Trabajo Practico #11

Objetivos:

1. Familiarizarse con el protocolo MQTT y cómo se utiliza para interactuar con Beebotte.
2. Obtener una comprensión práctica de la autenticación en servicios de IoT como Beebotte.
3. Aplicar conceptos de programación para comunicarse con sensores y servicios en la nube.
4. Desarrollar habilidades de resolución de problemas y depuración en el contexto de sistemas de IoT.

Desarrollo:

Se proporciona el siguiente código, que tiene como objetivo conectar un ESP32 a Beebotte usando MQTT para publicar lecturas de un sensor de humedad y temperatura DHT11. Aunque el código se compila y se ejecuta correctamente, y se conecta a la red WiFi y al broker MQTT sin problemas, los datos de temperatura y humedad no llegan a Beebotte. Debe resolver este ejercicio especificando cual es el motivo de la falla, y proponer una posible solución.

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define WIFI_SSID "your_wifi_ssid"
#define WIFI_PASS "your_wifi_password"
#define BEEBOTTE_HOST "mqtt.beebotte.com"
#define BEEBOTTE_PORT 1883
#define CHANNEL_NAME "your_channel_name"
#define CHANNEL_WRITE_KEY "your_token_channel" // Reemplaza con tu token Beebotte
#define TEMP_TOPIC CHANNEL_NAME "/temperature"
#define HUMIDITY_TOPIC CHANNEL_NAME "/humidity"
#define DHTPIN 2
```

```
17 #define DHTTYPE DHT11
18
19 // Prototipos de funciones
20 void onMessage(char *topic, byte *payload, unsigned int length);
21 boolean reconnect();
22
23 // Variables globales
24 WiFiClient wifiClient;
25 PubSubClient client(wifiClient);
26 DHT_Unified dht(DHTPIN, DHTTYPE);
27 uint32_t delayMS;
28
29 void setup()
30 {
31   // ... (resto del código) ...
32
33   client.setServer(BEEBOTTE_HOST, BEEBOTTE_PORT);
34   client.setCallback(onMessage);
35 }
36
37 void onMessage(char *topic, byte *payload, unsigned int length)
38 {
39   // Aquí es donde procesarías los mensajes entrantes si los hubiera.
40   // Por ahora, este código no hace nada con ellos.
41 }
42
43 void loop()
44 {
45   // ... (resto del código) ...
46
47   char tempStr[10];
48   dtostrf(event.temperature, 4, 2, tempStr);
49   if (client.publish(TEMP_TOPIC, tempStr))
50   {
51     Serial.println("Publicación de temperatura exitosa");
52   }
53   else
54   {
55     Serial.println("Publicación de temperatura fallida");
56   }
57
58   // ... (resto del código) ...
59
60   char humidityStr[10];
61   dtostrf(event.relative_humidity, 4, 2, humidityStr);
62   if (client.publish(HUMIDITY_TOPIC, humidityStr))
63   {
64     Serial.println("Publicación de humedad exitosa");
65   }
66   else
67   {
```

```
68      Serial.println("Publicación de humedad fallida");
69    }
70
71    delay(delayMS);
72  }
73
74  boolean reconnect()
75  {
76    char authString[50];
77    sprintf(authString, "token:%s", CHANNEL_WRITE_KEY);
78    if (client.connect("ESP32Client", authString, ""))
79    {
80      client.subscribe(TEMP_TOPIC);
81      client.subscribe(HUMIDITY_TOPIC);
82      return true;
83    }
84    else
85    {
86      return false;
87    }
88  }
```

Ejercicio

Identifique el problema con este código y proponga una posible solución. ¿Cómo se están transmitiendo actualmente los datos de temperatura y humedad? ¿Cómo espera Beebotte que se formateen estos datos? Basado en su entendimiento de cómo funciona Beebotte, describa un enfoque para resolver este problema.