IT/Go 언어 (Golang)

[Golang] 자체 MQTT 브로커 서버 구현 - 1

공대생의 차고 2020. 2. 26. 16:41

MQTT Broker 서버 구현

MQTT 브로커 서버 구현에 앞서, 만약 이론적 배경이 궁금하다면 아래 포스트를 먼저 읽고 오는 것을 추천한다.

링크 1: MQTT 원리 이론편

[통신 이론] MQTT, MQTT Protocol (M…

이 론 MQTT(Message Queueing Telemetry Transport)는 2016 년 국제 표준화 된 (ISO 표준 ISO/IEC PRF 20922) 발행-구독…

underflow101.tistory.com

링크 2: MQTT - Arduino(ESP32) 실전편

아래는 Golang으로 구현한 MQTT 브로커 서버이며,
eclipse.paho.mqtt 패키지를 기반으로 구현했다. [eclipse/paho.mqtt.golang 링크]

아래는 Depedencies(Requirement)이며, 터미널에 $ go get 명령어로 다운로드 받을 수 있다.

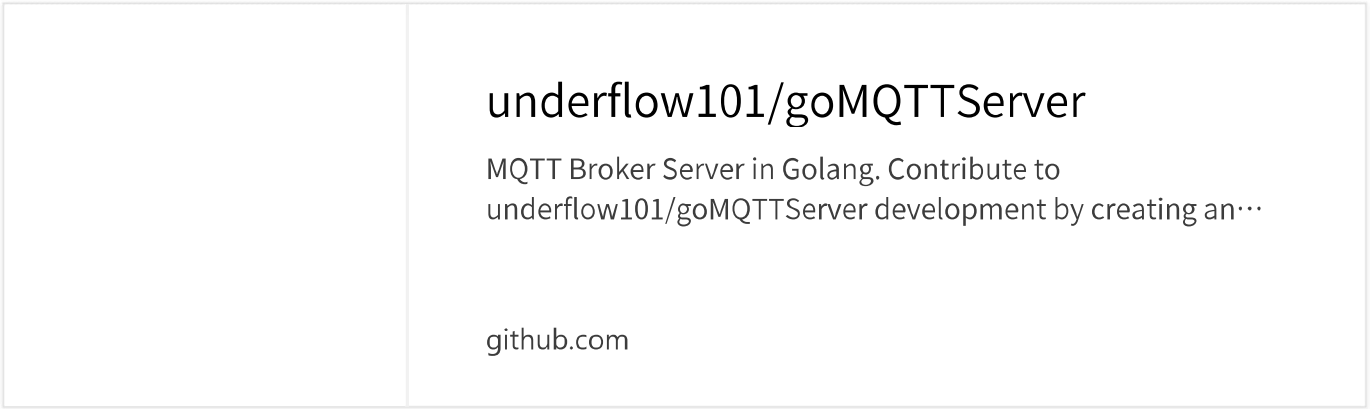
```
module github.com/underflow101/goMQTTServer

go 1.13

require (
    github.com/Shopify/sarama            v1.23.0
    github.com/bitly/go-simplejson       v0.5.0
    github.com/eclipse/paho.mqtt.golang v1.2.0
    github.com/gin-gonic/gin             v1.4.0
    github.com/labstack/gommon           v0.3.0
    github.com/patrickmn/go-cache        v2.1.0+incompatible
    github.com/satori/go.uuid            v1.2.0
    github.com/segmentio/fasthash        v1.0.1
    github.com/tidwall/gjson             v1.5.0
    go.uber.org/zap                      v1.14.0
    golang.org/x/net                     v0.0.0-20200222125558-5a598a2470a0
)
```

아래는 복붙을 위해 따로 기재해놓았다.

```
// Requirements

github.com/Shopify/sarama            v1.23.0
github.com/bitly/go-simplejson       v0.5.0
github.com/eclipse/paho.mqtt.golang v1.2.0
github.com/gin-gonic/gin             v1.4.0
github.com/labstack/gommon           v0.3.0
github.com/patrickmn/go-cache        v2.1.0+incompatible
github.com/satori/go.uuid            v1.2.0
github.com/segmentio/fasthash        v1.0.1
github.com/tidwall/gjson             v1.5.0
```

```
go.uber.org/zap                v1.14.0
golang.org/x/net                v0.0.0-20200222125558-5a598a2470a0
```

물론 $ go get 명령어로도 다운로드 받을 수 있으나, 어차피 go mod 차원에서 알아서 다운로드 받아주니 걱정할 필요는 없다.



위 링크에서 download in zip을 한 후, 터미널에서

```
$ go build main.go
$ ./main --host localhost -p 1883
```

위 명령어를 통해 실행시킬 수 있다.

실행시키면 아래와 같은 명령어가 뜨면 성공이다. (Ubuntu 18.04.4 LTS 기준)

잘 보이지 않을 것 같아 아래에 텍스트로 복사해서 붙여넣어보겠다.

```
bearpaek@bearpaek-System-Product-Name:~/data/git/dev/goMQTTServer$ go build
main.go
bearpaek@bearpaek-System-Product-Name:~/data/git/dev/goMQTTServer$ ./main --host
192.168.0.18 -p 1883
New Broker starting... Please wait...
{"level":"info","timestamp":"2020-02-
26T13:46:25.942+0900","logger":"broker","caller":"broker/broker.go:198","msg":"Sta
rt Listening client on ","hp":"192.168.0.18:1883"}
```

아래 코드는 Arduino IDE를 통해 MQTT 메시지를 발행하는 코드이다.
지속적으로 "Hello, MQTT World! I'm Bear :)" 를 발행시키며, Arduino IDE에서 ESP32 보드 패키지가 필요하고, PubSubClient 라이브러리를 다운로드 받아야한다.
소스 코드는 아래와 같다.

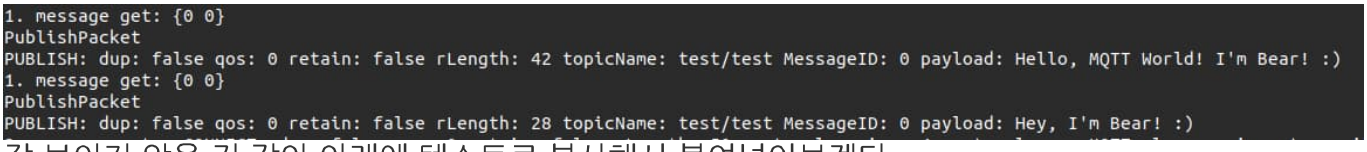
```
// MQTT Test

#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <PubSubClient.h>

#define WFSSID      "Your SSID"
#define PASSWORD    "Your PASSWORD"
#define mqtt_port   1883
#define mqtt_topic  "test/test"

// Put your IP Address in *
IPAddress mqtt_server(*, *, *, *);

const char* mqtt_message = "Hey, I'm from tistory!";
const char* j = 0;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int val = 0;

void setup() {
    Serial.begin(115200);
    //Serial.setDebugOutput(true);
    Serial.println();

    for(uint8_t t = 3; t > 0; t--) {
        Serial.printf("[SETUP] BOOT WAIT %d...\n", t);
        Serial.flush();
        delay(1000);
    }

    WiFi.mode(WIFI_STA);
    WiFi.begin(WFSSID, PASSWORD);
    Serial.print("Connecting to WiFi");
    while(WiFi.status() != WL_CONNECTED) {
        delay(300);
        Serial.print(".");
    }
```

```
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());

    client.setServer(mqtt_server, mqtt_port);

    client.setCallback(callback);
}

void loop() {
    client.loop();

    if(!client.connected()) {
        reconnect();
    }
    if(WiFi.status() != WL_CONNECTED) {
        WiFi.begin(WFSSID, PASSWORD);
        Serial.print("Reconnecting WiFi");
        while(WiFi.status() != WL_CONNECTED) {
            Serial.print(".");
        }
        Serial.println();
        Serial.println("WiFi Reconnected");
    }

    for(int i = 0; i < 1000; i++) {
        client.publish(mqtt_topic, "Hey, I'm from tistory!");
        client.publish(mqtt_topic, j);
        Serial.println("Sent message via MQTT! :)");
        j++;
        delay(3000);
    }
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");


    String msgText = "=> ";
```

```
    for (int i = 0; i < length; i++) {
      Serial.print((char)payload[i]);
      msgText += (char)payload[i];
    }
    Serial.println();
}


void reconnect() {
    while(!client.connected()) {
        Serial.println("Attempting MQTT connection...");
        String clientId = "ESP32Client-";
        clientId += String(random(0xffff), HEX);
        if(client.connect(clientId.c_str(),"ESP32client", "Connect")) {
            Serial.println("connected");
            client.publish(mqtt_topic, "Hello, MQTT World! I'm Bear! :)");
            delay(2000);
            client.publish(mqtt_topic, "Hey, I'm Bear! :)");
            client.subscribe(mqtt_topic);
            Serial.println("DONE connecting!");
        } else {
            Serial.print("failed, rc = ");
            Serial.print(client.state());
            Serial.println(" automatically trying again in 1 second");
            delay(1000);
        }
    }
}
```

주의할 점은, mqtt_server에 적는 IP Address는 127.0.0.1이나 localhost가 아닌, 실제 PC의 IP Address여야 한다.
(Windows에서는 ipconfig, MacOS/Linux에서는 ifconfig로 알 수 있다.)

만약 아래와 같은 메시지를 브로커가 뿜어내면 옳게 실행된 것이다.



잘 보이지 않을 것 같아 아래에 텍스트로 복사해서 붙여넣어보겠다.

```
// When ESP32 is connected with MQTT Broker
2. message get:  CONNECT: dup: false qos: 0 retain: false rLength: 50
protocolversion: 4 protocolname: MQTT cleansession: true willflag: false WillQos:
0 WillRetain: false Usernameflag: true Passwordflag: true keepalive: 15 clientId:
ESP32Client-6ae3 willtopic:  willmessage:  Username: ESP32client Password: Connect
{"level":"info","timestamp":"2020-02-
26T13:46:36.747+0900","logger":"broker","caller":"broker/broker.go:279","msg":"rea
d connect from ","clientID":"ESP32Client-6ae3"}
{"level":"error","timestamp":"2020-02-
26T13:46:36.747+0900","logger":"broker","caller":"broker/broker.go:617","msg":"sea
rch sub client error,  ","error":"Cannot publish to $ topics"}


// When ESP32 publishes MQTT message
1. message get: {0 0}
PublishPacket
PUBLISH: dup: false qos: 0 retain: false rLength: 42 topicName: test/test
MessageID: 0 payload: Hello, MQTT World! I'm Bear! :)
1. message get: {0 0}
PublishPacket
PUBLISH: dup: false qos: 0 retain: false rLength: 28 topicName: test/test
MessageID: 0 payload: Hey, I'm Bear! :)
```

즉, IoT 기기가 연결이 되면 2번 메시지를, MQTT message가 publish 되었다면 1번 메시지를 보여준다.

1번 메시지를 통해 MQTT 메시지의 Type은 무엇인지(위의 경우 PublishPacket), QoS는 몇인지(위의 경우 0), 발행자가 어느 토픽에 글을 발행했는지(위의 경우 test/test), 메시지 본문은 무엇인지(위의 경우 Hello, MQTT World! I'm Bear! :) 와 Hey, I'm Bear! :))에 대해 알 수 있다.

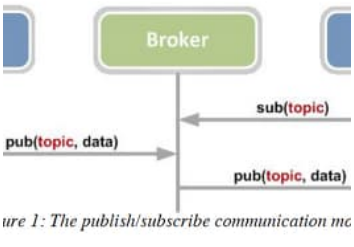
'IT > Go 언어 (Golang)' 카테고리의 다른 글

[Golang] 자체 MQTT 브로커 서버 구현 - 2 (0) 2020.02.26
[Golang] 고루틴과 채널(chan)을 이용한 피보나치 수열 (0) 2020.01.12
[Golang] Go Routine (고 루틴) - Golang의 꽃 [기본] (0) 2020.01.12

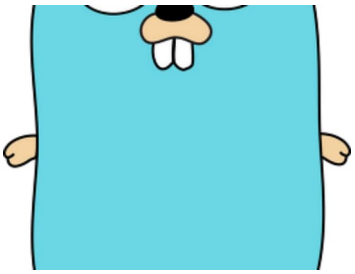
[Golang] Go Routine (고 루틴) - Golang의 꽃 [심화] (0) 2020.01.12

[Golang] 문자열과 문자열의 종류 (0) 2019.12.22

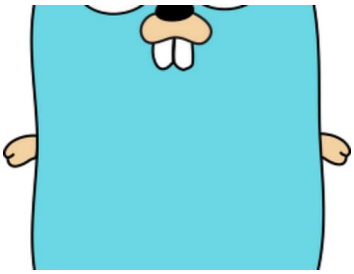
'IT/Go 언어 (Golang)' Related Articles



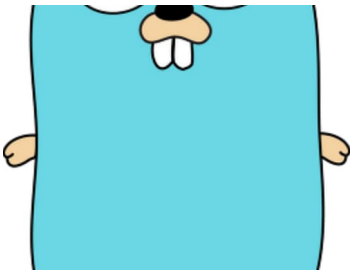
[Golang] 자체 MQTT 브로커 서버 구현 - 2



[Golang] 고루틴과 채널(chan)을 이용한 피보…



[Golang] Go Routine (고 루틴) - Golang의 …



[Golang] Go Routine (고 루틴) - Golang의 …

공대생의 차고

공대생의 차고 님의 블로그입니다.

구독하기

댓글 2

익명

비밀댓글입니다.

2021. 2. 2. 15:00

공대생의 차고

안녕하세요, wjddk4182님,
이메일 보내드렸습니다만, 혹시 못 보셨을 수도 있을 것 같아 댓글로도 남겨드립니다.

이메일 확인해주세요 :)

2021. 2. 10. 16:04 **답글**

이름

비밀번호

내용을 입력하세요.

등록