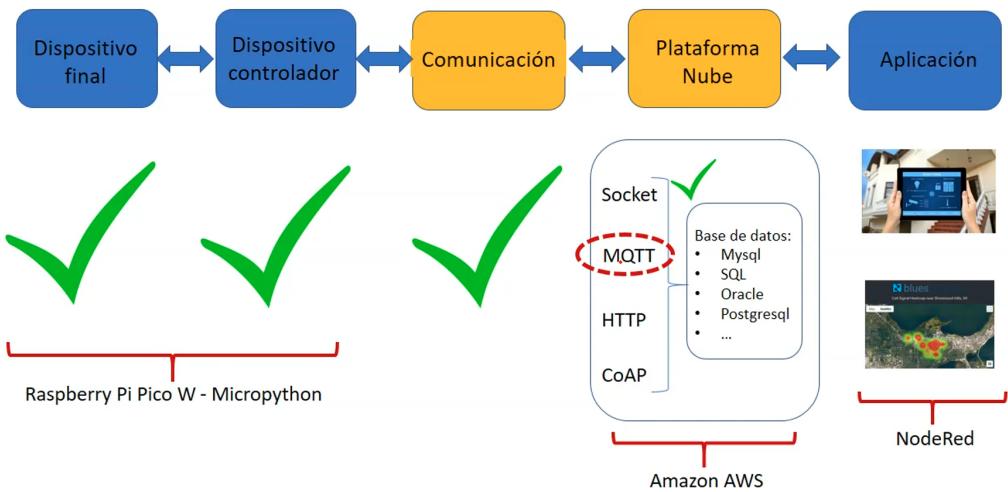


01_Manejo MQTT

March 7, 2025

0.1 Arquitectura IoT Protocolo MQTT

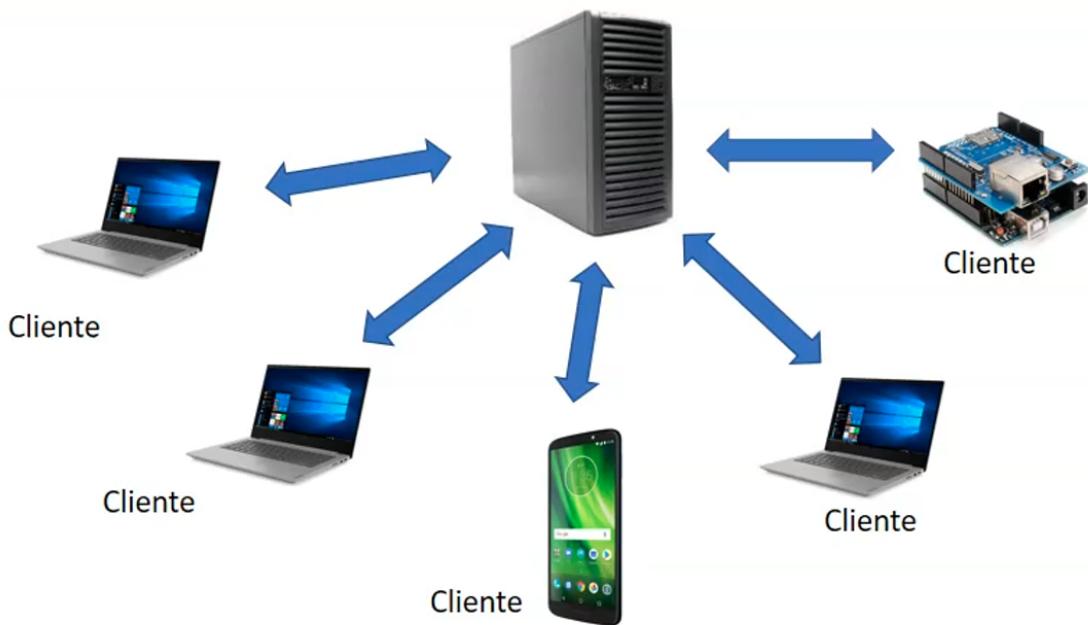
Arquitectura de IoT



0.1.1 Conceptos

- Funciona sobre TCP
- Usa el puerto **1883**
- Puede usar usuario y password para el mensaje
- Usa SSL/TLS para seguridad

Servidor MQTT

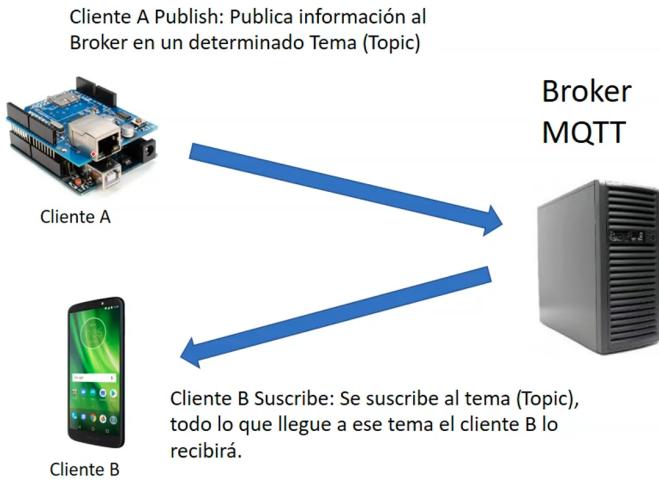


0.1.2 Elementos clave

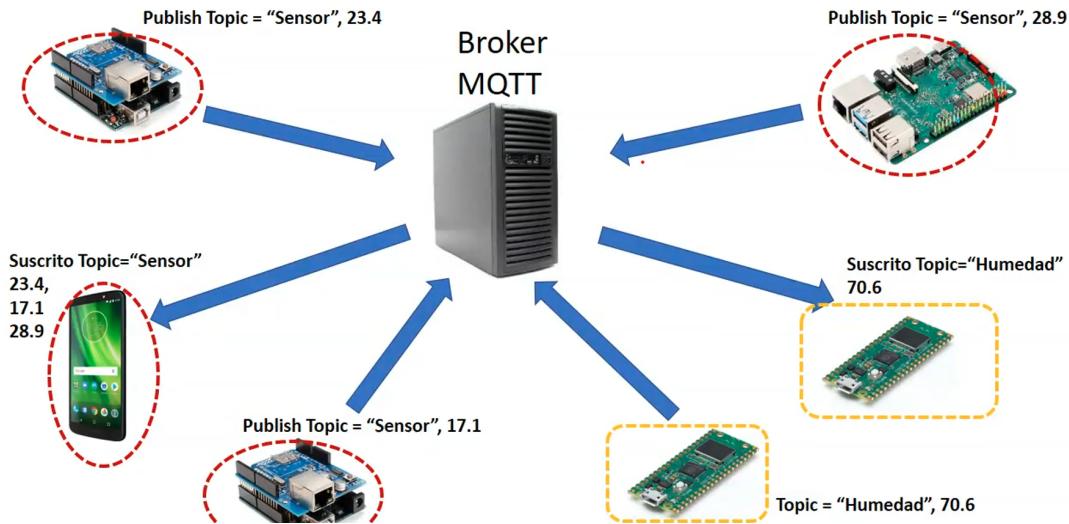
- Broker: Es el servidor MQTT
- Publish/Subscribe: Son las acciones del cliente (un cliente puede publicar en el servidor o puede suscribirse en el servidor)
- Topics: Los temas a los cuales se suscribirán y publicarán los clientes. Un cliente puede publicar en un determinado topic o puede suscribirse a un determinado topic, en términos prácticos, los topics son palabras o temas, por ejemplo temperatura o humedad



El broker por el momento lo vamos a instalar en un servidor o en la PC.



El cliente **A** publica en un topic, por ejemplo el topic **prueba**, de tal forma que el cliente **B** se subscribe al topic **prueba** y de esta manera, todo lo que el cliente **A** publique al topic **prueba** (cada vez) le llegará al cliente **B** y a todos los subscriptores del topic **prueba**. La condición es que tienan que estar en el mismo topic.



0.2 Ejemplo 01 Instalar el servidor MQTT en la PC y hacer pruebas

Para instalar mosquito buscar en google **mosquitto**

Download

Source

- mosquitto-2.0.20.tar.gz (GPG signature)
- Git source code repository (github.com)

Older downloads are available at <https://mosquitto.org/files/>

Binary Installation

The binary packages listed below are supported by the Mosquitto project. In many cases Mosquitto is also available directly from official Linux/BSD distributions.

Windows

- 64-bit: [mosquitto-2.0.20-install-windows-x64.exe](#)
- 32-bit: [mosquitto-2.0.20-install-windows-x86.exe](#)

Older installers can be found at <https://mosquitto.org/files/binary/>.

See also [README-windows.md](#) after installing.

Mac

Mosquitto can be installed from the homebrew project. See [brew.sh](#) and then use `brew install mosquitto`

Linux distributions with snap support

- `snap install mosquitto`

Debian

- Mosquitto is now in Debian proper. There will be a short delay between a new release and it appearing in Debian as part of the normal Debian procedures.
- There are also Debian repositories provided by the mosquitto project, as described at <https://mosquitto.org/2013/01/mosquitto-debian-repository>

Raspberry Pi

Mosquitto is available through the main repository.

There are also Debian repositories provided by the mosquitto project, as described at <https://mosquitto.org/2013/01/mosquitto-debian-repository/>

Ubuntu

Mosquitto is available in the Ubuntu repositories so you can install as with any other package. If you are on an earlier version of Ubuntu or want a more recent version of mosquitto, add the [mosquitto-dev PPA](#) to your repositories list - see the link for details. mosquitto can then be installed from your package manager.

Aplica para windows

Observación 1

Ir al archivo `mosquitto.conf`

Editar el archivo y adicionar al final
listener 1883 0.0.0.0
allow_anonymous true

Reiniciar

Observación 2

Acciones en el CMD Windows

```
cd ..  
cd ..  
cd "Program Files"  
cd mosquitto  
mosquitto -c mosquitto.conf  
mosquitto_sub -h ipBroker -t topic  
mosquitto_pub -h ipBroker -m "mensaje" -t topic
```

0.2.1 Instalación de Mosquitto en Debian 12

```
sudo apt update && sudo apt upgrade -y
sudo apt install mosquitto mosquitto-clients -y
sudo nano /etc/mosquitto/mosquitto.conf
mosquitto.conf ***
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

### AGREGAR ESTAS LINEAS AL FINAL DEL ARCHIVO ###
listener 1883 0.0.0.0
allow_anonymous true
#password_file /etc/mosquitto/passwd
```

```
sudo systemctl restart mosquitto
sudo systemctl status mosquitto
sudo systemctl enable mosquitto
```

La configuración anterior usa mosquito sin que se establezca un password, para información sobre la instalación con password, vea la siguiente liga al documento pdf.

Guia de Instalación Mosquitto en Debian 12

0.2.2 Probar mosquitto

En una terminal usando la dirección ip que tenga su computadora, ejecutar
mosquitto_sub -h 192.168.2.186 -t "test/topic"

La terminal se queda bloqueada para recibir los mensajes.

```
21:25 $ mosquitto_sub -h 192.168.2.168 -t "test/topic"
```

En otra terminal ejecutar el siguiente comando para publicar en el topico, recuerde usar la direccion ip que tiene su computadora.

```
mosquitto_pub -h 192.168.2.186 -t "test/topic" -m "Hola Mundo"
```

```
21:31 $ mosquitto_pub -h 192.168.2.186 -t "test/topic" -m "Hola Mundo"
21:32 $
```

Ahora si regresamos a la terminal en donde fue realizada la subscripcion se podrá visualizar el mensaje.

```
21:25 $ mosquitto_sub -h 192.168.2.186 -t "test/topic"
Hola Mundo
```

Se puede detener el comando usando la combinación de teclas **"ctrl+c"**

Es importante realizar correctamente el uso de las direcciones ip.

0.3 Implementación de MQTT en la nube

0.3.1 Amazon AWS

Amazon AWS va a dar por un año una serie de servicios gratis (free tier) con un consumo limitado de su uso. El servicio que interesa es E2C, de las cuales crearemos instancias (servidores virtuales), Key pair loggin (generar llaves para conexion, .pem para openSSH o .ppk para usar con PuTTY y cambiar de formatos entre llaves usando **puttygen.exe**), el cual no se comparte con nadie

0.3.2 Oracle Cloud Infraestructure

Asumiendo que ya tenemos una maquina virtual en la nube y que accederemos usando la consola de Linux usando SHH

```
ubuntu@instance-20240709-1306:~$ 23:05 $ ssh ubuntu@159.54.134.99
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.8.0-1015-oracle x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Sun Feb  9 05:07:09 UTC 2025

System load: 0.0      Processes:          115
Usage of /: 12.0% of 44.96GB  Users logged in:    0
Memory usage: 27%           IPv4 address for ens3: 10.0.0.128
Swap usage:  0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

53 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Thu Feb  6 21:32:36 2025 from 201.116.38.18
ubuntu@instance-20240709-1306:~$ ubuntu@instance-20240709-1306:~$ ubuntu@instance-20240709-1306:~$ ubuntu@instance-20240709-1306:~$
```

La dirección del servidor que se usa para pruebas en este momento es 159.54.134.99

Procedemos a realizar lo siguiente:

```
$sudo su
#apt-get update
#apt-get upgrade
```

Nota: en el caso de que se haya actualizado el kernel
#reboot

```
$sudo su
#apt-get install mosquitto
```

Ahora editamos el archivo de configuración de mosquitto

```
$sudo su
#cd /etc/mosquitto
#ls
#nano mosquitto.conf
```

Agregar las dos lineas al final del archivo, asi que el contenido del archivo se debe ver como:

mosquitto.conf

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883 0.0.0.0
allow_anonymous true
```

Ahora verificamos el contenido del archivo y reiniciamos el servicio de mosquitto

```
#cat mosquitto.conf
#/etc/init.d/mosquitto stop
#/etc/mosquitto# /etc/init.d/mosquitto start
```

En este momento, el servicio de mosquitto ya esta funcionando, pero al hacer la prueba esto no va a funcionar aún.

Desde la consola de una maquina diferente que tenga instalado mosquitto, vamos a ejecutar lo siguiente:

```
23:29 $ mosquitto_sub -h 159.54.134.99 -t "test/topic"
Error: Connection timed out
```

Como se puede observar, ahora da un error, puesto que no se puede conectar ya que la seguridad de la nube de oracle y el firewall lo está bloqueando. Para resolver lo anterior hay que **Abrir puertos** Loguearse en la cuenta de cloud.oracle.com.

Seleccionar "Dashboard."

The screenshot shows the Oracle Cloud Home page. At the top, there is a banner for a Free Tier account, a search bar, and a location dropdown set to Mexico Central (Queretaro). The main area has a sidebar with categories like Compute, Storage, Networking, Oracle Database, Databases, Analytics & AI, Developer Services, Identity & Security, Observability & Management, and Hybrid. Below the sidebar, there are three sections: 'Pinned' (Cloud Classic, Instances Recommended, Virtual cloud networks Recommended), 'Recently visited' (No items found), and a 'Service spotlight' section for Virtual cloud networks.

Seleccionar el virtual cloud network.

The screenshot shows the Virtual Cloud Networks page under the Networking category. It displays a table with one item: vcn-20231130-1502. The table columns include Name, State, IPv4 CIDR Block, IPv6 Prefix, Default Route Table, DNS Domain Name, and Created. The item vcn-20231130-1502 is listed with the state 'Available', IPv4 CIDR Block '10.0.0.0/16', and Default Route Table '1502'. The DNS Domain Name is 'vcn11301519.oraclevcn.com' and it was created on 'Thu, Nov 30, 2023, 21:19:47 UTC'. The table has a footer note 'Showing 1 item < 1 of 1 >'.

Seleccionar la subnet correspondiente.

An unexpected problem occurred. Refresh your browser.

You are using a Free Tier account. To access all services and resources, [upgrade](#) to a paid account.

[Learn more](#)

Seleccionar el security list.

⚠️ Free Tier account
You are using a Free Tier account. To access all services and resources, [upgrade](#) to a paid account.

[Learn more](#)

Seleccionar "Add Ingress Rules"

Default Security List for vcn-20231130-1502

Instance traffic is controlled by firewall rules on each instance in addition to this Security List

Add Ingress Rules	Edit	Remove					
Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
No	0.0.0.0	TCP	All	22		TCP traffic for ports: 22 SSH Remote Login Protocol	
No	0.0.0.0	ICMP			3, 4	ICMP traffic for 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set	

Realizar lo siguiente:

1. En la nueva regla, no active ni de click "Stateless."
2. Establezca "Source type" en CIDR. Establezca "Source CIDR" to 0.0.0.0/0. Es posible establecer algo menos general que 0.0.0.0/0, si deseamos restringir las direcciones ip de entrada a nuestro servidor, pero esa sería una configuración para un escenario particular.
3. Establezca el protocolo IP deseado, en este caso, TCP.
4. Deje establecido Source Port Range en All.
5. Establezca Dest Port Range al rango de puertos que desea abrir, en este caso, 1883.
6. Ingrese la descripción de la regla,

Ingress Rule 1

Allows TCP traffic for ports: all

Stateless (radio button)

Source Type: CIDR

Source CIDR: 0.0.0.0/0

IP Protocol: TCP

Source Port Range: All

Destination Port Range: 1883

Description: Open access to mosquitto server

7. Guarde la nueva regla de ingreso. Damos click en el botón de "Add Ingress Rules"

Ahora esa regla debe aparecer en el dashboard

The screenshot shows the Oracle Cloud Infrastructure (OCI) Security List Information page for a security list named 'SL'. The page includes a 'Free Tier account' notice, search bar, and navigation tabs. The main content area displays 'Security List Information' with details like OCID, creation date, and compartment. Below this is a 'Resources' section with 'Ingress Rules' and 'Egress Rules' tabs. The 'Ingress Rules' table lists four rules:

	Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
<input type="checkbox"/>	No	0.0.0.0/0	TCP	All	22			TCP traffic for ports: 22 SSH Remote Login Protocol
<input type="checkbox"/>	No	0.0.0.0/0	ICMP			3, 4		ICMP traffic for: 3. Destination Unreachable: Fragmentation Needed and Don't Fragment was Set
<input type="checkbox"/>	No	10.0.0.0/16	ICMP			3		ICMP traffic for: 3 Destination Unreachable
<input type="checkbox"/>	No	0.0.0.0/0	TCP	All	1883			TCP traffic for ports: 1883 Open access to mosquito server

At the bottom right of the table, it says 'Showing 4 items < 1 of 1 >'.

Una vez creada la regla, Tambien se debe configurar la máquina virtual, ya que incorporan firewalls internos (ya sea iptables o uncomplicated firewall, ufw) y se deben abrir los puertos, pero una vez realizado esto, el servicio funcionará.

Ahora desde dentro de la maquina virtual (por ser la infraestructura de oracle, usa iptables), abrimos el puerto con IPTABLES

```
sudo iptables -I INPUT 6 -m state --state NEW -p tcp --dport 1883 -j ACCEPT
```

Con esta regla ya deberíamos poder comunicarnos con el servidor de mosquitto.

(LO DE ESTOS PARÉNTESIS NO REALIZARLO EN ORACLE CLOUD

NOTA: EN CASO DE USAR otro servicio de nube, que utilice ufw:

```
firewall-cmd --permanent --zone=public --add-port=1883/tcp; firewall-cmd --reload  
)
```

Hacemos la prueba desde otra consola en otra maquina:

```
00:26 $ mosquitto_sub -h 159.54.134.99 -t "test/topic"
```

No habrá ningún mensaje porque al conectarse se queda en modo bloqueante esperando recibir mensajes.

The screenshot shows three terminal windows side-by-side. The leftmost terminal window has a dark header bar with the text "root@instance-20240709-1306:/home/ubuntu". Below the header, the command "mosquitto_sub -h 159.54.134.99 -t "test/topic"" is entered. The middle terminal window has a dark header bar with the text "Terminal". The rightmost terminal window has a dark header bar with the text "Terminal". There is a vertical scroll bar on the right side of the terminal area.

Ahora desde otra terminal en la maquina desde la que se hacen pruebas ejecutamos

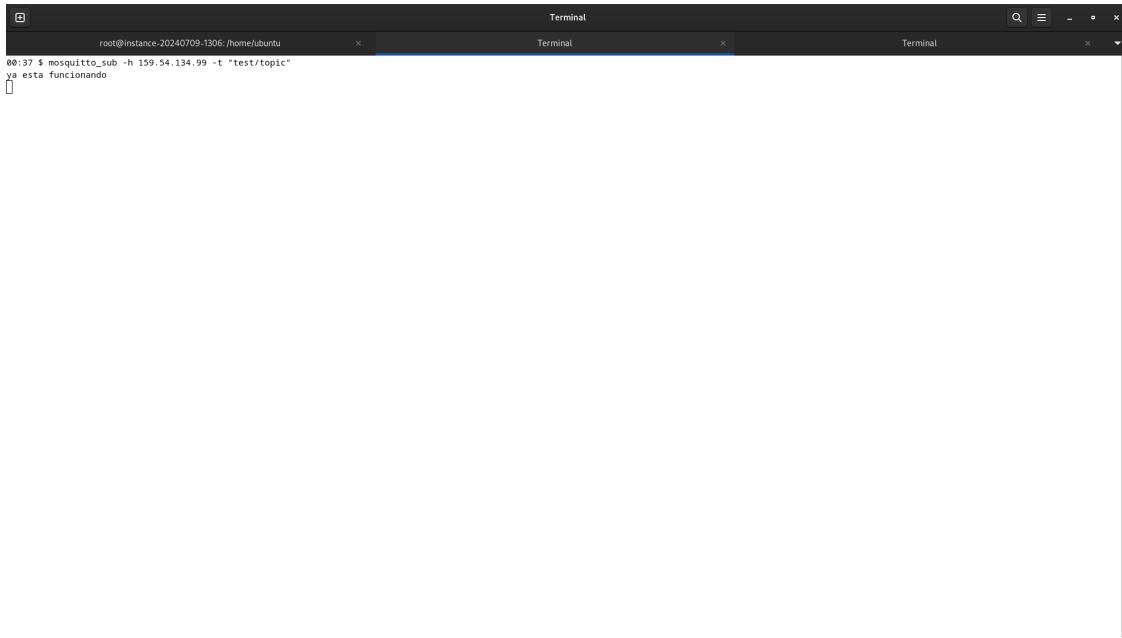
```
00:26 $ mosquitto_pub -h 159.54.134.99 -t "test/topic" -m "ya esta funcionando" 00:27 $
```

The screenshot shows three terminal windows side-by-side. The leftmost terminal window has a dark header bar with the text "root@instance-20240709-1306:/home/ubuntu". Below the header, the command "mosquitto_pub -h 159.54.134.99 -t "test/topic" -m "ya esta funcionando"" is entered. The middle terminal window has a dark header bar with the text "Terminal". The rightmost terminal window has a dark header bar with the text "Terminal". There is a vertical scroll bar on the right side of the terminal area.

Y podemos ver en la primer terminal, en la que se quedo en modo bloqueante lo siguiente:

```
00:26 $ mosquitto_sub -h 159.54.134.99 -t "test/topic"
```

ya esta funcionando



The screenshot shows three terminal windows side-by-side. The leftmost terminal window has a dark header with the text "root@instance-20240709-1306:/home/ubuntu". It contains the command "mosquitto_sub -h 159.54.134.99 -t "test/topic"" followed by the message "ya esta funcionando". The middle and right terminal windows have a light gray header labeled "Terminal" and are completely blank, indicating they are waiting for input or output.

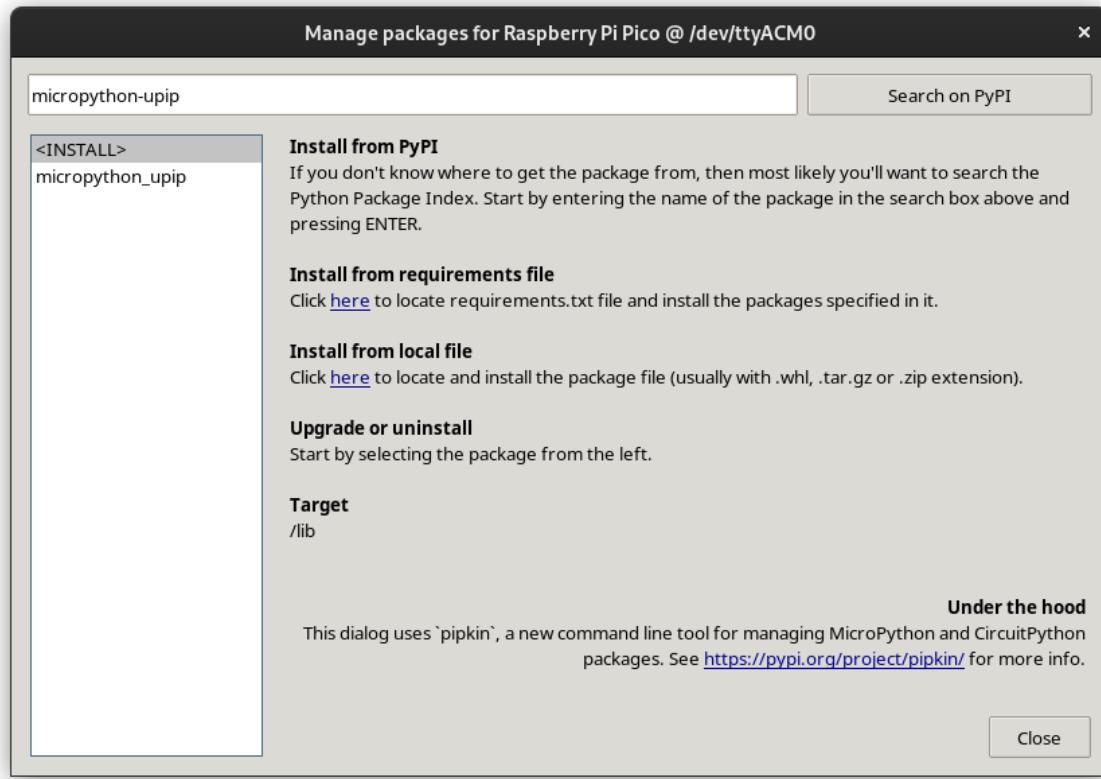
Dicha terminal seguirá en modo bloqueante esperando recibir mas mensaje y con eso aseguramos que el servicio está activo. Podemos terminar la subscripcion usando la combinación de teclas "Ctrl+c".

Ahora el Broker está en la nube. Recuerde que se tendrán que hacer configuraciones adicionales para que el servicio de mosquitto se ejecute cada vez que se reincie la máquina virtual y ademas tambien la regla de iptables requiere mas configuración en caso de que deseemos que sea permanente.

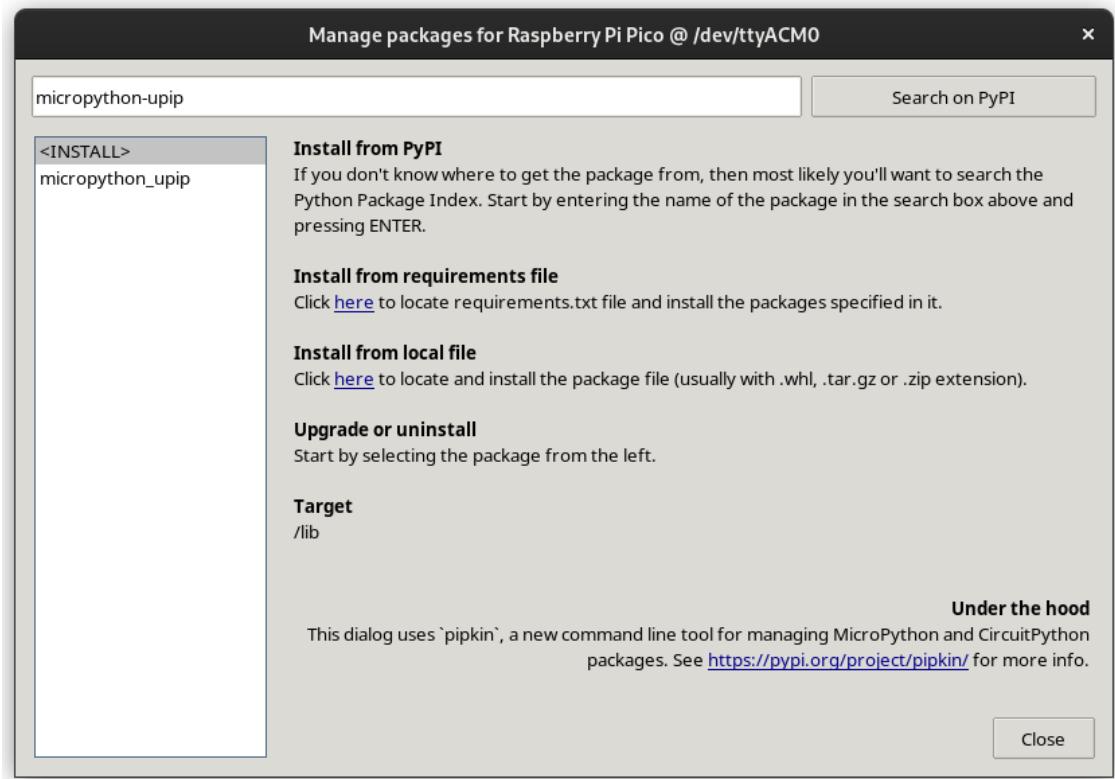
0.4 Instalar Libreria umqtt

Para instalar la librería umqtt en la pico, en caso de que no esté instalada

1. Instalar micropython-upip en Thonny, lo que permite manejar los paquetes.



2. Hacer que el Raspberry Pico W tenga internet (libreria network), para lo cual ejecutamos el código del archivo **06_ejemplo_02Conexion_wifi.py**



ofuscar

informacion de red en la imagen

Previo al siguiente paso, se puede observar que dentro del directorio lib, no se encuentra la libreria de umqtt

The screenshot shows the Thonny IDE interface. The top window is titled "Thonny -" and contains a menu bar with File, Edit, View, Run, Tools, Help. Below the menu is a toolbar with icons for file operations and a stop button. A tab labeled "06_ejemplo_02Conexion_wifi.py" is selected. The code in the editor is:

```
1 import network
2 import utime
3
4
5 wf = network.WLAN(network.STA_IF)
6 wf.active(True)
7
8 wf.connect('[REDACTED]', '[REDACTED]')
9 #wf.connect('[REDACTED]', '[REDACTED]')
10
11 # la conexión puede demorar unos segundos,
12 # Creamos un ciclo que termina cuando se ha realizado la conexión
13 while not wf.isconnected():
14     print(".")
15     utime.sleep(1) # cada segundo se imprime un punto
16
17 # al salir, imprimimos
18 print("Conectado al WiFi")
19 print(wf.ifconfig())
20
```

Below the editor is a "Shell" window with the title "Shell <". It contains the command ">>> %Run -c \$EDITOR_CONTENT" followed by several dots indicating output. At the bottom of the shell window, the text "Conectado al WiFi" and the IP configuration "('192.168.2.164', '255.255.255.0', '192.168.2.1', '192.168.2.207')". The status bar at the bottom right of the shell window says "MicroPython (Raspberry Pi Pico) • /dev/ttyACM0".

ofuscar

informacion de red en la imagen

3. En la shell de Thonny >>>import mip >>>mip.install("umqtt.robust")
 >>>mip.install("umqtt.simple")

The screenshot shows the Thonny IDE interface. The top menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations and a stop button. The left sidebar is titled 'Files' and shows two locations: 'This computer' and 'Raspberry Pi Pico'. Under 'Raspberry Pi Pico', there is a 'lib' folder containing 'micropython_upip-1.2.4.dist-info' and 'umqtt' which contains 'simple.mpy', 'upip.py', and 'upip_utarfile.py'. The main code editor window is titled '06_ejemplo_02_conexion_wifi.py' and contains the following Python code:

```
1 import network
2 import utime
3
4
5 wf = network.WLAN(network.STA_IF)
6 wf.active(True)
7
8 wf.connect('XXXXXXXXXX', 'XXXXXXXXXX')
#wf.connect('XXXXXXXXXX', 'XXXXXXXXXX')
9
10 # la conexión puede demorar unos segundos
11 # Creamos un ciclo que termina cuando se
12 while not wf.isconnected():
13     print(".")
14     utime.sleep(1) # cada segundo se impr
15
16 # al salir, imprimimos
17 print("Conectado al WiFi")
18 print(wf.ifconfig())
19
20
21
22
```

Below the code editor is a 'Shell' window showing the command and its output:

```
>>> mip.install("umqtt.simple")
Installing umqtt.simple (latest) from https://micropython.org/pi/v2 to /lib
Copying: /lib/umqtt/simple.mpy
Done
```

At the bottom right of the shell window, it says 'MicroPython (Raspberry Pi Pico) • /dev/ttyACM0'.

ofuscar

informacion de red en la imagen

4. Verificar que en la carpeta lib este umqtt del Raspberry Pi Pico W

The screenshot shows the Thonny IDE interface. The top window is titled "Thonny -" and contains a code editor with the file "06_ejemplo_02_conexion_wifi.py". The code imports network and utime modules, initializes a WLAN interface, connects to a WiFi network, and prints a message when connected. The bottom window is a "Shell" terminal showing MicroPython commands for installing the umqtt.simple and umqtt.robust libraries from the micropython.org/pi/v2 repository to the /lib directory.

```

import network
import utime

wf = network.WLAN(network.STA_IF)
wf.active(True)

wf.connect('...', '...')

# la conexión puede demorar unos segundos,
# Creamos un ciclo que termina cuando se ha realizado la conexión
while not wf.isconnected():
    print(".")
    utime.sleep(1) # cada segundo se imprime un punto

# al salir, imprimimos
print("Conectado al WiFi")

```

```

>>> mip.install("umqtt.simple")
Installing umqtt.simple (latest) from https://micropython.org/pi/v2 to /lib
Copying: /lib/umqtt/simple.mpy
Done

>>> mip.install("umqtt.robust")
Installing umqtt.robust (latest) from https://micropython.org/pi/v2 to /lib
Copying: /lib/umqtt/robust.mpy
Done

>>> %cd /lib/umqtt
>>> %cd /lib
>>>

```

ofuscar

informacion de red en la imagen

0.4.1 Descripción de la librería umqtt

```

//Librería
import umqtt.simple as mqtt

//Objeto
c=mqtt.MQTTClient("clienteid", brokermqtt)

//Funciones publicador y suscriptor
c.connect()
c.disconnect()

```

```

//Funciones publicador
c.publish(topic, mensaje)

//Funciones de suscriptor
def funcionC(topic, msg):
    print(topic+ " " + msg)

//El proceso en orden es el siguiente

c.callback(funcionC) #definir la funcion

#aqui va el c.connect()
c.subscribe(topic) #subscriber al topic
c.wait_msg() #esperar 1 mensaje, si se desea se puede poner dentro de un bucle para que se quede

```

0.5 Ejemplo 01 MQTT

Realizar un programa en el Raspberry Pi Pico que se suscriba a un servidor MQTT en el topic "prueba". Luego hacer las pruebas de enviar mensaje desde la App IoT MQTT Panel

07_ejemplo_01_mqtt_recepcion_mensajes.py

```

import network
import utime
import umqtt.simple as mqtt

### Funcion a realizar cuando se reciba un mensaje mqtt
def receptor(topic, msg):
    #print(topic + " " + msg) #imprimir topic y mensaje
    print(msg) # imprimir solo mensaje

### SECCION DE CONEXION CON wifi
wf = network.WLAN(network.STA_IF)
wf.active(True)

wf.connect('ISSD', 'password red')

# la conexión puede demorar unos segundos,
# Creamos un ciclo que termina cuando se ha realizado la conexión
while not wf.isconnected():
    print(".")


```

```

utime.sleep(1) # cada segundo se imprime un punto

# al salir, imprimimos
print("Conectado al WiFi")
print(wf.ifconfig())

### SECCION DE CONEXION CON mqtt
#el id "pico1" debe ser único, si hay mas picos, estos ids deben ser diferentes
#la ip del broker 159.54.134.99
c=MQTTClient("pico1", "159.54.134.99")
#define la funcion a realizar cuando recibimos un mensaje
c.set_callback(receptor)
#conectamos y suscribimos
c.connect()
c.subscribe('prueba')
#ciclamos la recepcion de mensajes
print("Ejecutando el suscriptor")
while True:
    c.wait_msg()

```

Your browser does not support the video tag.

0.6 Probar desde el teléfono

Buscar la App **iot mqtt panel** en el **PlayStore** de Google, del autor Rahul Kundu.

Al abrir la aplicación indica que no hay ninguna conexión

3:17



≡ Connections

You do not have any connection to communicate with MQTT broker. If you are using this application for the first time, we highly recommend to go through FAQ and User Guide from main menu.

SETUP A CONNECTION



En este ejemplo, se agrega la información de la imagen pero recuerde que debe establecer la información correspondiente a su configuración personal.

3:23

Bluetooth signal strength 8 Wi-Fi signal strength

Add Connection

Connection name *

oracle 20240709-1306

Client ID

celularRedmi

Broker Web/IP address *

159.54.134.99

Port *

1883

Network protocol

TCP

Add Dashboard

Additional options

CANCEL

CREATE

1 2 3 4 5 6 7 8 9 0

@ # \$ _ & - + () /

=\< * " ' : ; ! ? ⌂

ABC , 1234 . →

23

Se debe dar clic en agregar un dashboard

3:26

Bluetooth, Signal, WiFi, 14%

Add Connection

Connection name *

oracle 20240709-1306

Client ID

celularRedmi

Broker Web/IP address *

159.54.134.99

Port *

1883

Network protocol

TCP

Add

Dashboard name

PanelPrueba

Set as connection home

CANCEL

SAVE

PanelPrueba | Panelprueba | Panel prueba

q w e r t y u i o p
a s d f g h j k l ñ
z x c v b n m

?123 , .

25

Page Number: 25

Y dar clic en el botón de **CREATE**

Cuando el simbolo es color naranja, significa que se ha establecido la conexión. Ahora entramos al dashboard

3:27



≡ Connections

oracle 20240709-1306



Agregar un panel

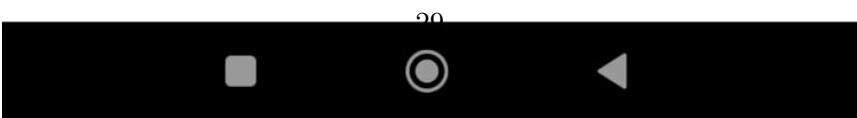


≡ PanelPrueba



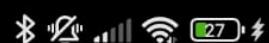
Current dashboard does not have any panel

ADD PANEL



y agregamos un control (hay controles que son publicadores, como text input, y suscriptores, como text log) para poder enviar información, seleccionamos text input

3:32



Select panel type to add



- | | | |
|--|-----------------------|--|
| | Button | |
| | Switch | |
| | Slider | |
| | Text Input | |
| | Text Log | |
| | Node Status | |
| | Combo Box | |
| | Radio Buttons | |
| | LED Indicator | |
| | Multi-State Indicator | |
| | Progress | |
| | Gauge | |
| | Color Picker | |
| | Date & Time Picker | |
| | Line Graph | |



Se establece el nombre como **texto**, el topic como **prueba**



← Add a Text Input panel

Panel name *

texto

Disable dashboard prefix topic



Topic *

prueba

Clear text on publish

Payload is JSON Data

Show sent timestamp

Confirm before publish

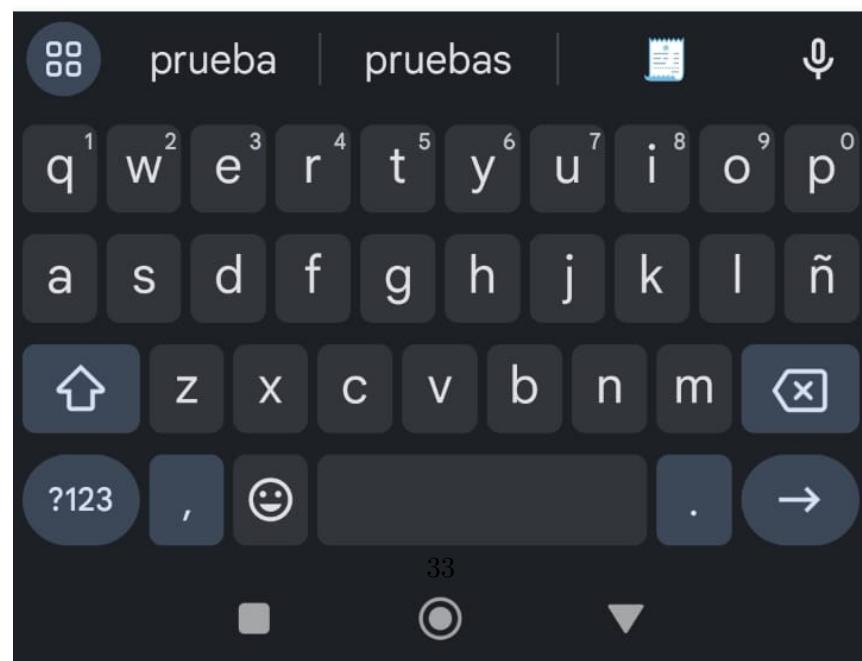
Retain

QoS

0 ▾

CANCEL

CREATE



Y se selecciona el botón de **CREATE**

Escribimos el mensaje **hola desde celular** y lo enviamos



← Add a Text Input panel

Panel name *

texto

Disable dashboard prefix topic



Topic *

prueba

Clear text on publish

Payload is JSON Data

Show sent timestamp

Confirm before publish

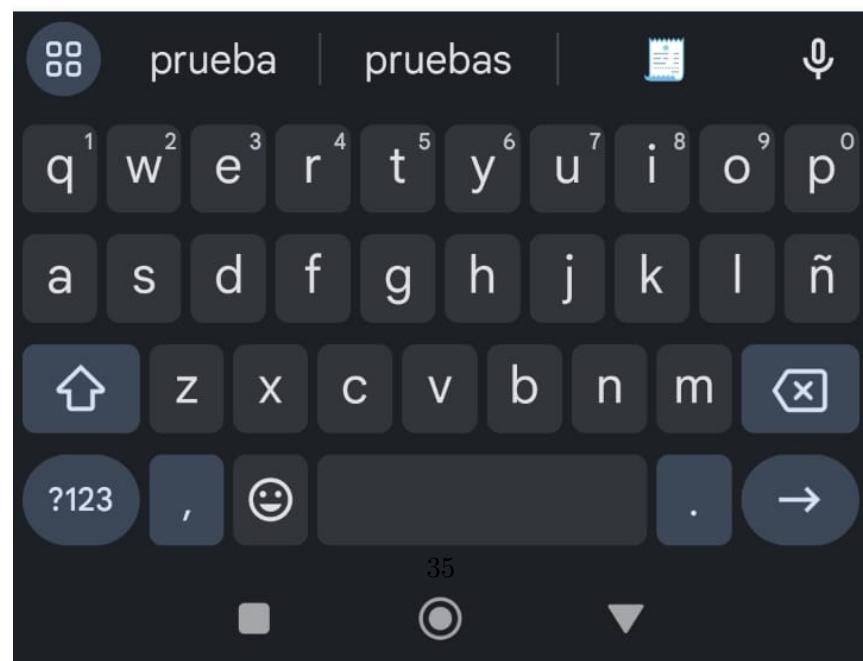
Retain

QoS

0 ▾

CANCEL

CREATE



Escribimos el mensaje a enviar



Y se verifica que haya sido recibido en la Raspberry Pi Pico W

The screenshot shows a terminal window with two panes. The left pane is a code editor with Python scripts for WiFi connection and MQTT communication. The right pane is a terminal window showing the output of the MQTT commands.

```
File Edit View Run Tools Help
06_ejemplo_02_conexion_wifi.py 07_ejemplo_01_mqtt_repcion_mensajes.py
1 import network
2 import utime
3 import umqtt.simple as mqtt
4
5 ### Funcion a realizar cuando se reciba un mensaje mqtt
6 def receptor(topic, msg):
7     #print(topic + " " + msg) #imprimir topic y mensaje
8     print(msg) # imprimir solo mensaje
9
10
11 ### SECCION DE CONEXION CON wifi
12 wf = network.WLAN(network.STA_IF)
13 wf.active(True)
14
15 wf.connect('████████', '████████')
16 #wf.connect('████████', '████████')
17
18 # la conexión puede demorar unos segundos,
19 # Creamos un ciclo que termina cuando se ha realizado la conexión
20 while not wf.isconnected():
21     print(".")

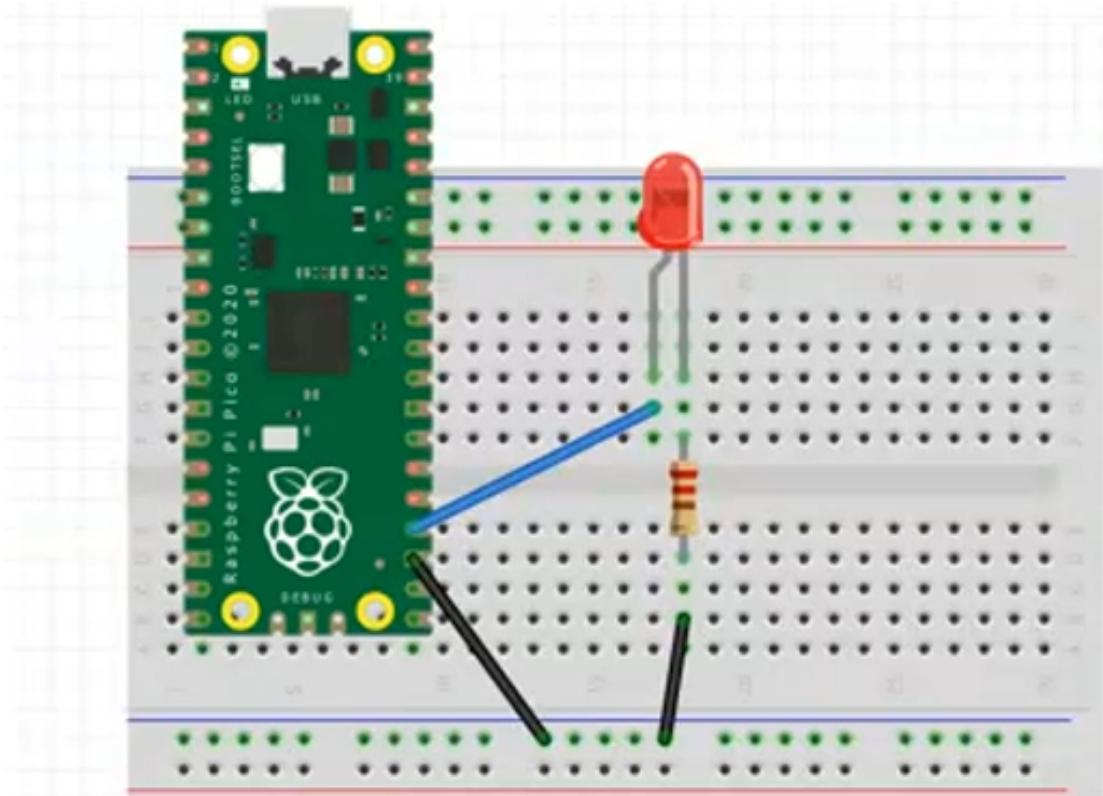
>>> %Run -c $EDITOR_CONTENT
Conectado al WiFi
('192.168.2.164', '255.255.255.0', '192.168.2.1', '192.168.2.267')
Ejecutando el suscriptor
b'holá'
b'holá'
b'holá desde celular'

ubuntu@instance-20240709-1306:~$ mosquitto_pub -h 159.54.134.99 -t "prueba" -m "holá"
ubuntu@instance-20240709-1306:~$ mosquitto_pub -h 159.54.134.99 -t "prueba" -m "holá"
ubuntu@instance-20240709-1306:~$
```

0.7 Ejemplo 02 MQTT

Hacer un programa en el Raspberry Pi Poco W que se suscriba a un servidor MQTT en el topic "prueba"

Conectar un led en el GPIO 18, hacer que si recibe la letra A encienda el led, con la letra B apaga el led. En el App crear dos botones y publicar A con el botón "Encender" y publicar B con el botón "Apagar"



07_ejemplo_02_mqtt_encender_led.py

```
import network
import utime
import umqtt.simple as mqtt
from machine import Pin

### Funcion a realizar cuando se reciba un mensaje mqtt
def receptor(topic, msg):
    #print(topic + " " + msg) #imprimir topic y mensaje
    # print(msg) # imprimir solo mensaje
    # print(msg.decode()) # Lo convierte a string
    m = msg.decode() # mensaje en string
    print(m)
    if m == 'A':
        led.on()
    elif m == 'B':
        led.off()
```

```

### SECCION DE CONEXION CON wifi
wf = network.WLAN(network.STA_IF)
wf.active(True)

wf.connect('ISSD', 'password red')

# la conexión puede demorar unos segundos,
# Creamos un ciclo que termina cuando se ha realizado la conexión
while not wf.isconnected():
    print(".")
    utime.sleep(1) # cada segundo se imprime un punto

# al salir, imprimimos
print("Conectado al WiFi")
print(wf.ifconfig())

### SECCION led
# led de la tarjeta pi pico
led = Pin('LED', Pin.OUT)
# led conectado al pin 18
#led = Pin(18, Pin.OUT)

### SECCION DE CONEXION CON mqtt
#el id "pico1" debe ser único, si hay mas picos, estos ids deben ser diferentes
#la ip del broker 159.54.134.99
c=mqtt.MQTTClient("pico1", "159.54.134.99")
#define la función a realizar cuando recibimos un mensaje
c.set_callback(receptor)
#conectamos y suscribimos
c.connect()
c.subscribe('prueba')
#ciclamos la recepción de mensajes
print("Ejecutando el suscriptor")
print("Esperando mensaje")
while True:
    c.wait_msg()

```

Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón +

12:53



≡ PanelPrueba



texto

hola desde celular



Selección **Button**

12:53



Select panel type to add



- | | | |
|--|-----------------------|--|
| | Button | |
| | Switch | |
| | Slider | |
| | Text Input | |
| | Text Log | |
| | Node Status | |
| | Combo Box | |
| | Radio Buttons | |
| | LED Indicator | |
| | Multi-State Indicator | |
| | Progress | |
| | Gauge | |
| | Color Picker | |
| | Date & Time Picker | |
| | Line Graph | |



Configurar el botón, con el atributo Panel name **Encender**, Tópic **prueba**, Payload **A**,

12:54



← Add a Button panel

Panel name *

Encender

 Disable dashboard prefix topic

Topic *

prueba

 No payload

Payload *

A|

Separate payload on release



Button color



Button size

Medium ▾

 Repeat publish until released Fit to panel width Use icons for button Payload is JSON Data Show sent timestamp Confirm before publish Retain

QoS

0 ▾

45



todos los demás atributos se dejan como están, y posteriormente dar clic en **Aceptar**

12:54

76%

← Add a Button panel

Disable dashboard prefix topic



Topic *

prueba

No payload



Payload *

A

Separate payload on release



Button color



Button size

Medium ▾

Repeat publish until released

Fit to panel width

Use icons for button

Payload is JSON Data

Show sent timestamp

Confirm before publish

Retain

QoS

0 ▾

CANCEL

CREATE

Ahora se creará el botón para apagar el led.

Configurar el botón, con el atributo Panel name **Apagar**, Tópic **prueba**, Payload **B**,

12:55

76%

← Add a Button panel

Panel name *

Apagar

Disable dashboard prefix topic



Topic *

prueba

No payload



Payload *

B

Separate payload on release



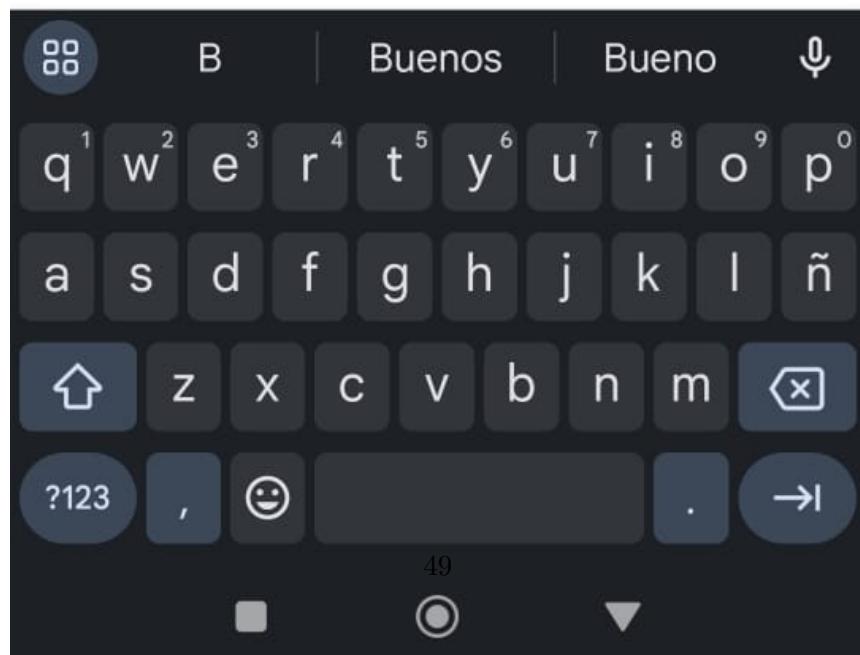
Button color



Button size

Medium ▾

Repeat publish until released



todos los demás atributos se dejan como están, y posteriormente dar clic en **Aceptar**

12:55



← Add a Button panel

Panel name *

Apagar

Disable dashboard prefix topic



Topic *

prueba

No payload



Payload *

B|

Separate payload on release



Button color



Button size

Medium ▾

Repeat publish until released

Fit to panel width

Use icons for button

Payload is JSON Data

Show sent timestamp

Confirm before publish

Retain

QoS

0 ▾



Your browser does not support the video tag.

0.8 Ejemplo 03 MQTT

Hacer un programa en el Raspberry Pi Poco W que pida un mensaje por consola, si el mensaje es “z”, terminar el programa, si es otro mensaje publicarlo en el servidor MQTT. En la App adicionar un control “Text Log” que se suscriba al mismo topic del Raspberry Pi Pico W, monitorear los mensajes que publica el Raspberry Pi Pico W.

07_ejemplo_03_mqtt_publicador_mensajes.py

```
import network
import utime
import umqtt.simple as mqtt

### SECCION DE CONEXION CON wifi
wf = network.WLAN(network.STA_IF)
wf.active(True)

wf.connect('ISSD', 'password red')

# la conexión puede demorar unos segundos,
# Creamos un ciclo que termina cuando se ha realizado la conexión
while not wf.isconnected():
    print(".")
    utime.sleep(1) # cada segundo se imprime un punto

# al salir, imprimimos
print("Conectado al WiFi")
print(wf.ifconfig())

### SECCION DE CONEXION CON mqtt
#el id "pico1" debe ser único, si hay mas picos, estos ids deben ser diferentes
#la ip del broker 159.54.134.99
c=mqtt.MQTTClient("pico1", "159.54.134.99")

#conectamos
c.connect()
#ciclamos la recepcion de mensajes
print("Ejecutando el suscriptor")
continuar = True
while continuar:
```

```
a = input('Escriba el mensaje: ')
if a == 'z':
    continuar = False
else:
    c.publish('prueba', a)

c.disconnect()
print('Fin del programa')
```

Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón +

11:09 ☺ ...



≡ PanelPrueba



texto

hola desde celular



ENCENDER



APAGAR



Seleccionar la opción **Text Log**

11:09 ⚡ ...



Select panel type to add



- | | | |
|--|-----------------------|--|
| | Button | |
| | Switch | |
| | Slider | |
| | Text Input | |
| | Text Log | |
| | Node Status | |
| | Combo Box | |
| | Radio Buttons | |
| | LED Indicator | |
| | Multi-State Indicator | |
| | Progress | |
| | Gauge | |
| | Color Picker | |
| | Date & Time Picker | |
| | Line Graph | |



Configurar el panel con el atributo Panel Name **Log**, el atributo Topic **prueba** y dar click en el botón **Create**

11:09 ⓘ ...



← Add a Text Log panel

Panel name *

Log

Disable dashboard prefix topic



Topic *

prueba

Show last message only

Max visible lines

10

Max persistence

100

Hide topic

Enable notification



Payload is JSON Data

Show received timestamp

QoS

0 ▾

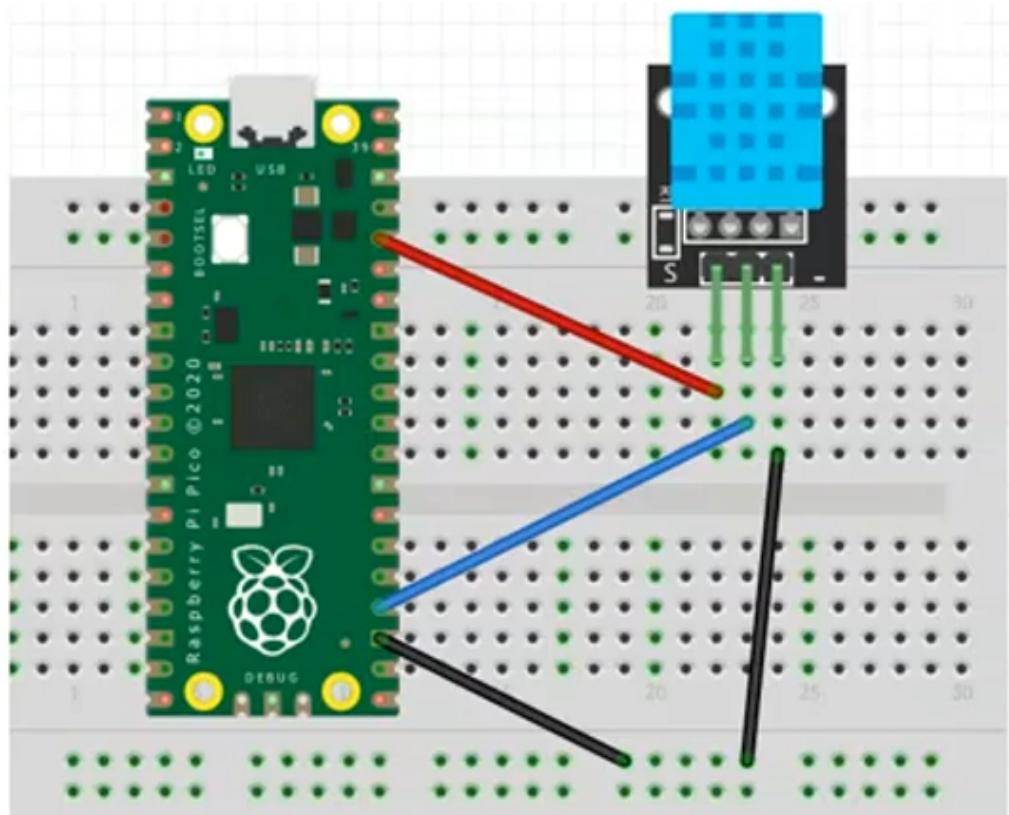
CANCEL

CREATE

Your browser does not support the video tag.

0.9 Ejemplo 04 MQTT

Hacer un programa en el Raspberry Pi Pico W que realice el monitoreo de la temperatura y humedad de un sensor DHT11, conectado en el GPIO18, crear el App con 2 gauge(medidor radial), uno para temperatura, otro para humedad.



```
//Librería
import dht

//Objeto
dht11 = dht.DHT11(Pin(x))

dht11.measure()
dht.temperature()
dht11.humidity()
```

07_ejemplo_04_mqtt_publicador_temperatura_humedad.py

```
import network
import utime
import umqtt.simple as mqtt
import dht
from machine import Pin

### SECCION DE CONEXION CON wifi
wf = network.WLAN(network.STA_IF)
wf.active(True)

wf.connect('ISSD', 'password red')

# la conexión puede demorar unos segundos,

### DEFINICIÓN DEL OBJETO DE INTERES
dht11 = dht.DHT11(Pin(18))

# Creamos un ciclo que termina cuando se ha realizado la conexión
while not wf.isconnected():
    print(".")
    utime.sleep(1) # cada segundo se imprime un punto

# al salir, imprimimos
print("Conectado al WiFi")
print(wf.ifconfig())

### SECCION DE CONEXION CON mqtt
#el id "pico1" debe ser único, si hay mas picos, estos ids deben ser diferentes
#la ip del broker 159.54.134.99
c=mqtt.MQTTClient("pico1", "159.54.134.99")

#conectamos
c.connect()
#ciclamos la recepcion de mensajes
print("Ejecutando el suscriptor")

while True:
    dht11.measure()
    c.publish('prueba', str(dht11.temperature()))
    c.publish('prueba1',str(dht11.humidity()))
    utime.sleep(2) # Envía cada dos segundos

c.disconnect()
```

```
print('Fin del programa')
```

Ejecutamos el programa

Probamos primero usando dos terminales en la computadora (temperatura y humedad respectivamente), una terminal ejecutando el comando:

```
mosquitto_sub -h 159.54.134.99 -t "prueba"
```

Y otra terminal ejecutando el comando:

```
mosquitto_sub -h 159.54.134.99 -t "prueba"
```

The screenshot shows a terminal window at the bottom left and a code editor window at the top right. The terminal window displays the command `mosquitto_sub -h 159.54.134.99 -t "prueba"`. The code editor window shows a Python script named `07_ejemplo_04_imprepcionador/temperatura_humedad.py`. The script uses the `MQTTClient` class to connect to a broker at `159.54.134.99` and publish temperature and humidity data from a DHT11 sensor every 2 seconds. The script also prints a message when it starts and when it exits.

```
18:22 $ mosquitto_sub -h 159.54.134.99 -t "prueba"
22
23
24
25
26
27
28
29 # al salir, imprimimos
30 print("Conectado al WiFi")
31 print(wf.ifconfig())
32
33 ### SECCION DE CONEXION CON mqtt
34 #el id "picol" debe ser único, si hay mas picos, estos ids deben ser diferentes
35 #la ip del broker 159.54.134.99
36 c=mqtt.MQTTClient("picol", "159.54.134.99")
37
38 #conectamos
39 c.connect()
40 #ciclamos la recepcion de mensajes
41 print("Ejecutando el suscriptor")
42
43 while True:
44     dht11.measure()
45     c.publish('prueba', str(dht11.temperature()))
46     c.publish('prueba', str(dht11.humidity()))
47     time.sleep(2) # Envía cada dos segundos
48
49 c.disconnect()
50 print('Fin del programa')
51
```

0.9.1 Eliminar el Text log panel creado en el ejemplo anterior

Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón +

Seleccionar la opción **Gauge**, el cual será para la temperatura

6:59 ☺ ...



Select panel type to add



- | | | |
|--|-----------------------|--|
| | Button | |
| | Switch | |
| | Slider | |
| | Text Input | |
| | Text Log | |
| | Node Status | |
| | Combo Box | |
| | Radio Buttons | |
| | LED Indicator | |
| | Multi-State Indicator | |
| | Progress | |
| | Gauge | |
| | Color Picker | |
| | Date & Time Picker | |
| | Line Graph | |

Configurar el panel con el atributo Panel Name **Temp**, el atributo Topic **prueba**, el atributo Payload min **0** y el atributo Payload max **50**, dar click en el botón **Create**

7:02 ⌂ ...



← Add a Gauge panel

Panel name *

Temp

 Disable dashboard prefix topic

Topic *

prueba

Payload min *

0

Payload max *

50

Unit

Factor

1

Arc color



16.67



33.33

 Enable notification Payload is JSON Data Show received timestamp

QoS

0 ▾

CANCEL

CREATE

Ahora para la humedad Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón + Seleccionar la opción **Gauge**, el cual será para la humedad

7:03 ☀ ...



≡ PanelPrueba



texto

hola desde celular



⋮

ENCENDER

⋮

APAGAR

⋮

Temp

22



⋮



66



Configurar el panel con el atributo Panel Name **Humedad**, el atributo Topic **prueba1**, el atributo Payload min **0** y el atributo Payload max **100**, dar click en el botón **Create**

7:04 ⓘ ...

* 53

← Add a Gauge panel

Panel name *

Humedad

 Disable dashboard prefix topic

Topic *

prueba1

Payload min *

0

Payload max *

100

Unit

Factor

1

Arc color



33.33



66.67

 Enable notification Payload is JSON Data Show received timestamp

QoS

0 ▾

CANCEL

CREATE

Quedando el panel de la siguiente manera

7:05 ☀ ...



≡ PanelPrueba



texto

hola desde celular



ENCENDER



APAGAR



Temp

24



Humedad

72



70



Your browser does not support the video tag.

0.10 Ejemplo 05 MQTT

Hacer un programa en el Raspberry Pi Poco W para controlar un servomotor mediante la aplicación con un control de tipo **slider** (barra de desplazamiento). En la App adicionar un control "Slider" que se suscriba al mismo topic del Raspberry Pi Pico W, monitorear los mensajes que publica el Raspberry Pi Pico W y controlar el servomotor.

0.10.1 Ahora crearemos primero la aplicación en el App del celular

Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón +

Seleccionar la opción **Slider**

7:35 ⚡ ...



Select panel type to add

X

- | | | |
|--|-----------------------|--|
| | Button | |
| | Switch | |
| | Slider | |
| | Text Input | |
| | Text Log | |
| | Node Status | |
| | Combo Box | |
| | Radio Buttons | |
| | LED Indicator | |
| | Multi-State Indicator | |
| | Progress | |
| | Gauge | |
| | Color Picker | |
| | Date & Time Picker | |
| | Line Graph | |

Configurar el panel con el atributo Panel Name **Control.servo**, el atributo Topic **prueba**, el atributo Payload min **0**, al atributo Payload max **180** y al atributo Slider step **2** y dar click en el botón **Create**

7:40 ⓘ ...



← Add a Slider panel

Panel name *

Control.servo

 Disable dashboard prefix topic

Topic *

prueba

Subscribe Topic



Payload min *

0

Payload max *

180

Slider step *

2

Factor

1

Unit

Slider Orientation

Horizontal



Slider color

 Dynamic color Enable notification Payload is JSON Data Show received timestamp Show sent timestamp Confirm before publish Retain

QoS

0



Quedando de la siguiente manera (nota, tal vez desee eliminar los paneles creados en el ejemplo anterior, ya que uno de los topic es el mismo y se modificará a medida que se modifique el slider)

7:42



≡ PanelPrueba



texto

hola desde celular



ENCENDER



APAGAR



Temp

0



Humedad

5



Control.servo: 0



76



07_ejemplo_05_mqtt_receptor_mensajes_control_servo.py

```
import network
import utime
import umqtt.simple as mqtt
from machine import PWM,Pin

### Funcion a realizar cuando se reciba un mensaje mqtt
def receptor(topic, msg):
    angulo = msg.decode() # decodificar el mensaje
    print(angulo)

    #calculamos el valor del duty cycle (es decir, la posición del servo)
    dc = 1638.375 + 36.4*int(angulo)
    pwm.duty_u16(int(dc))

### SECCION DE CONEXION CON wifi
wf = network.WLAN(network.STA_IF)
wf.active(True)

wf.connect('ISSD', 'password red')

# la conexión puede demorar unos segundos,
# Creamos un ciclo que termina cuando se ha realizado la conexión
while not wf.isconnected():
    print(".")
    utime.sleep(1) # cada segundo se imprime un punto

# al salir, imprimimos
print("Conectado al WiFi")
print(wf.ifconfig())

### SECCION DE CONEXION CON mqtt
#el id "pico1" debe ser único, si hay mas picos, estos ids deben ser diferentes
#la ip del broker 159.54.134.99
c=umqtt.MQTTClient("pico1", "159.54.134.99")

### DEFINIMOS EL OBJETO DEL SERVOMOTOR
pwm = PWM(Pin(7)) # probar con 18

pwm.freq(50) #frecuencia comun de trabajo del servo
              # es decir el periodo T=20ms
#####

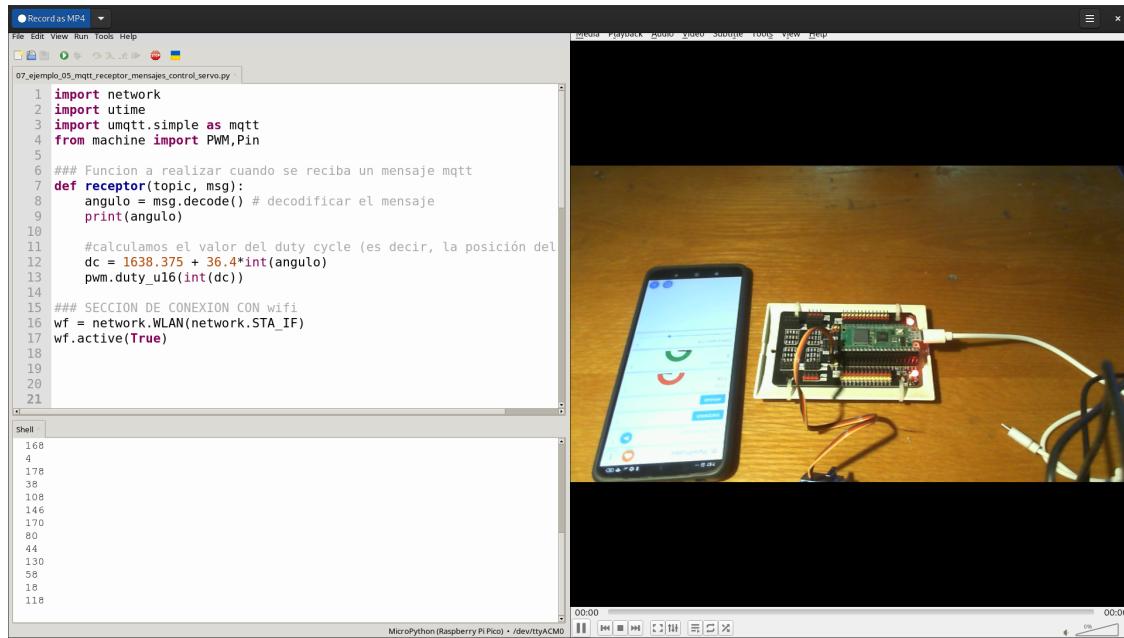
#define la funcion a realizar cuando recibimos un mensaje
c.set_callback(receptor)
```

```

#conectamos y suscribimos
c.connect()
c.subscribe('prueba')
#ciclamos la recepcion de mensajes
print("Ejecutando el suscriptor")
print("Esperando mensajes...")
while True:
    c.wait_msg()

```

Imagen funcionamiento



Your browser does not support the video tag.