ISBN 978–987–633–574–4

Sistema domótico de control de iluminación y procesamiento de datos mediante mqtt centralizado en la nube

Carlos Binker[1], Hugo Tantignone[1], Guillermo Buranits[1], Eliseo Zurdo[1], Diego Romero[1], Maximiliano Frattini[1], Lautaro Lasorsa[1]

[1]Universidad Nacional de La Matanza, Florencio Varela 1903 (B1754JEC) -- San Justo, Buenos Aires, Argentina
{cbinker, htantignone, gburanits, djromero}@unlam.edu.ar;
{ezurdo, mfrattini, llasorsa}@alumno.unlam.edu.ar

Resumen. Este trabajo aborda la implementación de un sistema de control de dispositivos a nivel hogareño, empleando componentes IOT. Dicha implementación le brindará al usuario un tablero de control (dashboard) a través del cual se logrará la interacción para operar sobre sus dispositivos, ya sea emitiéndoles órdenes a los mismos, o bien recibiendo datos provenientes de dichos elementos. Este procedimiento de envío y recepción de mensajes hacia o desde los dispositivos IOT, se llevará a cabo a través de un protocolo de manejo de cola de mensajes diseñado para funciones de telemetría como lo es el MQTT, basado en tópicos que permite publicación y suscripción. Por otro lado el dispositivo IOT utilizado (placa ESP-01 Relay y el ESP01s) incorpora como valor agregado el hecho que se puede conectar su relé de salida a una llave de combinación, permitiendo así un control independiente de la iluminación, es decir a través de la aplicación web (dashboard) o bien desde la tecla de combinación tradicional y además la carga a controlar no tiene por qué ser un dispositivo inteligente, tal como exigen otros sistemas comerciales.

Palabras claves: IOT, MQTT, ESP01s, web socket, dashboard

1 Introducción

Se presentará un caso de estudio en donde se propone un sistema domótico de control de iluminación hogareño por medio de switches a través de una aplicación web (*dashboard.php*). Para ello se desarrolló una maqueta como prototipo que simula una instalación eléctrica básica hogareña (ver Figura 1). El módulo de control (la placa ESP-01 Relay en la cual va insertado el microcontrolador ESP01s) se inserta junto con la fuente de alimentación de 220 V AC a 5 V DC en una pequeña caja plástica (construida con una impresora 3D), que tiene la ventaja que puede ser montada dentro de un bastidor de 10 cm x 5 cm (ver Figura 2). El dashboard posee también un espacio en donde pueden mostrarse los datos provenientes desde los dispositivos IOT. En el caso del estudio planteado, estos datos vendrán como valores generados desde el propio ESP01s [1] (datos enviados por cada cliente, de acuerdo al estudio). Estos

datos podrían ser por ejemplo el envío de variables provenientes de sensores, tales como la tensión y la corriente alterna, y a partir de estos valores se podría determinar por ejemplo la potencia aparente, la potencia activa, el coseno fi, etc. El acceso al dashboard se da a través de una pantalla de autenticación del usuario *(login.php)*, el cual previamente debió registrarse (*register.php*). El dashboard además permite al cliente registrar y borrar los dispositivos pertenecientes al usuario logueado (*devices.php*). Estas interfaces web fueron construidas a partir de la instalación de un panel de desarrollo web (*Flatkit)* [2], el cual fue instalado en nuestro VPS [3] en la nube (ver Figura 3). La IP de nuestro VPS es una IP pública y el dominio asociado es *c2ing076.tk*, también se instaló en el VPS MySQL [4] para generar las siguientes tablas: *users* y *devices* (ver Figura 4). Los requerimientos del VPS son: 1 GB de RAM, 10 GB de SSD y un núcleo. El SO instalado es Ubuntu 18.04 LTS [5]. El broker mqtt [6] utilizado para el estudio fue EMQX [7], dado su excelente desempeño profesional y además posee una licencia de uso gratuito que permite el control de hasta cien mil dispositivos. El estudio se realizó para dos clientes con un único dispositivo ESP01s por cada cliente, el cual envía y recibe datos desde el broker mqtt bajo diferentes tópicos.

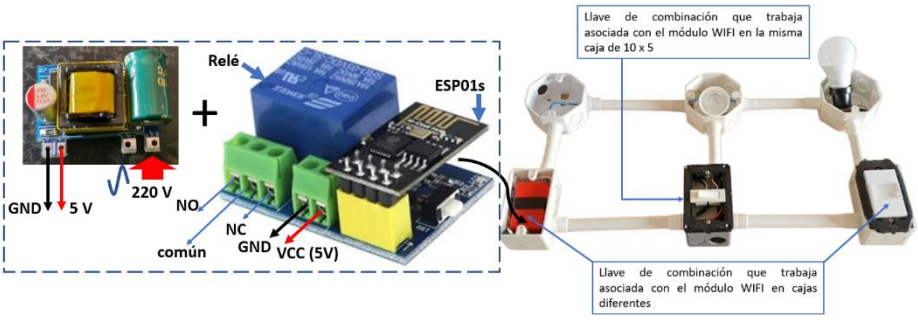

Figura 1. Placa ESP-01 Relay con ESP01s y fuente (izquierda) y maqueta prototipo (derecha)

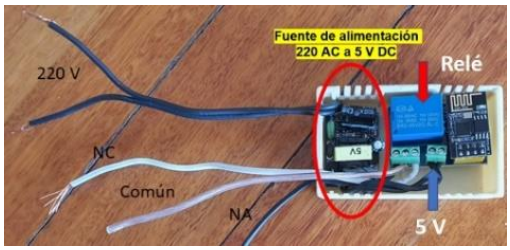

Figura 2. Caja contenedora del módulo Relay con el ESP01s más la fuente de alimentación

1.1 Hardware ESP01s

Se utilizó el microcontrolador ESP01 serie S que incorpora el chip ESP8266 de ESPRESSIF [8]. Para programar el dispositivo se utilizó Platformio [9] instalado en Vscode desde donde se confeccionó el proyecto (tanto la parte de PHP, JavaScript,

como la programación del 8266 misma). Se instaló SFTP [10] para la comunicación con el VPS. Esta placa de desarrollo con el ESP01s tiene conexión a WiFi, 2 puertos GPIO (I00 e I02). El WiFi es compatible con el protocolo 802.11 b/g/n y soporta autenticación WEP y WPA/WPA2 (Figura 5).

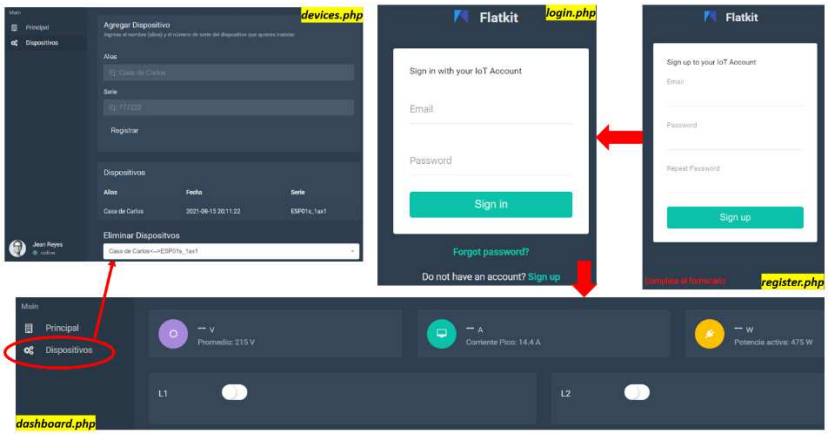

Figura 3. Vistas de los módulos register.php, login.php, dashboard.php y devices.php

Figura 4. Tablas users y devices de la base de datos admin_c2ing076 en MySQL

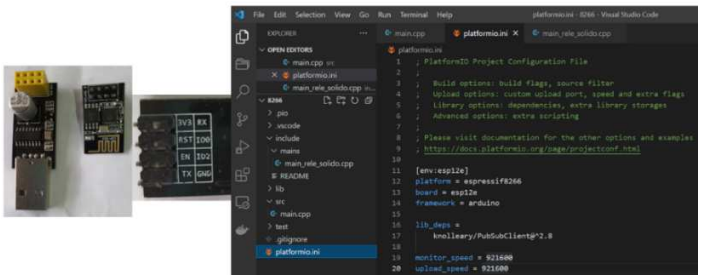

Figura 5. Programador ESP01s (izquierda), diagrama de pines (centro) y software Platformio

1.2 Placa ESP-01 Relay

El ESP01s fue montado sobre una placa que contiene un relé que permite controlar una carga para 220 V, en este caso una lámpara LED (podría ser cualquier dispositivo

hasta un máximo de 10 A). El relé está optoacoplado con la parte del microcontrolador, lo que permite una aislación galvánica entre los 220 V y la parte de control que funciona con sólo 3.3 V. El Relé se activa a través del pin IOO del ESP01s con un nivel bajo (LOW). El terminal común del relé está conectado normalmente al borne NC. Cuando se energiza la bobina se conecta al borne NA (Figura 6). Por otro lado recordemos que el relé está conectado en combinación con una llave inversora de tecla convencional, permitiendo así encender o apagar la luminaria tanto desde el dashboard como desde la llave física. Se hace énfasis en este punto, *ya que no es algo menor*, dado que la mayoría de estos módulos que pueden encontrarse de manera comercial son sólo un mero switch electrónico activado por wifi que sólo funciona como una llave de un punto y la carga a controlar es por lo general un dispositivo inteligente; en este caso de estudio podemos usar cualquier luminaria, que sólo cumpla con la máxima especificación de carga ya mencionada.

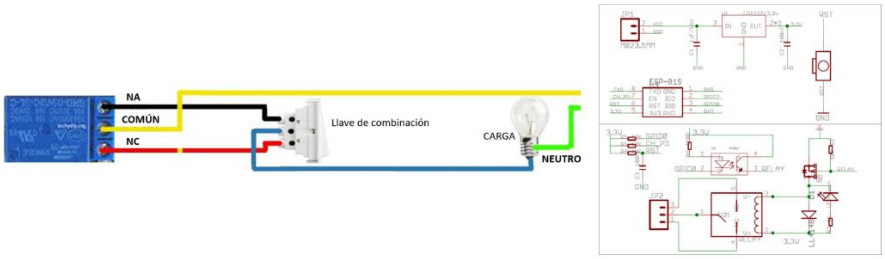

Figura 6. Diagrama circuital del módulo ESP-01 Relay y circuito de combinación con tecla

2 Descripción del broker MQTT EMQX

El broker mqtt, que residirá en el VPS, es el responsable de la administración de los mensajes provenientes tanto de los dispositivos IOT, como de las aplicaciones web. Estos mensajes se organizan bajo estructuras temáticas denominadas *tópicos* [11], [12], [13]. Cuando un dispositivo (por lo general a través de un elemento sensor) o una aplicación web envían información bajo determinado tópico, otros dispositivos u otras aplicaciones web podrán suscribirse a dichos tópicos. De esta manera, sólo aquellos dispositivos o aplicaciones web que se suscriban a determinados tópicos recibirán información de los mismos a través del broker mqtt. Por ende el mqtt es un protocolo denominado de *Publicación/Suscripción* y es el encargado de filtrar y administrar la distribución de mensajes entre dispositivos y aplicaciones (Figura 7). El broker elegido para esta investigación es el EMQX. Como se ha expresado en la introducción, la elección se debe al alto desempeño profesional que tiene frente a otros brokers (como por ejemplo Mosquitto, CloudMQTT, etc.) y por tener una licencia de uso gratuito capaz de soportar hasta 100K dispositivos (Figura 8). El EMQX se comunica con los dispositivos IOT por medio del puerto TCP 1883, en cambio con las aplicaciones web lo hace por medio de *web sockets*, empleando el puerto 8093 para *ws*, mientras que para *wss* (Websockets [14] sobre SSL/TLS) se emplea el puerto 8094. WebSocket es una tecnología que proporciona un canal de

ISBN 978–987–633–574–4

comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. En nuestro caso de estudio se emplearán los puertos 1883 (TCP) para comunicar el ESP01s con EMQX, mientras que la comunicación entre el *dashboard.php* y el EMQX se realizará a través del puerto 8094 por websocket seguro. También se ha instalado un certificado de seguridad SSL a través de Lets Encrypt [15] en nuestro VPS para el dominio utilizado en el estudio *c2ing076.tk*. Lógicamente que para que esto funcione se han habilitado las respectivas reglas en el firewall de nuestro VPS en cuanto a la habilitación de los puertos correspondientes. Hay otros puertos que también deberán configurarse para el correcto funcionamiento, como el 8883 para SSL y el 8090 para management. Por otro lado el puerto 18083 se emplea para conectarse al dashboard del EMQX (no confundir con nuestro panel *dashboard.php*). El dashboard del EMQX es un cliente web que permite administrar gráficamente el broker de una forma mucho más amigable y es provisto por EMQX y se accede por *http://c2ing076.tk:18083*. Los listeners pueden observarse en la Figura 9.

Figura 7. Ejemplo de tópicos y subtópicos. Operador mono nivel (+) y multinivel (#)

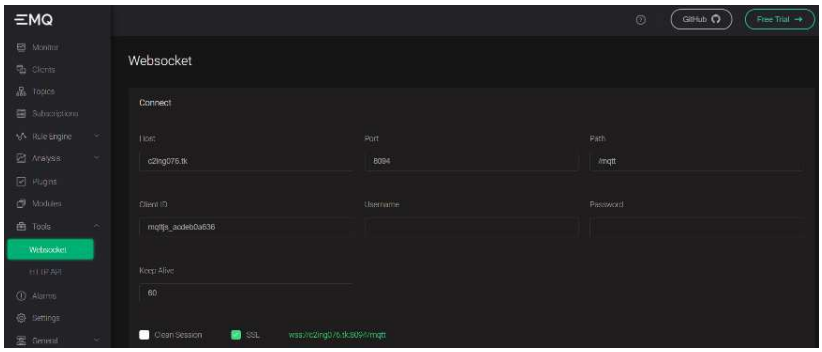

Figura 8. Dashboard (panel de control) correspondiente al EMQX. http://c2ing076.tk:18083

Tomando el ejemplo de tópicos de la Figura 7, el mismo puede interpretarse de la siguiente manera: existe un tópico principal denominado UNIVERSIDAD, la

universidad posee dos edificios (A y B), con tres pisos por edificio. A su vez en cada piso se han instalado sensores de humedad y de temperatura y en el Piso_PB se han agregado también módulos de medición de tensión eléctrica por su proximidad a los tableros de energía. Por ende, los edificios, como así también los pisos son subtópicos del tópico principal que como dijimos es UNIVERSIDAD. En el ejemplo, se está emitiendo un mensaje bajo el tópico UNIVERSIDAD/Edificio_B/Piso_2/temp enviando el valor de 25 °C. Por lo tanto el dispositivo que se suscriba a ese tópico recibirá todos los valores de temperatura sólo del Piso_2 del Edificio_B. También puede emplearse el símbolo monovalente **"+"** o el multivalente **"#"**. Como ejemplo del operador monovalente podemos indicar que si un dispositivo se suscribiera a UNIVERSIDAD/**+**/Piso_2/temp recibiría todos los mensajes de temperatura enviados de ambos edificios, pero sólo del Piso_2. En cambio como ejemplo de multivalente podríamos citar: UNIVERSIDAD/Edificio_A/**#**, en este caso se recibirían todas las variables sensadas de todos los pisos, pero solamente del Edificio_A.

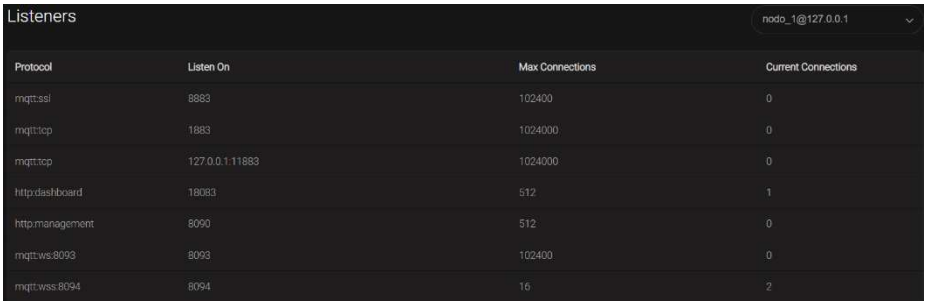

Figura 9. Configuración de los puertos de EMQX (listeners)

Otra característica muy importante de EMQX (en realidad de mqtt), es lo que se conoce como QOS (Quality of Service) [16], el cual puede tomar 3 valores posibles: 0, 1 ó 2. Un QOS igual a 0 significa que se confía en TCP para el envío de los mensajes, esto no significa que el mensaje va a llegar necesariamente al usuario, sí llegará al broker, pero éste lo enviará según la disponibilidad de la red, es decir si hubiera un fallo o el servidor se reiniciara por ejemplo, el mensaje no llegaría al cliente suscripto a un determinado tópico. De todas maneras es poco probable que suceda si la red es de buena calidad. En cambio si QOS es igual a 1, el broker intentará "*al menos 1 vez"* entregar el mensaje al usuario suscripto. Esto da la garantía que el mensaje será entregado al suscriptor del tópico, pero en el afán de cumplir con la meta de entrega, podría haber mensajes duplicados, es decir los suscriptores podrían recibir los mensajes en múltiples ocasiones. Esto también va a implicar mayor carga de tráfico y de procesamiento para el broker. Finalmente una QOS igual a 2 implica que sólo le llegará al suscriptor "*una sola vez"* el mensaje. Este último caso es importante en la situación en que el suscriptor no pueda recibir mensajes en múltiples ocasiones, pero implica un mayor nivel de procesamiento ya que el broker debe asegurarse que efectivamente el mensaje al suscriptor sólo se entregue en una oportunidad. Para finalizar debemos decir que el EMQX posee numerosos *plugins*,

que le añaden enorme funcionalidad. Por ejemplo puede autenticar clientes a través de una tabla de usuarios (mqtt_user) o bien ejecutar ACL (mqtt_ACL). Estas dos tablas vienen con una configuración por default y son provistas por EMQX.

3 Topología del sistema de control y análisis de tópicos

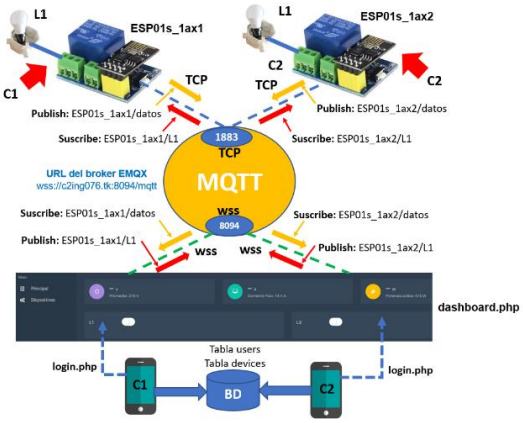

Figura 10. Topología de conexión para dos clientes (C1 y C2) con sus respectivos dispositivos

La topología de la figura 10 muestra un análisis para dos clientes con sus respectivos dispositivos ESP01s. Según puede observarse en el diagrama de la figura precedente, se utiliza la serie del dispositivo (almacenada en la tabla de la BD MySQL como *devices*) como parte principal del tópico. Cada dispositivo publica datos bajo el tópico **N°_de serie**/datos. Para el cliente 1 (C1) el número de serie es ESP01s_1ax1, mientras que para el cliente 2 (C2), la serie es ESP01s_1ax2. Esto permite fácilmente desde el código JavaScript del dashboard.php extraer mediante un método de Split (cuyo separador es el **"/"**) el tópico principal haciendo que los datos se transmitan únicamente al cliente logueado y a su dispositivo asociado.

3.1 Análisis de los procesos en el ESP01s y en la aplicación web (dashboard)

En el ESP01s se utilizan las siguientes funciones principales provistas a través de la librería *PubSubClient.h* [17]:

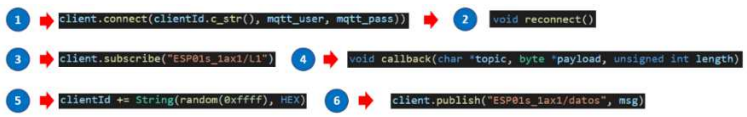

Figura 11. Funciones y métodos empleados en el ESP01s

Observando la figura 11, en **1** tenemos el método "*connect",* al cual debe pasársele el ClientId (ver **5** cómo se genera), el usuario y el password para conectarse al mqtt.

Esta función se incorpora dentro de **2** (función *reconnect*), y es en esta parte del código en donde se produce la suscripción del dispositivo a los diferentes tópicos. En este caso se hace con la función descripta en **3** bajo el tópico indicado "ESP01s_1ax1/L1" para C1 (cliente 1) y lo propio para C2 con el tópico "ESP01s_1ax2/L1". Finalmente en **4** se describe el método "*callback*", el cual recibirá los mensajes bajo los tópicos suscriptos y en función de esos mensajes procederá a excitar o no al relé que controla cada luminaria (ver figuras 12 y 13 en resultados obtenidos). La publicación por parte del ESP01s se realiza con un método dentro del loop (ver **6**). Un análisis similar se emplea desde el lado de JavaScript en los procesos y en la conexión a mqtt por parte del *dashboard.php*. Todas las funciones y métodos se basan en la librería *mqtt.js* [18].

4 Resultados obtenidos

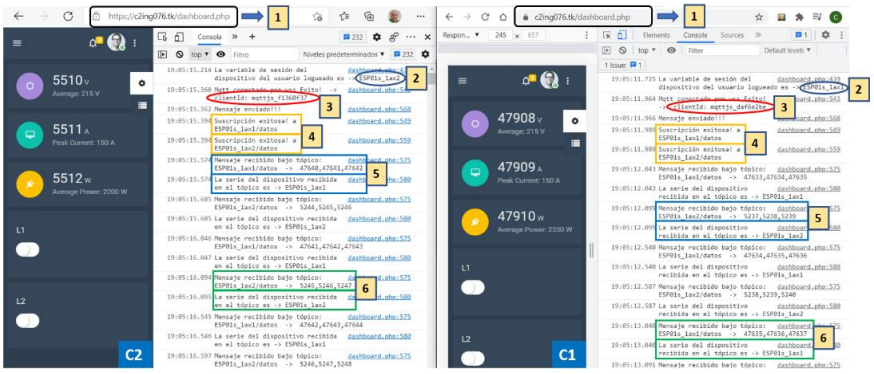

Figura 12. Captura de datos bajo tópicos ESP01s_1ax1 (C1) y ESP01s_1ax2 (C2)

En **1** observamos como ambos clientes (C1 y C2), efectivamente se loguean al mismo *dashboard.php*. En **2** observamos como es capturada la serie de cada dispositivo cuando el cliente se loguea, para ello se apela a las variables de sesión de PHP, dichas variables se obtienen desde la tabla *devices* de la base de datos tras corroborarse el logueo correcto del cliente en el archivo *login.php*. En **3** observamos como se genera un Id de cliente aleatorio en cada cliente logueado. Esto debe ser así porque si al cliente se le cae la conexión y se lograra de nuevo con el mismo ClientId, sería rechazado por el broker mqtt. Idéntica situación se da para el dispositivo ESP01s, en donde también apela a la generación de un ClientId aleatorio cada vez que se conecta al broker mqtt. En **4** se observa la suscripción que realiza el dashboard en todos los dispositivos de los clientes bajo el subtópico *datos*, en este caso para el estudio en cuestión se trata sólo de dos clientes, pero esto puede por supuesto extenderse a n clientes. En **5** se observa para cada uno de los clientes como al recepcionar datos en un tópico cuyo número de serie no se corresponde con el dispositivo del cliente, los mismos son ignorados. Finalmente en **6**, se observa como los valores recibidos bajo el tópico correcto impactan en el tablero html, indicando así los tres valores indicados.

ISBN 978–987–633–574–4

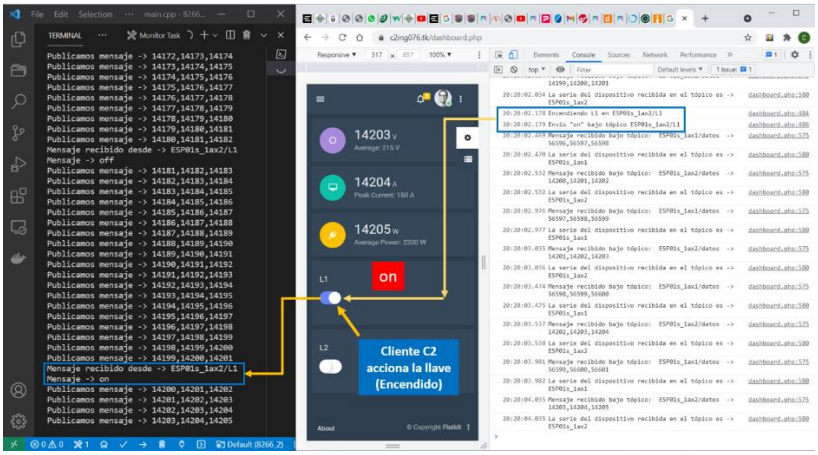

Figura 13. Encendiendo la luminaria bajo tópico ESP01s_1ax2/L1 (C2) bajo el mensaje ‘on’

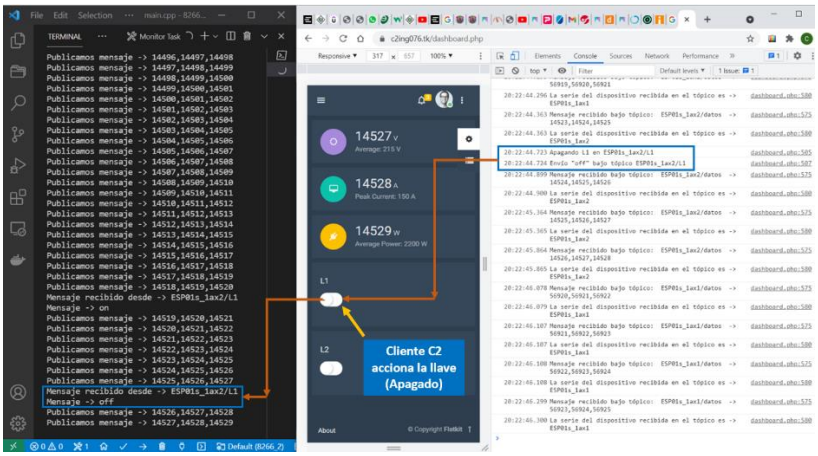

Figura 14. Apagando la luminaria bajo tópico ESP01s_1ax2/L1 (C2) ajo el mensaje ‘off’

5 Conclusiones y trabajo futuro

1. Este estudio contempla dos escenarios posibles en cuanto al tema de la instalación eléctrica se refiere. Por un lado un escenario que es el desarrollado en este estudio, en donde cada caja con su relé accionado por el microcontrolador puede ser ubicada en un bastidor de 10 cm x 5 cm, no alterando en absoluto la instalación eléctrica existente. Esto implica que cada usuario registre varios dispositivos.
2. El segundo escenario prevé la posibilidad de que con un único dispositivo por usuario (por ejemplo un ESP-32) que posee múltiples pines GPIO se exciten a varios relés, en donde cada luminaria a controlar se enmarcaría bajo un tópico. La complejidad acá radica en construir la placa, en alojar la misma en alguna caja (que podrá ir embutida o no, etc.); en síntesis esta es una buena alternativa pero quedaría

circunscripta a nuevas instalaciones eléctricas, o bien al deseo del usuario de modificar la instalación existente, cosa que no siempre es técnicamente factible.

Entre los trabajos futuros se puede mencionar:

- Incorporación de un motor de reglas implementado en Node.js, que esté a la espera de eventos preconfigurados, por ejemplo "*que se active un ventilador cuando la temperatura del recinto a controlar supere determinado umbral*".
- Configuración dinámica del dashboard por medio de los usuarios del sistema, de manera tal que se vayan incorporando los nuevos dispositivos de cada usuario al tablero de control.
- Suscripción dinámica de tópicos de todos los dispositivos de los usuarios (tarea a ejecutarse en el dashboard.php), esto se hace imprescindible conforme aumente la cantidad de usuarios del sistema.

6 Referencias

1. NodeMCU Connect Things EASY, http://www.nodemcu.com/index_en.html
2. https://themeforest.net/item/flatkit-app-ui-kit/13231484
3. https://www.ambit-bst.com/blog/definici%C3%B3n-de-iaas-paas-y-saas-en-qu%C3%A9-se-diferencian
4. https://es.wikipedia.org/wiki/MySQL. Fuente Wikipedia.
5. https://ubuntu.com/
6. Protocolo MQTT (Message Queue Telemetry Transport), http://mqtt.org.
7. EMQX. https://www.emqx.io/
8. Espressif System ESP8266 Series, https://www.espressif.com/. Ceja, J., Renteira, R., Ruelas, R., & Ochoa, G. (2017). Módulo ESP8266 y sus aplicaciones en el internet de las cosas. Revista de Ingeniería Eléctrica, 24-36.
9. https://platformio.org/install/integration
10. https://github.com/liximomo/vscode-sftp
11. Hands-On Internet of Things with MQTT: Build connected IoT devices with Arduino and MQ.
12. Implementación de middlewarepublicador/subscriptor para aplicaciones web de monitoreo. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).
13. Naik, N. (2017, October). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In 2017 IEEE international systems engineering symposium (ISSE) (pp. 1-7). IEEE.
14. Arquitectura de software con websocket para aplicaciones web multiplataforma. VI Workshop Innovación en Sistemas de Software (WISS). CACIC 2014.
15. https://letsencrypt.org/es/
16. https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels
17. https://github.com/knolleary/pubsubclient
18. https://github.com/mqttjs/MQTT.js