Các khái niệm trong MQTT ?

Nói lại cách giao tiếp thông thường khi một device kết nối với server, sẽ gửi POST request với giá trị của sensor(ở đây ví dụ ta sẽ gửi dữ liệu từ sensor lên server), sau đó sẽ là ngắt kết nối với server, đợi khoảng 1 thời gian nào đó, sau đó lặp lại các bước trên. Vậy mỗi lần cần gửi giá trị của sensor thì device lại phải thực hiện kết nối với server lại, việc kết nối nhiều lần ở đây ta thấy là không cần thiết. Ngược lại, nếu chúng ta muốn gửi data từ server về device, thì chỉ có cách duy nhất là device phải kết nối với server và phải thực hiện kiểm tra nếu có update dữ liệu mới. Đây rõ ràng không phải là cách hiệu quả nhất để gửi dữ liệu. Giao thức HTTP sẽ sử dụng phương thức giao tiếp là request-response, phương thức này thì tốt cho việc transfer các dữ liệu lớn nhưng nó không phải là protocol hữu hiệu nhất khi chúng ta chỉ cần gửi vài byte dữ liệu(giá trị sensor), và đó là lý do cần phải dùng MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức gởi dạng publish/subscribe sử dụng cho các thiết bị Internet of Things với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M. MQTT cũng là giao thức sử dụng trong Facebook Messager.

Các định nghĩa **“subscribe”, “publish”, “qos”, “retain”, “last will and testament (lwt)”** trong giao thức MQTT là gì ?

Publish, subscribe

Trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là mqtt client – gọi tắt là client) kết nối tới một MQTT server (gọi là broker). Mỗi client sẽ đăng ký một vài kênh (topic), ví dụ như “/client1/channel1”, “/client1/channel2”. Quá trình đăng ký này gọi là **“subscribe”**, giống như chúng ta đăng ký nhận tin trên một kênh Youtube vậy. Mỗi client sẽ nhận

được dữ liệu khi bất kỳ trạm nào khác gởi dữ liệu và kênh đã đăng ký. Khi một client gởi dữ liệu tới kênh đó, gọi là **“publish”**.

QoS

Ở đây có 3 tuỳ chọn *_QoS (Qualities of service)_ * khi “publish” và “subscribe”:

- **QoS0** Broker/client sẽ gởi dữ liệu đúng 1 lần, quá trình gởi được xác nhận bởi chỉ giao thức TCP/IP, giống kiểu đem con bỏ chợ.
- **QoS1** Broker/client sẽ gởi dữ liệu với ít nhất 1 lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
- **QoS2** Broker/client đảm bảm khi gởi dữ liệu thì phía nhận chỉ nhận được đúng 1 lần, quá trình này phải trải qua 4 bước bắt tay.

Bạn có thể tham khảo thêm về **QoS**

Một gói tin có thể được gởi ở bất kỳ QoS nào, và các client cũng có thể subscribe với bất kỳ yêu cầu QoS nào. Có nghĩa là client sẽ lựa chọn QoS tối đa mà nó có để nhận tin. Ví dụ, nếu 1 gói dữ liệu được publish với QoS2, và client subscribe với QoS0, thì gói dữ liệu được nhận về client này sẽ được broker gởi với QoS0, và 1 client khác đăng ký cùng kênh này với QoS 2, thì nó sẽ được Broker gởi dữ liệu với QoS2.

Một ví dụ khác, nếu 1 client subscribe với QoS2 và gói dữ liệu gởi vào kênh đó publish với QoS0 thì client đó sẽ được Broker gởi dữ liệu với QoS0. QoS càng cao thì càng đáng tin cậy, đồng thời độ trễ và băng thông đòi hỏi cũng cao hơn.

Retain

Nếu RETAIN được set bằng 1, khi gói tin được publish từ Client, Broker **PHẢI** lưu trữ lại gói tin với QoS, và nó sẽ được gởi đến bất kỳ Client nào

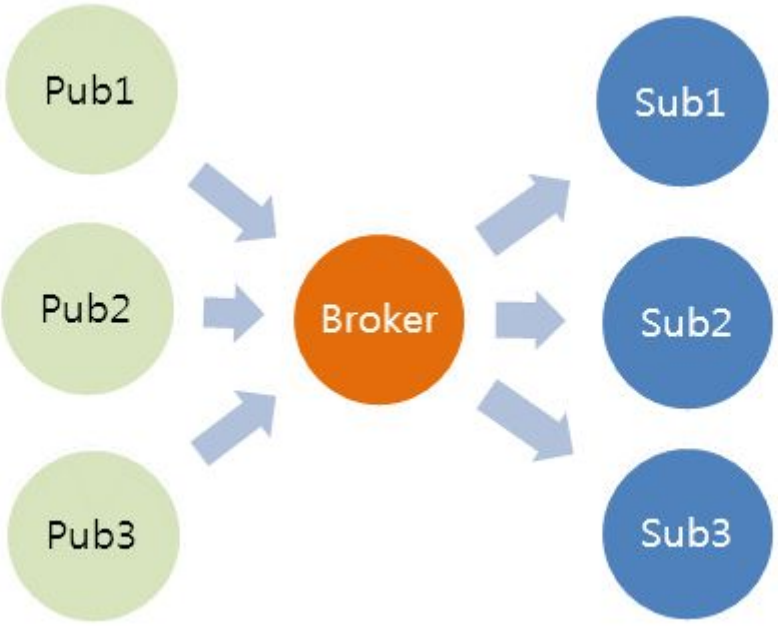
subscribe cùng kênh trong tương lai. Khi một Client kết nối tới Broker và subscribe, nó sẽ nhận được gói tin cuối cùng có RETAIN = 1 với bất kỳ topic nào mà nó đăng ký trùng. Tuy nhiên, nếu Broker nhận được gói tin mà có QoS = 0 và RETAIN = 1, nó sẽ huỷ tất cả các gói tin có RETAIN = 1 trước đó. Và phải lưu gói tin này lại, nhưng hoàn toàn có thể huỷ bất kỳ lúc nào.

Khi publish một gói dữ liệu đến Client, Broker phải se RETAIN = 1 nếu gói được gởi như là kết quả của việc subscribe mới của Client (giống như tin nhắn ACK báo subscribe thành công). RETAIN phải bằng 0 nếu không quan tâm tới kết quả của việc subscribe.

LWT

Gói tin LWT (last will and testament) không thực sự biết được Client có trực tuyến hay không, cái này do gói tin KeepAlive đảm nhận. Tuy nhiên gói tin LWT như là thông tin điều gì sẽ xảy đến sau khi thiết bị ngoại tuyến.

MQTT hoạt động như thế nào ?



MQTT rất nhẹ và nhanh, chỉ cần vài byte để kết nối tới server và kết nối này được giữ mọi lúc. Kết nối của nó sẽ dùng ít dữ liệu và thời gian hơn giao thức HTTP. Vậy MQTT hoạt động ra sao ?

Toàn bộ hệ thống của chúng ta sẽ bao gồm rất nhiều clients và 1 broker duy nhất, tất cả các device của chúng ta được gọi chung là clients, client cũng có thể là điện thoại, là laptop, là pc hoặc là esp8266. Mỗi device này chỉ kết nối với một broker duy nhất và chúng không kết nối với nhau.

Toàn bộ hệ thống sẽ dựa trên phương thức kết nối là publish – subcribe, mỗi client có thể là một publisher – gửi messages, hoặc là subcriber – lắng nghe message tới, hoặc đồng thời là cả 2 trong cùng 1 thời điểm.

Broker ở đây là loại server với nhiệm vụ là cho phép publish mesage từ publisher và chuyển tiếp nó tới subcriber.

Vậy làm thế nào để broker hiểu được để chuyển đúng message tới subscriber ? Chúng ta thử nghĩ nếu mỗi subscriber đều nhận được mọi message, trong khi đó có rất nhiều publisher gửi message trong hệ thống, nghĩa là subscriber sẽ nhận được 1 đống tin nhắn rác, để giải quyết vấn đề này thì cần có các topic.

Mỗi publisher sẽ gửi tới một hoặc nhiều topic và mỗi subscriber sẽ subscribe từ một hoặc nhiều topic có liên quan. Các topic này sẽ có dạng cấu trúc cây, và được phân chia bởi dấu /

Ví dụ 1: Chúng ta có nhiệt độ môi trường ở một số nơi khác nhau và tôi thì thích nhiệt độ ở trong vườn nhà mình. Tôi sẽ dùng smartphone để subscribe topic là `Sensors/Garden/Temperature`. Còn device dùng để đo nhiệt độ môi trường là ESP8266 sẽ được đặt ở trong vườn, kết nối với internet và publish giá trị nhiệt độ lên topic `Sensors/Garden/Temperature`, và lúc này thì smartphone của tôi sẽ nhận được thông tin giá trị nhiệt độ hiện tại.

Ví dụ 2: Một ví dụ khác khi bạn dùng với LWT và keep alive là ta có 1 cảm biến, nó gửi những dữ liệu quan trọng và rất không thường xuyên. Nó có publish trước với Broker một tin nhắn lwt ở topic

`/node/gone-offline` với tin nhắn `id` của nó. Và tôi cũng subscribe topic `/node/gone-offline`, sẽ gởi SMS tới điện thoại thôi mỗi khi nhận được tin nhắn nào ở kênh mà tôi subscribe.

Trong quá trình hoạt động, cảm biến luôn giữ kết nối với Broker bởi việc luôn gởi gói tin keepAlive. Nhưng nếu vì lý do gì đó, cảm biến này chuyển sang ngoại tuyến, kết nối tới Broker timeout do Broker không còn nhận được gói keepAlive.

Lúc này, do cảm biến của tôi đã đăng ký LWT, do vậy broker sẽ đóng kết nối của Cảm biến, đồng thời sẽ publish một gói tin là Id của cảm biến vào kênh `/node/gone-offline`, dĩ nhiên là tôi cũng sẽ nhận được tin nhắn báo rằng cảm biến yêu quý của mình đã ngoại tuyến.

Tóm gọn: Ngoài việc đóng kết nối của Client đã ngoại tuyến, gói tin LWT có thể được định nghĩa trước và được gởi bởi Broker tới kênh nào đó khi thiết bị đăng ký LWT ngoại tuyến.

Một ví dụ minh họa sống động về MQTT pub sub lên các topic bạn có thể xem tại đây

Thực hành 2

Tiếp tục từ bài thực hành 1 thì chúng ta sẽ phát triển thêm một chút nữa, xây dựng hệ thống với 2 client là eps8266 và laptop(pub sub lên broker dùng mqttlen) và broker là cloudMQTT

ESP8266 sẽ thực hiện publish data lên topic esp/test sau mỗi 10s và subcribe data từ broker

```
#include <ESP8266WiFi.h> #include <PubSubClient.h>
const char* ssid = "wifi_name"; const char* password
= "pass"; const char* mqttServer =
"m12.cloudmqtt.com"; const int mqttPort = 10769;
const char* mqttUser = "your_username"; const char*
```

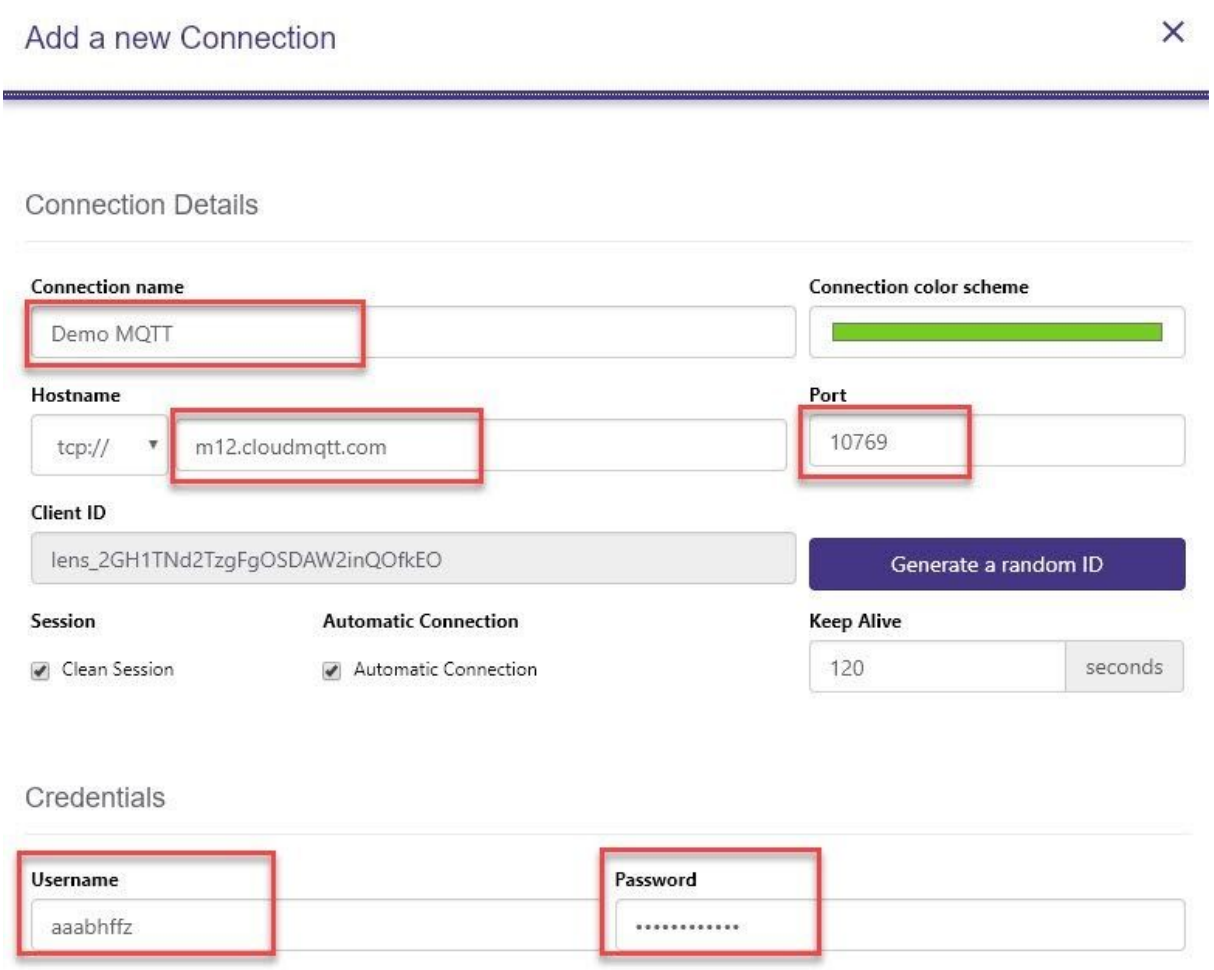
```
mqttPassword = "your_password"; long lastMsg = 0;
char msg[50]; int value = 0; WiFiClient espClient;
PubSubClient client(espClient); void setup() {
Serial.begin(115200); WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.println("Connecting to WiFi.."); }
Serial.println("Connected to the WiFi network");
client.setServer(mqttServer, mqttPort);
client.setCallback(callback); while
(!client.connected()) { Serial.println("Connecting
to MQTT..."); if (client.connect("ESP8266Client",
mqttUser, mqttPassword )) {
Serial.println("connected"); } else {
Serial.print("failed with state ");
Serial.print(client.state()); delay(2000); } } //Pub
Hello to esp/test topic and sub this topic
client.publish("esp/test", "Hello from ESP8266");
client.subscribe("esp/test"); } void callback(char*
topic, byte* payload, unsigned int length) {
Serial.print("Message arrived in topic: ");
Serial.println(topic); Serial.print("Message:"); for
(int i = 0; i < length; i++) {
Serial.print((char)payload[i]); } Serial.println();
Serial.println("-----------------------"); } void
loop() { client.loop(); long now = millis(); if (now
- lastMsg > 8000) { lastMsg = now; ++value; snprintf
(msg, 75, "hello world #%ld", value); //
```

```
client.publish(mqtt_topic_pub, msg); if (value < 10)
{ Serial.print("Publish message: ");
Serial.println(msg); client.publish("esp/test",
msg); } } }
```

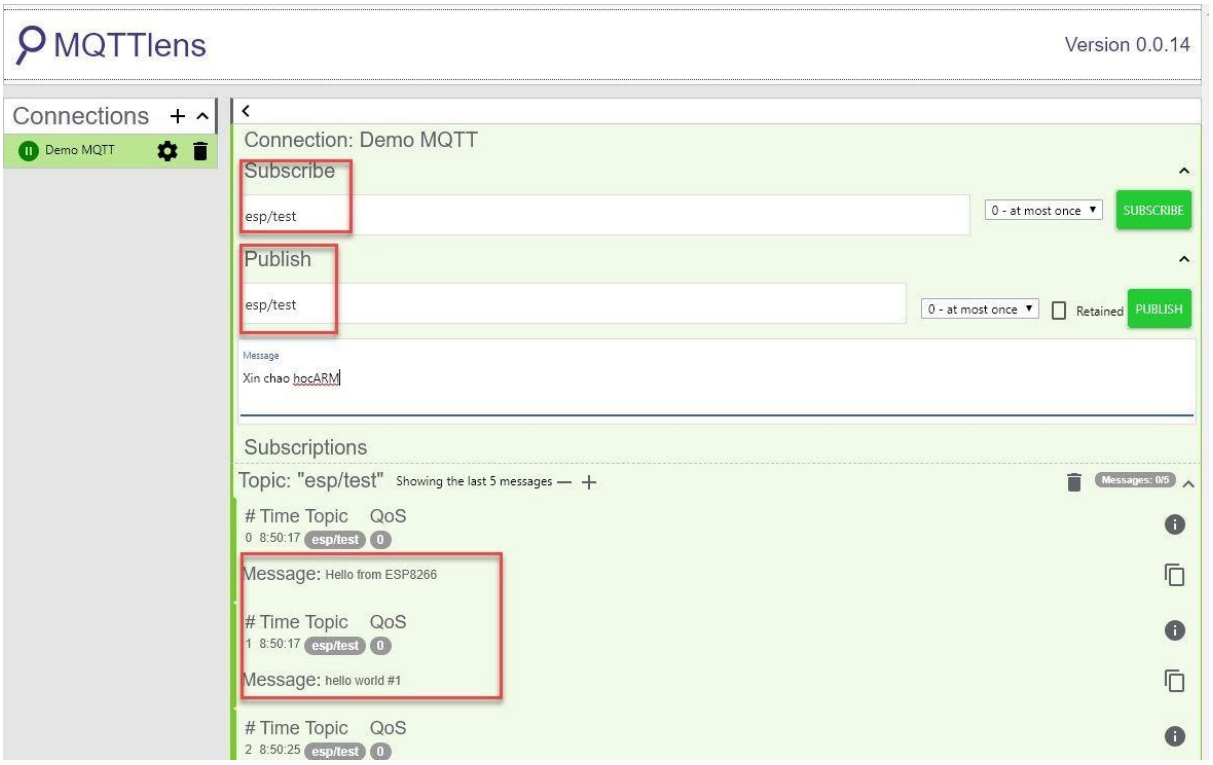
Arduino

PC thực hiện publish/subscribe data lên topic esp/test bằng mqttlen

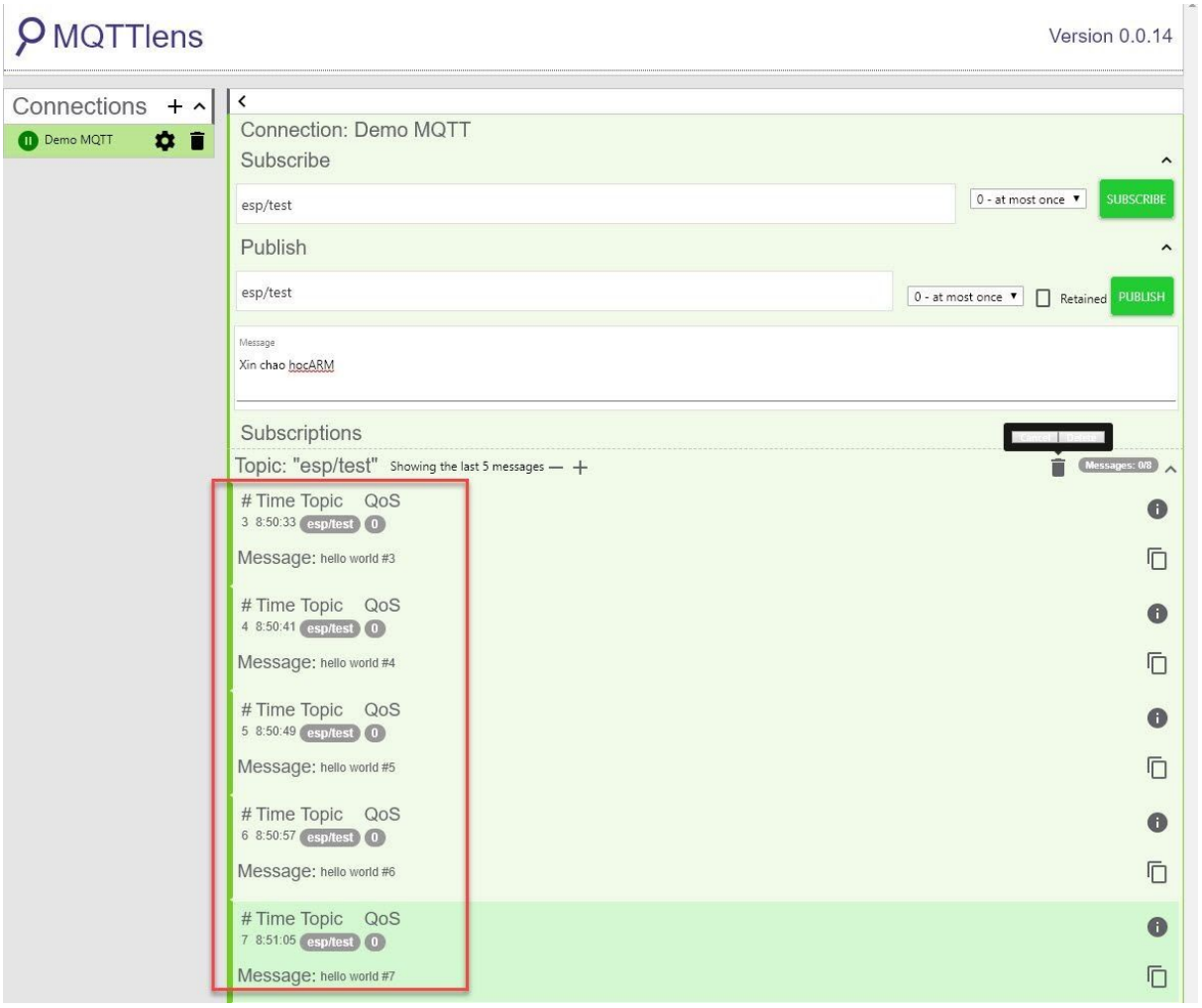
- Cấu hình MQTTlens, bạn tạo mới một connection với tên bất kỳ, hostname là tên server cloudmqtt với port là 10769, lưu ý phần credentials nhập username và password trên cloudmqtt của bạn vào



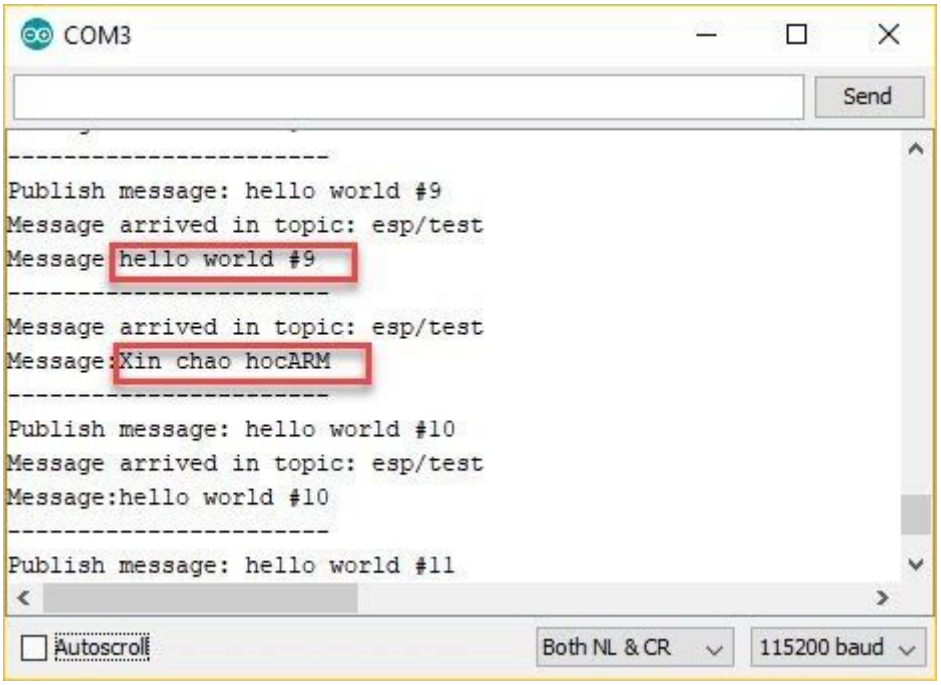
- Publish/Subcribe với MQTTlen: để nhận được message đưa lên broker tại topic esp/test thì ta nhập topic này tại phần subscribe, sau đó ấn nút subscribe, và để publish message lên topic esp/test trên broker thì ta nhập topic và nội dung message, sau đó ấn publish.



Phần subscriptions sẽ là thông tin của các message gửi lên topic, ở đây ta có thể thấy thông tin từ esp8266 gửi lên broker thông qua topic là esp/test



Kết quả sau khi publish và subcribe với ESP8266: ta thấy các message publish lên sẽ có dạng hello world #x, khi có một message được publish từ mqttlens thì ESP sẽ nhận được tin với nội dung Xin chao hocARM


COM3
Send
Publish message: hello world #9
Message arrived in topic: esp/test
Message hello world #9
Message arrived in topic: esp/test
Message:Xin chao hocARM
Publish message: hello world #10
Message arrived in topic: esp/test
Message:hello world #10
Publish message: hello world #11
Autoscroll
Both NL & CR
115200 baud