SIMATIC IOT2000 连接阿里云

作者：　吴腾飞
日期：　2017/10/26

Catalog

1 任务

1.1 概览

介绍

本应用案例主要介绍如何使用 SIMATIC IOT2000 与阿里云进行连接及数据传输。同时，本案例也介绍了如何通过 IOT2000 IO 模块采集 IO 信号。

目标

通过本文档，您将学习到如何

- 使用 node-red MQTT and IO 模块
- 与阿里云连接及传输数据
- 在阿里云中订阅及使用数据

2 应用案例

2.1 预备知识

1. 在阿里云注册账号，具体内容请参考这里
2. 在阿里云中创建 ECS 实例，并安装 Linux 操作系统
3. 了解如何使用 IOT2000 IO 模块，具体内容请参考这里
4. IOT2000 IO 模块与传感器正确连接
5. 了解 node-red 的基本用法
6. 了解 python 及 flask 相关内容

2.2 配置 node-red 程序

配置 node-red IO 节点

1. 选择“intel_gpio digital input”节点，并拖拽至编辑面板中

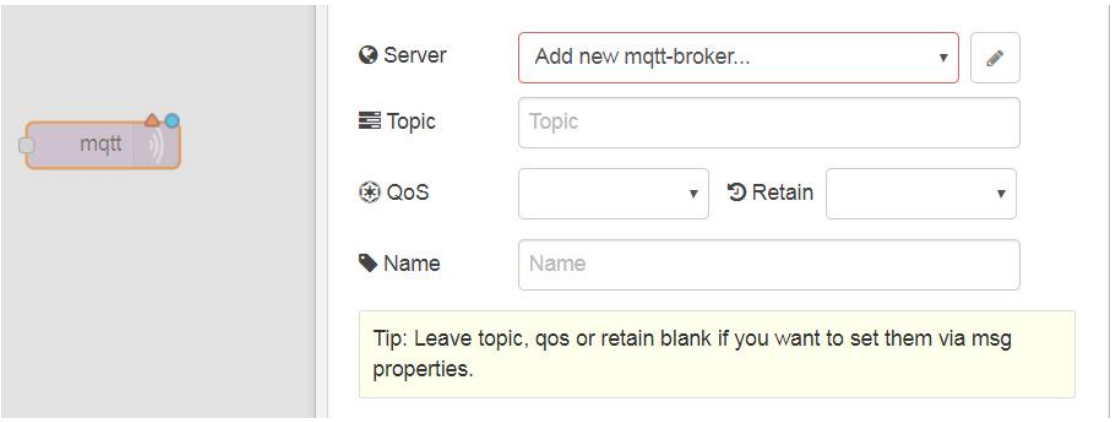
2. 双击该节点

3. 选择合适的 IO 管脚及中断触发模式。

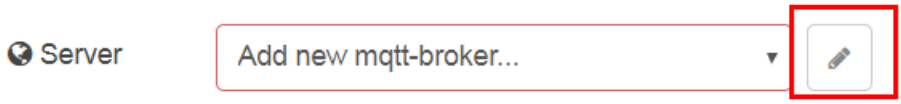
配置 node-red mqtt 节点

1. 选择“mqtt output”节点，并拖拽至编辑面板中

2. 双击该节点

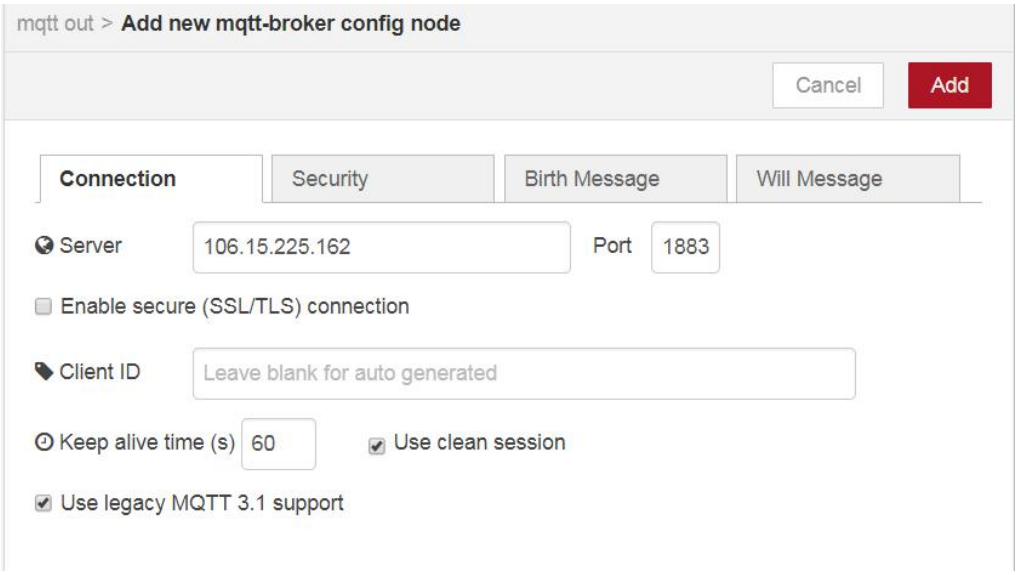


3. 点击编辑按钮，添加新的 mqtt-broker

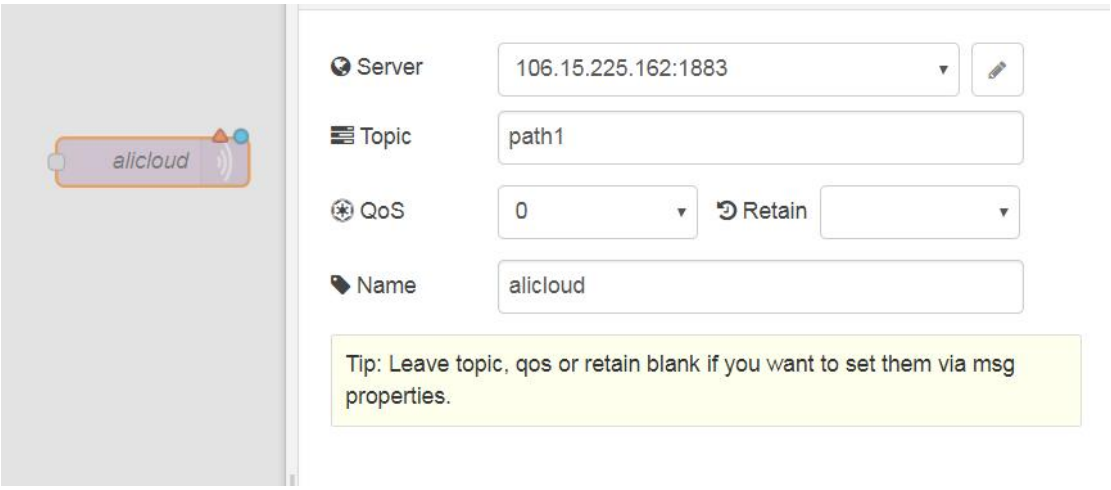


4. 填写 mqtt-broker 信息

在下一节我们将会介绍如何在阿里云中部署 mqtt-broker。现在，我们仅需要填入 ECS 服务器的 IP 地址即可。

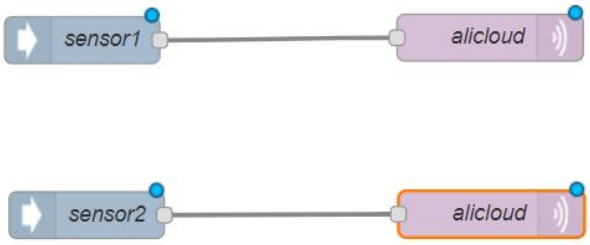


5. 填写 mqtt 节点信息



连接两个节点

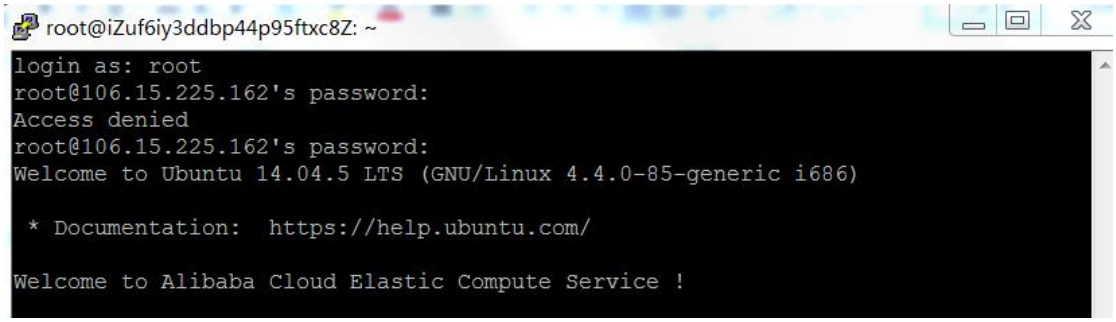
重复以上过程并添加第二个传感器



点击 **Deploy** 按钮，部署 **node-red** 程序

2.3 在阿里云中部署 mqtt-broker

使用 **putty** 登录阿里云 **ECS** 服务器



安装 **mosquitto**

Mosquitto 是一款开源的 mqtt-broker，在本应用案例中我们将使用 mosquitto 作为 mqtt-broker。Mosquitto 的最新版本为 1.4.14。

在 putty 中输入以下命令进行编译安装：

wget http://mosquitto.org/files/source/mosquitto-1.4.14.tar.gz

tar zxvf mosquitto-1.4.14.tar.gz

cd mosquitto-1.4.14

make

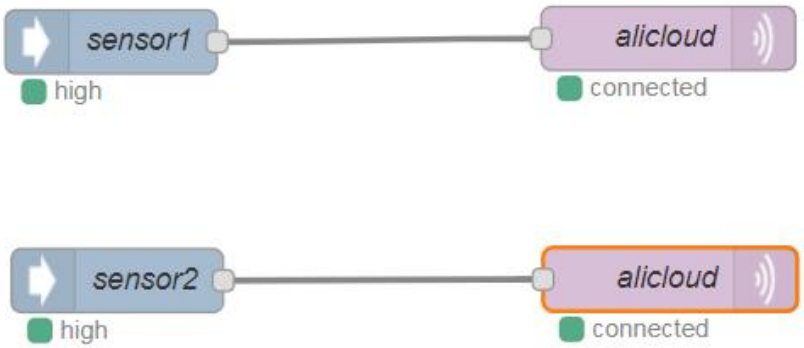
make install

运行 Mosquitto

在 putty 中运行 mosquitto –v 命令，即可启动 Mosquitto

```
root@iZuf6iy3ddbp44p95ftxc8Z:~# mosquitto -v
1509001227: mosquitto version 1.4.14 (build date 2017-08-14 16:56:24+0800) starting
1509001227: Using default config.
1509001227: Opening ipv4 listen socket on port 1883.
1509001227: Opening ipv6 listen socket on port 1883.
1509001228: New connection from 218.90.84.189 on port 1883.
1509001228: New client connected from 218.90.84.189 as mqtt_2b18050f.dde0aa (c1, k60).
1509001228: Sending CONNACK to mqtt_2b18050f.dde0aa (0, 0)
```

在 node-red 的编辑面板中，我们可以看到“mqtt output”节点已经成功的连接至阿里云中的 mqtt-broker。



2.4 在 python 应用中订阅及使用数据

Mqtt 采用发布者/订阅者的模式工作。在本应用案例中，node-red 作为发布者向 mqtt-broker 发布传感器数据，python 应用作为订阅者向 mqtt-broker 订阅传感器数据。

在 python 中，paho-mqtt 是一款优秀的 mqtt 客户端扩展库。在本应用案例中，我们将采用 paho-mqtt 作为 mqtt 客户端用于向 mqtt-broker 订阅数据。

安装 paho-mqtt

在 putty 中运行 pip install paho-mqtt，即可安装 paho-mqtt。

结合 flask 的 python 应用示例代码

```
from flask import Flask, render_template, request
import json
import paho.mqtt.client as mqtt

MqttData = {
	"path1": 0,
	"path2": 0
}


def on_connect(client, userdata, flags, rc):
	# subscribe topic
	client.subscribe("path1")
	client.subscribe("path2")

def on_message(client, userdata, msg):
	# parse json message
	topic = msg.topic
	print topic
	if topic == "path1":
		MqttData['path1'] = MqttData['path1'] + 1
	elif topic == "path2":
		MqttData['path2'] = MqttData['path2'] + 1
	print MqttData

def create_app():
	# mqtt part
	client = mqtt.Client()
	#client.username_pw_set(username, password)
	client.on_connect = on_connect
	client.on_message = on_message
	client.connect('localhost')
	client.loop_start()

	app = Flask(__name__)
	return app
```

```
application = create_app()

@application.route('/', methods = ['GET', 'POST'])
def marbledemo():
	'''
	this function is for marble demo
	'''
	if request.method == "POST":
		data = {
			'pathNum1': MqttData['path1'],
			'pathNum2': MqttData['path2']
		}
		return json.dumps(data)
	else:
		return render_template('marbledemo.html')

if __name__ == '__main__':
	# flask part
	application.run()
```