



ITESO, Universidad Jesuita de Guadalajara

Entrega 1 Proyecto
IoT para Linux Embebido

Rodrigo Ramos Romero IE715082
Daniel Jiram Rodriguez Padilla IE703331
12/09/2025

Zamora Davalos Ricardo

Proyecto: MQTT con reconocimiento de comandos por voz

Tabla de Contenidos

Aplicación	3
Definición de la entrega	3
Resultados	4

Definición de la entrega

Esta entrega tiene como meta ser la base del resto del proyecto. Estamos creando la interfaz que se usara por el resto de los módulos. Esta interfaz consiste en la parte de MQTT del proyecto, que permitirá la conexión e interacción de todos los dispositivos para la aplicación. Se puede definir en estas metas:

- Utilizar Mosquitto para la creación del Broker de MQTT a través del cual se comunicará la aplicación y todos los dispositivos.
- Probar este bróker con comandos desde otra CMD en el host de Windows
- Crear la aplicación dentro de “MQTT Panel” en el celular y probar la comunicación con este bróker de MQTT
- Lograr la conexión con el celular e interactuar con este utilizando comandos de Publish y Subscribe.

Aplicación

Para esta entrega, consta de 2 archivos y un Makefile para la construcción del proyecto. MQTT_Project.c y MQTT_Project.h, ejecutable y su librería.

MQTT_Project.h

Contiene los tópicos para subscribirnos, el QoS, Keep alive, puerto del Servidor (1883) y la IP del host. Básicamente, todo lo que necesita una aplicación de MQTT para poder ser editado rápidamente.

MQTT_Project.c

Contiene toda la funcionalidad MQTT del proyecto. Desde las conexiones con el Broker, hasta el código para subscribirnos y publicar. Se junto todo en un mismo código, ya que se espera que el proyecto final tenga varios ejecutables y librerías. Tratamos de encapsular totalmente MQTT en solo estos dos códigos.

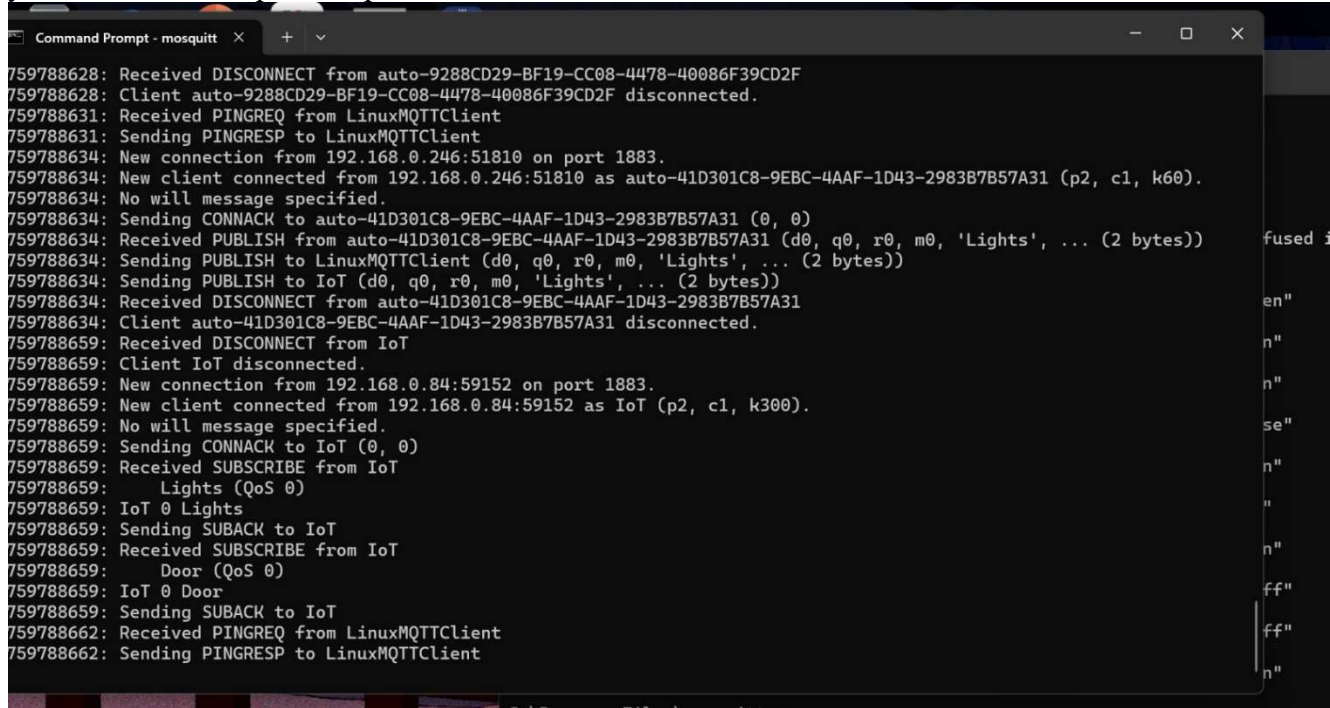
Algo muy importante para este código, es que se utilizó una implementación muy parecida a lo entregado en la Practica 2, se utilizaron apuntadores en vez de estructuras. Esto nos pareció necesario, ya que se localizo que el problema principal en los errores de conexiones era el uso de estructuras. MQTT espera tramas exactas de datos, pero estas estructuras pueden generar padding en su compilación, lo que hace que el Broker nunca pueda llegar a una conexión con la aplicación.

La aplicación funciona de esta manera:

1. Manda el paquete de conexión creado con apuntadores.
2. Espera el acknowledge, que el bróker haya permitido establecer una conexión.
3. Hace las subscripciones a los tópicos seleccionados.
4. Entra al loop de main, donde espera mensajes de MQTT en los tópicos subscritos, o un input del usuario para poder mandar un mensaje propio de Publish.

Resultados

Las pruebas se realizaron primero probando el bróker creado de MQTT con Mosquitto, usando pings, subscribes y publishes desde consola en esta primera foto vemos el servidor funcionando perfectamente y como se necesita para la aplicación.



```
759788628: Received DISCONNECT from auto-9288CD29-BF19-CC08-4478-40086F39CD2F
759788628: Client auto-9288CD29-BF19-CC08-4478-40086F39CD2F disconnected.
759788631: Received PINGREQ from LinuxMQTTClient
759788631: Sending PINGRESP to LinuxMQTTClient
759788634: New connection from 192.168.0.246:51810 on port 1883.
759788634: New client connected from 192.168.0.246:51810 as auto-41D301C8-9EBC-4AAF-1D43-2983B7B57A31 (p2, c1, k60).
759788634: No will message specified.
759788634: Sending CONNACK to auto-41D301C8-9EBC-4AAF-1D43-2983B7B57A31 (0, 0)
759788634: Received PUBLISH from auto-41D301C8-9EBC-4AAF-1D43-2983B7B57A31 (d0, q0, r0, m0, 'Lights', ... (2 bytes))
759788634: Sending PUBLISH to LinuxMQTTClient (d0, q0, r0, m0, 'Lights', ... (2 bytes))
759788634: Sending PUBLISH to IoT (d0, q0, r0, m0, 'Lights', ... (2 bytes))
759788634: Received DISCONNECT from auto-41D301C8-9EBC-4AAF-1D43-2983B7B57A31
759788634: Client auto-41D301C8-9EBC-4AAF-1D43-2983B7B57A31 disconnected.
759788659: Received DISCONNECT from IoT
759788659: Client IoT disconnected.
759788659: New connection from 192.168.0.84:59152 on port 1883.
759788659: New client connected from 192.168.0.84:59152 as IoT (p2, c1, k300).
759788659: No will message specified.
759788659: Sending CONNACK to IoT (0, 0)
759788659: Received SUBSCRIBE from IoT
759788659: Lights (QoS 0)
759788659: IoT 0 Lights
759788659: Sending SUBACK to IoT
759788659: Received SUBSCRIBE from IoT
759788659: Door (QoS 0)
759788659: IoT 0 Door
759788659: Sending SUBACK to IoT
759788662: Received PINGREQ from LinuxMQTTClient
759788662: Sending PINGRESP to LinuxMQTTClient
```

Figura 1: MQTT Bróker corriendo en Windows CMD

Teniendo un bróker de MQTT funcional, el siguiente paso es preparar la interfaz. La meta del proyecto es lograr una interfaz completa con el celular, imprimiendo información de estados en este, y permitiendo actualizar estados y salidas del sistema total desde el celular. Se utilizó la aplicación “MQTT Panel” y esta es la interfaz creada:

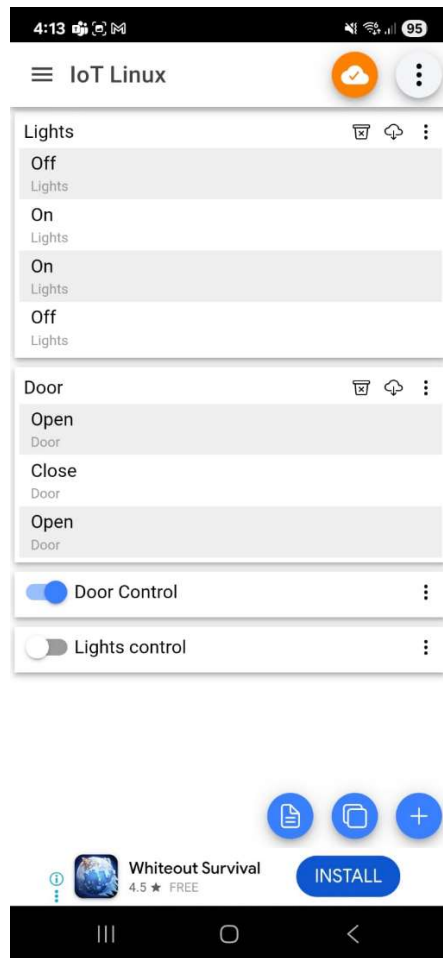
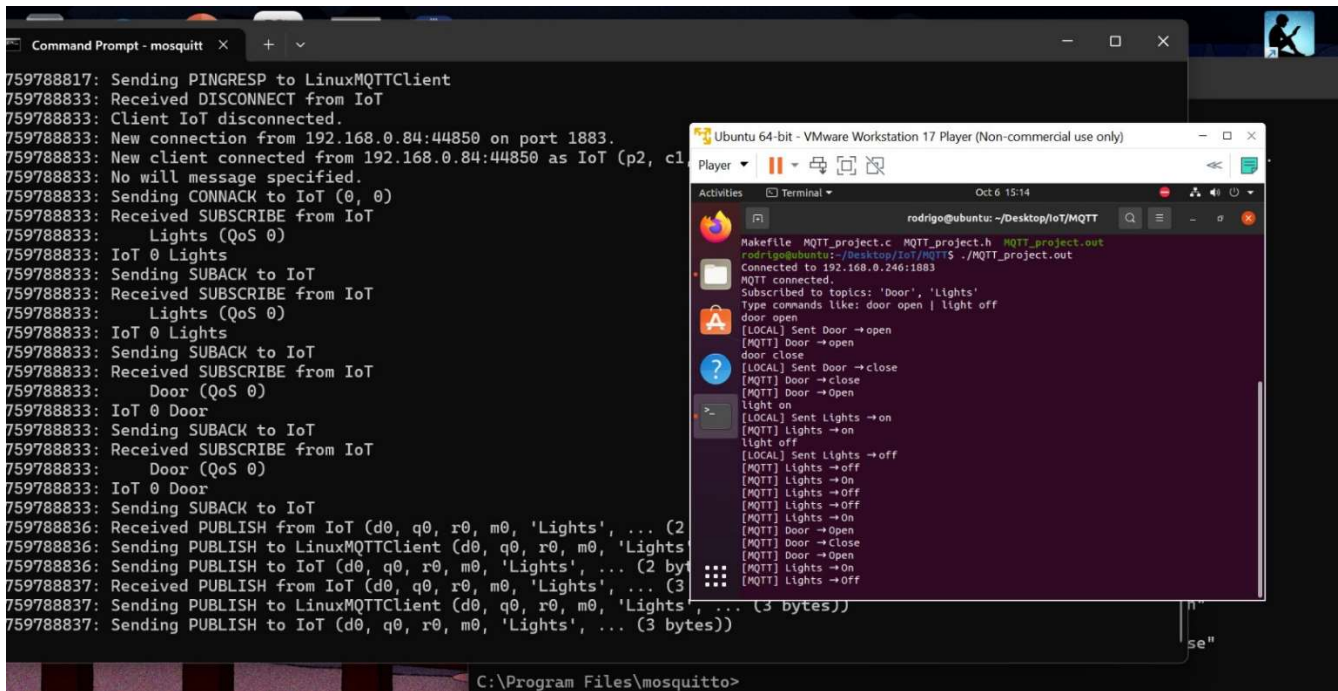


Figura 2: Aplicación de apoyo en el celular, MQTT Panel

Aquí podemos ver ya la aplicación funcionando. Imprimimos los últimos 3 estados o cambios de los tópicos “Door” y “Lights”. Los switches nos permiten actualizar manualmente el estado.



The image shows two terminal windows. The left window, titled 'Command Prompt - mosquitto', displays logs from the MQTT broker. It shows a client connecting from 192.168.0.84:44850, subscribing to 'Lights' and 'Door' topics, and receiving PUBLISH messages. The right window, titled 'Ubuntu 64-bit - VMware Workstation 17 Player', shows a terminal for 'rodrigo@ubuntu: ~/Desktop/IoT/MQTT'. It displays the execution of a Makefile, connection to the broker, subscription to topics, and a series of commands like 'door open', 'door close', 'light on', and 'light off' being sent to the broker.

```
759788817: Sending PINGRESP to LinuxMQTTClient
759788833: Received DISCONNECT from IoT
759788833: Client IoT disconnected.
759788833: New connection from 192.168.0.84:44850 on port 1883.
759788833: New client connected from 192.168.0.84:44850 as IoT (p2, cl
759788833: No will message specified.
759788833: Sending CONNACK to IoT (0, 0)
759788833: Received SUBSCRIBE from IoT
759788833: Lights (QoS 0)
759788833: IoT 0 Lights
759788833: Sending SUBACK to IoT
759788833: Received SUBSCRIBE from IoT
759788833: Lights (QoS 0)
759788833: IoT 0 Lights
759788833: Sending SUBACK to IoT
759788833: Received SUBSCRIBE from IoT
759788833: Door (QoS 0)
759788833: IoT 0 Door
759788833: Sending SUBACK to IoT
759788833: Received SUBSCRIBE from IoT
759788833: Door (QoS 0)
759788833: IoT 0 Door
759788833: Sending SUBACK to IoT
759788836: Received PUBLISH from IoT (d0, q0, r0, m0, 'Lights', ... (2
759788836: Sending PUBLISH to LinuxMQTTClient (d0, q0, r0, m0, 'Lights
759788836: Sending PUBLISH to IoT (d0, q0, r0, m0, 'Lights', ... (2 by
759788837: Received PUBLISH from IoT (d0, q0, r0, m0, 'Lights', ... (3
759788837: Sending PUBLISH to LinuxMQTTClient (d0, q0, r0, m0, 'Lights', ... (3 bytes))
759788837: Sending PUBLISH to IoT (d0, q0, r0, m0, 'Lights', ... (3 bytes))
```

```
Makefile MQTT_project.c MQTT_project.h MQTT_project.out
rodrigo@ubuntu:~/Desktop/IoT/MQTT$ ./MQTT_project.out
Connected to 192.168.0.246:1883
MQTT connected.
Subscribed to topics: 'Door', 'Lights'
Type commands like: door open | light off
door open
[LOCAL] Sent Door → open
[MQTT] Door → open
door close
[LOCAL] Sent Door → close
[MQTT] Door → close
[MQTT] Door → Open
light on
[LOCAL] Sent Lights → on
[MQTT] Lights → on
light off
[LOCAL] Sent Lights → off
[MQTT] Lights → off
[MQTT] Lights → On
[MQTT] Lights → Off
[MQTT] Lights → Off
[MQTT] Lights → On
[MQTT] Door → Open
[MQTT] Door → Close
[MQTT] Door → Open
[MQTT] Lights → On
[MQTT] Lights → Off
```

Figura 3: Aplicación funcionando en Consola

Finalmente, la aplicación funciona como fue esperado y definido en las metas. Se puede actualizar en la misma aplicación el estado de los dos tópicos o funcionalidades. Ahora se controla con comandos, pero luego se controlará por el micrófono usando “Speech to Text”. Podemos ver que también la aplicación responde frente a comandos recibidos por el celular, y que el bróker funciona bien y detecta todo lo que esta pasando en la interfaz.