



# ITESO, Universidad Jesuita de Guadalajara

Entrega 2 Proyecto  
IoT para Linux Embebido

Rodrigo Ramos Romero IE715082  
Daniel Jiram Rodriguez Padilla IE703331  
12/11/2025

Zamora Dávalos 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

La segunda entrega de proyecto añade la funcionalidad del Hardware que necesita el proyecto, micrófono y botones. Se trabaja sobre la implementación de MQTT de la entrega 1, en esta entrega se puede escuchar publicaciones a los 4 tópicos del proyecto, y grabar el comando de voz que será implementado en la entrega 3.

### Entrega 1

- 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.

### Entrega 2

- Agregar implementación del micrófono.
- Botón “Play” para grabar un comando de voz en formato .wav “Comando.wav”. graba por 2 segundos.
- Botón “Next” para escuchar el “Comando.wav” y verificar que se haya grabado lo que queremos.
- Todo esto encima de la aplicación MQTT de la entrega 1

## Aplicación

Se creció el proyecto a 5 archivos en total. Buttons .c y .h, MQTT .c y .h, main.c

### MQTT.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.c

Contiene toda la funcionalidad MQTT del proyecto. Desde las conexiones con el Broker, hasta el código para subscribirnos y publicar. Se junta 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.

### Buttons.c

Dependiendo del Ev.value detecto al presionar el botón hace una función en específico:

- Play: Grabar comando de voz por 2 segundos, guardar como “Comando.wav”.
- Next: Reproducir “Comando.wav”, únicamente para que el usuario verifique que se grabó lo que esperaba.

## Buttons.h

Únicamente contiene el prototipo de la función de botones para ser llamada por el main

## Main.c

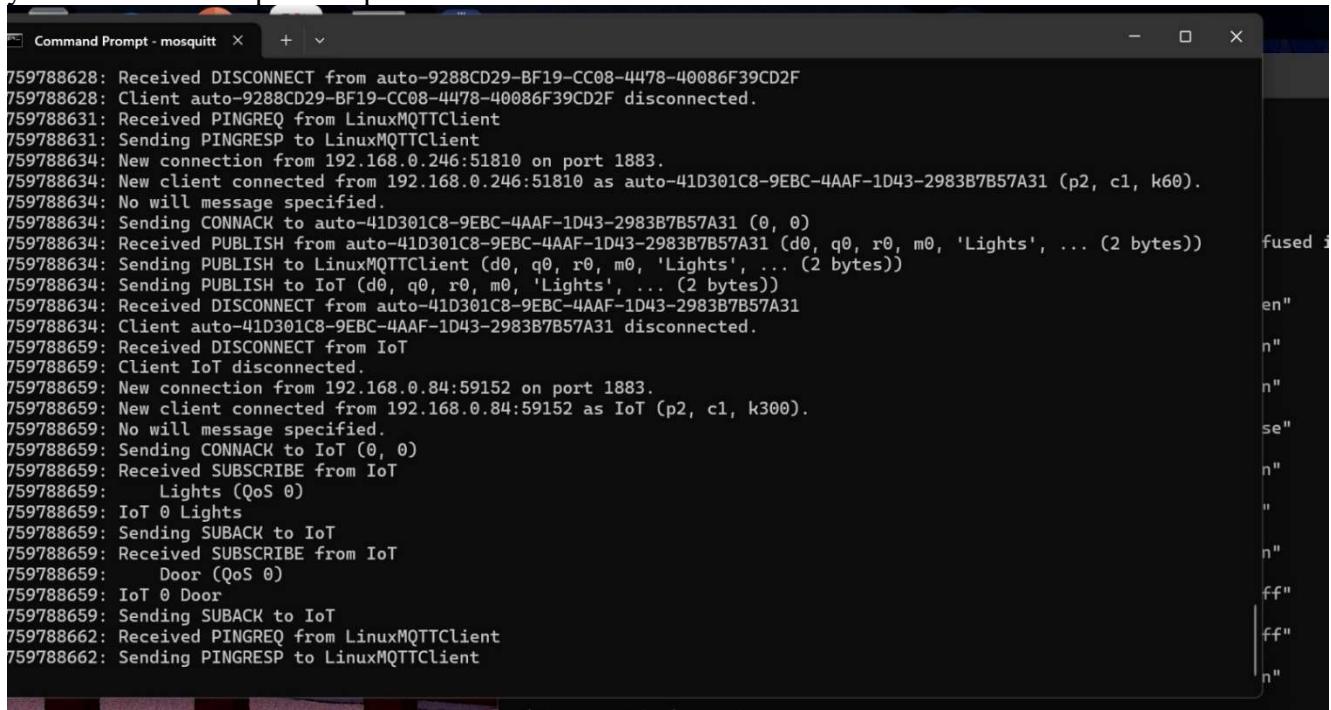
Llama las funciones de MQTT para inicializar la conexión al bróker, entra a un loop para seguir conectado al bróker mientras escucha los botones.

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 subscriptos, o un input del usuario para poder mandar un mensaje propio de Publish.
5. Escucha a los botones mientras está en el loop, graba o reproduce el comando.

## 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 ésta es la interfaz creada:

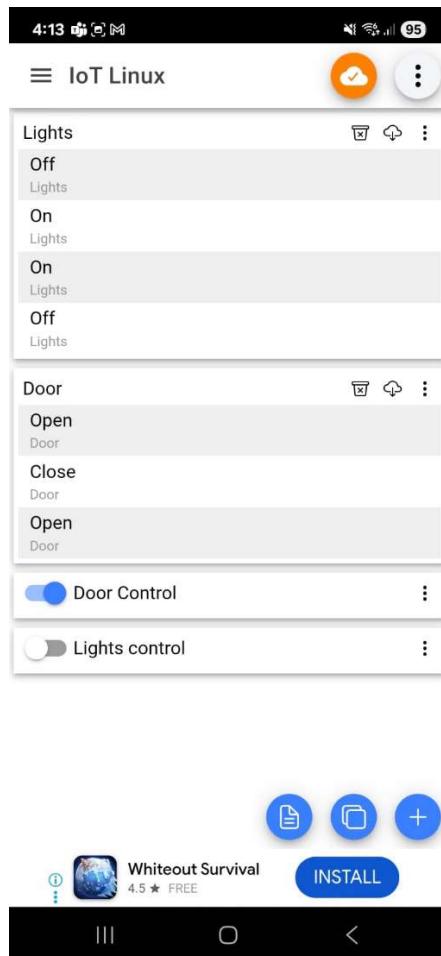


Figura 2: Aplicación de apoyo en el celular, MQTT Panel

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.

```

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 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 byt
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))

```

*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.

Debido a que los resultados de la entrega 2 son de voz, no se agregaron nuevas fotos como evidencia.