树莓派连接阿里云物联网平台记录

原创

Raindy_nightmare 于 2018-09-20 17:28:28 发布 11963 收藏 56

分类专栏： 就先这么放着吧 文章标签： 物联网 阿里云物联网云平台 树莓派 坑爹阿里云 新手博主

本文链接：https://blog.csdn.net/qq_19961917/article/details/82771629

就先这么放着吧 专栏收录该内容

3 篇文章 0 订阅

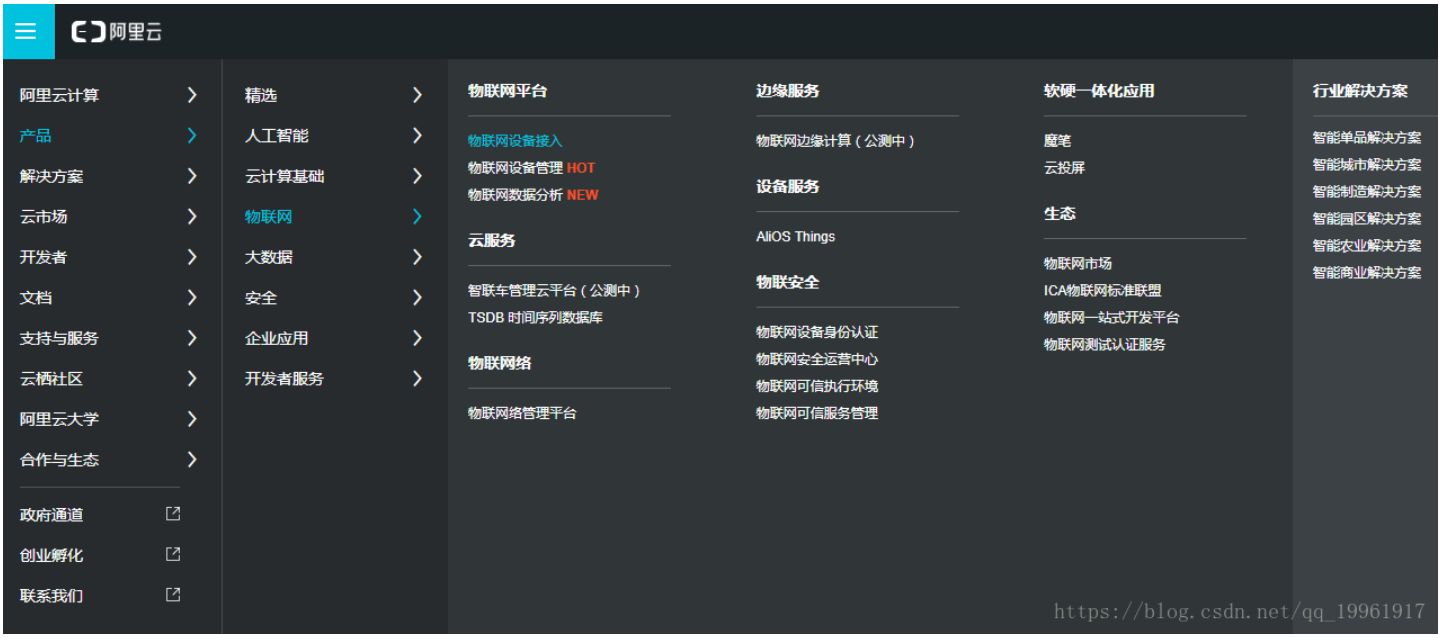
订阅专栏

接触阿里云物联网平台

手中有一台树莓派3B，想接入物联网云平台试试，最终决定先从阿里云下手。本文写作时间为2018年9月19日。

寻找阿里云物联网平台入口

see it?



进入管理控制台

登录账号，进入物联网设备接入栏目，开通物联网平台服务。准备使用高级版产品。对于使用基础版还是高级版，没有仔细研究，鉴于高级版也有每日免费使用的额度，为什么不薅用呢？计费请参照下文。

关于计费

目前分为两种版本：基础版和高级版，收费的标准分为设备管理费用和传输消息费用，官方文档中的介绍是这样的：

> 物联网平台分为基础版与高级版。基础版仅传输消息需要付费，高级版传输消息与设备管理需要收费。两者均为按量付费，后付费，不设最低费用。

对于将要使用的高级版，设备管理和传输消息都有免费的额度以供测试使用。

消息收费

> 每月前100万条消息数免费赠送，当月1号凌晨生效，不累计到下月。当月累计超出100万条的消息数部分开始计费。

设备收费

> 每天每个账号拥有10个免费日活设备赠额。当天累计超出10个设备的部分开始计费。

最新计费标准请参照阿里云官方文档-产品计费

创建产品和设备

根据文档balabalabala介绍，需要先创建一个**产品**（产品上限数量是1000个），然后再创建一个**设备**添加到产品上（单个产品最多可以添加50万台设备），然后在管理控制台的**设备管理-设备**菜单中，点击创建好的设备进入详情界面，这一堆参数里一会要用到的有**ProductKey**、**DeviceName**和**DeviceSecret**，这三个参数名为“三元组认证参数”。创建产品和设备的具体信息参照官方文档创建产品（高级版）和创建设备。

备注：本文创建产品为高级版，节点类型为网关，设备类型为无，数据格式为Alink JSON，是否接入网关为否。新建的设备创建到该产品上。

新增一个属性功能

在产品的管理界面中可以找到功能定义选项卡，创建产品的时候设备类型选择的是无，所以这里不会自动创建任何功能。现在点击新增，手动创建一个功能。

功能分为属性、服务和事件三种类型，这三种功能的含义可在物模型文档中查看。

比如想让设备上传温度，就新增一个属性，名称为温度，标识符为temp，数据类型为float，取值范围为-20~50，分辨率为1，单位为摄氏度℃，读写类型为读写。

物模型与Alink上传格式

物模型是高级版中描述某物理实体的数字模型，简称为TSL。**物模型和Alink不是一回事。**

阿里物联网平台定义了一种基于Json的数据请求格式，叫Alink，在设备上拼凑成这种格式的数据进行上报，云平台才能知道你上传的数据是什么意思。

基于Alink编写刚刚创建的温度属性的数据格式准备用于设备端使用SDK上传数据，该格式可参考Alink格式请求数据格式

下面的数据中注意temp是上一节定义的温度属性的标识符，value的值是float型，所以不加双引号，time是**精确到毫秒**时间戳，如果**位数少了是不识别的**，时间戳类型为整形，同样不加双引号。其他字段按照样例来是没有问题的。

```
{
  "id": "123",
  "version": "1.0",
  "params": {
    "temp": {
      "value": 20,
      "time": 1524448722000
    },
  "method": "thing.event.property.post"
}
```

关于Topic

正如字面意思，Topic就是主题，可以理解为消息的分类，高级版中系统自动建好了几个topic，包括一些系统定义的topic，我们也可以自己定义topic。这里是关于topic的文档介绍

至此云平台方面的准备工作就做好了。

设备端（树莓派）操作

树莓派系统

Raspbian 9

下载C-SDK

阿里云开发了一套SDK用于设备端向云平台发送数据。现在支持C、Java、Android、IOS、HTTP/2。对于树莓派系统则下载C版本的SDK，可参考高级版快速开始-开发设备。

登录系统，切换到合适的目录，执行如下命令，将设备端SDK代码从Github克隆到本地。

```
git clone https://github.com/aliyun/iotkit-embedded
```

查看所有SDK版本

```
cd iotkit-embedded
```

```
git branch -r
```

切换到最新SDK版本号

```
git checkout SDK版本号
```

修改C-SDK

编辑make.setting文件，设置功能开关，如下

```
FEATURE_MQTT_COMM_ENABLED = y
FEATURE_MQTT_DIRECT = y
FEATURE_MQTT_DIRECT_NOTLS = n
FEATURE_MQTT_DIRECT_NOITLS = y
FEATURE_COAP_COMM_ENABLED = n
FEATURE_HTTP_COMM_ENABLED = y
FEATURE_SUBDEVICE_ENABLED = n
FEATURE_CMP_ENABLED = y
FEATURE_DM_ENABLED = y
FEATURE_SERVICE_OTA_ENABLED = y
```

我们使用MQTT上传数据，编辑sample/mqtt/目录下的mqtt-example.c，把include下面那几组PRODUCT_KEY、DEVICE_NAME和DEVICE_SECRET改成云平台设备控制面板中的相应参数。

新定义一个宏，一会往这个topic上发消息

```
#define TOPIC_POST "/sys/"PRODUCT_KEY"/"DEVICE_NAME"/thing/event/property/post
#define TOPIC_POST_FMT "/sys/%s/%s/thing/event/property/post"
```

在mqtt_client_secure函数中，添加变量

```
char topic_post[IOTX_URI_MAX_LEN] = {0};
```

添加语句

```
HAL_Snprintf(topic_post,IOTX_MAX_LEN,TOPIC_POST_FMT,__product_key,__device_id2);
```

在mqtt_client这个函数中，可以找到发布消息的模板，注释/*Initialize topic information*/下面就是，搭载消息的变量是msg_pub，只需要更改它的内容即可。注释掉strcpy函数，改为如下(argv参数为运行程序时传入的参数，由于获取传感器数据的程序python编写的，所以可以在调用C程序的时候传入参数)：

```
sprintf(msg_pub,"{\"id\":\"123\",\"version\":\"1.0\",\"params\":{\"temp\":{\"value\":%s,\"time\":%s}},\"method\"
:\"thing.event.property.post\"}",user_argv[1],user_argv[2]);
```

将IOT_MQTT_Publish函数中第二个参数改为

```
TOPIC_POST
```

编译SDK

坑爹的地方来啦！！阿里云开始演我了！！！
下面的报错花了我很长时间，最后确定为SDK对树莓派系统的兼容性问题。
在SDK根目录下执行

```
//清除之前编译生成的程序
sudo make distclean
```

然后编译

```
sudo make
```

然后疯狂报错

```
cmp/cmp-example.o：在函数‘cmp_client’中：
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:120：对‘IOT_CMP_Init’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:124：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:134：对‘IOT_CMP_Register’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:138：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:145：对‘IOT_CMP_Yield’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:148：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:174：对‘IOT_CMP_Send’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:182：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:187：对‘IOT_CMP_Yield’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:190：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:201：对‘IOT_CMP_Unregister’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:204：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:209：对‘IOT_CMP_Yield’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:212：对‘IOT_CMP_Deinit’未定义的引用
/home/raindy/al/iotkit-embedded/sample/cmp/cmp-example.c:219：对‘IOT_CMP_Deinit’未定义的引用
collect2: error: ld returned 1 exit status
/home/raindy/al/iotkit-embedded/build-rules/_rules-prog.mk:44: recipe for target 'cmp-example' failed
make[3]: *** [cmp-example] Error 1
/home/raindy/al/iotkit-embedded/build-rules/_rules-prog.mk:61: recipe for target 'cmp-example' failed
make[2]: *** [cmp-example] Error 2
/home/raindy/al/iotkit-embedded/build-rules/_rules-submods.mk:118: recipe for target 'sample' failed
make[1]: *** [sample] Error 2
/home/raindy/al/iotkit-embedded/build-rules/_rules-submods.mk:5: recipe for target 'sub-mods' failed
make: *** [sub-mods] Error 2
```



```
/home/raindy/al/iotkit-embedded/src/cmp/Link-CMP/src/iotx_cmp_common.c:61:19: error: ‘string_TIMESTAMP’ defined but not
 used [-Werror=unused-const-variable=]
 static const char string_TIMESTAMP[] CMP_READ_ONLY = "2524608000000";
                   ^~~~~~~~~~~~~~~~
/home/raindy/al/iotkit-embedded/src/cmp/Link-CMP/src/iotx_cmp_common.c:60:19: error: ‘string_MD5_METHOD’ defined but no
t used [-Werror=unused-const-variable=]
 static const char string_MD5_METHOD[] CMP_READ_ONLY = "hmacmd5";
                   ^~~~~~~~~~~~~~~~
cc1: all warnings being treated as errors
/home/raindy/al/iotkit-embedded/build-rules/_rules-flat.mk:87: recipe for target 'Link-CMP/src/iotx_cmp_common.o' faile
d
make[2]: *** [Link-CMP/src/iotx_cmp_common.o] Error 1
```


提工单，经过两天陆续跟阿里云5位售后工程师亲切交谈，他们告诉我，把第二个图中报错的变量注释掉。ok那就注释掉吧，

然而事情没有那么简单，注释完之后又报错

```
                                    makefile
/home/raindy/al/iotkit-embedded/src/dm/src/dm_thing_manager.c:33:19: error: 'string_method_name_thing_
dsl_post_reply' defined but not used [-Werror=unused-const-variable=]
 static const char string_method_name_thing_dsl_post_reply[] __DM_READ_ONLY__ = METHOD_NAME_THING_DSL_
POST_REPLY;
                   ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
cc1: all warnings being treated as errors
/home/raindy/al/iotkit-embedded/build-rules/_rules-flat.mk:87: recipe for target 'src/dm_thing_manager
.o' failed
make[2]: *** [src/dm_thing_manager.o] Error 1
```


注释完重新编译，还是有error，这个报错还是变量定义未使用，那就继续注释掉。事情远远没有那么简单。继续报错。

```
/home/raindy/al/iotkit-embedded/src/dm/src/dm_thing.c: In function 'install_cjson_item_data_type':
/home/raindy/al/iotkit-embedded/src/dm/src/dm_thing.c:598:57: error: format '%lu' expects argument of type 'long uns
igned int', but argument 4 has type 'size_t {aka unsigned int}' [-Werror=format=]
             snprintf(_key_index, KEY_BUFF_SIZE - 1, "[%lu]", index + 1);
                                                         ^
At top level:
/home/raindy/al/iotkit-embedded/src/dm/src/dm_thing.c:65:19: error: 'string_true' defined but not used [-Werror=unus
ed-const-variable=]
 static const char string_true[] __DM_READ_ONLY__ = "true";
                   ^~~~~~~~~~~
cc1: all warnings being treated as errors
/home/raindy/al/iotkit-embedded/build-rules/_rules-flat.mk:87: recipe for target 'src/dm_thing.o' failed
make[2]: *** [src/dm_thing.o] Error 1
```


嗯，这次不仅报了个变量未使用的错误，还有个数据类型报错？？？谁知道还有多少错，而且编译一次好几分钟，被阿里云演的透彻。没办法，继续注释掉未使用的变量，把%lu改成%u，再编译，这次居然不报错了。成功生成了output文件夹，下面有release目录，再下面有bin目录，bin下面有各种demo程序，太tm美妙了。

运行mqtt的demo程序，加上loop参数让它持续运行，就可以在控制台看到设备激活并在线的状态。

```
sudo ./output/release/bin/mqtt-example loop
```

另外，在跟阿里云交流的时候顺手开了一台ubuntu系统的vps，按照文档给的流程走，敲敲打打命令，几分钟就连上了，没有遇到任何报错阻碍，简直不要太真实。

接下来，将通过树莓派运行python程序给mqtt的demo程序传参，将真实数据发送到云平台。

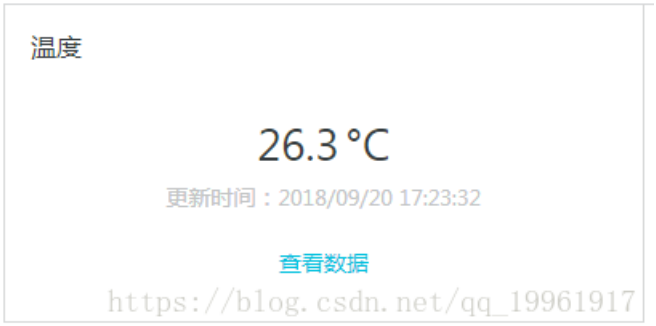
写Python获取数据并传参

传感器使用DHT22温湿度传感器，连接树莓派的Pin 13（BCM编码为GPIO27）引脚。此处只将代码贴出来，详细步骤将在另一篇文章中分析。

```
import Adafruit_DHT
import time
import os
sensor = Adafruit_DHT.DHT22
pin = 27
while True:
 try:
  time.sleep(3)
  hu, temp = Adafruit_DHT.read_retry(sensor, pin)
  t = time.time()
  os.system("sudo /(SDK根目录)/output/release/bin/mqtt-example {0:0.1f} {1}".format(temp,int(round(t*1000))))
 except RuntimeError as e:
  print("error\n{0}".format(e))
 except:
  print("error\nFailed to read sensor data!")
```

查看控制台

查看控制台中设备控制面板中运行状态选项卡，即可看到最新数据



还可以查看历史数据