Tài liệu mô tả cấu trúc & hướng dẫn sử dụng hệ thống API

Link github: https://github.com/dangnguyenquang/VTC

Author: dangnguyenquang

Collaborators: NMD555

A. Thông tin server và broker

- Nichietsu_server ’s link: http://nodejs-api-iot.app-provider.xyz
- VTC_server ’s link: https://ialert.vnpt.vn/apivtc/conn/v1/message
- Token server VTC: dg-fDVjn2UCCp0VlJB9IIj3eeDnTud-crV9DroTKRB4OCkxC9qzqxbfH-0uor604hu6hsStS_hQUmVnmzeMfJuI4QdXbsMF88CU6PzLmt0A=
- mqtt_broker: iot-solar.nichietsuvn.com
- mqtt_port: 1884
- mqtt_username : guest
- mqtt_password : 123456a@

B. Thông tin hệ thống :

I. Cấu trúc folder và file:

1. Folder**:**
   - device: gồm có những file liên quan đến việc thiết bị gửi data qua mqtt, database, server VTC (file api.js và catchMes.js).x
   - device-sync: có file đồng bộ thiết bị khi có thiết bị mới.
   - json-server: gồm có file chạy server của Nichietsu.
   - doc: gồm có file tài liệu chứa những mô tả cấu trúc, yêu cầu, nhiệm vụ của hệ thống.
   - python: có file thêm thiết bị mới vào database.
2. File**:**
   a. File gửi API từ thiết bị (path: /device/api.js): Có nhiệm vụ gửi API alive từ thiết bị lên server VTC theo interval, thông báo lập tức cho server VTC biết khi nhận được thông tin cháy. (URL: api/message)
   b. File gửi bản tin đồng bộ thiết bị (path: device-sync/apidevice.js): Bắt buộc phải được kích hoạt thủ công khi có một thiết bị mới thêm vào hệ thống. (URL: api/device)

c. File server (json-server/app.js): là file server xử lý những yêu cầu API từ VTC bao gồm :

- Kích hoạt/Ngưng dịch vụ (URL: api/active)
- Lấy thông tin thiết bị (URL: api/deviceinfo)
- Gửi yêu cầu test định kỳ (URL: api/requestdevice)
- Cấu hình wifi (URL: api/config)
- Đồng bộ thiết bị (URL: api/syncdevice)

d. File bắt data từ thiết bị qua MQTT (path: device/catchMes.js): Có nhiệm vụ nhận data từ thiết bị và thêm vào db.

e. File thêm thiết bị mới (path : python/importdeviceInfo.py): Được sửa đổi và kích hoạt khi có nhu cầu thêm thiết bị mới.

II. Cấu trúc database (mongodb://localhost:27017):

Có 2 collection:

1. deviceData:

```
id: String, // ID Thiết bị
payload: String, // data nhận được
timestamp: Date // Thời điểm nhận được
```

payload: INPUT-1 (có cháy)

INPUT-0 (trạng thái bình thường)

BUTTO-1 (công tắc được nhấn)

2. deviceInfo:

```
deviceID: String,
serialNumber: String,
IMEI: String,
ngaySanXuat: String,
feeStatus: String,
wifiSSID: String,
wifiPASS: String,
connectStatus: String,
interval: Number,
lastUpdate: Date
```

Những thông tin trên được nạp từ file importdeviceInfo.py

III. Mô tả hệ thống hoạt động:

Nhận bản tin từ quản lý thiết bị (api.js) (được tự động gửi đi sau mỗi 15s):

1. catchMes.js sẽ ghi nhận lại những tin nhắn từ thiết bị vào VTC/deviceData
2. File api.js sẽ lấy thông tin interval từ VTC/deviceInfo và data từ VTC/deviceData nếu (thời gian hiện tại - timestamp > 3*interval) thì thông báo cho server VTC thiết bị không phản hồi, ngược lại thì thông báo thiết bị còn sống
3. Nếu có hỏa hoạn, api.js ngay lập tức nhận thông tin từ mqtt và báo lên server VTC

   Lưu ý : để tránh trường hợp thiết bị còn hoạt động nhưng bị lỗi 1 vài thời điểm nên gửi đi không đúng theo interval. Ta lấy interval*3.

   Trong api.js có 3 arr cần chú ý :

   deviceIds: Chứa id của các thiết bị được lấy ra từ db.

   topics: Chứa topic của các thiết bị để sub vào mqtt broker.

   checkLastedMess: Chứa thông tin tin nhắn cuối cùng của từng thiết bị (cần để check trạng thái bình thường sau khi cháy của bản tin kind : 1/ state : 3)

   Các mảng trên sẽ thay đổi nếu nhận thấy có sự thay đổi trong db.

Đồng bộ thiết bị (api.js): api.js sẽ lấy dữ liệu của tất cả thiết bị trong VTC/deviceInfo và gửi đi.

Đồng bộ thiết bị (api.js): khi nhận được yêu cầu app.js sẽ lấy dữ liệu của tất cả thiết bị trong VTC/deviceInfo và gửi đi

Kích hoạt ngưng dịch vụ (app.js) : app.js tìm device trong VTC/deviceInfo theo deviceId nhận được từ res và thay đổi feeStatus.

Các status được chấp nhận là: 0 (Đăng ký), 1 (Kích hoạt), 2 (Ngưng dịch vụ), 3 (Hủy dịch vụ). Nếu ngoài các status trên app.js sẽ log ra : ‘Sai status’.

Thông tin thiết bị (app.js) :

1. Sau khi nhận được yêu cầu app.js sẽ tìm thông tin thiết bị trong VTC/deviceInfo dựa trên deviceId, nếu không tìm thấy deviceId log ra : ‘Không tìm thấy deviceId’ (mục đích lấy thông tin của thiết bị).
2. Sau đó, app.js sẽ pub 1 mess có nội dung : PING đến topic có tên của thiết bị cần truy xuất thông tin. Thiết bị sẽ trả về thông tin trạng thái hiện tại (có cháy

hay không) nếu sau 10s thiết bị không trả về, kết luận thiết bị không còn hoạt động (mục đích lấy trạng thái hiện tại của thiết bị).

3. Gửi API gồm các thông tin lấy được của thiết bị đi.

Gửi yêu cầu xuống thiết bị (app.js)

1. app.js sau khi nhận res sẽ check xem deviceId có tồn tại trong db hay không, nếu không có, log ra 'deviceId không tồn tại'.
2. app.js sẽ lấy thông tin interval từ VTC/deviceInfo và data từ VTC/deviceData nếu (thời gian hiện tại - timestamp > 3*interval) thì thông báo cho server VTC thiết bị không phản hồi, ngược lại thì thông báo thiết bị còn sống.
3. Gửi bản tin định kỳ đến VTC.

Cấu hình wifi (app.js): Sau khi nhận được res sẽ check xem deviceId có tồn tại trong db hay không, nếu không, log ra 'Không tìm thấy deviceId'. Thay đổi ssid, pwd của thiết bị trong db.

IV. Các loại bản tin của VTC

1. Bản tin báo cháy (kind: 1):

   Báo cháy khi xảy ra sự cố (state: 1)

   Bình thường khi giải quyết xong sự cố (state: 3)

2. Bản tin cập nhật trang thái thiết bi (kind: 2):

   Online (state: 1) / Offline (state: 2)

3. Bản tin kiểm tra định kì (kind: 3):

   Online (state: 1) / Offline (state: 2)

4. Bản tin test thiết bị bằng nút bấm (kind: 4):

   State mặc định là 1

Lưu ý: hệ thống phải chạy đồng thời 3 file: **catchMes.js, app.js, api.js.**

Khi cần thêm thiết bị mới vào hệ thống: chạy file **importdeviceInfo.py** để thêm device vào db, chạy file **apidevice.js** để đồng bộ thiết bị bên VTC.

Mô hình hóa hệ thống