体红外	R	监测是否周边有人体活动：1 表示检测到附近有人，0 表示未检测到有人体活动
		D0(OD0/CD0)	主动上报使能	RW	D0 的 bit0~bit7 对应 A0~A7 主动上报使能，0 表示不允许主动上报，1 表示允许主动上报
		D1(OD1/CD1)	开关控制	RW	D1 的 bit 位控制设备： bit0~bit1：LED 照明控制，LED 关(00)、LED 一级光(01)、LED 二级光(10)、LED 三级光(11) bit2：电机开关控制，关闭电机(0)、打开电机(1)
		V0	主动上报时间间隔	RW	A0~A7 主动上报时间间隔，默认 30，单位：S

2. 应用接口

针对 Android 移动应用程序开发，智云平台提供应用接口库：libWsnDroid-20230206.jar (app\libs\libWsnDroid-20230206.jar)，用户只需要在编写 Android 应用程序时，先导入该 jar 包，然后在代码中调用相应的方法即可。以下重点介绍实时连接、历史数据、摄像监控、用户数据接口。

2.1 实时连接接口

实时连接接口基于智云平台的消息推送服务，消息推送服务通过利用云端与客户端之间建立稳定、可靠的长连接来为开发者提供向客户端应用推送实时消息服务。智云消息推送服务针对物联网行业特征，支持多种推送类型：传感实时数据、执行控制命令、地理位置信息、SMS 短信消息等，同时提供用户信息及通知消息统计信息，方便开发者进行后续开发及运营。

基于 Android 的接口如下：

函数	参数说明	功能
new WSNRTConnect(String myZCloudID, String myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	建立实时数据实例，并初始化智云 ID 及密钥
connect()	无	建立实时数据服务连接
disconnect()	无	断开实时数据服务连接
setRTConnectListener(){ onConnect() onConnectLost(Throwable arg0) onMessageArrive(String mac, byte[] dat)	mac:传感器的 MAC 地址 dat: 发送的消息内容	设置监听，接收实时数据服务推送过来的消息： onConnect:连接成功操作 onConnectLost:连接失败操作 onMessageArrive:数据接收操作

}		
sendMessage(String mac, byte[] dat)	mac:传感器的 MAC 地址 dat: 发送的消息内容	发送消息
setServerAddr(String sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.2 历史数据接口

历史数据基于智云数据中心提供的智云数据库接口开发，智云数据库采用 Hadoop 后端分布式数据库集群，并且多机房自动冗余备份，自动读写分离，开发者不需要关注后端机器及数据库的稳定性、网络问题、机房灾难、单库压力等各种风险。物联网传感器数据可以在智云数据库永久保存，通过提供的简单的 API 编程接口可以完成与云存储服务器的数据连接、数据访问存储、数据使用等。

基于 Android 的接口如下：

函数	参数说明	功能
new WSNHistory(String myZCloudID, String myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化历史数据对象，并初始化智云 ID 及密钥
queryLast1H(String channel);	channel:传感器数据通道	查询最近 1 小时的历史数据
queryLast6H(String channel);	channel:传感器数据通道	查询最近 6 小时的历史数据
queryLast12H(String channel);	channel:传感器数据通道	查询最近 12 小时的历史数据
queryLast1D(String channel);	channel:传感器数据通道	查询最近1天的历史数据
queryLast5D(String channel);	channel:传感器数据通道	查询最近5天的历史数据
queryLast14D(String channel);	channel:传感器数据通道	查询最近14天的历史数据
queryLast1M(String channel);	channel:传感器数据通道	查询最近1月(30天) 的历史数据
queryLast3M(String channel);	channel:传感器数据通道	查询最近3月(90天) 的历史数据
queryLast6M(String channel);	channel:传感器数据通道	查询最近6月(180天) 的历史数据
queryLast1Y(String channel);	channel:传感器数据通道	查询最近1年(365天) 的历史数据
query();	无	获取所有通道最后一次数据
query(String channel);	channel:传感器数据通道	获取该通道下最后一次数据
query(String channel, String start, String end);	channel:传感器数据通道 start:起始时间 end:结束时间 时间为 ISO 8601 格式的日期, 例如: 2010-05-20T11:00:00Z	通过起止时间查询指定时间段的历史数据(根据时间范围默认选择采样间隔)
query(String channel, String start, String end, String interval);	channel:传感器数据通道 start:起始时间 end:结束时间 interval:采样点的时间间隔, 详见后续说明 时间为 ISO 8601 格式的日期, 例如: 2010-05-20T11:00:00Z	通过起止时间查询指定时间段指定时间间隔的历史数据
setServerAddr(String sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.3 摄像监控接口

智云平台提供对 IP 摄像头的远程采集控制接口，支持远程对视频图像进行实时采集、图像抓拍、控制云台转动等操作。

基于 Android 的接口如下：

函数	参数说明	功能
<code>new WSNCamera(String myZCloudID, String myZCloudKey);</code>	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化摄像头对象，并初始化智云 ID 及密钥
<code>initCamera(String myCameralP, String user, String pwd, String type);</code>	myCameralP: 摄像头 外网域名/IP 地址 user:摄像头用户名 pwd:摄像头密码 type:摄像头类型(F-Series、F3-Series、H3-Series) # 以上参数从摄像头手册获取	设置摄像头域名、用户名、密码、类型等参数
<code>openVideo();</code>	无	打开摄像头
<code>closeVideo();</code>	无	关闭摄像头
<code>control(String cmd);</code>	cmd:云台控制命令，参数如下： UP: 向上移动一次 DOWN: 向下移动一次 LEFT: 向左移动一次 RIGHT: 向右移动一次 HPATROL: 水平巡航转动 VPATROL: 垂直巡航转动 360PATROL: 360 度巡航转动	发指令控制摄像头云台转动
<code>checkOnline();</code>	无	检测摄像头是否在线
<code>snapshot();</code>	无	抓拍照片
<code>setCameraListener(){ onOnline(String myCameralP, boolean online) onSnapshot(String myCameralP, Bitmap bmp) onVideoCallBack(String myCameralP, Bitmap bmp) }</code>	myCameralP: 摄像头 外网域名/IP 地址 online:摄像头在线状态 (0/1) bmp:图片资源	监听摄像头返回数据： onOnline:摄像头在线状态返回 onSnapshot:返回摄像头截图 onVideoCallBack: 返回实时的摄像头视频图像
<code>freeCamera(String myCameralP);</code>	myCameralP: 摄像头 外网域名/IP 地址	释放摄像头资源
<code>setServerAddr(String sa)</code>	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.4 用户数据接口

智云用户数据接口提供私有的数据库使用权限，实现多客户端间共享的私有数据进行存储、查询和使用。私有数据存储采用 key-value 型数据库服务，编程接口更简单高效。

基于 Android 的接口如下：

函数	参数说明	功能
new WSNProperty(String myZCloudID,String myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化用户数据对象，并初始化智云 ID 及密钥
put(String key,String value);	key:名称 value:内容	创建用户应用数据
get();	无	获取所有的键值对
get(String key);	key:名称	获取指定key的value值
setServerAddr(String sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

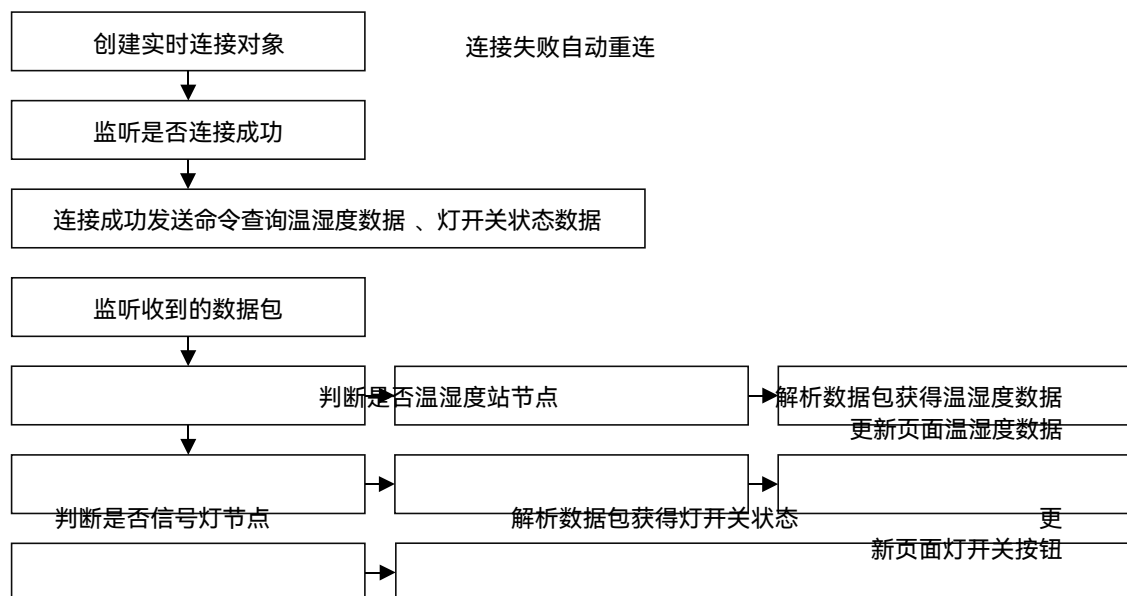
3. 程序分析

3.1 实时连接

RTConnectDemo-Droid 是一个简单的实时连接应用接口示例，实现了温湿度传感器数据的实时采集、灯光的开关控制等功能。

● 程序逻辑:

RTConnectDemo-Droid 程序逻辑如下：



● 程序分析:

RTConnectDemo-Droid 程序内容如下：

1) 主界面布局文件(app\src\main\res\layout\activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```

android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

```

```

<TextView
    android:paddingTop="30dp"
    android:paddingLeft="20dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="实时连接示例"
    android:visibility="gone"/>

```

```

<RelativeLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    >

```

```

<TextView android:id="@+id/tv_wd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="当前温度: ____

```

```

°C" />

```

```

<TextView android:id="@+id/tv_sd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="当前湿度: ____ %"
    android:layout_below="@id/tv_wd"
    />

```

```

<TextView android:id="@+id/tv_kg"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="灯光开关: "
    android:layout_below="@id/tv_sd"
    />

```

```

<Button android:id="@+id/btn_light"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

        android:text=""
        android:background="@mipmap/an_off"
        android:layout_below="@id/tv_sd"
        android:layout_toRightOf="@id/tv_kg"
        android:onClick="onSetLight"/>

    </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

2) 主程序文件(app\src\main\java\com\zonesion\rtconnectdemo\MainActivity.java)

```

package com.zonesion.rtconnectdemo;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Looper;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.zhiyun360.wsn.droid.WSNRTConnect;
import com.zhiyun360.wsn.droid.WSNRTConnectListener;

import java.text.DecimalFormat;

public class MainActivity extends AppCompatActivity implements WSNRTConnectListener{

    private WSNRTConnect wRTConnect; // 创建 WSNRTConnect 实例
    private String mMac = "00:11:22:33:44:55:66:77"; // 温湿度传感器节点 IEEE 地址
    private String ID = "123456789"; // 用户账号
    private String KEY = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // 用户密钥

    private DecimalFormat frum = new DecimalFormat("##.0"); // 温度值文本格式化:float 保留一位小数
    private String TEMPERATURE = "温度值: 0.0°C";
    private String HUMIDITY = "湿度值: 0.0%";

    private TextView temperatureTextView;
    private TextView humidityTextView;
    private Button mButtonLight;
    String D1 = "0";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

```

        temperatureTextView = (TextView) findViewById(R.id.tv_wd);
        humidityTextView = (TextView) findViewById(R.id.tv_sd);
        mButtonLight = (Button) findViewById(R.id.btn_light);

        wRTConnect = new WSNRTConnect(ID, KEY); // 创建 wRTConnect 链接, 并初始化
        wRTConnect.setServerAddr("api.zhiyun360.com:28081"); // 设置智云服务地址
        wRTConnect.setRTConnectListener(this);

        wRTConnect.connect(); // 与智云服务建立链接

    }

    public void onSetLight(View v){
        final String cmd;
        if (D1.equals("0")) {
            cmd = "{OD1=3,D1=?}";
        } else {
            cmd = "{CD1=3,D1=?}";
        }
        new Thread() {
            public void run() {
                wRTConnect.sendMessage(mMac, cmd.getBytes());
            }
        }.start();
    }

    @Override
    // 处理接收到的无线数据包
    public void onMessageArrive(String mac, byte[] dat) {
        onSensorData(mac, dat);
    }

    @Override
    // 智云服务断开链接的处理
    public void onConnectLost(Throwable arg0) {
        Toast.makeText(MainActivity.this, "断开连接",
            Toast.LENGTH_SHORT).show();
    }

    @Override
    // 智云服务链接建立成功处
    public void onConnect() {
        // TODO Auto-generated method stub
        Toast.makeText(MainActivity.this, "连接网关成功",
            Toast.LENGTH_SHORT).show();
        initWSN();
    }

```

```

    }
    private void initWSN(){

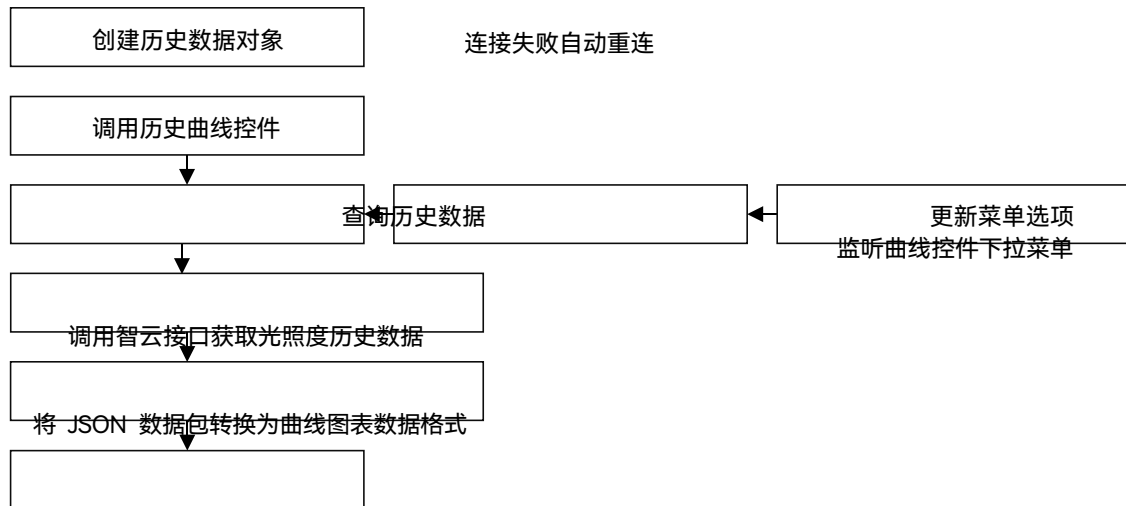
        new Thread(){
            public void run() {
                Looper.prepare();
                if (mMac.length() > 0) {
                    wRTConnect.sendMessage(mMac,
                        "{A1=?,A2=?,D1=?}".getBytes()); // 查询当前传感器数
                    值Toast.makeText(getApplicationContext(),
                        "{A1=?,A2=?,D1=?}", Toast.LENGTH_SHORT).show();
                }
                Looper.loop();
            };
        }.start();
    }

    @Override
    public void onDestroy() {
        wRTConnect.disconnect(); // 断开智云链
        接super.onDestroy();
    }

    // 解析收到的数据，获取当前温湿度值
    public void onSensorData(String mac, byte[] dat) {
        System.out.println("mac:" + mac + ",dat:" + new String(dat));
        String data = new String(dat);
        // data = data.substring(0);
        if (data.charAt(0) != '{') { // 判断首字符是否为 '{
            ' return;
        }
        if (data.charAt(data.length() - 1) != '}') // 判断尾字符是否为 '}'
            ' return;
        data = data.substring(1, data.length() - 1); // 截取{}内的字符串

        // 数据格式 A0=25,A1=34.0
        String[] tags = data.split(","); // 以','来分割字符串
        for (String tag : tags) { // 循环提取每项键值对，比如 A0=25
            String[] cv = tag.split("="); // 以'='来分割字符串
            if (cv.length != 2)
                continue;
            if (mMac.equalsIgnoreCase(mac)) { // 判断 Mac 地址
                if (cv[0].equals("A1")) { // 判断参数 A0
                    float v = Float.parseFloat(cv[1]);
                    TEMPERATURE = "当前温度: " + fnum.format(v) + "°C";
                }
            }
        }
    }

```

● 程序分析:

HistoryDemo-Droid 程序内容如下:

1) 主界面布局文件(app\src\main\res\layout\activity_main.xml)

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >
        <TextView android:id="@+id/tv_h"
            android:paddingTop="30dp"
            android:paddingLeft="20dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="30sp"
            android:text="历史数据示例"
            android:visibility="gone"/>

        <TextView android:id="@+id/tv_gz"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="25sp"
            android:layout_centerHorizontal="true"
            android:text="光照强度: ____ Lx"
            android:layout_below="@id/tv_h"
            />

        <Spinner
  
```

```

        android:id="@+id/sp_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_gz"

    ></Spinner>

    <LinearLayout
        android:id="@+id/layout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"

        android:layout_below="@id/sp_time">

    </LinearLayout>
</RelativeLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

2) 主程序文件(app\src\main\java\com\zonesion\historydemo\MainActivity.java)

```

package com.zonesion.historydemo;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.LinearLayout;
import android.widget.SimpleAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.zhiyun360.wsn.droid.WSNHistory;

import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.model.TimeSeries;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;
import org.json.JSONArray;

```

```

import org.json.JSONObject;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private String mMac = "00:11:22:33:44:55:66:77"; // 温湿度传感器节点 IEEE 地址
    private String ID = "123456789"; // 用户账号
    private String KEY = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // 用户密钥

    private String channel = mMac+"_A0"; // 数据通道
    private GraphicalView mGrapView; // 曲线显示视图
    private List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>(); // 历史数据
    list private String historyResult; // 历史数据
    private String[] at; // 时间存放数组
    private double[] value; // 数据存放数组
    private Date[] date; // 时间存放数组(格式化)

    LinearLayout layout;
    XYMultipleSeriesRenderer renderer;
    List<Date[]> x = new ArrayList<Date[]>(); // 点的横坐标(时间)
    List<double[]> values = new ArrayList<double[]>(); // 点的纵坐标(数值)

    private Date curDate = new Date(System.currentTimeMillis()); // 当前时间
    private SimpleDateFormat simpleDateFormat;

    private Handler mHandler;
    private WSNHistory wHistory; // 声明 WSNHistory 实例

    Spinner mSpinnerTime;
    private String[] starArray = {"最近 1 小时","最近 6 小时","最近 12 小时","最近 1 天","最近 1 周","最近 2 周","最近 1 月","最近 3 月","最近半年","最近 1 年"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
    }
}

```

```

        ArrayAdapter<String>
        =
        new
        ArrayAdapter<String>(this,android.R.layout.simple_spinner_dropdown_item,starArray);

        mSpinnerTime = (Spinner)findViewById(R.id.sp_time);
        mSpinnerTime.setAdapter(starAdapter);

        wHistory = new WSNHistory();
        wHistory.setIdKey(ID, KEY);
        wHistory.setServerAddr("api.zhiyun360.com:8080");

        // 设置画布及曲线属性
        layout = (LinearLayout) findViewById(R.id.layout1);
        renderer = setRenderer();

        // 查询近五天的历史数据
        SimpleDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss"); // 转换时间格式
        mHandler = new Handler() { // 消息处理

            @Override
            public void handleMessage(Message msg) {
                // TODO Auto-generated method stub
                list = (List<HashMap<String, Object>>) msg.obj;
                if (msg.what == 0) {

                    if (list.size() <= 0) {
                        showNullLine(curDate, 5); // 显示空的曲线

                    } else {
                        getAxis();
                        showLine(date, value); // 显示 5 天的数据曲线图
                        ((TextView)findViewById(R.id.tv_gz)).setText(" 光 照 强 度 :
"+list.get(list.size()- 1).get("Value")+"Lx");
                    }
                }
                super.handleMessage(msg);
            }
        };

        mSpinnerTime.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {

                new getHistoryDataAsyn().execute(new String(""+i));
                Toast.makeText(getApplicationContext(), "正在查询近"+starArray[i]+"数据! ",
                    Toast.LENGTH_LONG).show();
            }
        });

```

```

    }
    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {

    }
});
mSpinnerTime.setSelection(0);
}

// 将查询得到的 JSON 数据填充进 list
public List<HashMap<String, Object>> getList(String resultJson) {
    ArrayList<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();
    try {
        JSONObject jsonObj = new JSONObject(resultJson);
        JSONArray datapoints = jsonObj.getJSONArray("datapoints");
        for (int i = 0; i < datapoints.length(); i++) {
            JSONObject jsonObj = datapoints.getJSONObject(i);
            HashMap<String, Object> map = new HashMap<String, Object>();
            map.put("Value", jsonObj.getString("value"));
            map.put("At", jsonObj.getString("at"));
            list.add(map);
        }

    } catch (Exception e) { e.
        printStackTrace();
        // TODO: handle exception
    }
    return list;
}

// 将 list 中的数据填充进 Date[] date(时间存放数组) 和 double[] value(数据存放数组)
private void getAxis() {
    at = new String[list.size()];
    value = new double[list.size()];
    date = new Date[list.size()];
    for (int i = 0; i < list.size(); i++) {
        HashMap<String, Object> map = list.get(i);
        at[i] = (String) map.get("At");
        //try {
        try {
            date[i] = SimpleDateFormat.parse(at[i]);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        // } catch (ParseException e) {

```

```

        // TODO Auto-generated catch block
        //      e.printStackTrace();
        //    }
        value[i] = Double.parseDouble((String) map.get("Value"));
    }
}

public void showLine(Date[] date, double[] value) { // 显示指定时间内的曲线数据
    x.clear();
    values.clear();

    x.add(date);
    values.add(value);
    double minValue = value[0];
    double maxValue = value[0];
    for (int i = 0; i < value.length; i++) { // 得出数据中的最小值和最大值(确定纵坐标) if
        (value[i] < minValue) {
            minValue = value[i];
        }
        if (value[i] > maxValue) {
            maxValue = value[i];
        }
    }
    double scale = (maxValue - minValue) / 6;
    if (scale < 0.1) scale = 6;
    setChartSettings(renderer, x.get(0)[0].getTime(),
        x.get(0)[date.length - 1].getTime(), (minValue - scale),
        (maxValue + scale));

    showGraphicalView();
}

public void showNullLine(Date date, long daySumNum) { // 显示空曲线
    setChartSettings(renderer, (date.getTime() - daySumNum * 24 * 60 * 60
        * 1000), date.getTime(), -5, 30);
    mGrpView = ChartFactory.getTimeChartView(getBaseContext(),
        buildDateDataset(x, values), renderer, "M/d-H:mm");
    showGraphicalView();
}

// 设置 layout 的视图, 显示
GraphicalView private void
showGraphicalView() {
    mGrpView = ChartFactory.getTimeChartView(getBaseContext(),
        buildDateDataset(x, values), renderer, "M/d-H:mm");
    LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(

```

```

        LinearLayout.LayoutParams.MATCH_PARENT, LinearLayout.LayoutParams.MATCH_PARENT);
        lp.weight = 1;
        layout.removeAllViews();
        layout.addView(mGrapView, lp);
    }

```

// XYMultipleSeriesRenderer 的基本属性设置

```

private XYMultipleSeriesRenderer setRenderer() {
    // TODO Auto-generated method stub
    XYMultipleSeriesRenderer renderer = new XYMultipleSeriesRenderer(); // 坐标绘制
    renderer.setAxisTitleTextSize(16); // 坐标轴标题字体大小
    renderer.setLabelsTextSize(15); // 数轴标签字体大小
    renderer.setMargins(new int[] { 20, 50, 20, 20 }); // 图形四边距(上, 左, 下, 右)
    renderer.setXLabels(10); // x 轴的近似坐标数
    renderer.setYLabels(10); // y 轴的近似坐标数
    renderer.setMarginsColor(Color.WHITE); // 四边距颜色
    renderer.setZoomButtonsVisible(true); // 显示变焦按钮(放大, 缩小)
    renderer.setChartTitle("历史数据曲线图"); // 图表标题
    renderer.setXTitle("时间"); // x 轴标题
    renderer.setYTitle("数值"); // y 轴标题
    renderer.setAxesColor(Color.BLACK); // x, y 轴颜色
    renderer.setLabelsColor(Color.GREEN); // x, y 轴标题颜色
    // renderer.setXLabelsColor(Color.BLACK);
    renderer.setShowGrid(true); // 显示曲线图中的表格
    // renderer.setGridColor(Color.BLACK);
    // 设置曲线图背景颜色
    renderer.setApplyBackgroundColor(true);
    renderer.setBackgroundColor(Color.WHITE);

    XYSeriesRenderer r = new XYSeriesRenderer(); // 曲线图中每个点的绘制
    r.setColor(Color.rgb(30, 144, 255)); // 设置点的颜色
    // r.setPointStyle(PointStyle.CIRCLE); //设置点的样式
    r.setFillPoints(false);
    r.setLineWidth(1); // 设置曲线的宽度(粗细)

    renderer.addSeriesRenderer(r); // 将要绘制的点添加到坐标绘制中
    return renderer;
}

```

// 构建和时间有关的 XYMultipleSeriesDataset

```

private XYMultipleSeriesDataset buildDateDataset(List<Date[]> xValues,
                                                List<double[]> yValues) {
    XYMultipleSeriesDataset dataset = new XYMultipleSeriesDataset();
    TimeSeries series = new TimeSeries("历史数据"); // 构建时间序列
    if (xValues.size() <= 0) {

```

```

        series.add((curDate.getTime() - 5 * 24 * 60 * 60 * 1000), 0);
    } else {
        Date[] xV = xValues.get(0);
        double[] yV = yValues.get(0);
        int seriesLength = xV.length;
        for (int k = 0; k < seriesLength; k++) {
            series.add(xV[k], yV[k]);
        }
    }
    dataset.addSeries(series);
    return dataset;
}

// XYMultipleSeriesRenderer 的横坐标和纵坐标设置
private void setChartSettings(XYMultipleSeriesRenderer renderer,
                               double xMin, double xMax, double yMin, double yMax) {
    renderer.setXAxisMin(xMin); // x 轴最小值
    renderer.setXAxisMax(xMax); // x 轴最大值
    renderer.setYAxisMin(yMin);
    renderer.setYAxisMax(yMax);
}

```

```

class getHistoryDataAsync extends AsyncTask<String, String, Boolean> {
    MyDialog dialog = new MyDialog(MainActivity.this);

    // 在 onPreExecute()完成后立即执行，用于执行较为费时的操作
    @Override
    protected Boolean doInBackground(String... arg0) {
        try {
            if (arg0[0].equals("0")) {
                historyResult = wHistory.queryLast1H(channel);
            } else if (arg0[0].equals("1")) {
                historyResult = wHistory.queryLast6H(channel);
            } else if (arg0[0].equals("2")) {
                historyResult = wHistory.queryLast12H(channel);
            } else if (arg0[0].equals("3")) {
                historyResult = wHistory.queryLast1D(channel);
            } else if (arg0[0].equals("4")) {
                historyResult = wHistory.queryLast5D(channel);
            } else if (arg0[0].equals("5")) {
                historyResult = wHistory.queryLast14D(channel);
            } else if (arg0[0].equals("6")) {
                historyResult = wHistory.queryLast1M(channel);
            } else if (arg0[0].equals("7")) {
                historyResult = wHistory.queryLast3M(channel);
            }
        }
    }
}

```

```

        }else if (arg0[0].equals("8")) {
            historyResult = wHistory.queryLast6M(channel);
        }else {
            historyResult = wHistory.queryLast1Y(channel);
        }
        list = getList(historyResult);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return true;
}

// 当后台操作结束时，此方法将会被调用
@Override
protected void onPostExecute(Boolean paramBoolean) {
    dialog.cancel();
    mHandler.obtainMessage(0, list).sendToTarget();
}

// 在 execute(para)方法被调用后立即执行
@Override
protected void onPreExecute() {
    dialog.show();
}
}
}

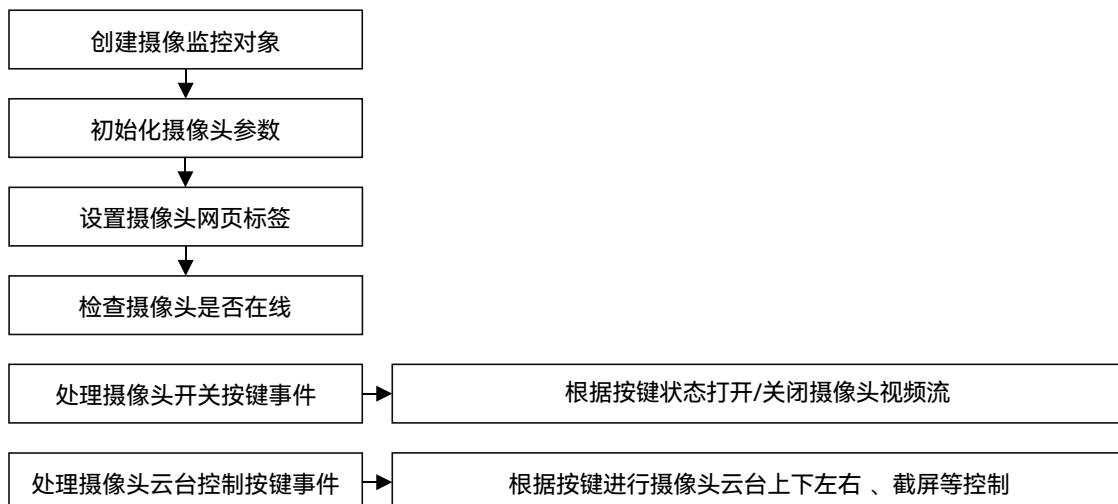
```

3.3 摄像监控

CameraDemo-Droid 是一个简单的摄像监控应用接口示例，实现了网络实时摄像头视频流监控和摄像头云台转动控制等功能。

● 程序逻辑:

CameraDemo-Droid 程序逻辑如下:



● 程序分析:

CameraDemo-Droid 程序内容如下:

1) 主界面布局文件(app\src\main\res\layout\activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >

        <TextView
            android:id="@+id/tv_manual"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="clip_vertical"
            android:text="@string/s_tv_manual"
            android:textSize="15sp"
            android:textColor="@android:color/white"
            android:visibility="visible" />

        <ImageView
            android:id="@+id/iv_video_display"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"

            android:layout_centerInParent="true"
            />
    </FrameLayout>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >

        <RelativeLayout
            android:layout_width="180dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical">

            <ImageButton
                android:id="@+id/ib_on_off"
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:background="@mipmap/button_on"
        android:onClick="onOff" />

```

```

<ImageButton
    android:id="@+id/ib_up"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/ib_on_off"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="30dp"
    android:background="@mipmap/button_up_disable"
    android:onClick="buttonClick" />

```

```

<ImageButton
    android:id="@+id/ib_down"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/ib_up"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="30dp"
    android:background="@mipmap/button_down_disable"
    android:onClick="buttonClick" />

```

```

<ImageButton
    android:id="@+id/ib_left"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/ib_up"
    android:layout_marginLeft="10dp"
    android:background="@mipmap/button_left_disable"
    android:onClick="buttonClick" />

```

```

<ImageButton
    android:id="@+id/ib_right"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignTop="@id/ib_left"
    android:layout_marginRight="10dp"
    android:background="@mipmap/button_right_disable"
    android:onClick="buttonClick" />

```

```

        <ImageButton
            android:id="@+id/ib_snapshot"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/ib_down"
            android:layout_marginTop="30dp"
            android:layout_centerHorizontal="true"
            android:background="@mipmap/button_snapshot_disable"
            android:onClick="buttonClick" />
    </RelativeLayout>
</FrameLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

2) 主程序文件(app\src\main\java\com\zonesion\camerademo\MainActivity.java)

```

package com.zonesion.camerademo;

import androidx.appcompat.app.AppCompatActivity;
import android.Manifest;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import com.zhiyun360.wsn.droid.WSNCamera;
import java.sql.Date;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Locale;

public class MainActivity extends AppCompatActivity implements WSNCamera.WSNCameraListener {
    private String ID = "12345678";// 用户账号
    private String KEY = "ABCDEFGHJKLMNOPQRSTUVWXYZ";// 用户密钥
    private String gw_ip = "127.0.0.1:28083";
    private String cam = "Camera:1";// 摄像头地址 1
    private ImageView ivVideoDisplay;// 视频监控区域图像视图
    private ImageButton ibOnOff;// 开关按钮
    private ImageButton ibUp;// 向上按钮
    private ImageButton ibDown;// 向下按钮
    private ImageButton ibLeft;// 向左按钮
    private ImageButton ibRight;// 向右按钮

```

```

private ImageButton ibSnapshot;// 抓拍按钮
private TextView tvMan;// 使用说明显示
boolean bOnOff;

WSNCamera mWSNCamera;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mWSNCamera = new WSNCamera(ID, KEY);
    mWSNCamera.initCamera(gw_ip, cam);
    mWSNCamera.setCameraListener(this);
    // 通过资源 id 找到视图控件对象
    ibOnOff = (ImageButton) findViewById(R.id.ib_on_off);
    ibUp = (ImageButton) findViewById(R.id.ib_up);
    ibDown = (ImageButton) findViewById(R.id.ib_down);
    ibLeft = (ImageButton) findViewById(R.id.ib_left);
    ibRight = (ImageButton) findViewById(R.id.ib_right);

    ibSnapshot = (ImageButton) findViewById(R.id.ib_snapshot);
    ivVideoDisplay = (ImageView) findViewById(R.id.iv_video_display);
    tvMan = (TextView) findViewById(R.id.tv_manual);
}
@Override
protected void onDestroy(){
    mWSNCamera.closeVideo();
    super.onDestroy();
}
// 处理开关方法
public void processOnOff() {
    if (bOnOff) {
        mWSNCamera.closeVideo();
        ivVideoDisplay.setVisibility(View.GONE);
        tvMan.setVisibility(View.VISIBLE);// 设置使用说明显示为可见
        ibOnOff.setImageDrawable(getResources().getDrawable(
            R.mipmap.button_on));// 开关按钮图片设置为开
        // 其他按钮图片设置为失效
        ibUp.setImageDrawable(getResources().getDrawable(
            R.mipmap.button_up_disable));
        ibDown.setImageDrawable(getResources().getDrawable(
            R.mipmap.button_down_disable));
        ibLeft.setImageDrawable(getResources().getDrawable(
            R.mipmap.button_left_disable));
        ibRight.setImageDrawable(getResources().getDrawable(

```

```

        R.mipmap.button_right_disable));

        ibSnapshot.setImageDrawable(getResources().getDrawable(
            R.mipmap.button_snapshot_disable));
        bOnOff = false; // 开关标志设为
关 } else {
    mWSNCamera.openVideo();
    ivVideoDisplay.setVisibility(View.VISIBLE);
    tvMan.setVisibility(View.INVISIBLE); // 设置使用说明显示为不可见

    ibOnOff.setImageDrawable(getResources().getDrawable(
        R.mipmap.button_off)); // 开关按钮图片设置为关
    // 其他按钮图片设置为使能
    ibUp.setImageDrawable(getResources().getDrawable(
        R.mipmap.button_up));
    ibDown.setImageDrawable(getResources().getDrawable(
        R.mipmap.button_down));
    ibLeft.setImageDrawable(getResources().getDrawable(
        R.mipmap.button_left));
    ibRight.setImageDrawable(getResources().getDrawable(
        R.mipmap.button_right));
    ibSnapshot.setImageDrawable(getResources().getDrawable(
        R.mipmap.button_snapshot));
    bOnOff = true; // 开关标志设为
开 }
}

// 视频流输出回调方
法@Override
public void onVideoCallBack(WSNCamera cam, Bitmap bitmap) {
    ivVideoDisplay.setImageBitmap(bitmap);
}

public void onOff(View view) { p
    rocessOnOff(); // 处理开关
}

// 在布局文件中设置的其他按钮点击方
法public void buttonClick(View view) {
    if (bOnOff) {
        switch (view.getId()) {
            case R.id.ib_up:
                mWSNCamera.control("UP");
                break;
            case R.id.ib_down:
                mWSNCamera.control("DOWN");

```

```

        break;
    case R.id.ib_left:
        mWSNCamera.control("LEFT");
        break;
    case R.id.ib_right:
        mWSNCamera.control("RIGHT");
        break;
    case R.id.ib_snapshot:
        if (writeSd) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        Bitmap bmp = mWSNCamera.snapshot();
                        saveBitmap(bmp);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        } else {
            MainActivity.this.requestPermissions(
                new
String[] {Manifest.permission.WRITE_EXTERNAL_STORAGE}, 222);
        }
        break;
    default:
        break;
    }
}

boolean writeSd = false;

private boolean hasAllPermissionsGranted( int[] grantResults) {
    for (int grantResults : grantResults) {
        if (grantResult == PackageManager.PERMISSION_DENIED) {
            return false;
        }
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if (!hasAllPermissionsGranted(grantResults)) {
        return;
    }
}

```

```

switch (requestCode) {
    case 222:
        //Toast.makeText(getApplicationContext(), " 已申请权限", Toast.LENGTH_SHORT).show();
        writeSd = true;

        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Bitmap bmp = mWSNCamera.snapshot();

                    saveBitmap(bmp);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).start();
        break;
    }
}

// 保存图片方法
public boolean saveBitmap(Bitmap bitmap) {
    final String sPictureName = new SimpleDateFormat("yyyy-MM-dd-HH-mm-ss",
        Locale.CHINA).format(new Date(System.currentTimeMillis()))
        + ".jpeg";// 用日期时间作为图片名, 图片扩展名为 jpeg
    File fSDCardPath = Environment.getExternalStorageDirectory();// 获取 SDCard 路
    径final File fDirectory = new File(fSDCardPath, "VideoMonitor");// 图片存放目录
    try {
        if (!fDirectory.exists()) {
            fDirectory.mkdir();// 创建文件目录
        }
        File file = new File(fDirectory, sPictureName);
        if (!file.exists()) {
            file.createNewFile();// 创建文件
        }
        FileOutputStream out = new FileOutputStream(file);// 文件输出流
        bitmap.compress(Bitmap.CompressFormat.JPEG, 90, out);// 按 90%的图片压缩率将图片输出到文件中
        out.flush();// 写缓冲
        out.close();// 关闭文件输出流
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(MainActivity.this,
                    " 图片" + sPictureName + "保存在" + fDirectory.toString(),
                    Toast.LENGTH_LONG).show();
            }
        });
    }
}

```

```

        return true;
    } catch (IOException e) {
        e.printStackTrace();
    }
    return false;
}
}

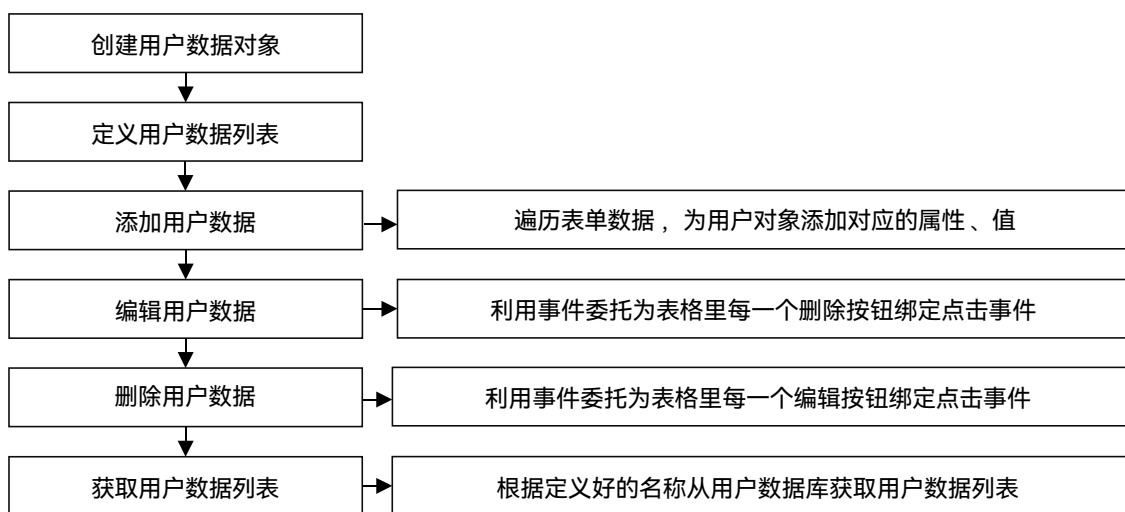
```

3.4 用户数据

WSNPropertyDemo-Droid 是一个简单的用户数据应用接口示例，实现了用户数据的“增删改”的功能。

● 程序逻辑:

WSNPropertyDemo-Droid 程序逻辑如下:



● 程序分析:

WSNPropertyDemo-Droid 程序内容如下:

1) 主界面布局文件(app\src\main\res\layout\activity_main.xml)

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView android:id="@+id/tv_title"
            android:paddingTop="30dp"

```

```

        android:paddingLeft="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="用户数据示例"
        android:visibility="gone"/>

<EditText android:id="@+id/et_id"
    android:layout_width="80dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/tv_title"
    android:hint="卡号"/>
<EditText android:id="@+id/et_name"
    android:layout_width="80dp"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/et_id"
    android:layout_below="@id/tv_title"
    android:hint="姓名"/>
<RadioGroup
    android:id="@+id/rg"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_below="@id/tv_title"
    android:layout_toRightOf="@id/et_name"
    android:layout_alignBottom="@id/et_name"
    >
<RadioButton android:id="@+id/cb_nan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="男"/>
<RadioButton android:id="@+id/cb_nv"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:text="女"/>
</RadioGroup>
<Button
    android:id="@+id/btn_add"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/rg"
    android:layout_below="@id/tv_title"
    android:onClick="addUser"

```

```

        android:text="添加"/>

        <ListView android:id="@+id/lv_user"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/et_id"
            />
    </RelativeLayout>

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

2) 主程序文件(app\src\main\java\com\zonesion\wsnpropertydemo\MainActivity.java)

```

package com.zonesion.wsnpropertydemo;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RadioGroup;
import android.widget.SimpleAdapter;
import android.widget.Toast;
import com.zhiyun360.wsn.droid.WSNProperty;
import com.zhiyun360.wsn.droid.WSNRTConnect;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {
    private String ID = "123456789"; // 用户账号
    private String KEY = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // 用户密钥

    WSNProperty mWSNProperty;
    List<Map<String,Object>> mList = new ArrayList();
    MySimpleAdapter mSa ;

```

```

ListView mLvUser;

JSONArray mJaUser = new JSONArray();

EditText mETId, mETName;
RadioGroup mRGSex;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mETId = (EditText)findViewById(R.id.et_id);
    mETName = (EditText)findViewById(R.id.et_name);
    mRGSex = (RadioGroup)findViewById(R.id.rg);

    mWSNProperty = new WSNProperty(ID, KEY);
    mWSNProperty.setServerAddr("api.zhiyun360.com:8080");

    mSa = new MySimpleAdapter(this, mList, R.layout.item,
        new String[]{"id", "name", "sex"}, new int[] {R.id.tv_id, R.id.tv_name, R.id.tv_sex});

    mLvUser = (ListView)findViewById(R.id.lv_user);
    mLvUser.setAdapter(mSa);
    new loadDataAsyn().execute(new String(""));
}

public void addUser(View v){

    String id = mETId.getText().toString().trim();
    String name = mETName.getText().toString().trim();
    String sex = "男";
    if (R.id.cb_nv == mRGSex.getCheckedRadioButtonId()){
        sex = "女";
    }
    if (id.length() == 0 || name.length() == 0){
        Toast.makeText(this, "卡号或用户名不能为空", Toast.LENGTH_SHORT).show();
        return;
    }
    JSONObject jo = new JSONObject();
    try {
        jo.put("id", id);
        jo.put("name", name);
        jo.put("sex", sex);
        mJaUser.put(mJaUser.length(), jo);
        String a = mJaUser.toString();
    }
}

```

```

        new saveDataAsyn().execute(a);
        mETId.setText("");
        mETName.setText("");
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

class MySimpleAdapter extends SimpleAdapter{
    public MySimpleAdapter(Context c, List l, int la, String[] m, int[] r){
        super(c, l, la, m, r);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = super.getView(position, convertView, parent);
        Button btn_del = (Button)v.findViewById(R.id.btn_del);
        btn_del.setTag(position);
        btn_del.setOnClickListener(mOnDelClickListener);
        Button btn_edit = (Button)v.findViewById(R.id.btn_edit);
        btn_edit.setTag(position);
        btn_edit.setOnClickListener(mOnEditClickListener);

        return v;
    }
}

View.OnClickListener mOnEditClickListener = new View.OnClickListener()
{ @Override
    public void onClick(View view) {
        final int pos = (int)view.getTag();
        AlertDialog.Builder dlg = new AlertDialog.Builder(MainActivity.this);
        final View v = LayoutInflater.from(MainActivity.this).inflate(R.layout.edit, null);
        final EditText et_name =(EditText) v.findViewById(R.id.et_name);
        final EditText et_id = (EditText)v.findViewById(R.id.et_id);
        final RadioGroup rg = v.findViewById(R.id.rg);
        et_name.setText((String)mList.get(pos).get("name"));
        et_id.setText((String)mList.get(pos).get("id"));
        String sex = (String)mList.get(pos).get("sex");
        if (sex.equals("女")){
            rg.check(R.id.cb_nv);
        }

        dlg.setTitle("修改数据");
        dlg.setView(v);
        dlg.setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override

```

```

        public void onClick(DialogInterface dialogInterface, int i) {
            String name = et_name.getText().toString().trim();
            String id = et_id.getText().toString().trim();
            String sex = "男";
            if (R.id.cb_nv == rg.getCheckedRadioButtonId()) sex="女";

            if (id.length() == 0 || name.length() == 0){
                Toast.makeText(MainActivity.this, "卡号或用户名不能为空", Toast.LENGTH_SHORT).show();
                return;
            }

            try {
                JSONObject ja = mJaUser.getJSONObject(pos);
                ja.put("name", name);
                ja.put("id", id);
                ja.put("sex", sex);
                String a = mJaUser.toString();
                new saveDataAsyn().execute(a);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });

    dlg.show();
}

};

View.OnClickListener mOnDelClickListener = new
View.OnClickListener() { @Override
public void onClick(View view) {
    JSONArray ja = new JSONArray();
    int pos = (int)view.getTag();
    for (int i=0; i<mJaUser.length(); i++){
        if (i != pos) {
            JSONObject jo = null;
            try {
                jo = mJaUser.getJSONObject(i);
                ja.put(jo);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
    String a = ja.toString();

```

```

        new saveDataAsyn().execute(a);
    }
};

class saveDataAsyn extends AsyncTask<String, String, Boolean> {
    MyDialog dialog = new MyDialog(MainActivity.this);

    // 在 onPreExecute()完成后立即执行，用于执行较为费时的操作
    @Override
    protected Boolean doInBackground(String... arg0) {

        try {
            mWSNProperty.put("userList", arg0[0]);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }

    // 当后台操作结束时，此方法将会被调用
    @Override
    protected void onPostExecute(Boolean paramBoolean) {
        dialog.cancel();
        new loadDataAsyn().execute(new String(""));
    }

    // 在 execute(para)方法被调用后立即执行
    @Override
    protected void onPreExecute() {
        dialog.show();
    }
}

```

```

class loadDataAsyn extends AsyncTask<String, String, Boolean> {

    MyDialog dialog = new MyDialog(MainActivity.this);

    // 在 onPreExecute()完成后立即执行，用于执行较为费时的操作
    @Override
    protected Boolean doInBackground(String... arg0) {

        try {
            String r = mWSNProperty.get("userList");
            mList.clear();
            if (r!=null && r.length()>0){

```

```

JSONArray ja = new JSONArray(r);
mJaUser = ja;
for (int i=0; i<ja.length(); i++){
    JSONObject jo = ja.getJSONObject(i);
    String id = jo.getString("id");
    String name = jo.getString("name");
    String sex = jo.getString("sex");

    Map<String,Object> it = new HashMap<String, Object>();
    it.put("id", id);
    it.put("name", name);
    it.put("sex", sex);
    mList.add(it);
}

} catch (Exception e) { e
    .printStackTrace();
}

return true;
}

// 当后台操作结束时，此方法将会被调用
@Override
protected void onPostExecute(Boolean paramBoolean) {
    dialog.cancel();
    mSa.notifyDataSetChanged();
}

// 在 execute(para)方法被调用后立即执行
@Override
protected void onPreExecute() {
    dialog.show();
}
}
}

```

四、实验步骤

1. 实验部署

本实验基于 xLab 实验平台来构建一个智云接口项目，可通过硬件物元仿真平台软件创建虚拟的硬件，也可以使用 xLab 实验平台真实的硬件(本实验默认通过硬件物元仿真平台软件创建)。

Android Studio 软件的路径: DISK-Packages\09-Android 应用技术\android-studio\Android.zip，软件

的安装与使用可参考《Android 应用技术》课程。

2. 原型搭建

2.1 创建传感器

可参考实验 4 的原型仿真步骤来创建传感器。

2.2 创建项目

可参考实验 4 的原型仿真步骤来创建项目。

2.3 运行项目

可参考实验 4 的原型仿真步骤来运行项目。

3. 应用体验

3.1 实时连接

1) 打开本实验目录下的工程 “RTConnectDemo-Droid”。

2) 用记事本打开MainActivity.java(app\src\main\java\com\zonesion\rtconnectdemo\MainActivity.java), 填写授权的智云帐号、密钥和节点的 MAC 地址:



```
public class MainActivity extends AppCompatActivity implements WSNRTConnectListener{

    private WSNRTConnect wRTConnect; // 创建WSNRTConnect实例
    private String mMac = "01:12:4B:00:DD:87:68:42"; // 传感器节点IEEE地址
    private String ID = " "; // 用户账号
    private String KEY = " "; // 用户密钥

    private DecimalFormat fnum = new DecimalFormat("##.0"); // 温度值文本格式化:float保留一位小数
    private String TEMPERATURE = "温度值: 0.0°C";
    private String HUMIDITY = "湿度值: 0.0%";

    private TextView temperatureTextView;
    private TextView humidityTextView;
    private Button mButtonLight;
    String D1 = "0";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        temperatureTextView = (TextView) findViewById(R.id.tv_wd);
        humidityTextView = (TextView) findViewById(R.id.tv_sd);
        mButtonLight = (Button) findViewById(R.id.btn_light);

        wRTConnect = new WSNRTConnect(ID, KEY); // 创建wRTConnect链接, 并初始化
    }
}
```

3) 用 Android Studio 软件打开实时连接示例程序 RTConnectDemo-Droid, 查看实时数据信息及 RGB 灯的开关操作。



3.2 历史数据

- 1) 打开本实验目录下的工程 “HistoryDemo-Droid”文件夹。
- 2) 用记事本打开 MainActivity.java(app\src\main\java\com\zonesion\historydemo\MainActivity.java) , 修改智云帐号、密钥和 ZigBee 无线节点的MAC 地址:

```

MainActivity.java - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
public class MainActivity extends AppCompatActivity {

    private String mMac = "01:12:4B:00:DD:87:68:42" // 传感器节点IEEE地址
    private String ID = " "; // 用户账号
    private String KEY = " "; // 用户密钥

    private String channel = mMac + "_A0"; // 数据通道
    private GraphicalView mGrapView; // 曲线显示视图
    private List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>> (); // 历史数据list
    private String historyResult; // 历史数据
    private String[] at; // 时间存放数组
    private double[] value; // 数据存放数组
    private Date[] date; // 时间存放数组 (格式化)

    LinearLayout layout;
    XYMultipleSeriesRenderer renderer;
    List<Date[]> x = new ArrayList<Date[]> (); // 点的横坐标 (时间)
    List<double[]> values = new ArrayList<double[]> (); // 点的纵坐标 (数值)

    private Date curDate = new Date(System.currentTimeMillis()); // 当前时间
    private SimpleDateFormat simpleDateFormat;

    private Handler mHandler;
    private WSNHistory wHistory; // 声明WSNHistory实例

    Spinner mSpinnerTime;

```

- 3) 用 Android Studio 软件打开历史数据示例程序 HistoryDemo-Droid, 查看光照强度实时数据信息及历史曲线信息, 可以通过曲线的下拉选项选择查看其它时间的历史数据曲线。



3.3 摄像监控

- 1) 将 USB 摄像头插入电脑 USB 接口。
- 2) 打开软件 ZXBeeGW 软件，选择 Camera 选项卡，在类型下拉菜单下，可以查看当前网关服务软件所支持的摄像头类型，如下图所示：



上图中，表明该软件支持 WEBCAMERA(笔记本摄像头和 USB 摄像头)、HK-DVR(海康数字硬盘录像机)、EasyN(易视 IP 摄像头)。

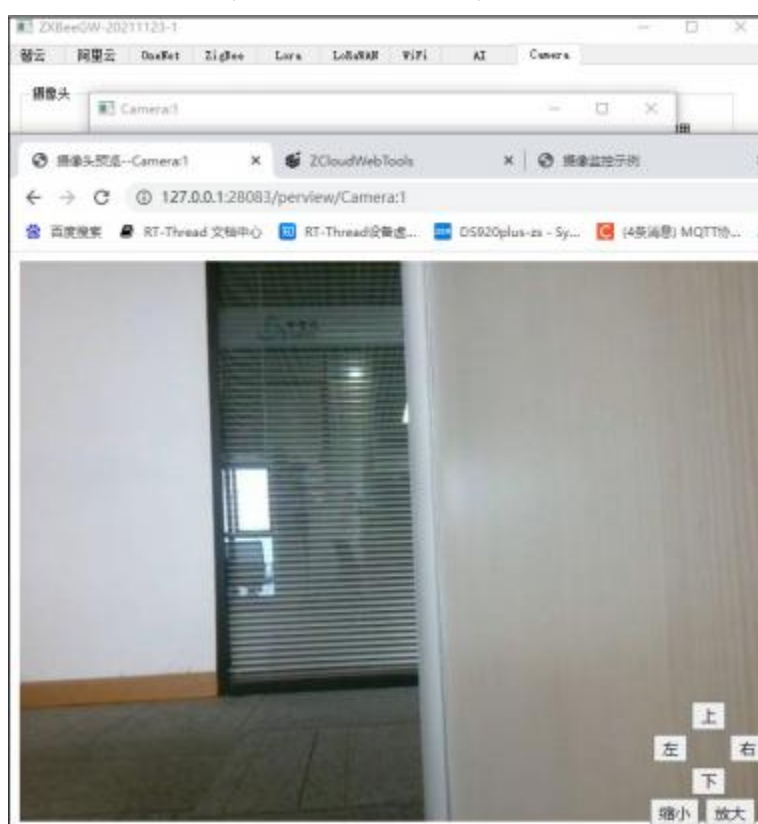
- 3) 类型选择：选择 WEBCAMERA(笔记本或 USB 摄像头)，不需填写地址、用户名和密码，勾选“启用”选项，如下图所示：



4) 点击“高级”选项，弹出对话框，如下图所示：



点击对话框上的“摄像头测试”按钮，软件会打开浏览器，并显示当前摄像头的画面，如下图所示：



5) 上述操作只是测试了摄像头和 PC 网关服务软件的连通性，还需要开启智云服务。选择“智云”选项卡，填入我司授权的用户账号和用户密钥(ID/KEY)，服务器地址填写“api.zhiyun360.com”，并启动远程服务、本地服务，即可将视频数据转发到云端数据中心和本地局域网内。



6) 用记事本打开 MainActivity.java (app\src\main\java\com\zonesion\camerademo\MainActivity.java), 修改智云帐号、密钥和摄像头的地址(这里默认使用内网地址):

```

MainActivity.java - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

private String ID = " "; // 用户账号
private String KEY = " "; // 用户密钥

private String gw_ip = "192.168.100.200:28083";
private String cam = "Camera:1"; // 摄像头地址1

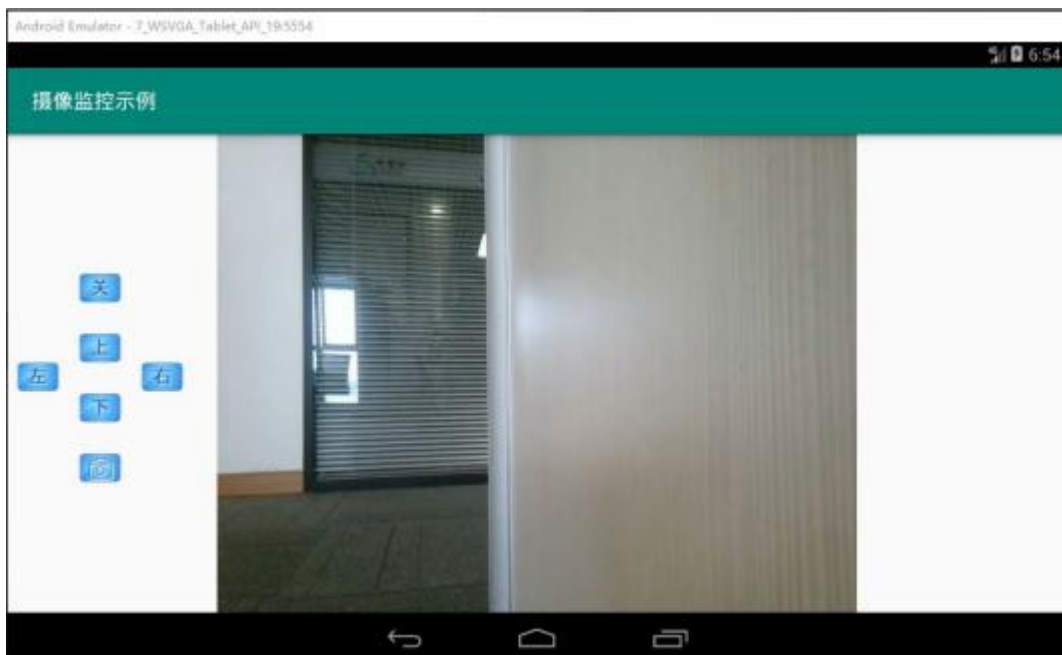
private ImageView ivVideoDisplay; // 视频监控区域图像视图
private ImageButton ibOnOff; // 开关按钮
private ImageButton ibUp; // 向上按钮
private ImageButton ibDown; // 向下按钮
private ImageButton ibLeft; // 向左按钮
private ImageButton ibRight; // 向右按钮

private ImageButton ibSnapshot; // 抓拍按钮
private TextView tvMan; // 使用说明显示
boolean bOnOff;

WSNCamera mWSNCamera;
@Override

```

7) 用 Android Studio 软件打开摄像监控示例程序 CameraDemo-Droid, 查看摄像头视频流数据:



3.4 用户数据

1) 用记事本打开 MainActivity.java 文件

(app\src\main\java\com\zonesion\wsnpropertydemo\MainActivity.java), 修改智云帐号、密钥和智云服务器地址:

```

MainActivity.java - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

public class MainActivity extends AppCompatActivity {
    private String ID = " "; // 用户账号
    private String KEY = " "; // 用户密钥

    WSNProperty mWSNProperty;

    List<Map<String,Object>> mList = new ArrayList();
    MySimpleAdapter mSa;
    ListView mLvUser;

    JSONArray mJaUser = new JSONArray();

    EditText mETId, mETName;
    RadioGroup mRGSex;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mETId = (EditText)findViewById(R.id.et_id);
        mETName = (EditText)findViewById(R.id.et_name);
        mRGSex = (RadioGroup)findViewById(R.id.rg);
    }
}
Windows (CRLF) 第 1 行, 第 1 列 100%

```

2) 用户数据接口支持 key-value 字段数据的增删查改操作。用 Android Studio 软件打开摄像监控示例程序 WSNPropertyDemo-Droid, 查看用户数据列表。



3) 添加用户数据：点击“添加用户”，输入卡号和姓名并选择性别，然后点击“添加”进行数据的添加。



下图是添加后的列表：



4) 编辑用户数据：下面来对第二条数据进行编辑：点击“编辑”按钮。

用户数据示例

卡号

姓名

☒男 ☐女

添加

123	zonesion	男	编辑	删除
456	小明	男	编辑	删除

我们可以对卡号、姓名以及性别进行编辑(我这里改变了名字):

修改数据

卡号: 456

姓名: 小智

性别: ☒男 ☐女

确定

点击“确定”就完成了编辑操作。

下图是编辑好的列表:

用户数据示例

卡号

姓名

☒男 ☐女

添加

123	zonesion	男	编辑	删除
456	小智	男	编辑	删除

5) 删除用户数据: 下面来对第二条数据进行删除操作: 点击“删除”按钮。

用户数据示例

卡号

姓名

☒男 ☐女

添加

123	zonesion	男	编辑	删除
456	小智	男	编辑	删除

下图是删除后的列表:



五、常见问题

1. 智云帐号、智云密钥、设备地址全部填写正确，应用才能接收到数据。
2. 使用电脑自带摄像头不支持上下左右云台控制。

6 智云 uni-app 应用接口

一、实验目的

1. 了解智云 uni-app 应用编程接口。
2. 掌握实时连接、历史数据、摄像监控、用户数据接口的编程。

二、实验环境

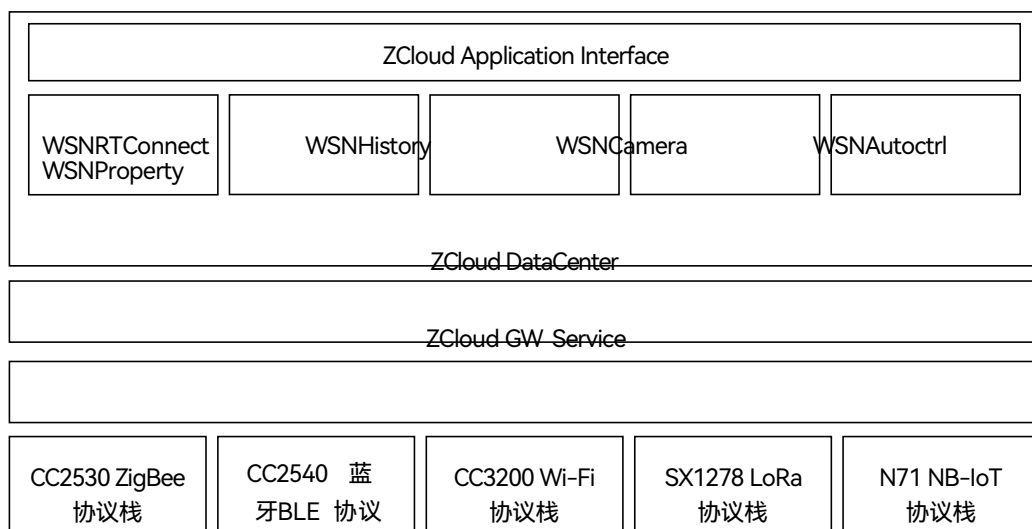
硬件环境:	PC 机 Pentium 处理器双核 2GHz 以上, 内存 4GB 以上
操作系统:	Windows7 64 位及以上操作系统
开发软件:	智云硬件物元仿真平台 HSIMS、ZCloudWebTools、ZXBeeGW4、Chrome 浏览器、HBuilderX 前端开发工具
实验配件:	USB 摄像头

三、实验内容

1. 智云框架

1.1 应用框架

智云平台提供五大应用接口供开发者使用, 包括: 实时连接(WSNRTConnect)、历史数据(WSNHistory)、摄像监控(WSNCamera)、自动控制(WSNAutoctrl)、用户数据(WSNProperty), 详细逻辑图如下:



1.2 功能设计

本实验通过 uni-app 的示例来介绍智云 API 接口的使用:

- 1) 实时连接示例: 实时采集温湿度传感器的温度和湿度数据; 远程控制灯的开关。
- 2) 历史数据示例: 实时采集光照度传感器的光照度数据; 获取光照度传感器的历史数据及曲线。

3) 摄像监控示例：调用智云接口进行远程视频监控及控制。4) 用户数据示例：调用智云接口对数据进行增加、修改和删除操作。相关的设备采用示例节点，使用到的协议如下：

节点名称	TYPE	参数	含义	读写权限	说明
示例节点	500	A0	光强值	R	光强值，整型：0~18000，单位：Lux
		A1	温度值	R	温度值，浮点型：0.1 精度，-40.0~70.0，单位：℃
		A2	湿度值	R	湿度值，浮点型：0.1 精度，0~100.0，单位：%RH
		A3	人体红外	R	监测是否周边有人体活动：1 表示检测到附近有人，0 表示未检测到有人体活动
		D0(OD0/CD0)	主动上报使能	RW	D0 的 bit0~bit7 对应 A0~A7 主动上报使能，0 表示不允许主动上报，1 表示允许主动上报
		D1(OD1/CD1)	开关控制	RW	D1 的 bit 位控制设备： bit0~bit1：LED 照明控制，LED 关(00)、LED 一级光(01)、LED 二级光(10)、LED 三级光(11) bit2：电机开关控制，关闭电机(0)、打开电机(1)
		V0	主动上报时间间隔	RW	A0~A7 主动上报时间间隔，默认 30，单位：S

2. 应用接口

针对 uni-app 应用开发，智云平台提供 JavaScript 接口库，用户直接调用相应的接口即可完成简单 uni-app 应用的开发。以下重点介绍实时连接、历史数据、摄像监控、用户数据接口。

2.1 实时连接

函数	参数说明	功能
new WSNRTConnect(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	建立实时数据实例，并初始化智云 ID 及密钥
connect()	无	建立实时数据服务连接
disconnect()	无	断开实时数据服务连接
onConnect()	无	监听连接智云服务成功
onConnectLost()	无	监听连接智云服务失败
onMessageArrive(mac, dat)	mac:传感器的MAC地址 dat: 发送的消息内容	监听收到的数据
sendMessage(mac, dat)	mac:传感器的MAC地址 dat: 发送的消息内容	发送消息
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.2 历史数据

函数	参数说明	功能
----	------	----

new WSNHistory(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化历史数据对象，并初始化智云ID及密钥
queryLast1H(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1小时的历史数据
queryLast6H(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近6小时的历史数据
queryLast12H(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近12小时的历史数据
queryLast1D(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1天的历史数据
queryLast5D(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近5天的历史数据
queryLast14D(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近14天的历史数据
queryLast1M(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1月(30天)的历史数据
queryLast3M(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近3月(90天)的历史数据
queryLast6M(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近6月(180天)的历史数据
queryLast1Y(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	查询最近1年(365天)的历史数据
query(cal);	cal:回调函数(处理历史数据)	获取所有通道最后一次数据
query(channel, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据)	获取该通道下最后一次数据
query(channel, start, end, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据) start:起始时间 end:结束时间 时间为ISO 8601格式的日期，例如：2010-05-20T11:00:00Z	通过起止时间查询指定时间段的历史数据(<u>根据时间范围默认选择采样间隔</u>)

query(channel, start, end, interval, cal);	channel:传感器数据通道 cal:回调函数(处理历史数据) start:起始时间 end:结束时间 interval:采样点的时间间隔, 详见后续说明 时间为ISO 8601格式的日期, 例如: 2010-05-20T11:00:00Z	通过起止时间查询指定时间段指定时间间隔的历史数据
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.3 摄像监控

函数	参数说明	功能
new WSNCamera(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化摄像头对象, 并初始化智云ID及密钥
initCamera(myCameralP, user, pwd, type);	myCameralP:摄像头外网域名/IP地址 user:摄像头用户名 pwd:摄像头密码 type:摄像头类型(F-Series、F3-Series、H3-Series、ZCamera) # 以上参数从摄像头手册获取	设置摄像头域名、用户名、密码、类型等参数
openVideo();	无	打开摄像头
closeVideo();	无	关闭摄像头
control(cmd);	cmd:云台控制命令, 参数如下: UP: 向上移动一次 DOWN: 向下移动一次 LEFT: 向左移动一次 RIGHT: 向右移动一次 HPATROL: 水平巡航转动 VPATROL: 垂直巡航转动 360PATROL: 360度巡航转动	发指令控制摄像头云台转动
checkOnline(cal);	cal:回调函数(处理检查结果)	检测摄像头是否在线
snapshot();	无	抓拍照片
setDiv(divID);	divID:网页标签	设置展示摄像头视频图像的标签
freeCamera(myCameralP);	myCameralP:摄像头外网域名/IP地址	释放摄像头资源
setServerAddr(sa)	sa:数据中心服务器地址及端口	设置/改变数据中心服务器地址及端口号

2.4 用户数据

函数	参数说明	功能
new WSNProperty(myZCloudID, myZCloudKey);	myZCloudID:智云账号 myZCloudKey:智云密钥	初始化用户数据对象, 并初始化智云 ID 及密钥

put(key, value, cal);	key:名称 value:内容 cal:回调函数	创建用户应用数据
get(cal);	cal:回调函数	获取所有的键值对
get(key, cal);	key:名称 cal:回调函数	获取指定key的value值
setServerAddr(sa)	sa:数据中心服务器地址及 端口	设置/改变数据中心服务器 地址及端口号

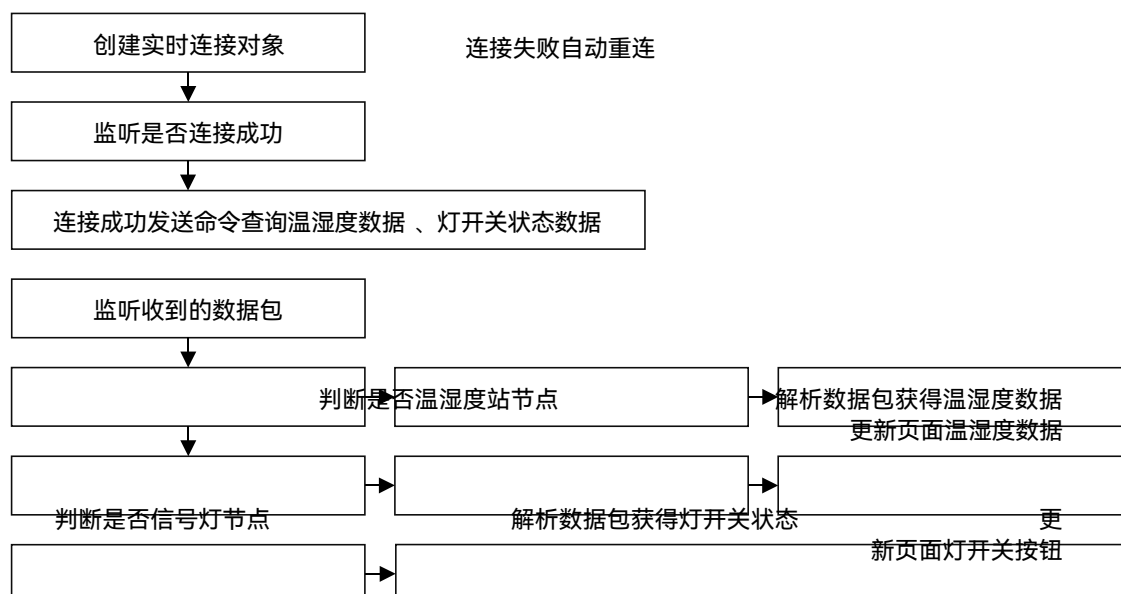
3. 程序分析

3.1 实时连接

RTConnectDemo-uniapp 是一个简单的实时连接应用接口示例，实现了温湿度传感器数据的实时采集、灯光的开关控制等功能。

● 程序逻辑:

RTConnectDemo-uniapp 程序逻辑如下:



● 程序分析:

RTConnectDemo-uniapp 程序内容如下:

1) 页面入口文件(pages\index\index.vue)

```

<template>
  <view class="content">
    <view class="box">
      <view class="item">
        <view class="img_box">
          <image src="@/static/temp.png" mode="widthFix"></image>
        </view>
        <view class="text">温度: {{temp}}°C</view>
      </view>
      <view class="item">
        <view class="img_box">

```

```

        <image src="@/static/humi.png" mode="widthFix"></image>
    </view>
    <view class="text">湿度: {{humi}}%</view>
</view>
</view>
<view class="btn_box">
    <image :src="require('@/static/${ switchVal ? 'on' : 'off'}.png)'"
mode="widthFix" @click="switchLight"></image>
</view>
<view class="state">{{state}}</view>
</view>

</template>

<script>

import { WSNRTConnect } from '@/static/WSN/WSNRTConnect.js'

export default {
  data() {
    return {
      id: "123456789", //智云帐号
      key: "ABCDEFGHJKLMNOPQRSTUVWXYZ", //智云密钥
      addr: "api.zhiyun360.com", //智云服务器地址
      mac: '00:12:4B:00:1F:5F:F2:88', //节点地址
      rtc: null, //创建用户数据服务
    }
  }
}

```

对象

```
state: '数据服务连接失败...', //服务
连接状态
temp: 0, //温度
humi: 0, //湿度
switchVal: false, //开关
    }
},
onLoad() {
    this.getConnect()
    //与智云服务连接, 并监听
和解析实时数据并显示
},
methods: {
    /**
     * 与智云服务连接, 并监听和解析实时数据并显示
     */
    getConnect(){
        this.rtc = new WSNRTConnect(this.id, this.key) //创建数据连接服务对象
        this.rtc.setServerAddr(this.addr) //设置服务器地址
        this.rtc.connect() //数据推送服务连接
        this.rtc.onConnect = () => { //连接成功回调函数
            this.state = '数据服务连接成功...'
            console.log('数据服务连接成功...')
            this.rtc.sendMessage(this.mac, '{A1=?,A2=?,D1=?}') //查询温湿度、灯光
            初始值
        }
        this.rtc.onConnectLost = () => { //数据服务掉线回调函数
            console.log('数据服务连接失败...')
            this.state = '数据服务连接失败...'
        };
        this.rtc.onmessageArrive = (mac, data) => { //消息处理回调函数
            if (data[0] == '{' && data[data.length - 1] == '}') {
                let its = data.slice(1,-1).split(',') //将后台返回的 json 对象截取(去掉
                {}符号) 后以“,”分割为数组
                for (let i = 0; i < its.length; i++) { //循环遍历数组的每一个值
                    let t = its[i].split("=") //将每个值以“=”分割为数组 t[0]
                    为参 t[1]为值
                    if (t.length != 2) continue // 满足条件时结束当前循

```

环

```
//console.log({mac},its);
if (mac == this.mac) { //判断是否是节点的数据
    if (t[0] == 'A1') { //判断参数 A1
        this.temp = t[1]
    }
    if (t[0] == 'A2') { //判断参数 A2
        this.humi = t[1]
    }
    if (t[0] == 'D1') { //判断参数 D1
        if(t[1] & 3) {
            this.switchVal = true
        } else {
            this.switchVal = false
        }
    }
}
}
}
},

/*****
* 灯光开关
*****/

switchLight() {
    if(this.switchVal) {
        this.rtc.sendMessage(this.mac, '{CD1=3,D1=?}') //当前开启状态则发
    } else {
        this.rtc.sendMessage(this.mac, '{OD1=3,D1=?}') //当前关闭状态则发
    }
}

}

}

</script>

<style scoped lang="scss">
.content {
    height: calc(100vh - 44px);
    display: flex;
    flex-direction: column;
    align-items: center;
```

```

justify-content: space-evenly;
.box {
  .item {
    width: 100vw;
    display: flex;
    justify-content: center;
    align-items: center;

    .img_box {
      flex: 3;
      display: flex;
      justify-content: flex-end;
      image {
        width: 100rpx;
      }
    }
    .text {
      flex: 5;
      padding: 30rpx 0rpx 30rpx 50rpx;
      font-size: 60rpx;
      font-weight: bold;
    }
  }
}
.btn_box {
  image {
    width: 300rpx;
  }
}
.state {
  height: 100rpx;
  line-height: 100rpx;
  position: absolute;
  bottom: 0;
  right: 50rpx;
}
}
</style>

```

2) 入口文件 main.js
import App from './App'

```

// #ifndef VUE3
import Vue from 'vue'
Vue.config.productionTip = false
App.mpType = 'app'
const app = new Vue({
  ...App
})
app.$mount()
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue'
export function createApp() {
  const app = createSSRApp(App)
  return {
    app
  }
}
// #endif

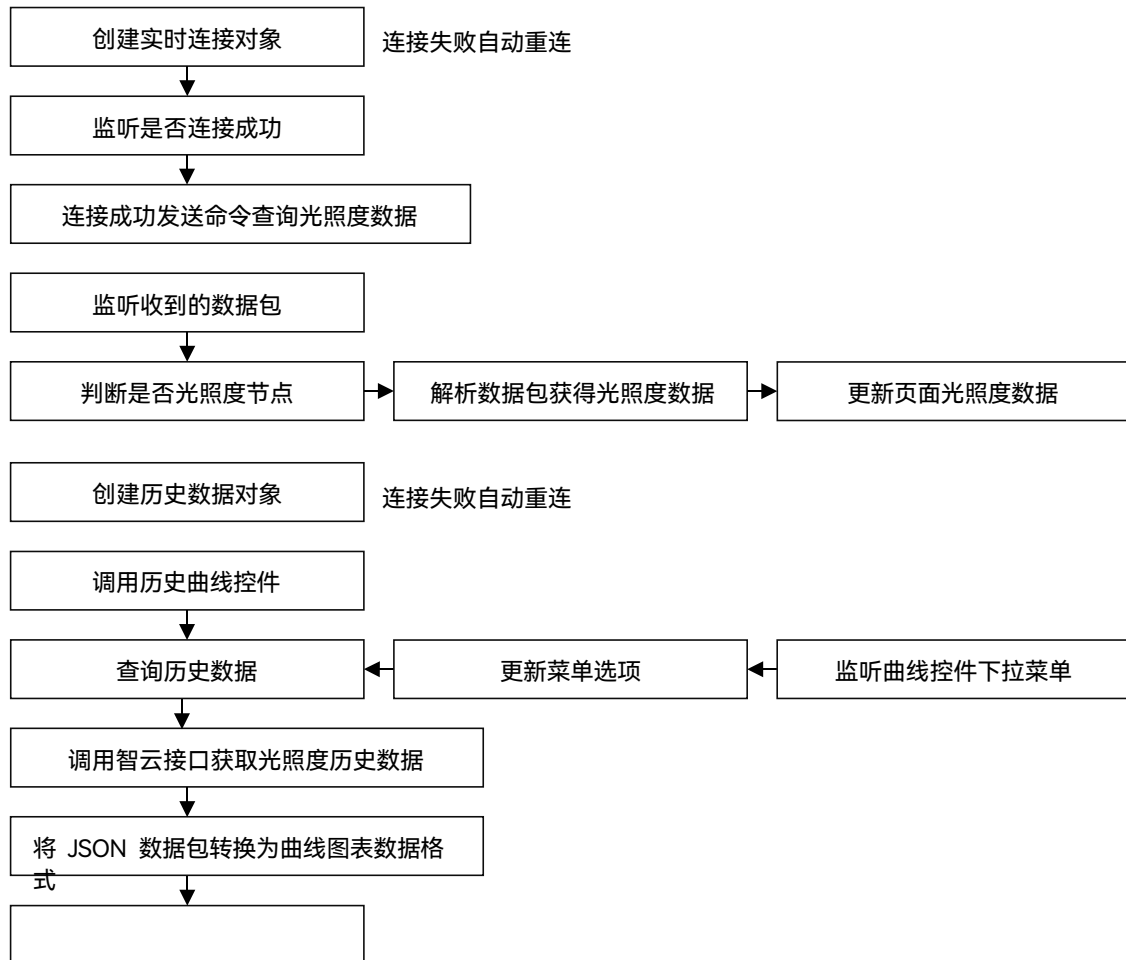
```

3.2 历史数据

HistoryDemo-uniapp 是一个简单的历史数据应用接口示例，实现了光照度传感器数据的实时采集、光照度历史数据曲线展示等功能。

- **程序逻辑:**

HistoryDemo-uniapp 程序逻辑如下:



● 程序分析:

HistoryDemo-uniapp 程序内容如下:

1) 页面入口文件(pages\index\index.vue)

```

<template>

    <view class="content">
        <div id="container"></div>
        <uni-section title="历史数据时间段" type="line">
            <uni-data-select
                v-model="time"
                :localdata="range"
                @change="change"
                :clear="false"
            ></uni-data-select>
        </uni-section>
        <view v-if="isImgShow">
            <image src="../../static/logo.png" style="display: none;"
                @error="Highcharts.onClick" @load="Highcharts.onClick" :data-option="opt" mode=""></image>
        </view>
    </view>

</template>

```

```

<script>
import { WSNHistory } from '@static/WSN/WSNHistory.js' //引入 WSNHistory 历史数

据接口

export default {
  data() {
    return {
      id: "123456789", //智云帐号
      key: "ABCDEFGHJKLMNOPQRSTUVWXYZ", //智云密钥
      addr: "api.zhiyun360.com", //智云服务器地址
      channel: '00:12:4B:00:22:FE:C9:16_A7', //查询历史数据的通道(由
mac+下划线+数据编号组成)
      myHisData: null, //创建用户数据服务对象
      isImgShow: false, //判断图片是否渲染

      opt: [], //历史数据
      time: 'queryLast1H', //历史数据查询的默认时
间段
      range: [ //选择查询历史数据
        { value: 'queryLast1H', text: "最近 1 小时" },
        { value: 'queryLast6H', text: "最近 6 小时" },
        { value: 'queryLast12H', text: "最近 12 小时" },
        { value: 'queryLast1D', text: "最近 1 天" },
        { value: 'queryLast14D', text: "最近 2 周" },
        { value: 'queryLast1M', text: "最近 1 月" },
        { value: 'queryLast3M', text: "最近 3 月" },
        { value: 'queryLast6M', text: "最近半年" },
        { value: 'queryLast1Y', text: "最近 1 年"
      }, ],
    }
  },
  onReady() {
    this.getConnect() //与智云服务连接, 并监听

和解析实时数据并显示

  },
  methods: {
    /*****
    * 根据下拉选项获取对应时间段数据
    *****/
    change(e) {
      setTimeout(() => {
        this.getConnect()
      }, 200);
    },
  },

```

```

/*****
* 与智云服务连接，并监听和解析实时数据并显示
*****/

对象
    getConnect(){
        this.myHisData = new WSNHistory(this.id, this.key) //创建数据连
        接服务

        this.myHisData.setServerAddr(this.addr) //设置服务器地址
        this.myHisData[this.time](this.channel, res => {
            // console.log(res) // 输
            出查询到的历史数据
            if (res.datapoints.length > 0) {
                let data = this.DataAnalysis(res); // JSON 数据转化为图表数
                据
                this.opt = JSON.stringify([...eval(data)])

                //显示图片，然后通过设置 image 属性来传递数据:data-option="opt"
                setTimeout(() => {
                    this.isImgShow = true;
                }, 200)
                setTimeout(() => {
                    this.isImgShow = false;
                }, 500)
            }
        });
    },

/*****
* 将 JSON 格式的数据转换成[x1,y1],[x2,y2],[x3,y3]...格式的数组(历史数据图表相关)
*****/

    DataAnalysis(data) {
        let str = ""
        let value = null
        data.datapoints.forEach((item, index) => {
            value = Date.parse(item.at)
            if (item.value.indexOf("http") != -1) {
                str = str + '[' + value + ',' + item.value + "],";
            } else {
                str = str + '[' + value + ',' + item.value + "],";
            }
        })
        return "[" + str + "]";
    },
}

</script>

```

```

<script module="Highcharts" lang="renderjs">
    let Highchart = {};

    let number = 0;
    export default {
    data() {
        return {
            highOpt: {
                //图标配置项
                chart: {
                    zoomType: 'x'
                },
                title: {
                    text: "
                },
                subtitle: {
                    text: document.ontouchstart === undefined ?
                        ' 鼠标拖动可以进行缩放' : '手势操作进行缩放'
                },
                xAxis: {
                    type: 'datetime',
                    dateTimeLabelFormats: {
                        millisecond: '%H:%M:%S.%L',
                        second: '%H:%M:%S',
                        minute: '%H:%M',
                        hour: '%H:%M',
                        day: '%m-%d',
                        week: '%m-%d',
                        month: '%Y-%m',
                        year: '%Y'
                    }
                },
                tooltip: {
                    dateTimeLabelFormats: {
                        millisecond: '%H:%M:%S.%L',
                        second: '%H:%M:%S',
                        minute: '%H:%M',
                        hour: '%H:%M',
                        day: '%Y-%m-%d',
                        week: '%m-%d',
                        month: '%Y-%m',
                        year: '%Y'
                    },
                },
                yAxis: {
                    title: {
                        text: "
                    }
                }
            }
        }
    }
    }

```

```

    },

    legend: {
        enabled: false
    },

    plotOptions: {
        area: {
            fillColor: {
                linearGradient: {
                    x1: 0,
                    y1: 0,
                    x2: 0,
                    y2: 1
                },
                stops: []
            },
            marker: {
                radius: 2
            },
            lineWidth: 1,
            states: {
                hover: {
                    lineWidth: 1
                }
            },
            threshold: null
        }
    },

    series: [{
        type: 'spline',
        name: 'value',
        data: []
    }]
},
seriesData: [],
//图表数据
}
},
watch: {
    /* 监听到图表数据改变，重新渲染图表 */
    seriesData(n) {
        if (n.length) {
            this.highOpt.series[0].data = n;
            this.initEcharts()
        }
    }
},
mounted() {

```

```

    /* 模板加载完后，引入 highcharts.js */
    this.$nextTick(() => {
        this.isLoadHighCharts()
    })
},

methods: {
    /* 引入 highcharts.js */
    isLoadHighCharts(){
        const script = document.createElement('script')
        //view 层的页面运行在 www 根目录，其相对路径相对于 www 计算
        script.src = 'static/highcharts.js'
        document.head.appendChild(script)
    },
    /* 渲染图表 */
    initEcharts() {
        let interval = setInterval(() => {
            if (Highcharts || number > 20) {
                clearInterval(interval);
                if (!Highcharts) return
                if (!this.highOpt.plotOptions.area.fillColor.stops.length) {
                    let stops = [
                        [0, new Highcharts.getOptions().colors[0]],
                        [1,
                            new
Highcharts.Color(Highcharts.getOptions().colors[0]).setOpacity(0).get('rgba')]
                    ]
                    this.highOpt.plotOptions.area.fillColor.stops = stops;
                }
                Highchart.container = Highcharts.chart('container', this.highOpt);
            } else {
                number++
            }
        }, 500)
    },

    /* 监听通过 image 属性传递的数据赋值给图表数据 */
    onClick(e) {
        let optionJson = e.currentTarget.dataset.option;
        this.seriesData = JSON.parse(optionJson)
    },
}
}

</script>

<style scoped lang="scss">

```

```

.content {
  height: calc(100vh - 44px);
  display: flex;
  flex-direction: column;
  align-items: center;
  background-color: #fff;

  .highcharts-container {
    width: 750rpx;
    height: 600rpx;
  }

  #container {
    width: 100%;
    height: 600rpx;
    border-bottom: 1rpx solid gainsboro;
  }

  /deep/.uni-section {
    width: 750rpx;
    margin-top: 50%;
    display: flex;
    justify-content: space-evenly;
    align-items: center;

    .uni-section-content {
      width: 300rpx;
    }
  }
}

```

</style>

2) 入口文件 main.js

```

import App from './App'

// #ifndef VUE3
import Vue from 'vue'
Vue.config.productionTip = false
App.mpType = 'app'
const app = new Vue({
  ...App
})
app.$mount()
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue'
export function createApp() {
  const app = createSSRApp(App)

```

```

    return {
      app
    }
  }
}
// #endif

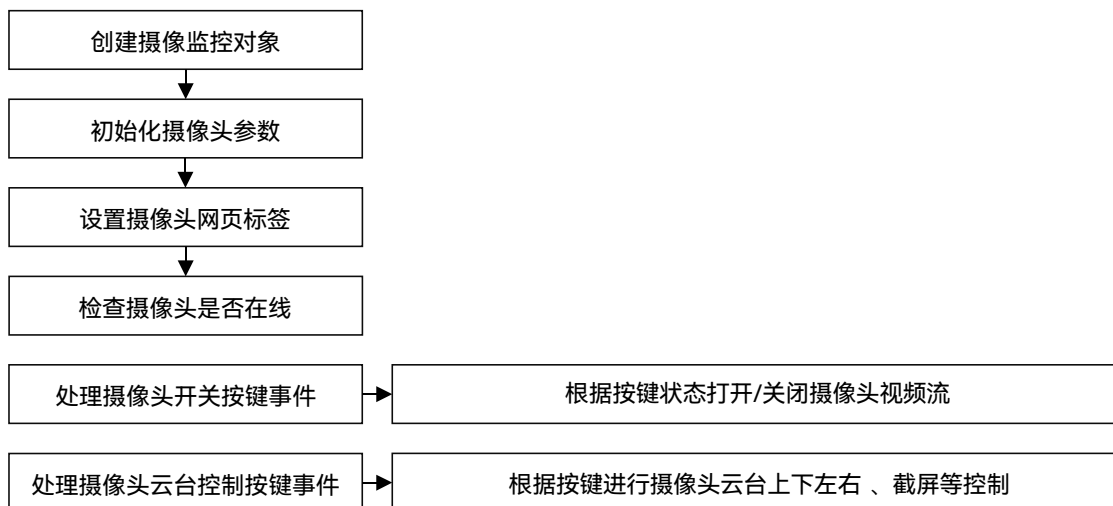
```

3.3 摄像监控

CameraDemo-uniapp 是一个简单的摄像监控应用接口示例，实现了网络实时摄像头视频流监控和摄像头云台转动控制等功能。

● 程序逻辑:

CameraDemo-uniapp 程序逻辑如下:



● 程序分析:

CameraDemo-uniapp 程序内容如下:

1) 页面入口文件(pages\index\index.vue)

```

<template>

  <view class="content">
    <image :src="imgSrc" mode="widthFix"></image>
    <wf-yaokong @btn_click="btn_click"></wf-yaokong>
  </view>

</template>

<script>

  import { WSNCamera } from '@static/WSN/WSNCamera.js' //引入 SDK 文件
  import yaokong from "@components/wf-yaokong/wf-yaokong.vue"//引入遥控器组件

  export default {
    data() {
      return {
        id: "123456789",
        //智云帐号
        key: "ABCDEFGHJKLMNOPQRSTUVWXYZ", //智云密钥
        IP: "127.0.0.1:28083",
        //网关地址(默认使用内网地址，为空时使用外网地址)
      }
    }
  }

```

地址，为空时使用外网地址)

```

        myCamera: null,
        //创建用户数据服务对象
        cameraID: "Camera:1",
        //定义用户数据列表
        imgSrc: '/static/bg.jpg',
        //视频默认图片
        videoSwitch: false,
        //视频开关状态(默认为关闭)
    },
    comments: {
        yaokong
        // 注册遥控器
    },
    onLoad() {
        this.myCamera = new WSNCamera(this.id, this.key) //创建摄像监控服务对象
        this.myCamera.initCamera(this.IP, this.cameraID) //摄像头初始化
    },
    methods: {
        /**
         * 遥控器点击事件(根据点击不同按钮接收到的值做相应操作)
         */
        btn_click(type) {
            if (type === 'center') {
                if(!this.videoSwitch) //判断当前视频为关闭状态
                {
                    this.imgSrc = this.myCamera.openVideo() //打开摄像头, 接收返回的链接
                    this.videoSwitch = true //更新视频开关状态
                } else {
                    //判断当前视频为开启状态
                    this.imgSrc = this.myCamera.closeVideo()//关闭摄像头, 接收返回的链接
                    this.videoSwitch = false //更新视频开关状态
                }
            }
            //控制摄像头上下左右移动
            if(type === 'top') this.myCamera.control("UP")
            if(type === 'bottom') this.myCamera.control("DOWN")
            if(type === 'left') this.myCamera.control("LEFT")
            if(type === 'right') this.myCamera.control("RIGHT")
        },
    }
}

```

```
</script>
```

```
<style lang="scss">
```

```
.content {  
  
  height: calc(100vh - 44px);  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  justify-content: space-between;  
  padding-bottom: 300rpx;  
  image {  
    width: 750rpx;  
    height: auto;  
  }  
}
```

```
</style>
```

2) 入口文件 main.js

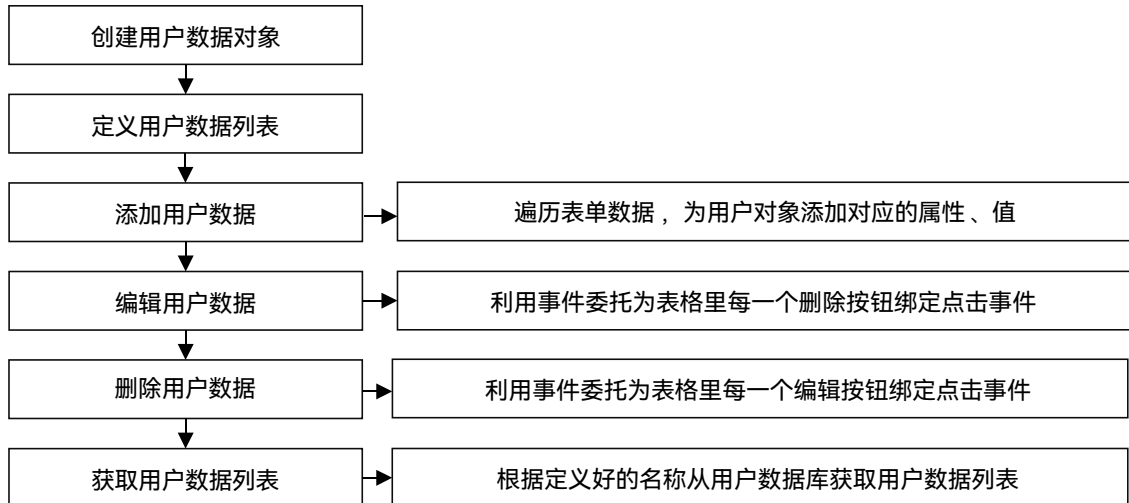
```
import App from './App'  
  
// #ifndef VUE3  
import Vue from 'vue'  
Vue.config.productionTip = false  
App.mpType = 'app'  
const app = new Vue({  
  ...App  
})  
app.$mount()  
// #endif  
  
// #ifdef VUE3  
import { createSSRApp } from 'vue'  
export function createApp() {  
  const app = createSSRApp(App)  
  return {  
    app  
  }  
}  
// #endif
```

3.4 用户数据

PropertyDemo-uniapp 是一个简单的用户数据应用接口示例，实现了用户数据的“增删改”的功能。

● 程序逻辑:

PropertyDemo-uniapp 程序逻辑如下:



● 程序分析:

PropertyDemo-uniapp 程序内容如下:

1) 页面入口文件(pages\index\index.vue)

```

<template>

  <view class="content">
    <uni-table ref="table" border stripe emptyText="暂无更多数据">
      <uni-tr>
        <uni-th width="80" align="center">ID</uni-th>
        <uni-th width="80" align="center">姓名</uni-th>
        <uni-th width="80" align="center">性别</uni-th>
        <uni-th width="150" align="center">设置</uni-th>
      </uni-tr>
      <uni-tr v-for="(item, index) in userList" :key="index">
        <uni-td align="center">{{ item.id }}</uni-td>
        <uni-td align="center">{{ item.name }}</uni-td>
        <uni-td align="center">{{ item.sex == 0 ? '男' : '女' }}</uni-td>
        <uni-td align="center">
          <view class="uni-group">
            <button class="uni-button" size="mini" type="primary"
              @tap="editModal(index, 'modal2')>修改</button>
            <button class="uni-button" size="mini" type="warn"
              @click="dltUser(index)>删除</button>
          </view>
        </uni-td>
      </uni-tr>
    </uni-table>

    <button class="cu-btn bg-green shadow" @tap="modalName = 'modal1'">添加用户</button>

    <view class="cu-modal" :class="modalName && 'show'">
      <view class="cu-dialog">
        <view class="cu-bar justify-end">

```

```

<view class="content">{{ modalName == 'modal1' ? '添加用户' : '编辑用户'
}}</view>

<view class="action" @tap="hideModal">
  <text class="culcon-close text-red"></text>
</view>
</view>
<view class="padding-xl">
  <!-- 基础用法，不包含校验规则 -->
  <uni-forms ref="baseForm" :modelValue="user">
    <uni-forms-item label="ID" required>
      <uni-easyinput v-model="user.id" placeholder="请输入 ID" />
    </uni-forms-item>
    <uni-forms-item label="姓名" required>
      <uni-easyinput v-model="user.name" placeholder="请输入姓名" />
    </uni-forms-item>
    <uni-forms-item label="性别" required>
      <uni-data-checkbox v-model="user.sex" :localdata="sexs" />
    </uni-forms-item>
  </uni-forms>
</view>
<view class="cu-bar justify-end">
  <view class="action">
    <button class="cu-btn line-green text-green" @tap="hideModal"> 取消
  </button>

    <button class="cu-btn bg-green margin-left"
@tap="addUser" v- if="modalName=='modal1'">确定</button>
    <button class="cu-btn bg-green margin-left" @tap="editUser" v-else>确定</button>

  </view>
</view>
</view>
</view>
</view>
</view>

</template>

<script>
import { WSNProperty } from '@static/WSN/WSNProperty.js'

export default {
  data() {
    return {
      id: "123456789",
      //智云帐号
      key: "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
      //智

```

对象

索引

框

据

```
addr: "api.zhiyun360.com", //智
云服务器地址

myProperty: null,
//创建用户数据服务

userList: [{
//定义用户数据列表
    id: 123,
    name: 'zonesion',
    sex: 0
}],
editIndex: 0,
//定义当前编辑用户

modalName: null,
//定义当前显示模块

user: {
// 默认表单数

    id: "",
    name: "",
    sex: 0,
},
sexs: [{
//单选数据源

    text: '男',
    value: 0
}, {
    text: '女',
    value: 1
}],
}
},
onLoad() {
    this.myProperty = new WSNProperty(this.id, this.key)//创建用户数据服务对象
    this.myProperty.setServerAddr(this.addr) //设置服务
    器地址
    this.getUserList()
    //获取用户数据
},
watch: {
    //监听当前显示的模块框，如果值为 null(关闭)，则重置表单类容
    modalName(newVal, oldVal) {
        if(newVal === null) {
            this.user = {
                id: "",
                name: "",
            }
        }
    }
}
```

```

        sex: 0
      }
    }
  },

  methods: {
    /**
     * 添加用户数据
     */
    addUser() {
      let { id, name, sex } = this.user
      if (id && name) {
        //判断用户对象的属性是否存在
        this.userList.push(this.user) //将用户数据添加至列表
        this.myProperty.put('userList', JSON.stringify(this.userList), res => { //更新用户数据
          this.getUserList() //重新获取用户数据列表渲染页面
          uni.showToast({
            title: '添加用户成功',
            duration: 2000
          });
        } else {
          uni.showToast({
            title: '信息不能为空',
            duration: 2000
          });
        }
        this.modalName = null
        //关闭模态框
      },

    /**
     * 删除用户数据
     */
    dltUser(index) {
      uni.showModal({
        title: '用户管理',
        content: '是否确定删除该用户? ',
        success: res => {
          if (res.confirm) {
            this.userList.splice(index, 1) //从列表删除选中的数据
            this.myProperty.put('userList', JSON.stringify(this.userList), res => {
              //更新用户数据
            }
          }
        }
      })
    }
  }
}

```

渲染页面

```
this.getUserList() //重新获取用  
户数据列表
```

```
uni.showToast({  
  title: '删除用户成功',  
  duration: 2000  
});
```

```
});  
} else if (res.cancel) {  
  console.log('用户点击取消');  
}
```

```
}  
});  
},
```

```
/******
```

```
* 修改用户数据
```

```
******/
```

```
editModal(index, modalName) {  
  this.user = this.userList[index]  
  this.editIndex = index  
  this.modalName = modalName  
},
```

```
editUser() {  
  let { id, name, sex } = this.user  
  if (id && name) {  
    //判断用户对象的属性是否
```

存在

```
this.userList[this.editIndex] = this.user //将用户数据添加至列表  
this.myProperty.put('userList', JSON.stringify(this.userList), res => { //更
```

新用户数据

```
this.getUserList() //重新  
获取用户数据列表渲染
```

渲染页面

```
uni.showToast({  
  title: '编辑用户成功',  
  duration: 2000  
});  
});  
} else {  
  uni.showToast({  
    title: '信息不能为空',  
    duration: 2000  
  });  
}  
this.modalName = null  
//关闭模态框
```

```

    },

    /**
     * 获取(查询) 用户数据
     */
    getUserList() {

        this.myProperty.get('userList', res => { //根据定义好的名称从用户
            数据库

            获取用户数据列表

            if (res && res.length > 0) { //判断数据是否存
                在

                this.userList = res //有则赋
                值给用户列表对页
                面进行重新渲染

            }
        });
    },

    /**
     * 关闭模态框
     */
    hideModal() {
        this.modalName = null
        //关闭模态框
    },
}

</script>

<style>

.content {
    height: calc(100vh - 44px);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-between;
    padding-bottom: 100px;
}

.uni-group {
    display: flex;
    align-items: center;
}

/deep/.uni-forms-item {
    display: flex;

```

2) 入口文件 main.js

```
import App from './App'

// #ifndef VUE3
import Vue from 'vue'
Vue.config.productionTip = false
App.mpType = 'app'

const app = new Vue({
  ...App
})
app.$mount()
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue'
export function createApp() {
  const app = createSSRApp(App)
  return {
    app
  }
}

// #endif
```

四、实验步骤

1. 实验部署

本实验基于 xLab 实验平台来构建一个智云接口项目，可通过硬件物元仿真平台软件创建虚拟的硬件，也可以使用 xLab 实验平台真实的硬件(本实验默认通过硬件物元仿真平台软件创建)。

uni-app 采用 HBuilderX 工具开发(DISK-Packages\10-Web 应用技术\HBuilderX.zip), 工具的安装与使用可参考《Web 应用技术》课程。

2. 原型搭建

2.1 创建传感器

可参考实验 4 的原型仿真步骤来创建传感器。

2.2 创建项目

可参考实验 4 的原型仿真步骤来创建项目。

2.3 运行项目

可参考实验 4 的原型仿真步骤来运行项目。

3. 应用体验

3.1 实时连接

- 1) 打开本实验目录下的工程“RTConnectDemo-uniapp”。
- 2) 用记事本打开 index.vue 文件(pages\index\index.vue), 填写授权的智云帐号、密钥和节点的 MAC地址



```
<script>
import { WSNRTConnect } from '@static/WSN/WSNRTConnect.js'

export default {
  data() {
    return {
      id: [REDACTED], //智云帐号
      key: [REDACTED], //智云密钥
      addr: "api.zhiyun360.com", //智云服务器地址
      mac: "01:12:4B:00:DD:87:6B:42", //节点地址
      rtc: null,

      state: "数据服务连接失败...", //服务连接状态
      temp: 0,

      humi: 0,

      switchVal: false,
    }
  },
  onLoad() {
    this.getConnect()
  },
}
//与智云服务连接, 并监听和解析实时数据并显示
```

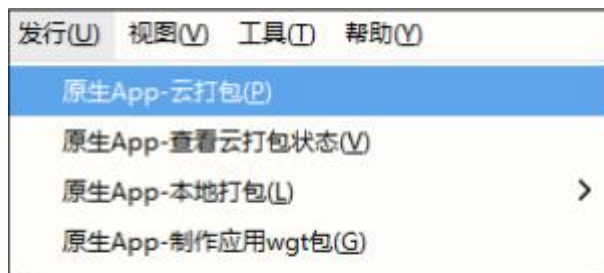
- 3) 在 HBuilderX 前端开发工具上打开 RTConnectDemo-uniapp 项目, 选择运行到 Chrome 浏览器上, 进入浏览器界面之后按下“F12”, 选择“”手机视图。查看实时数据信息及 RGB 灯的开关操作。



4) 调试完成后可打包成android app安装到手机使用:

打包流程:

- a. 配置打包信息。点击发行-->原生 App 云打包。



- b. 使用 DCloud 公用证书, Android 手机勾选 Android 选项。



- c. 如果“制作状态”栏显示“打包成功, 下载完成”则表示云端打包完成, 可点击“打开下载目录”查看下载的安装包。

- d. 生成的 APK 文件保存在本地。

- e. Android 设备通过数据线连接到电脑, Android 设备上确认文件管理功能, 电脑可以访问Android设备存储器。从电脑复制 apk 文件到 Android 设备存储器中, 在 Android 设备上找到apk 文件并安装。

3.2 历史数据

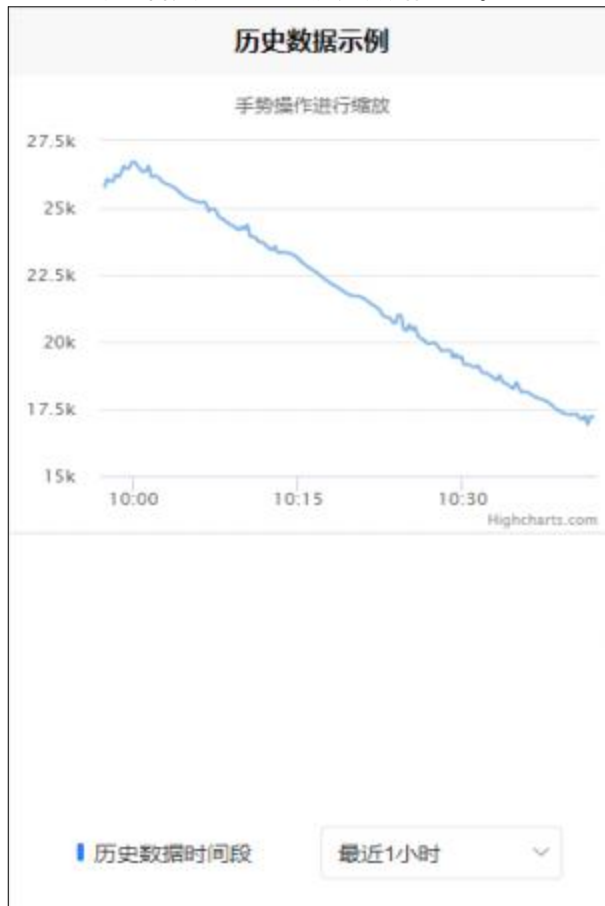
- 1) 打开本实验目录下的工程“HistoryDemo-uniapp”文件夹。

- 2) 用记事本打开 index.vue 文件(pages\index\index.vue), 修改智云帐号、密钥和 ZigBee 无线节点的 MAC 地址:



3) 在 HBuilderX 前端开发工具上打开 HistoryDemo-uniapp 项目，选择运行到 Chrome 浏览器上，进入浏览器界面之后按下“F12”，选择“”手机视图。查看光照强度实时数据信息及历史曲线信息，可

4) 可以通过曲线的下拉选项选择查看其它时间的历史数据曲线。

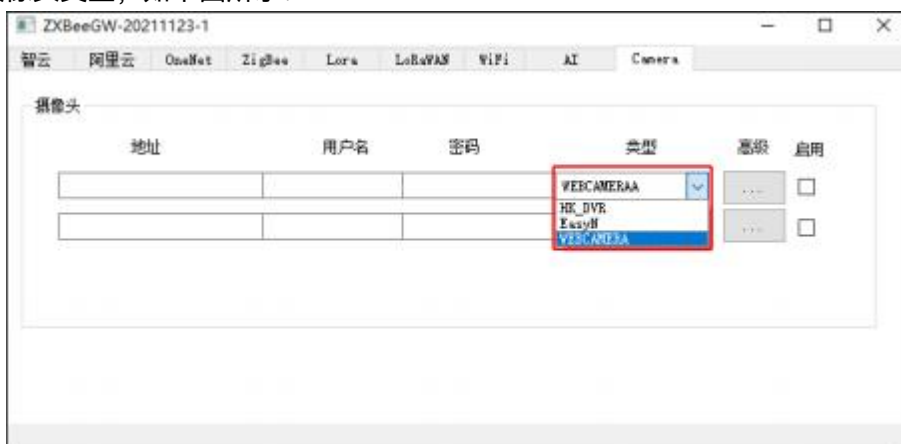


5) 调试完成后可打包成android app安装到手机使用(打包流程可参照2.1 实时连接步骤)。

3.3 摄像监控

1) 将 USB 摄像头插入电脑 USB 接口。

2) 打开软件 ZXBeegW 软件，选择 Camera 选项卡，在类型下拉菜单下，可以查看当前网关服务软件所支持的摄像头类型，如下图所示：



上图中，表明该软件支持 WebCAMERA(笔记本摄像头和 USB 摄像头)、HK-DVR(海康数字硬盘录像机)、EasyN(易视 IP 摄像头)。

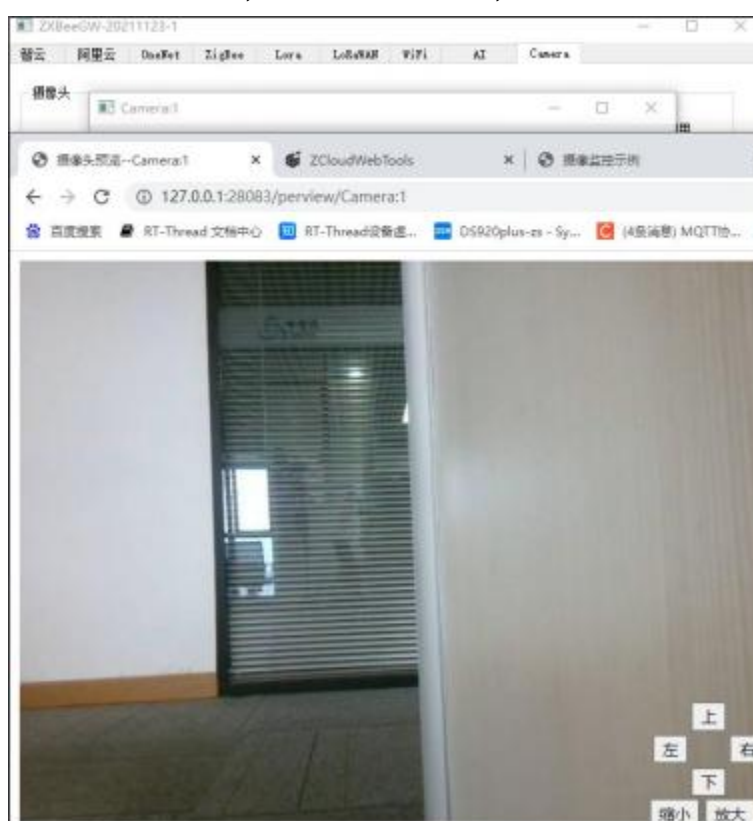
3) 类型选择: 选择 WEBCAMERA(笔记本或 USB 摄像头), 不需填写地址、用户名和密码, 勾选“启用”选项, 如下图所示:



4) 点击 高级 选项, 弹出对话框, 如下图所示:



点击对话框上的 摄像头测试 按钮, 软件会打开浏览器, 并显示当前摄像头的画面, 如下图所示:



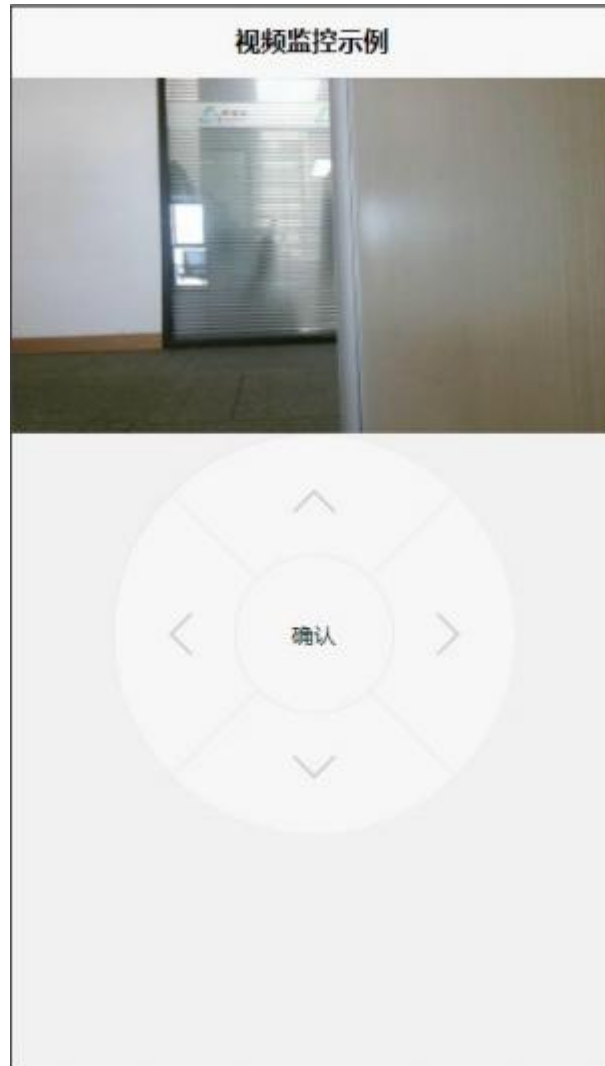
5) 上述操作只是测试了摄像头和 PC 网关服务软件的连通性, 还需要开启智云服务。选择 智云 选项卡, 填入我司授权的用户账号和用户密钥(ID/KEY), 服务器地址填写 api.zhiyun360.com , 并启动远程服务、本地服务, 即可将视频数据转发到云端数据中心和本地局域网内。



6) 用记事本打开 index.vue 文件(pages\index\index.vue), 修改智云帐号、密钥和摄像头的地址(这里默认使用内网地址):



7) 在 HBuilderX 前端开发工具上打开 CameraDemo-uniapp 项目, 选择运行到 Chrome 浏览器上, 进入浏览器界面之后按下“F12”, 选择“”手机视图。查看摄像头视频流数据:



8) 调试完成后可打包成android app安装到手机使用(打包流程可参照2.1 实时连接步骤)。

3.4 用户数据

1) 用记事本打开 index.vue 文件(pages\index\index.vue), 修改智云帐号和密钥:

```

index.vue - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

export default {
  data() {
    return {
      id: " ", //智云帐号
      key: " ", //智云密钥
      addr: "api.zhiyun360.com", //智云服务器地址
      myProperty: null,

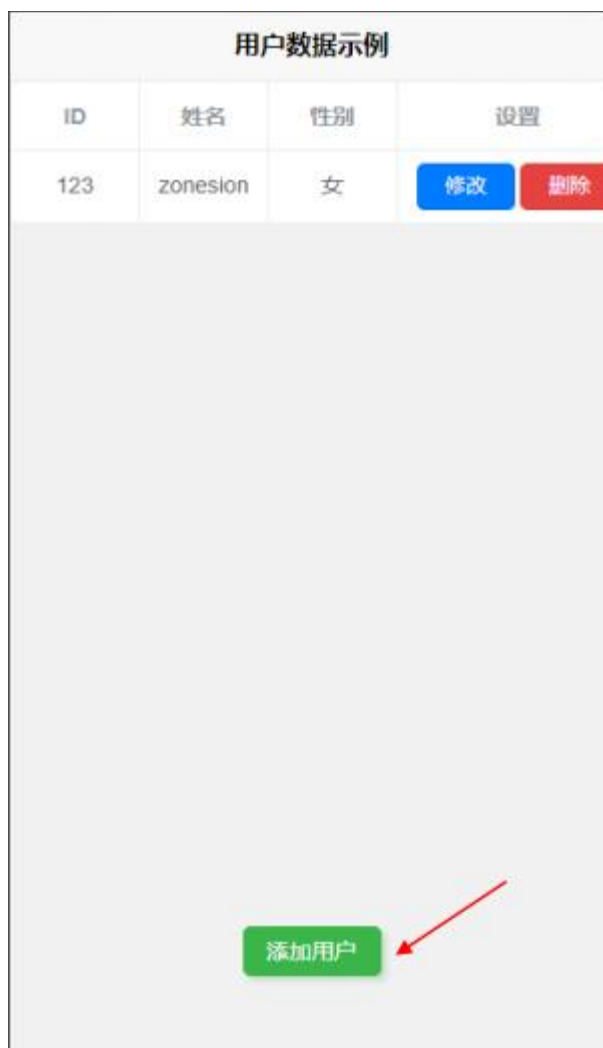
      //创建用户数据服务对象
      //定义用户数据列表
      userList: [{
        id: 123,
        name: 'zonesion',
        sex: 0
      }],
      editIndex: 0,
      modalName: null,
      //定义当前编辑用户索引
      //定义当前显示模块框
      //默认表单数据
      user: {
        id: "",
        name: "",
        sex: 0,
      }
    }
  }
}

```

第 70 行, 第 25 列 100% Windows (CRLF) UTF-8

2) 用户数据接口支持 key-value 字段数据的增删查改操作。在 HBuilderX 前端开发工具上打开

PropertyDemo-uniapp 项目，选择运行到 Chrome 浏览器上，进入浏览器界面之后按下“F12”，选择“”手机视图。查看用户数据列表。



3) 添加用户数据：点击“添加用户”，输入卡号和姓名并选择性别，然后点击“确认”进行数据的添加。



添加成功会有弹窗提示“添加用户成功”，下图是添加后的列表：

用户数据示例

ID	姓名	性别	设置
123	zonesion	女	修改 删除
456	小明	男	修改 删除

✓

添加用户成功

添加用户

4) 编辑用户数据：下面来对第二条数据进行编辑：点击“修改”按钮。

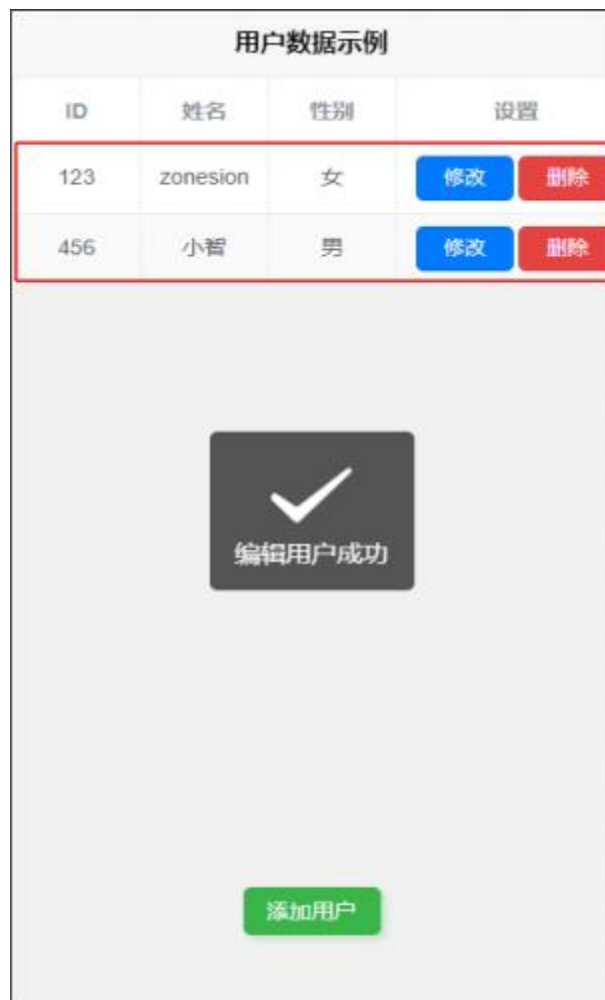
ID	姓名	性别	设置
123	zonesion	女	修改 删除
456	小明	男	修改 删除

我们可以对卡号、姓名以及性别进行编辑(我这里改变了名字)：



A dialog box for editing a user. It contains three fields: *ID with value 456, *姓名 (Name) with value 小智, and *性别 (Gender) with radio buttons for 男 (Male) and 女 (Female). The 男 option is selected. At the bottom right, there are two buttons: 取消 (Cancel) and 确定 (Confirm). The 确定 button is highlighted with a red box.

点击“确定”就完成了编辑操作。同时会有弹窗提示：“编辑用户成功”。下图是添加后的列表：



The user data table is shown with two rows. The first row has ID 123, name zonesion, and gender 女. The second row has ID 456, name 小智, and gender 男. Both rows have 修改 (Edit) and 删除 (Delete) buttons. The 删除 button for the second row is highlighted with a red box. Below the table is a large gray area with a dark gray box containing a white checkmark and the text 编辑用户成功 (Edit user successful). At the bottom is a green button labeled 添加用户 (Add user).

ID	姓名	性别	设置
123	zonesion	女	修改 删除
456	小智	男	修改 删除

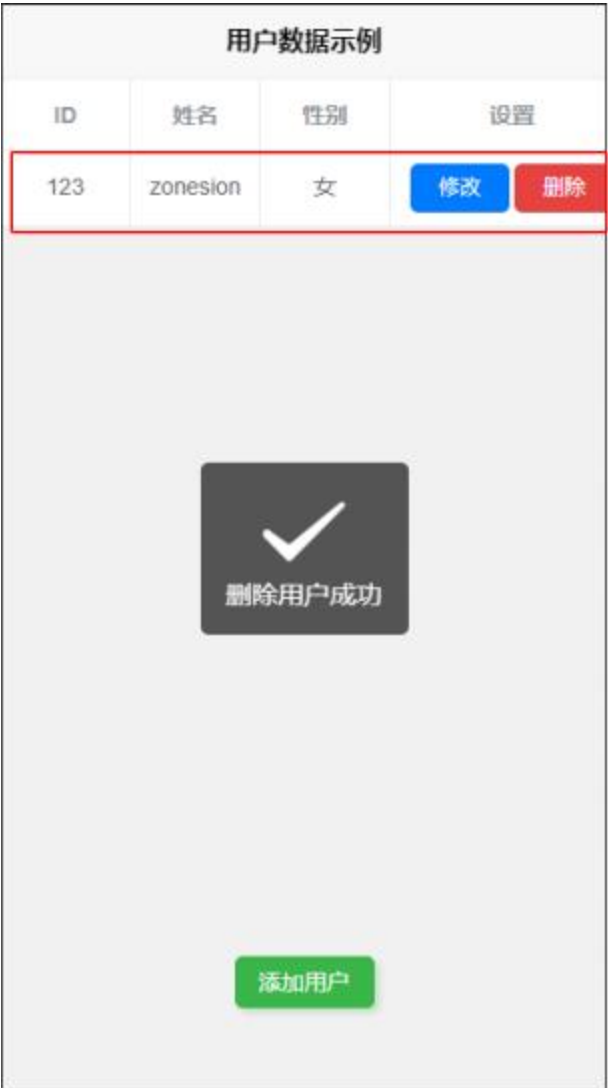
5) 删除用户数据：下面来对第二条数据进行删除操作：点击“删除”按钮。

ID	姓名	性别	设置
123	zonesion	女	修改 删除
456	小智	男	修改 删除

随后会有弹框提示，点击“确定”按钮：



确定之后会有弹窗提示：“编辑用户成功”，下图是删除后的列表：



6) 调试完成后可打包成android app安装到手机使用(打包流程可参照2.1 实时连接步骤)。

五、常见问题

1. 智云帐号、智云密钥、设备地址全部填写正确，应用才能接收到数据。

2. 使用电脑自带摄像头不支持上下左右云台控制。