

# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## FACULTAD DE CIENCIAS

### S6 | Interrupciones y MQTT



**Máquinas Digitales**

**Hugo Miranda Cano**

## Interrupciones

- Video del monitor serie mostrando el conteo de veces que se ha presionado el botón, usando el programa "IntButton". Se tiene que ver que se está presionando el botón.

### Video en GiHub

- Responder: ¿Qué ventaja tienen las interrupciones contra el polling? ¿Por qué es más adecuado usarlas para leer un botón?

Una interrupción es una suspensión temporal de la ejecución de un proceso para pasar a ejecutar una subrutina de servicio de interrupción, la cual no es parte del programa, posteriormente se reanuda el programa.

Una de las ventajas que tiene las interrupciones del polling es que en las interrupciones no se requiere estar sondeando o revisando si algo va mal, cosa contraria con el polling. En el caso del programa realizado en la clase, al utilizar polling el contador se actualizaba incrementando considerablemente el número de conteos cuando se apretó el botón de las que realmente se presionó. podríamos decir que el tiempo de respuesta no era inmediato, sin en cambio, al utilizar una interrupción, se ejecutaba el pedazo de código haciendo que el botón no tuviera ese ruido y por ende las pulsaciones o actualizaciones eran mas fieles, después de ejecutarse el código el programa se reanudaba teniendo así la mejor actualización del contador.

- Explicar qué es el rebote del botón (bouncig) y explicar sus técnicas de mitigación: Por software y por hardware.

Para entrar en contexto: el bouncing es la señal de ruido producido por pasar de un estado a otro estado.

Podemos retirar o quitar este ruido de dos maneras primero debounce por software y segundo por hardware. Comenzaremos por el debounce por hardware

El aplicar un debounce por hardware no incrementa el tiempo de ejecución del programa, es una solución más factible, pero la desventaja es que se vuelve más complejo nuestro esp32. Para aplicar el debounce ya mencionado se coloca un condensador (1 microfaradio) en paralelo con el dispositivo.

Ahora la segunda manera de quitar ese ruido es el debounce por software tiene la ventaja de no requerir componentes adicionales. Se resuelve únicamente modificando el código del programa, una desventaja es que incrementa el tiempo de ejecución del programa y la complejidad del programa.

La manera mas sencilla de aplicar debounce por software es comprobar el tiempo entre disparos de la interrupción, si el tiempo es inferior a un determinado umbral de tiempo simplemente ignoramos la interrupción. Para aplicar el debounce por software, se modifica la función ISR.

- (Opcional) Responder: ¿Cuál es la duración máxima default de una función que se invoca por interrupción en ESP32? ¿Se puede usar un Serial.println() dentro de un ISR?
- Push del programa "IntButton"

## MQTT

- Captura con las pruebas de alcance del endpoint del servidor MQTT (Broker MQTT) de Amazon. Las pruebas se hacen desde la terminal del sistema operativo: 1) ping, 2) traceroute(Linux) o tracert(Windows) y 3) telnet. Explicar cuál es el propósito de cada una.

Telnet es un programa que permite la comunicación directa con la consola del otro lado. (Mensaje sin cifrar).

Ping sirve para ver si se puede conectar a determinado punto de enlace.

```
da00:ff00::3d0:7ac): [icmp_seq=1 ttl=48 time=91.3 ms
d
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2728]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\OwnerPC>ping a9wczfioqpq2-ats.iot.us-east-1.amazonaws.com
La solicitud de ping no pudo encontrar el host a9wczfioqpq2-ats.iot.us-east-1.amazonaws.com. Compruebe el nombre y vuelva a intentarlo.
C:\Users\OwnerPC>ping a9wczfioqpq2-ats.iot.us-east-1.amazonaws.com
Haciendo ping a a9wczfioqpq2-ats.iot.us-east-1.amazonaws.com [2406:da00:ff00::22c2:acb3] con 32 bytes de datos:
Respuesta desde 2406:da00:ff00::22c2:acb3: tiempo=89ms
Respuesta desde 2406:da00:ff00::22c2:acb3: tiempo=89ms
Respuesta desde 2406:da00:ff00::22c2:acb3: tiempo=90ms
Respuesta desde 2406:da00:ff00::22c2:acb3: tiempo=90ms
Estadísticas de ping para 2406:da00:ff00::22c2:acb3:
Paquetes: enviados = 4, recibidos = 4, perdidos = 0
(0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
Mínimo = 89ms, Máximo = 90ms, Media = 89ms
C:\Users\OwnerPC>
```

Tracert(Windows) nos indica que servidores y routers hay entre mi computadora y el servidor.(servidores intermedios) también nos indican el tiempo entre routers.

```
Administrador: Símbolo del sistema
programa o archivo por lotes ejecutable.
C:\WINDOWS\system32>tracert a9wczfioqpq2-ats.iot.us-east-1.amazonaws.com
Trazo a la dirección a9wczfioqpq2-ats.iot.us-east-1.amazonaws.com [2406:da00:ff00::3df:690e] sobre un máximo de 30 saltos:
 1  3 ms  7 ms  4 ms  2806-107e-0021-722b-de8d-b7ff-fe74-ca80.ipv6.infinitem.net.mx [2806:107e:21:722b:de8d:b7ff:fe74:ca80]
 2  *  *  *  Tiempo de espera agotado para esta solicitud.
 3  48 ms  50 ms  48 ms  2806-100f-0000-0004-0000-0000-002e.ipv6.infinitem.net.mx [2806:100f:0:4:2e]
 4  49 ms  48 ms  50 ms  2806-100f-0000-0004-0000-0000-002d.ipv6.infinitem.net.mx [2806:100f:0:4:2d]
 5  *  *  *  Tiempo de espera agotado para esta solicitud.
 6  47 ms  54 ms  773 ms  2620:107:4000:fff:28d
 7  106 ms  91 ms  89 ms  2620:107:4000:8001:66
 8  *  *  *  Tiempo de espera agotado para esta solicitud.
 9  94 ms  97 ms  90 ms  2620:107:4000:cfff:f202:d6db
10  *  *  *  Tiempo de espera agotado para esta solicitud.
11  240 ms  94 ms  91 ms  2620:107:4000:c5c0:f3fd:14
12  92 ms  92 ms  90 ms  2620:107:4000:a611:f000:2434
13  91 ms  94 ms  97 ms  2620:107:4000:cfff:f201:d691
14  105 ms  104 ms  95 ms  2406:da00:ff00::3df:690e
Trazo completa.
C:\WINDOWS\system32>
```

- Captura del objeto Axolote\_{tu\_nombre} en la consola de administración de IoT Core.

**Objetos (3)** Información

Un objeto de IoT es una representación y un registro del dispositivo físico en la nube. El dispositivo físico necesita un registro de objeto para poder trabajar con AWS IoT.

<input type="checkbox"/>	Nombre	Tipo de objeto
<input type="checkbox"/>	<a href="#">Axolote_Hugo</a>	<a href="#">Plataforma-Axolote-Embebed-Cloud</a>
<input type="checkbox"/>	<a href="#">Axolote_Aldo</a>	<a href="#">Plataforma-Axolote-Embebed-Cloud</a>
<input type="checkbox"/>	<a href="#">AtmosphericProbe_Albert_CDMX</a>	<a href="#">SensorAtmosferico</a>

**Axolote\_Hugo** Información

**Detalles del objeto**

Nombre	Axolote_Hugo	Tipo	<a href="#">Plataforma-Axolote-Embebed-Cloud</a>
ARN		Grupo de facturación	

Subimos certificados al esp32 sketch data upload

```

AWS_MQTT Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda

AWS_MQTT
//
Conectar ESP32 a AWS
https://www.todonomaker.com
Colaboración: Néstor Coencho
Todos los derechos reservados.
*/

//Añadir bibliotecas
#include "SPIFFS.h"
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <Adafruit_BMP085.h>

SPIFFS Image Uploaded
Connecting.....
Chip is ESP32-D0WD-V3 (revision 3)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 94:b5:55:f9:f9:90
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash will be erased from 0x00290000 to 0x003effff...
Compressed 1441792 bytes to 7281...
Writing at 0x00290000... (100 %)
Wrote 1441792 bytes (7281 compressed) at 0x00290000 in 6.7 seconds (effective 1723.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

```

- Captura de la terminal serie donde se muestre que el ESP envía lecturas al servidor MQTT.

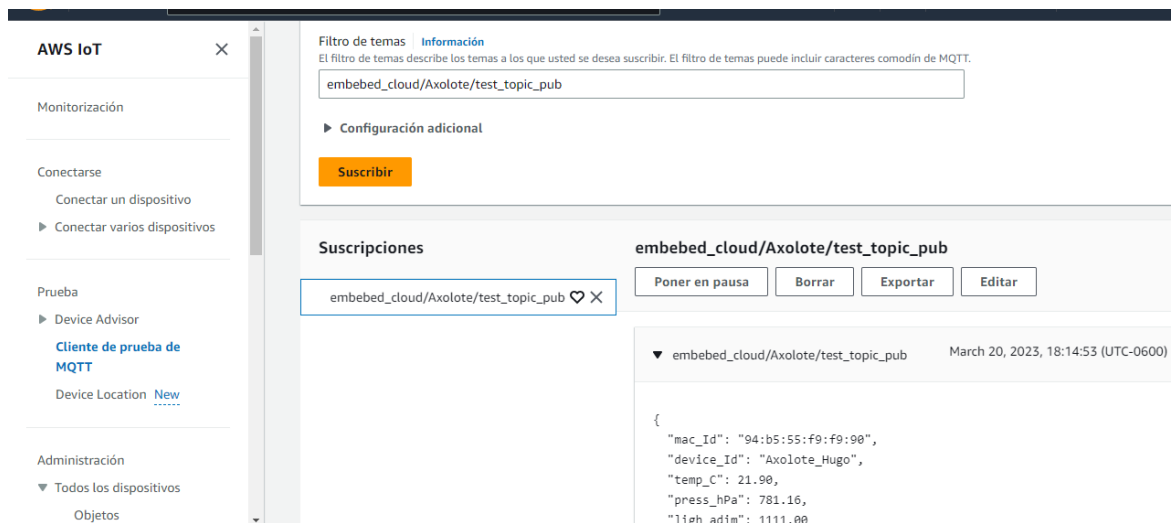
The screenshot shows the Arduino IDE interface with the 'ESP32 Dev Module' selected. The sketch 'AWS\_MQTT.ino' is open, displaying code for initializing an ESP32 to connect to AWS IoT Core. The code includes comments in Spanish and uses the `aws_iot_arduino` library. The `setup` function initializes the serial port at 115200 baud, the BMP sensor, and the LED pin. The `loop` function (partially visible) publishes sensor data to the MQTT topic `embebido_cloud/Axolote/test_topic_pub`.

The Serial Monitor at the bottom shows the output of the sketch, displaying a series of JSON messages published to the MQTT topic. Each message contains sensor data for a device named 'Axolote\_Hugo'.

```
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.13, "ligh_adim": 1081.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.18, "ligh_adim": 1087.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.16, "ligh_adim": 1035.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.14, "ligh_adim": 1106.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.13, "ligh_adim": 1136.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.08, "ligh_adim": 1178.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.18, "ligh_adim": 1103.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.19, "ligh_adim": 1083.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.23, "ligh_adim": 1127.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.23, "ligh_adim": 1056.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.13, "ligh_adim": 1065.00}
Publicando mensaje: {"mac_Id": "94:b5:55:f9:f9:90", "device_Id": "Axolote_Hugo", "temp_C": 21.90, "press_hPa": 781.19, "ligh_adim": 1088.00}
```

- Captura del cliente de prueba de IoT Core leyendo los mensajes que llegan al servidor MQTT.

Creando un cliente de prueba



- Video que muestre que se envía un mensaje desde el cliente de prueba hacia el tópic al que está suscrito el ESP. En la terminal serial debe verse que el ESP recibe el mensaje y lo despliega.

Captura de mensaje de cliente a esp32

AWS IoT

Monitorización

Conectarse

Conectar un dispositivo

Conectar varios dispositivos

Prueba

Device Advisor

Ciente de prueba de MQTT

Device Location [New](#)

Administración

Todos los dispositivos

Detalles de la conexión

Puede actualizar los detalles de la conexión. Para ello, elija Disconnect (Desconectar) y realice las actualizaciones en la página Establish connection to continue (Establecer conexión para continuar).

Conectado

Suscribirse a un tema

Publicar en un tema

Nombre del tema

El nombre del tema identifica el mensaje. La carga del mensaje se publicará en este tema con una calidad del servicio (QoS) de 0.

Q embebed\_cloud/Axolote/test\_topic\_sub

Carga del mensaje

{

"message": "Lo saludamos desde la consola de AWS IoT"

}

Configuración adicional

Publicar

AWS\_MQTT | Arduino IDE 2.0.3

File Edit Sketch Tools Help

ESP32 Dev Module

AWS\_MQTTino

debug\_custom.json

18 const char\* ssid = SSID;

19 const char\* password = PASSWORD;

20

21 //Servidor MQTT

22 const char\* mqtt\_server = AWS\_MQTT\_SERVER;

23 const int mqtt\_port = 8883;

24 String clientId = "Axolote";

25 const char\* PUBLISH\_TOPIC = "embebed\_cloud/Axolote/test\_topic\_pub";

26 const char\* SUBSCRIBE\_TOPIC = "embebed\_cloud/Axolote/test\_topic\_sub";

27

28 String Read\_rootca;

29 String Read\_cert;

30 String Read\_privatekey;

Output Serial Monitor

Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.00,"press\_hPa":781.43,"ligh\_adim":1105.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.36,"ligh\_adim":1125.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.39,"ligh\_adim":1195.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.31,"ligh\_adim":1230.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.34,"ligh\_adim":1130.00}

Lo saludamos desde la consola de AWS IoT

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.34,"ligh\_adim":1078.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.34,"ligh\_adim":1059.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.33,"ligh\_adim":1085.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.36,"ligh\_adim":1057.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.29,"ligh\_adim":1104.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.32,"ligh\_adim":1138.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.28,"ligh\_adim":1093.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.37,"ligh\_adim":1079.00}

Publicando mensaje: {"mac\_id":"94:b5:55:f9:f9:90","device\_id":"Axolote\_Hugo","temp\_C":21.80,"press\_hPa":781.39,"ligh\_adim":1067.00}

Segundo mensaje

The image shows two screenshots related to AWS IoT and Arduino.

The top screenshot is the AWS IoT console. The left sidebar shows the navigation menu with options like Monitorización, Conectarse, Prueba, and Administración. The main area displays 'Detalles de la conexión' with a 'Conectado' status. Below this, there are tabs for 'Suscribirse a un tema' and 'Publicar en un tema'. The 'Publicar en un tema' tab is active, showing a form to publish a message. The 'Nombre del tema' field contains 'embebed\_cloud/Axolote/test\_topic\_sub'. The 'Carga del mensaje' field contains a JSON object: 

```
{  "message": "Hola grupo de maquinas digitales 2023"}
```

. A 'Publicar' button is at the bottom.

The bottom screenshot is the Arduino IDE. The top bar shows 'AWS\_MQTT | Arduino IDE 2.0.3'. The 'Tools' menu is open, showing 'ESP32 Dev Module' selected. The code editor shows the following code:

```
18 const char* ssid = SSID;
19 const char* password = PASSWORD;
20
21 //Servidor MQTT
22 const char* mqtt_server = AWS_MQTT_SERVER;
23 const int mqtt_port = 8883;
24 String clientId = "Axolote_";
25 const char* PUBLISH_TOPIC = "embebed_cloud/Axolote/test_topic_pub";
26 const char* SUBSCRIBE_TOPIC = "embebed_cloud/Axolote/test_topic_sub";
27
28 String Read_rootca;
29 String Read_cert;
30 String Read_privatekey;
```

The 'Serial Monitor' window is open, showing a list of messages being published. The messages are in JSON format, containing fields like 'mac\_id', 'device\_id', 'temp\_C', 'press\_hPa', and 'ligh\_adim'. The last message is: 

```
{ "mac_id": "94:b5:55:f9:f9:90", "device_id": "Axolote Hugo", "temp_C": 21.70, "press_hPa": 781.38, "ligh_adim": 1174.00 }
```

- Push con el programa "aws\_mqtt".