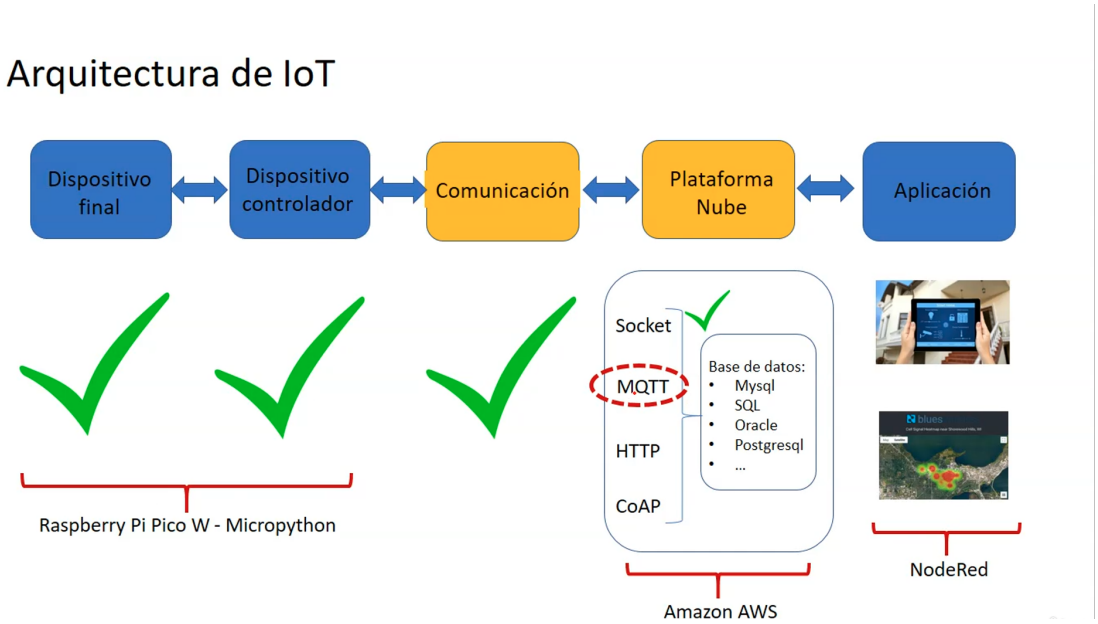
01_Manejo MQTT

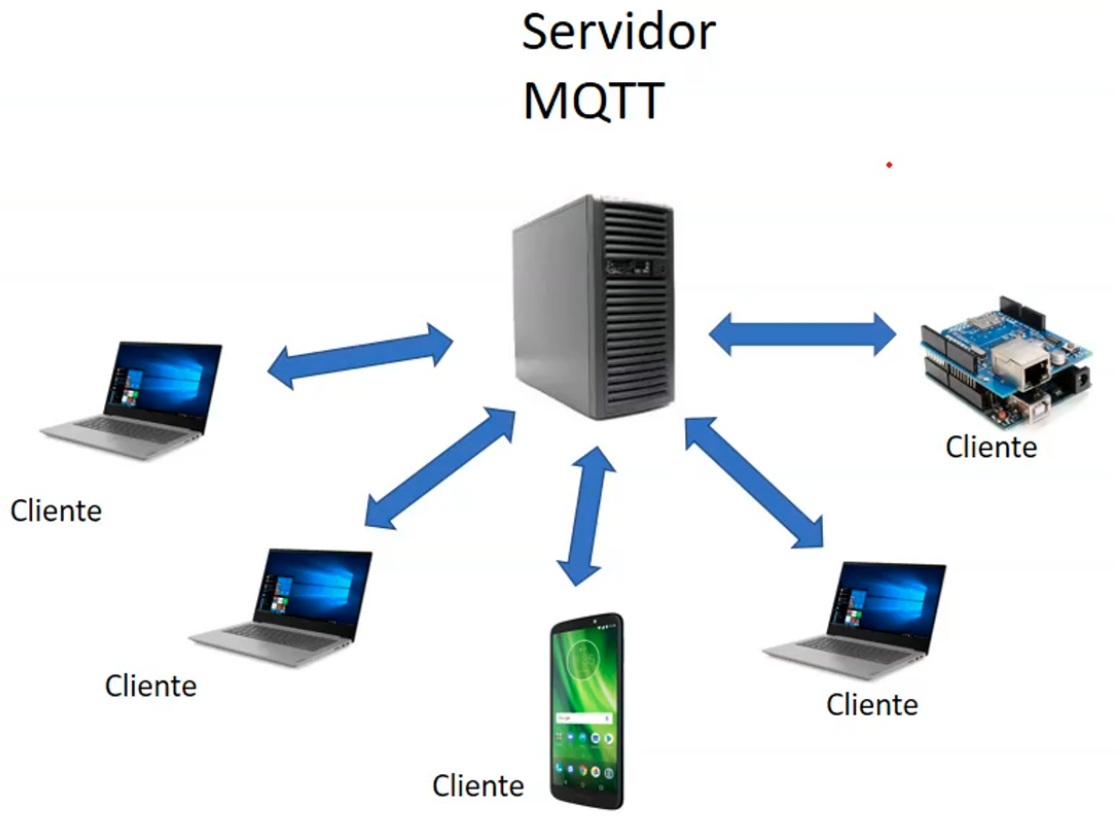
March 7, 2025

0.1 Arquitectura IoT Protocolo MQTT



0.1.1 Conceptos

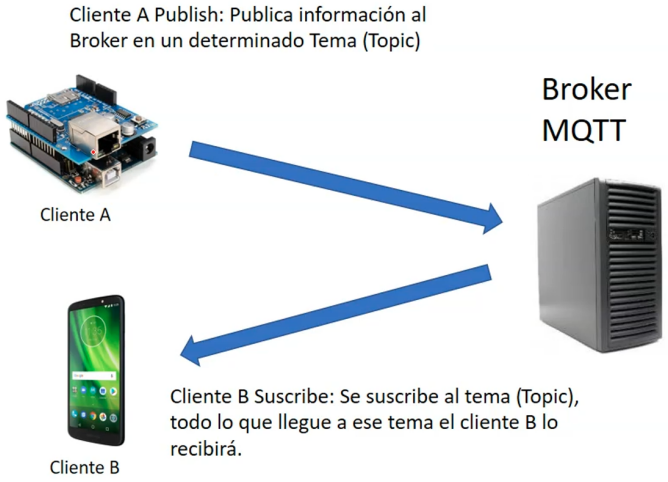
- Funciona sobre TCP
- Usa el puerto **1883**
- Puede usar usuario y password para el mensaje
- Usa SSL/TLS para seguridad



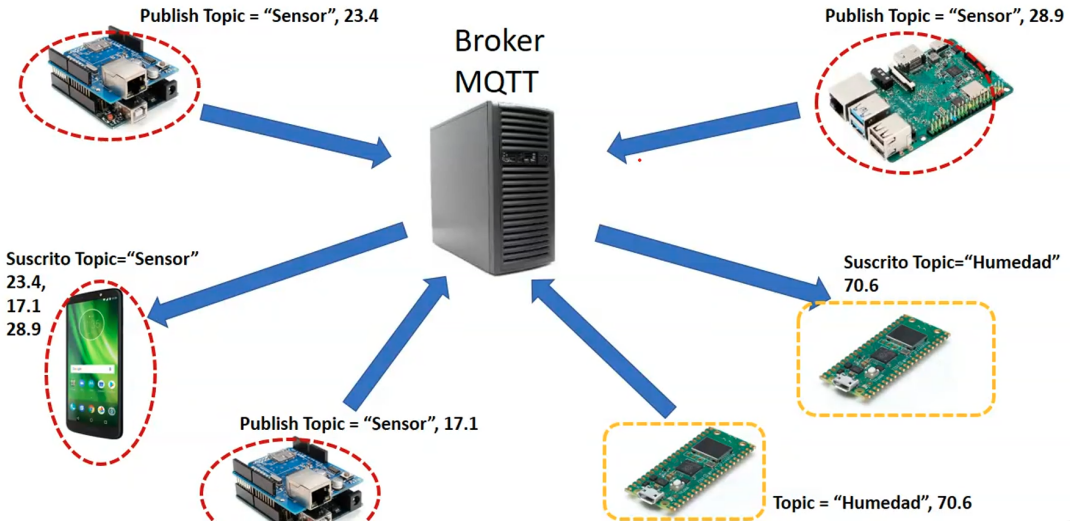
0.1.2 Elementos clave

- 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 terminos 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.

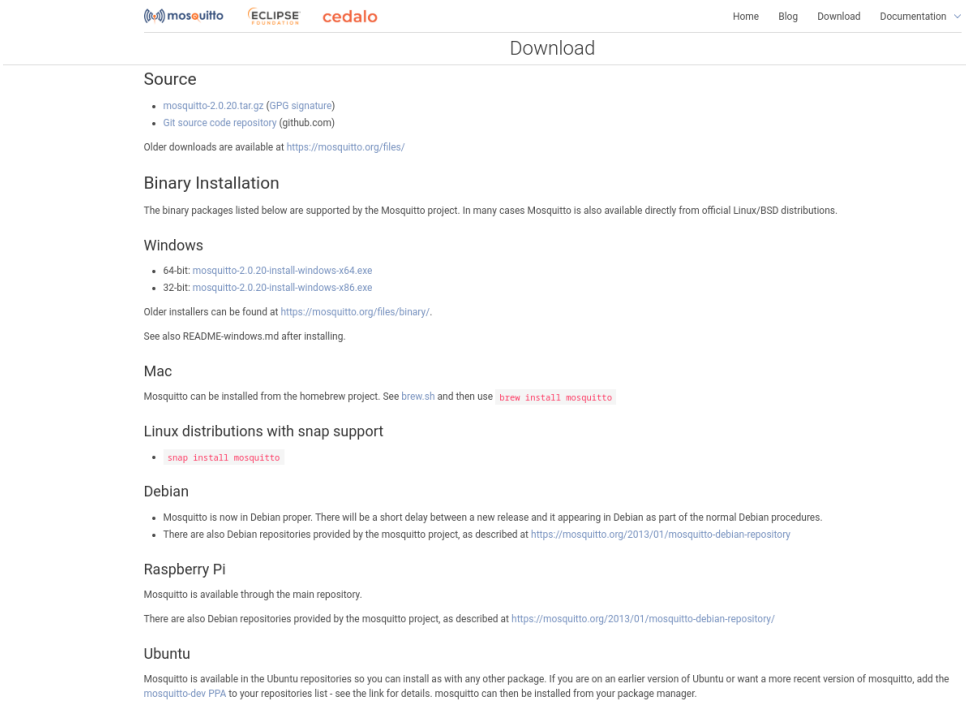


El cliente **A** publica en un topic, por ejemplo el topic **prueba**, de tal forma que el cliente **B** se subscribe al topic **prueba** y de esta manera, todo lo que el cliente **A** publique al topic **prueba** (cada vez) le llegará al cliente **B** y a todos los sobscriptores del topic **prueba**.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



mosquitto ECLIPSE FOUNDATION cedalo

Home Blog Download Documentation

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_anonimous 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 sonbre 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 subscripción 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.

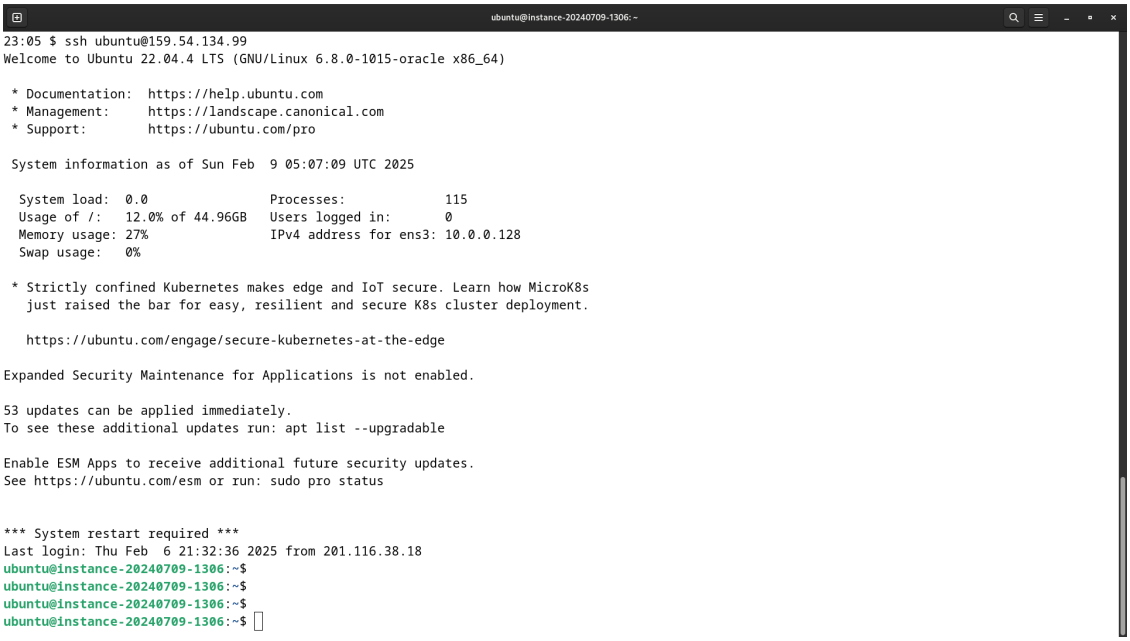
0.3 Implementación de MQTT en la nube

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



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.

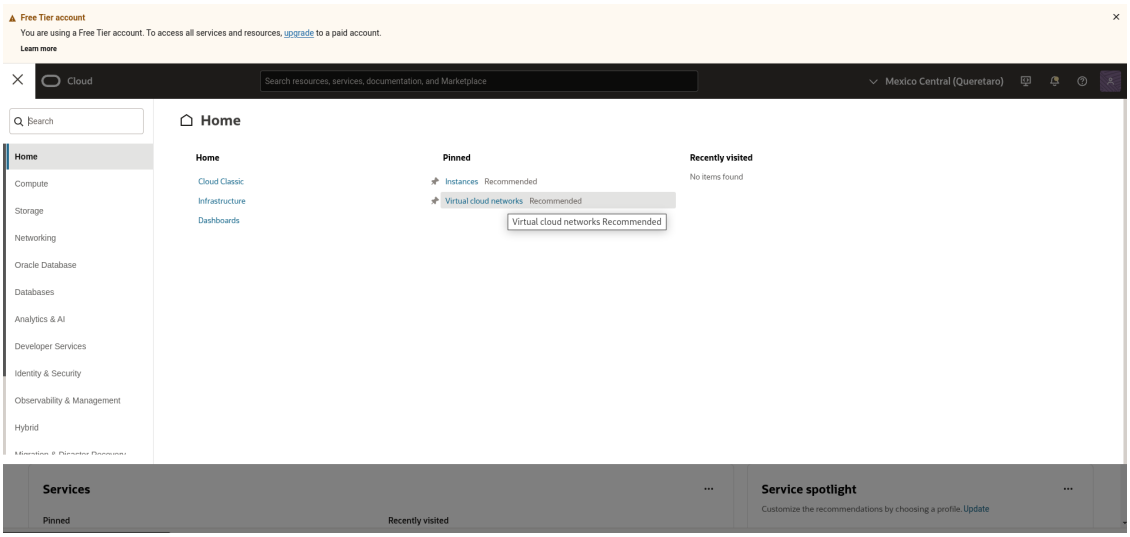
Desde la consola de una maquina diferente que tenga instalado mosquitto, vamos a ejecutar lo siguiente:

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

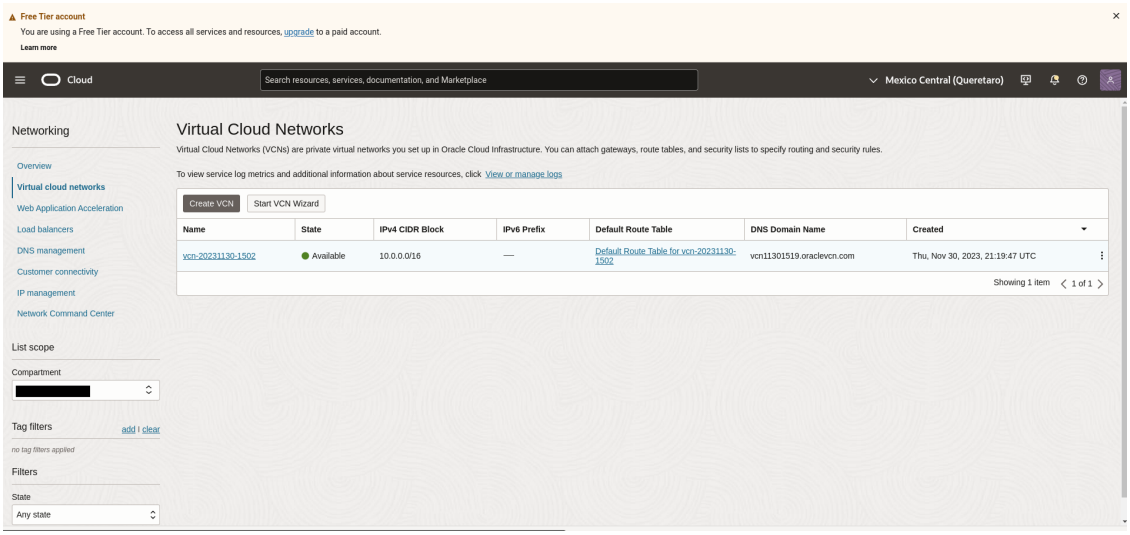
Como se puede observar, ahora da un error, puesto que no se puede conectar ya que la seguridad de la nube de oracle y el firewall lo está bloqueando. Para resolver lo anterior hay que **Abrir puertos**

Loguearse en la cuenta de cloud.oracle.com.

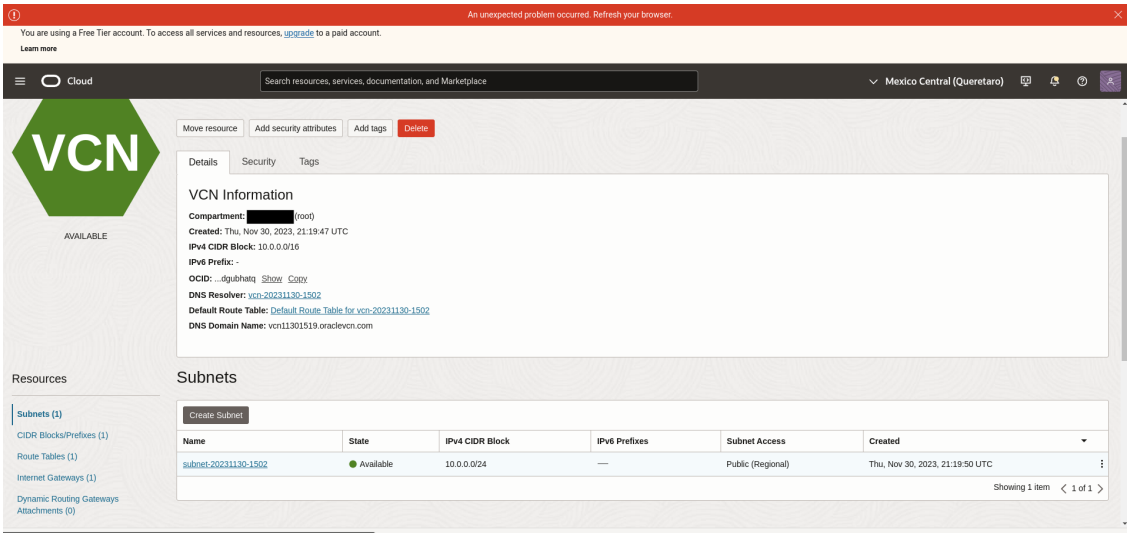
Seleccionar ”Dashboard.”



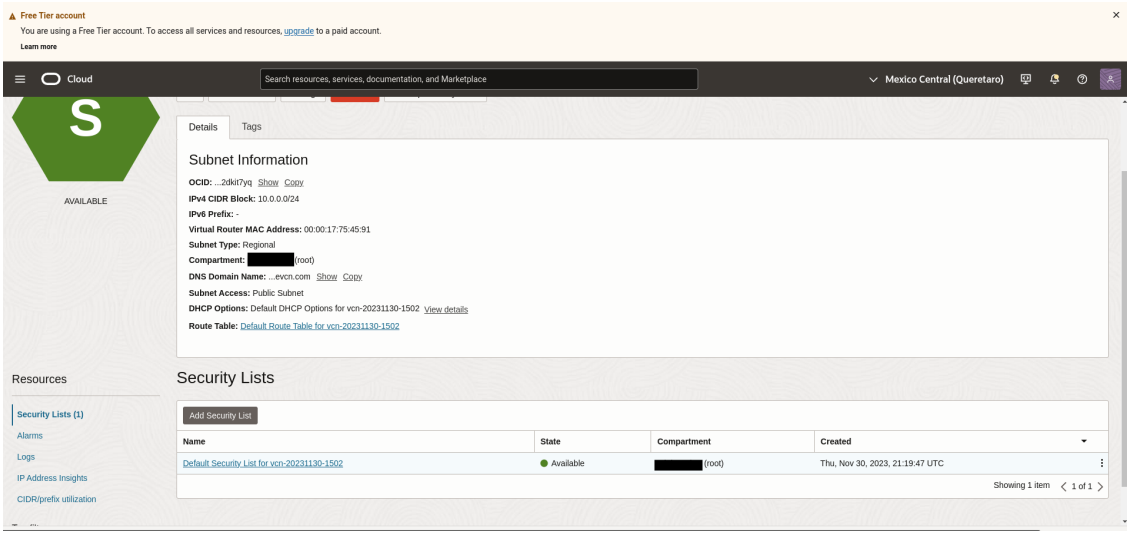
Seleccionar el virtual cloud network.



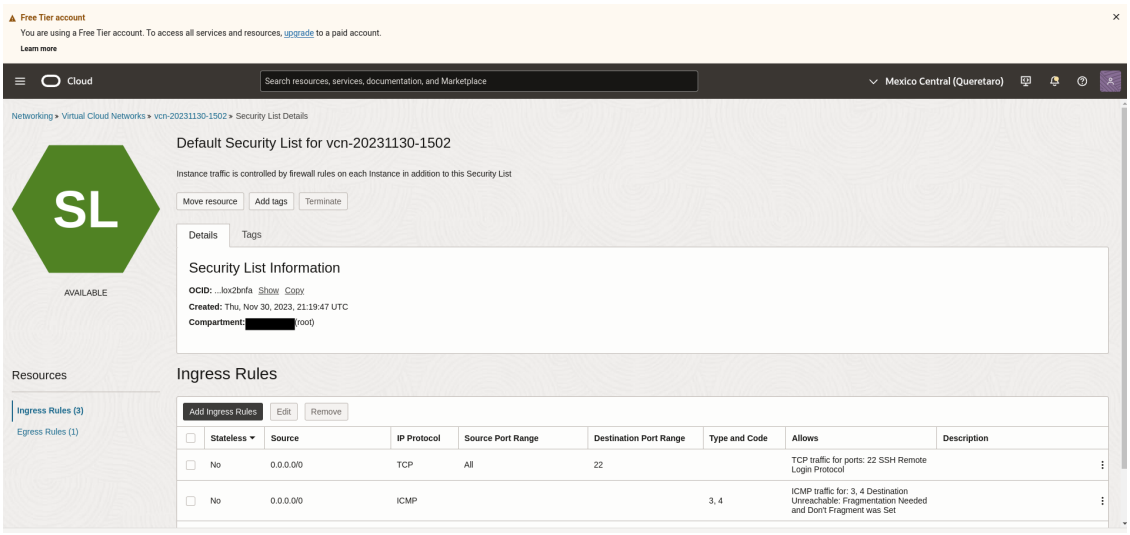
Seleccionar la subnet correspondiente.



Seleccionar el security list.

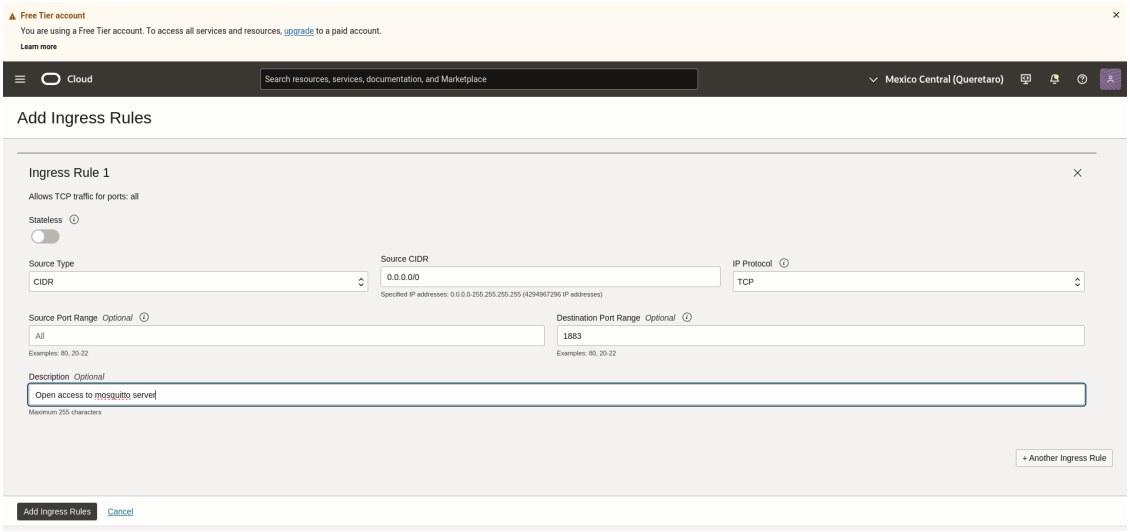


Seleccionar ”Add Ingress Rules”



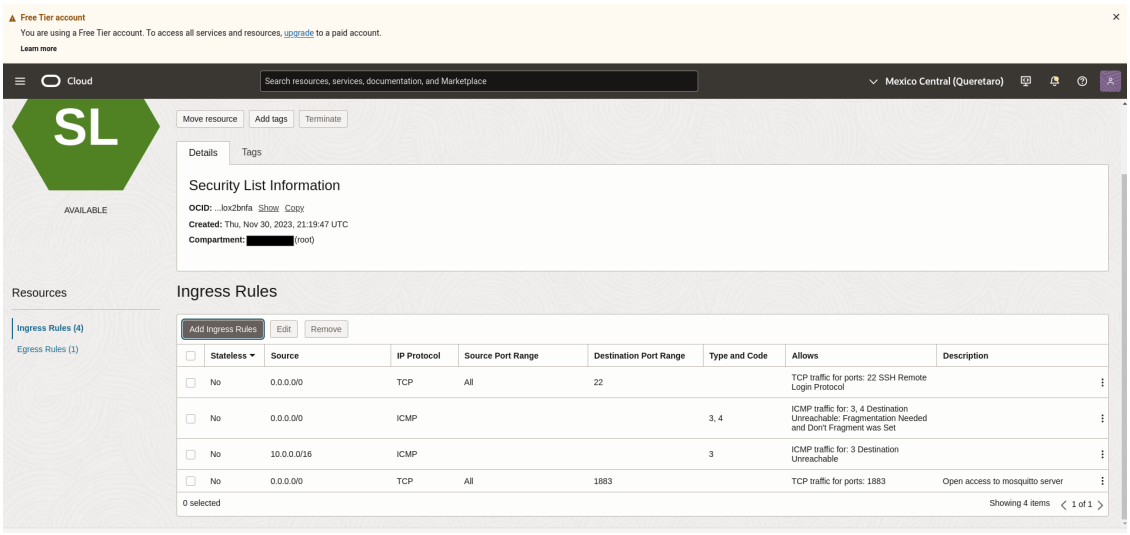
Realizar lo siguiente:

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,



7. Guarde la nueva regla de ingreso. Damos clck en el boton de ”Add Ingress Rules”

Ahora esa regla debe aparecer en el dashboard



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:

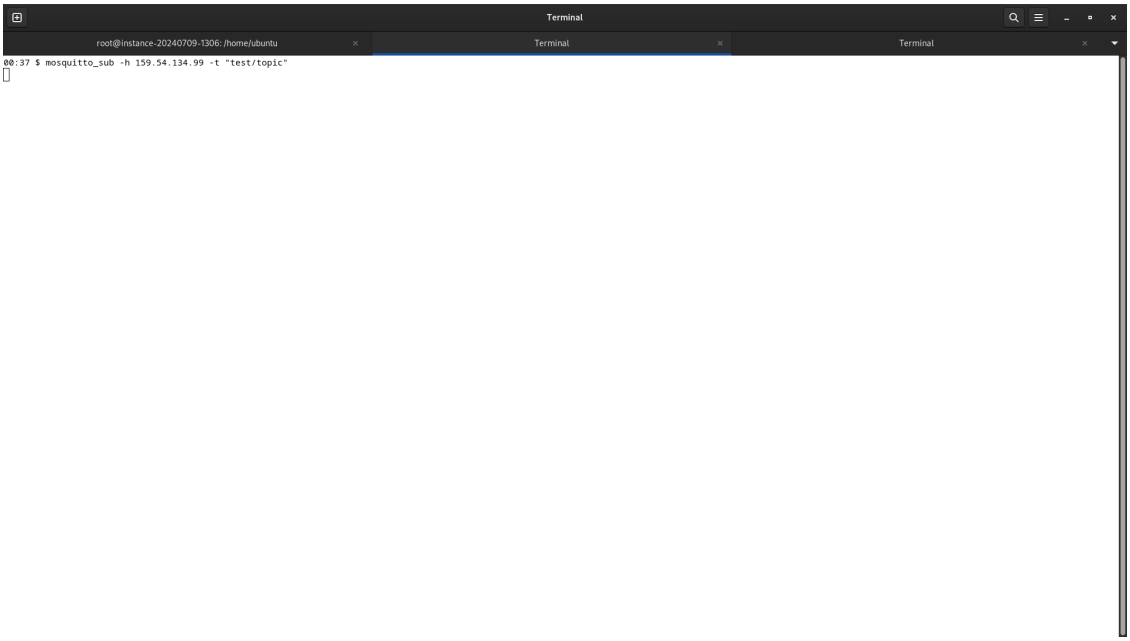
firewall-cmd --permanent --zone=public --add-port=1883/tcp; firewall-cmd --reload

)

Hacemos la prueba desde otra consola en otra maquina:

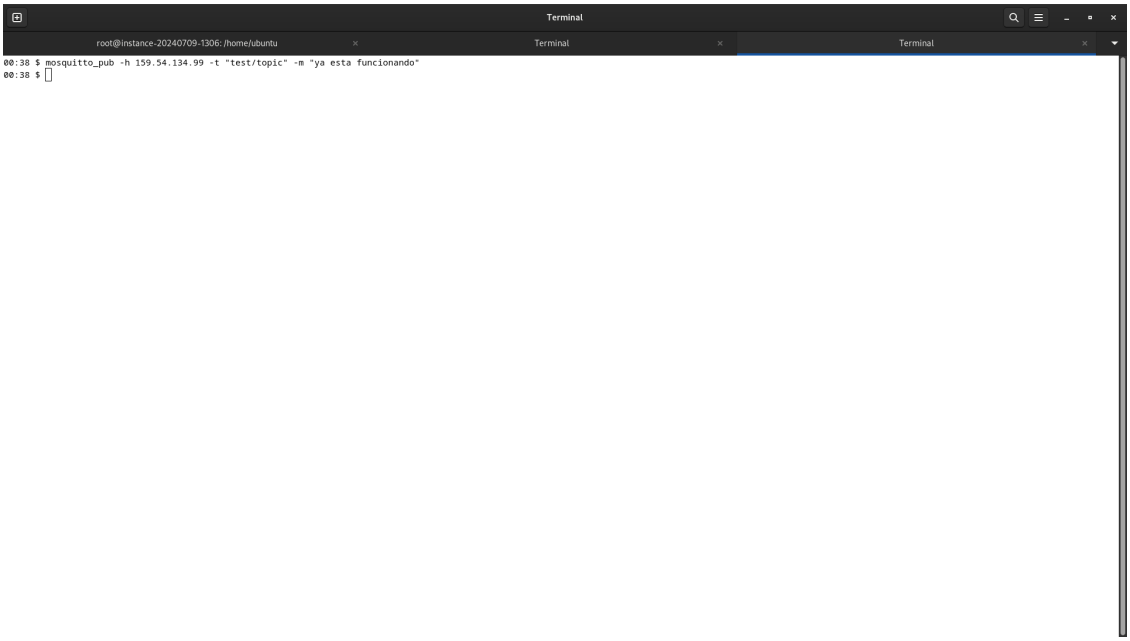
00:26 $ mosquitto_sub -h 159.54.134.99 -t ”test/topic”

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



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

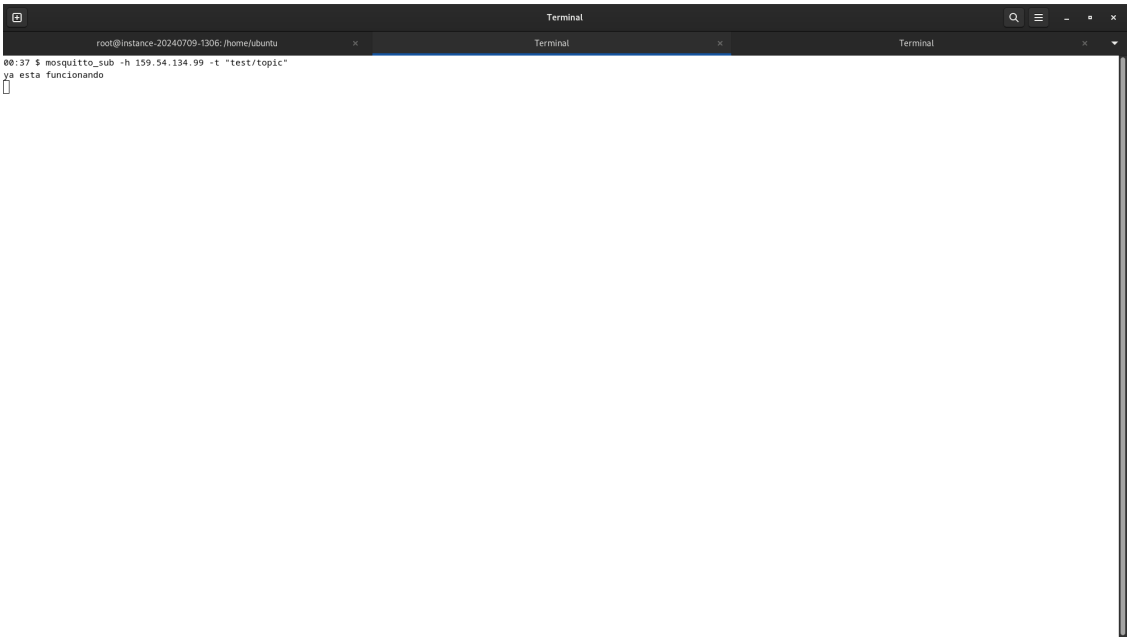
00:26 $ mosquitto_pub -h 159.54.134.99 -t ”test/topic” -m ”ya esta funcionando” 00:27 $



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

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

ya esta funcionando



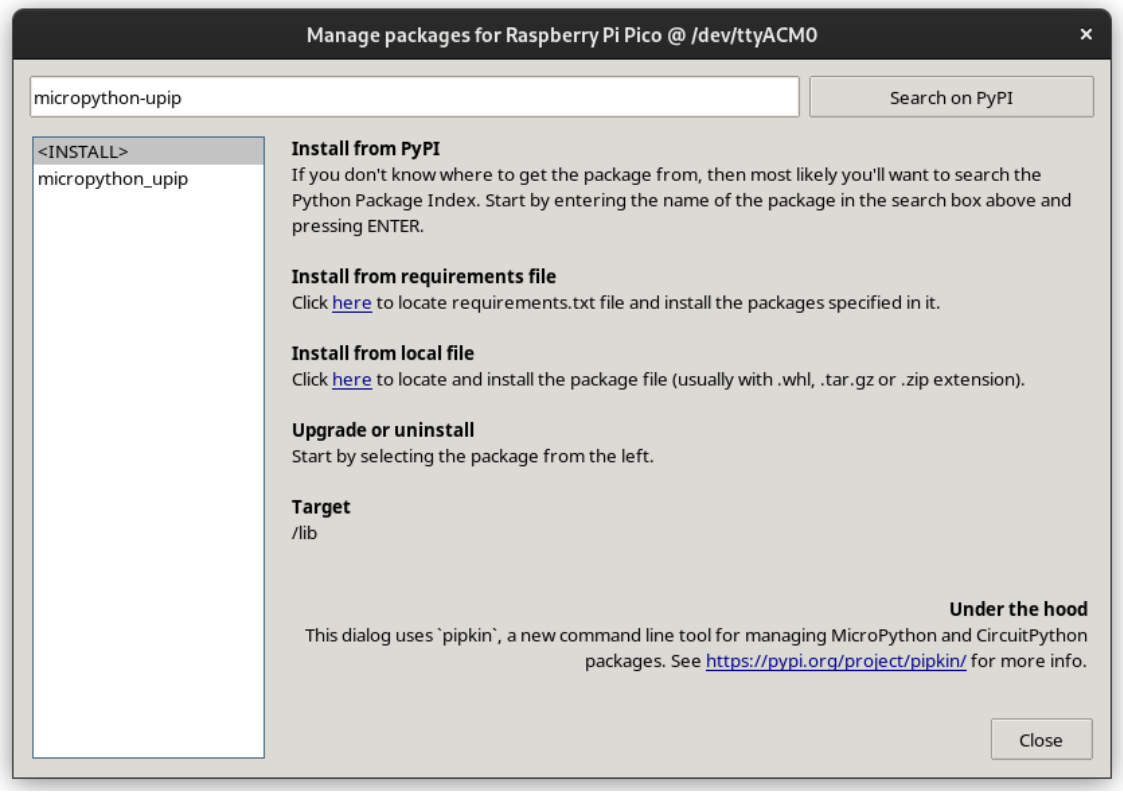
Dicha terminal seguirá en modo bloqueante esperando recibir mas mensaje y con eso aseguramos que el servicio está activo. Podemos terminar la subscripcion usando la combinación de teclas "Ctrl+c".

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.

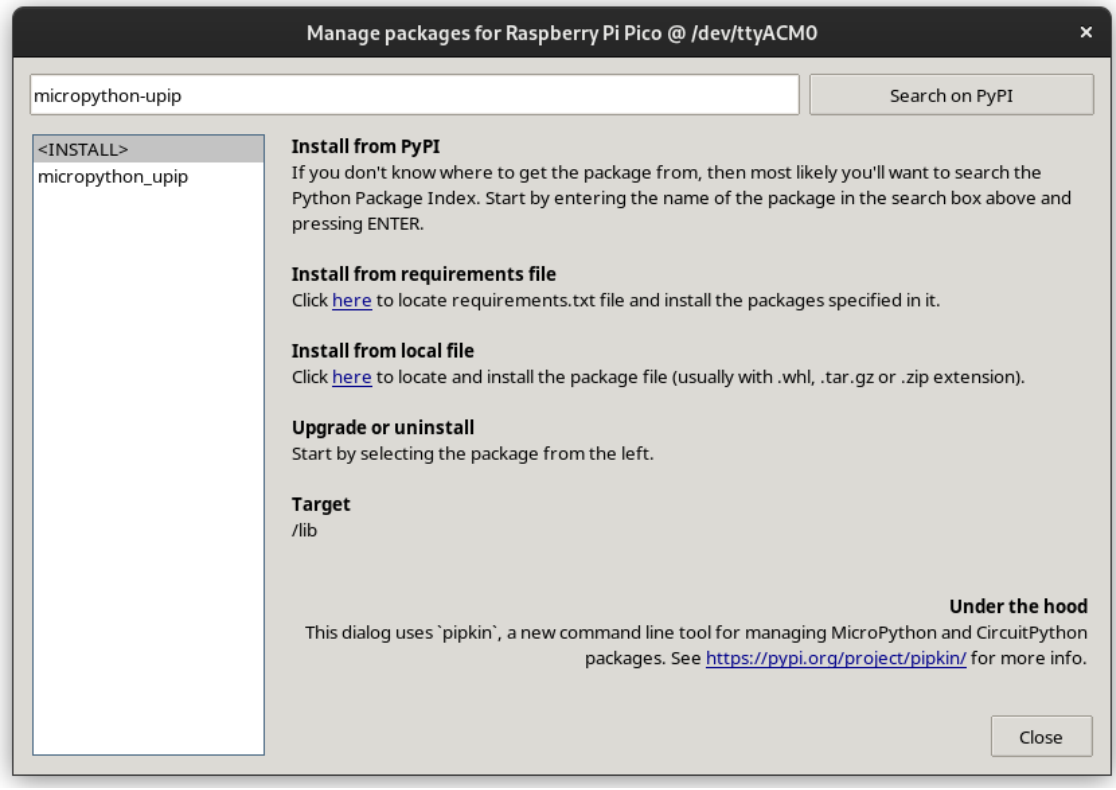
0.4 Instalar Libreria umqtt

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.



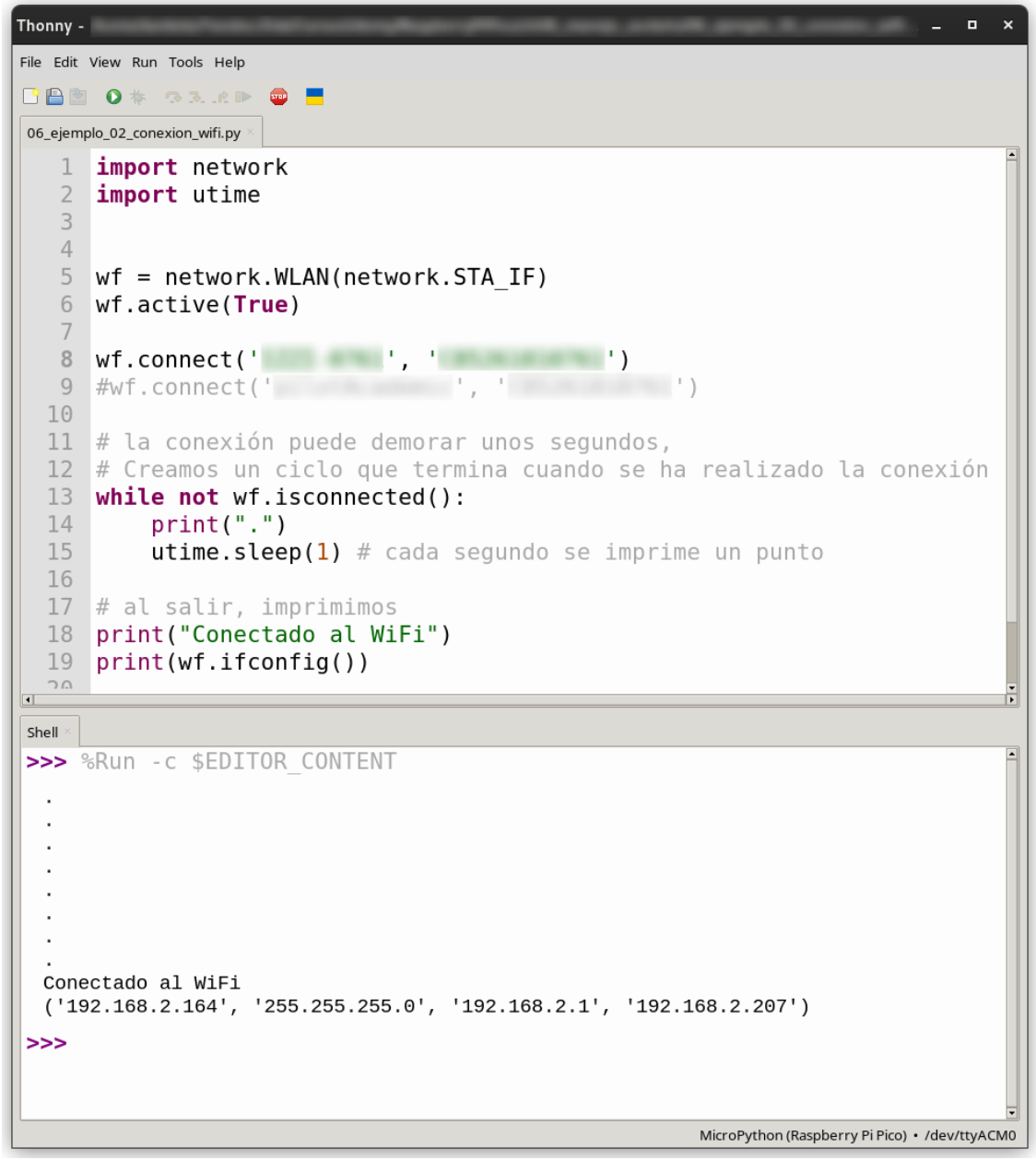
2. Hacer que el Raspberry Pico W tenga internet (libreria network), para lo cual ejecutamos el código del archivo **06_ejemplo_02_conexion_wifi.py**



ofuscar

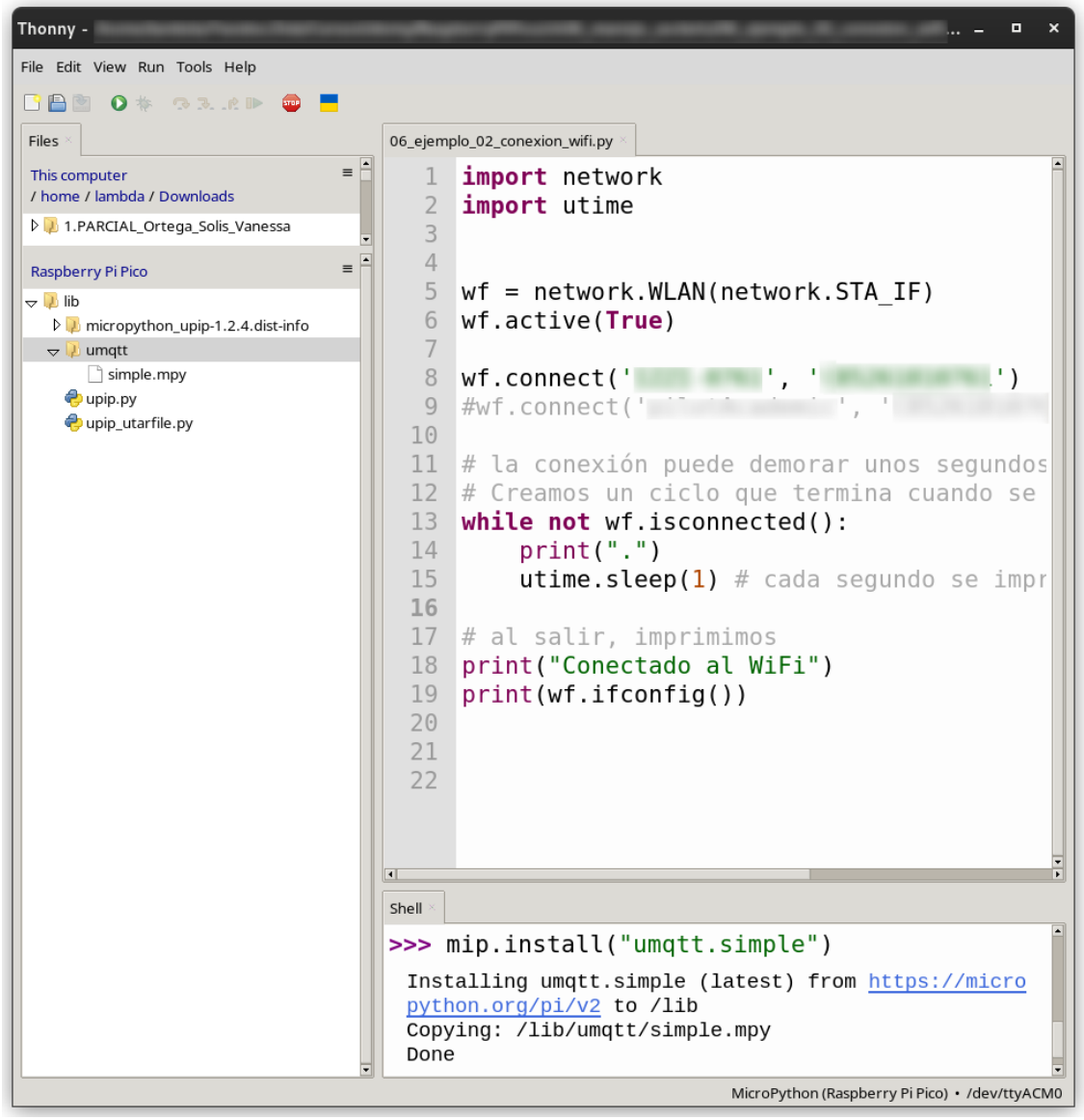
informacion de red en la imagen

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



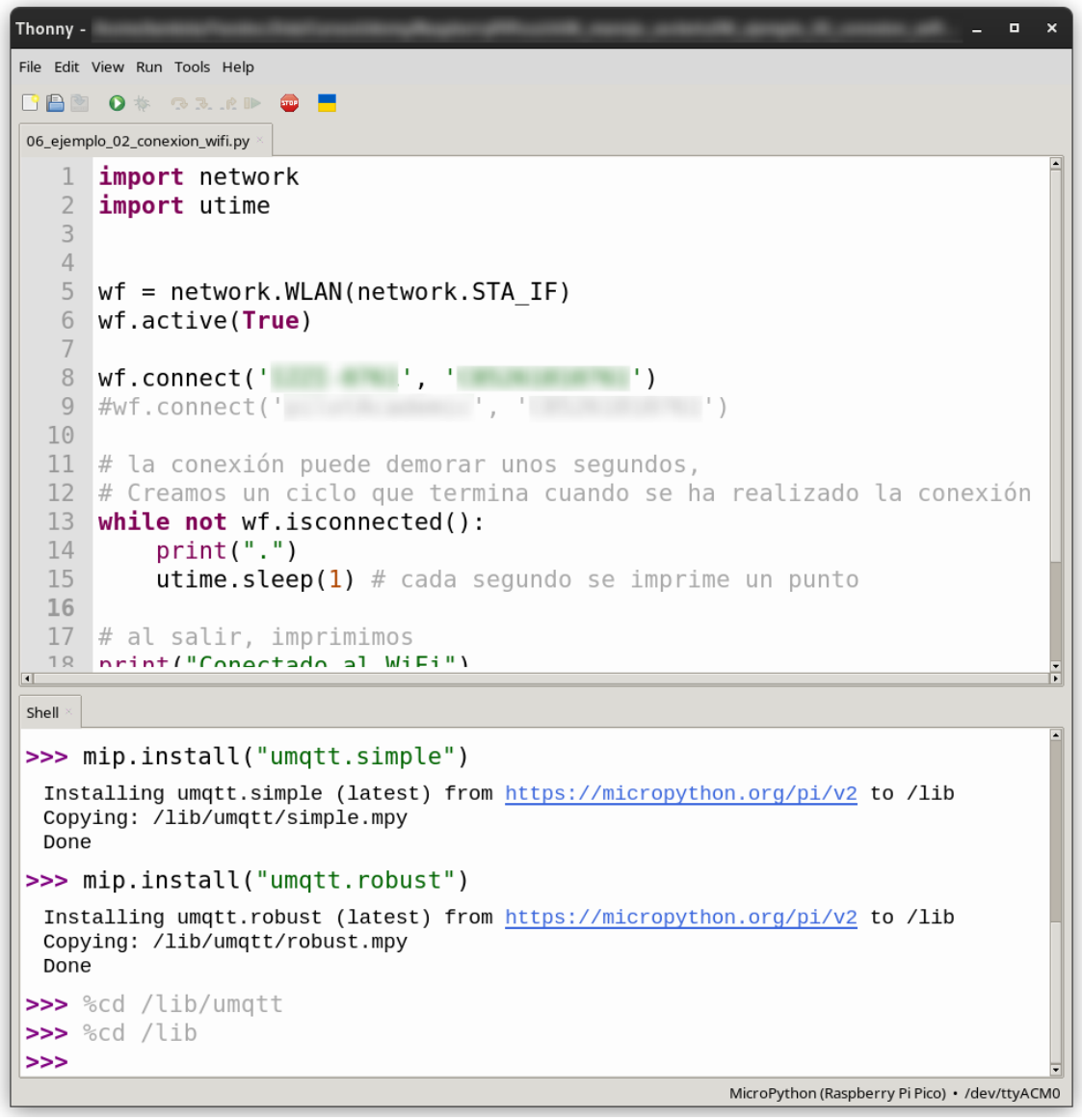
ofuscar informacion de red en la imagen

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



ofuscar informacion de red en la imagen

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



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 quec
```

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(".")
```

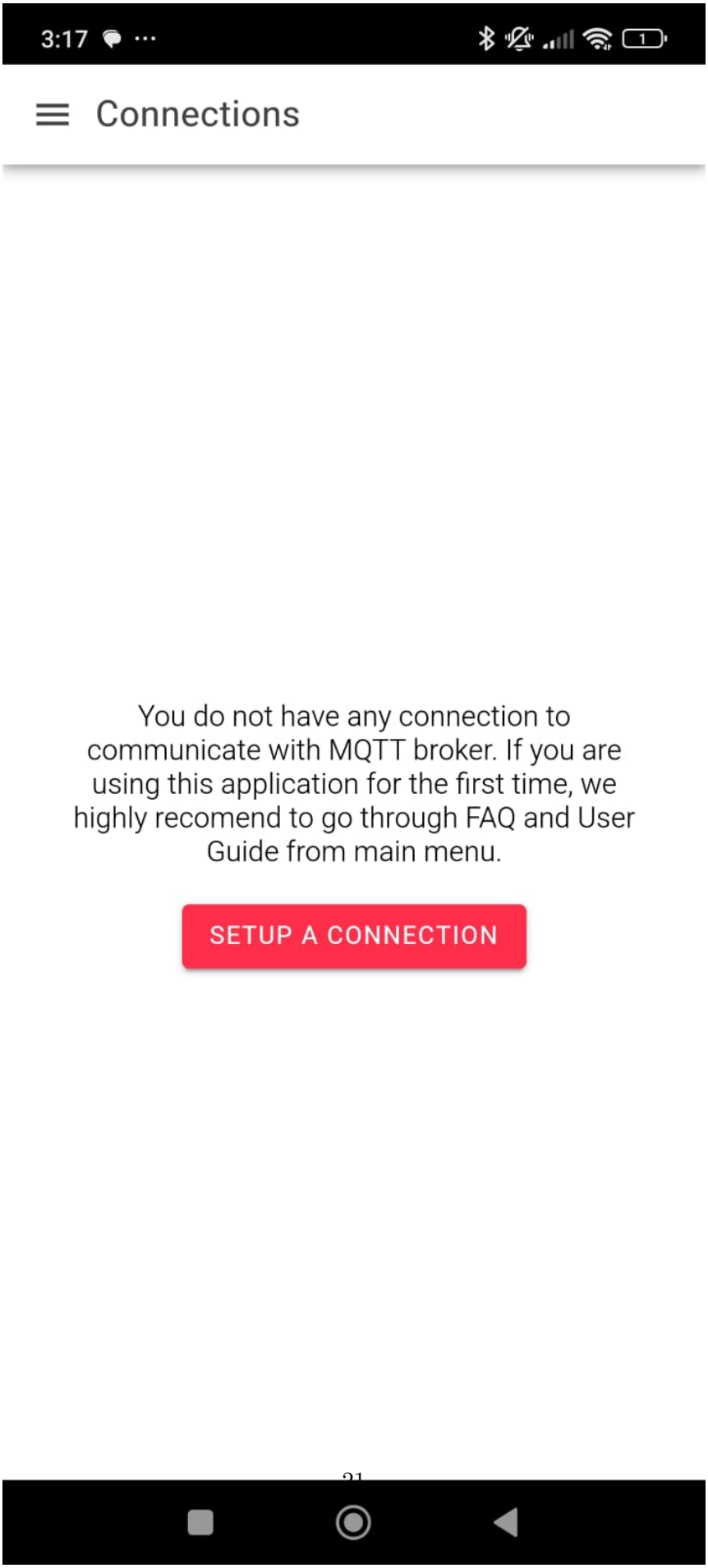
```
    utime.sleep(1) # cada segundo se imprime un punto

# 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=mqtt.MQTTClient("pico1", "159.54.134.99")
#definimos 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.

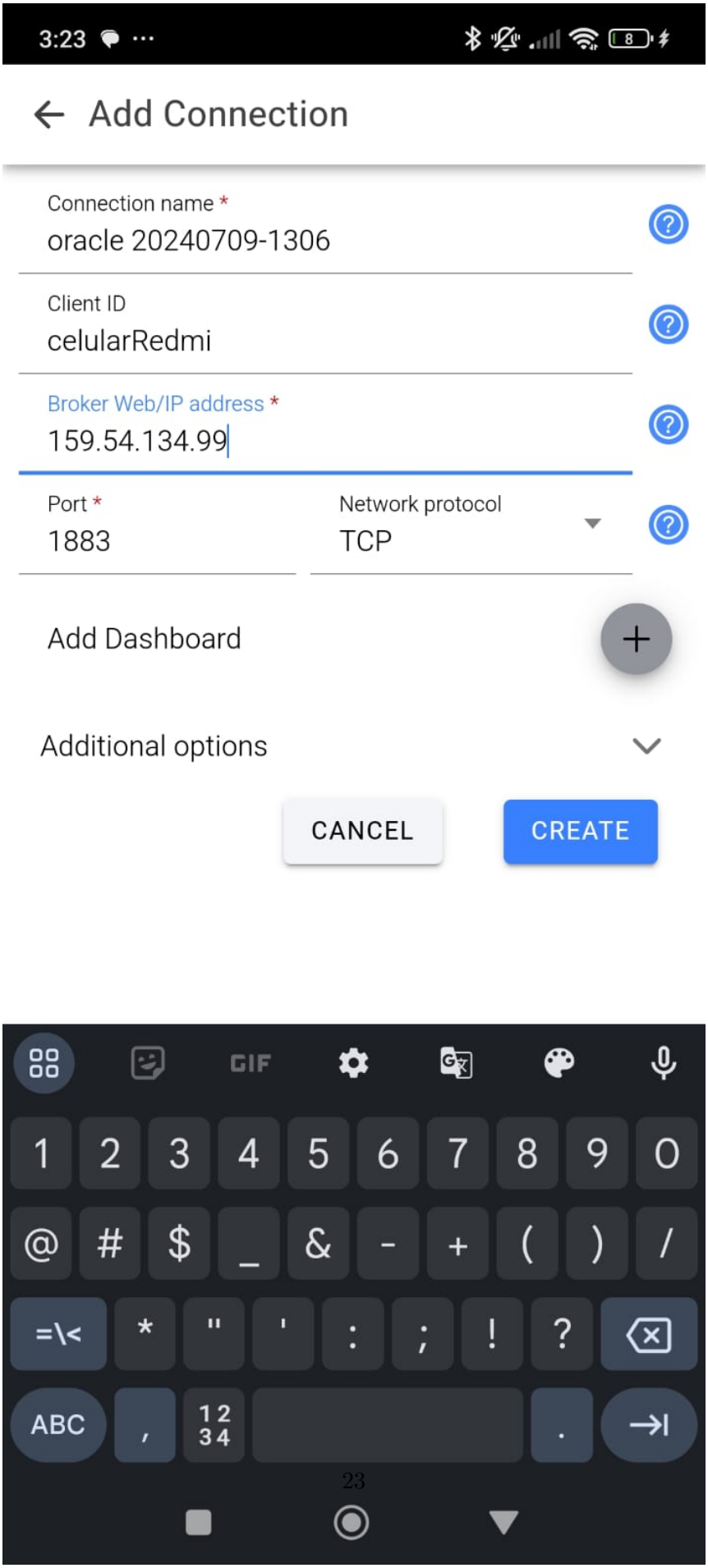
0.6 Probar desde el teléfono

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

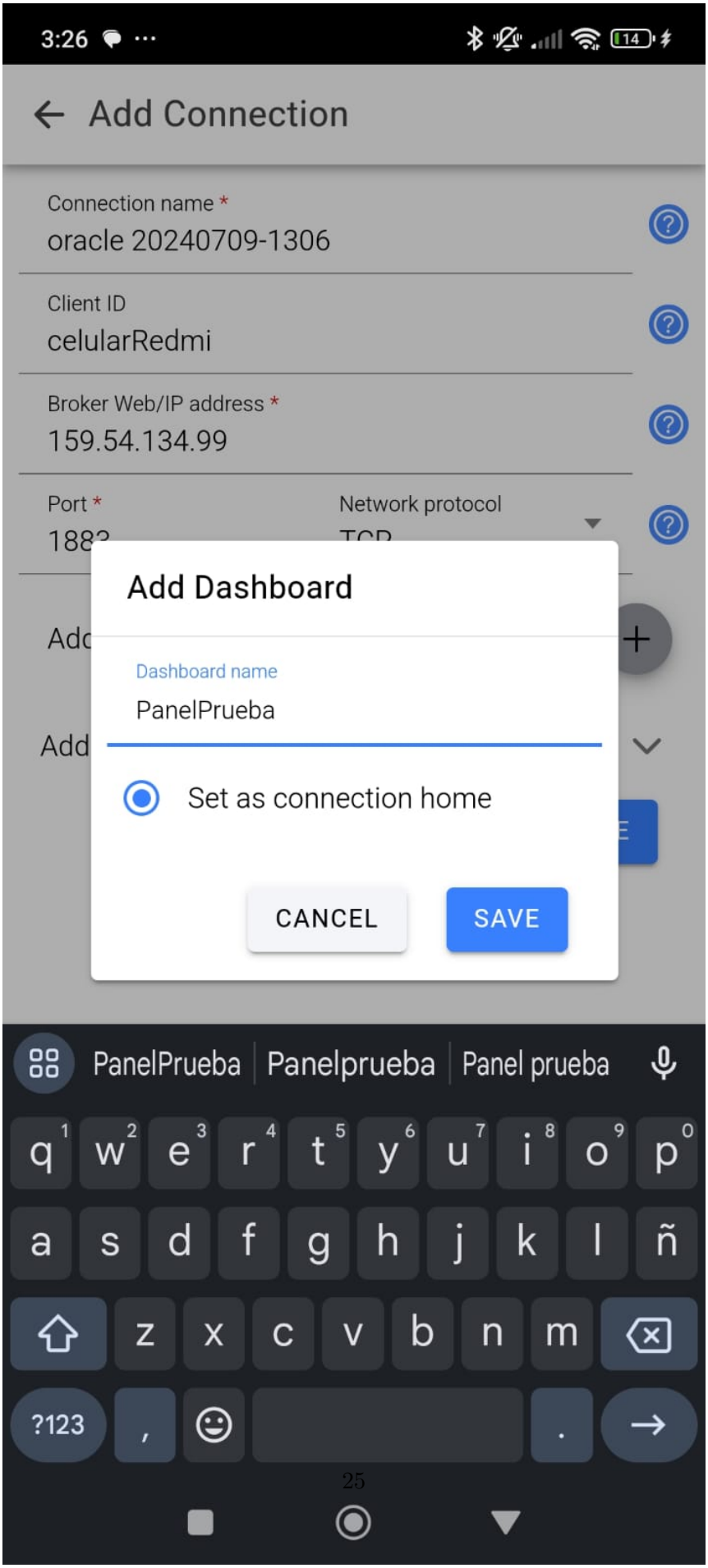
Al abrir la aplicación indica que no hay ninguna conexion


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 recomend 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
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

Se debe dar clic en agregar un dashboard

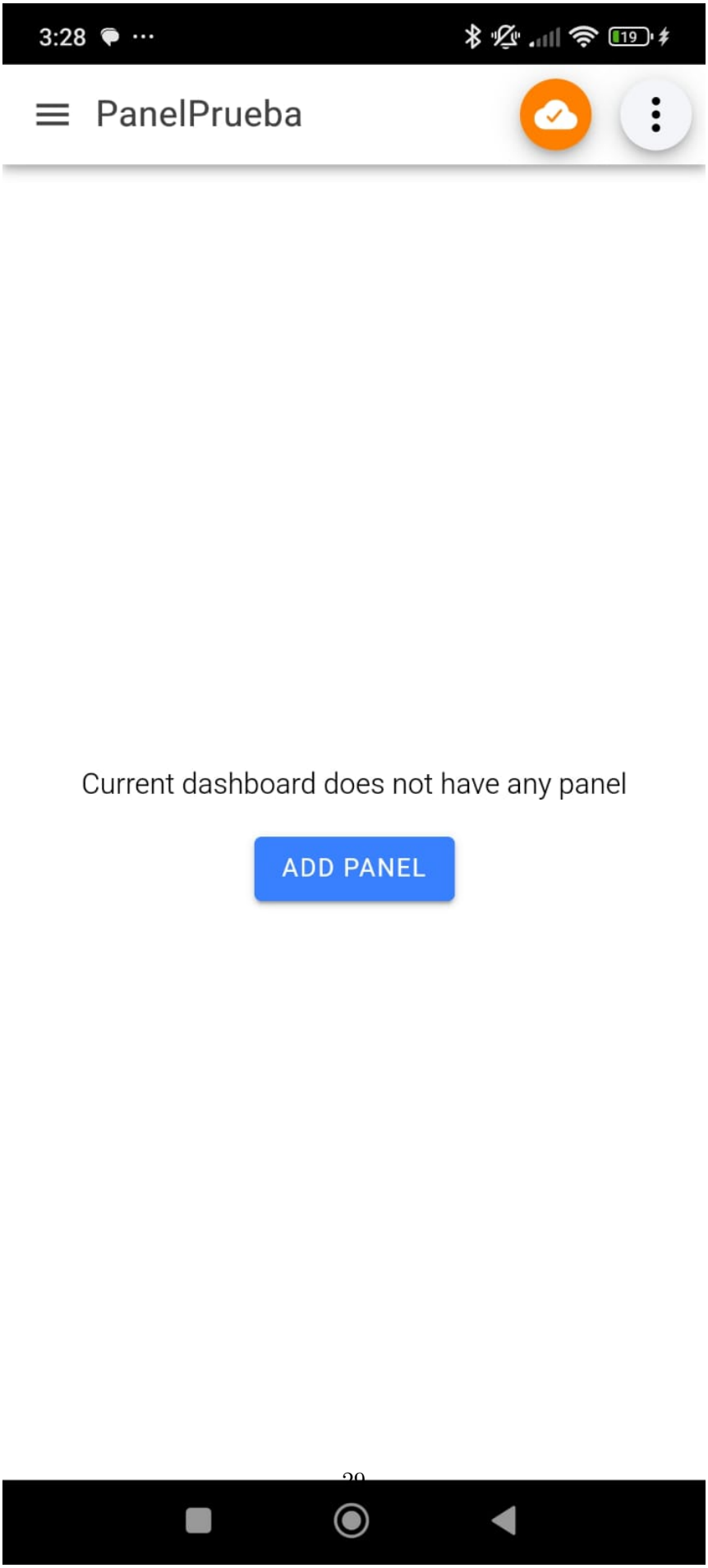

3:26
Add Connection
Connection name *
oracle 20240709-1306
Client ID
celularRedmi
Broker Web/IP address *
159.54.134.99
Port *
Network protocol
Add Dashboard
Dashboard name
PanelPrueba
Set as connection home
CANCEL
SAVE
PanelPrueba
Panelprueba
Panel prueba

Y dar clic en el botón de **CREATE**

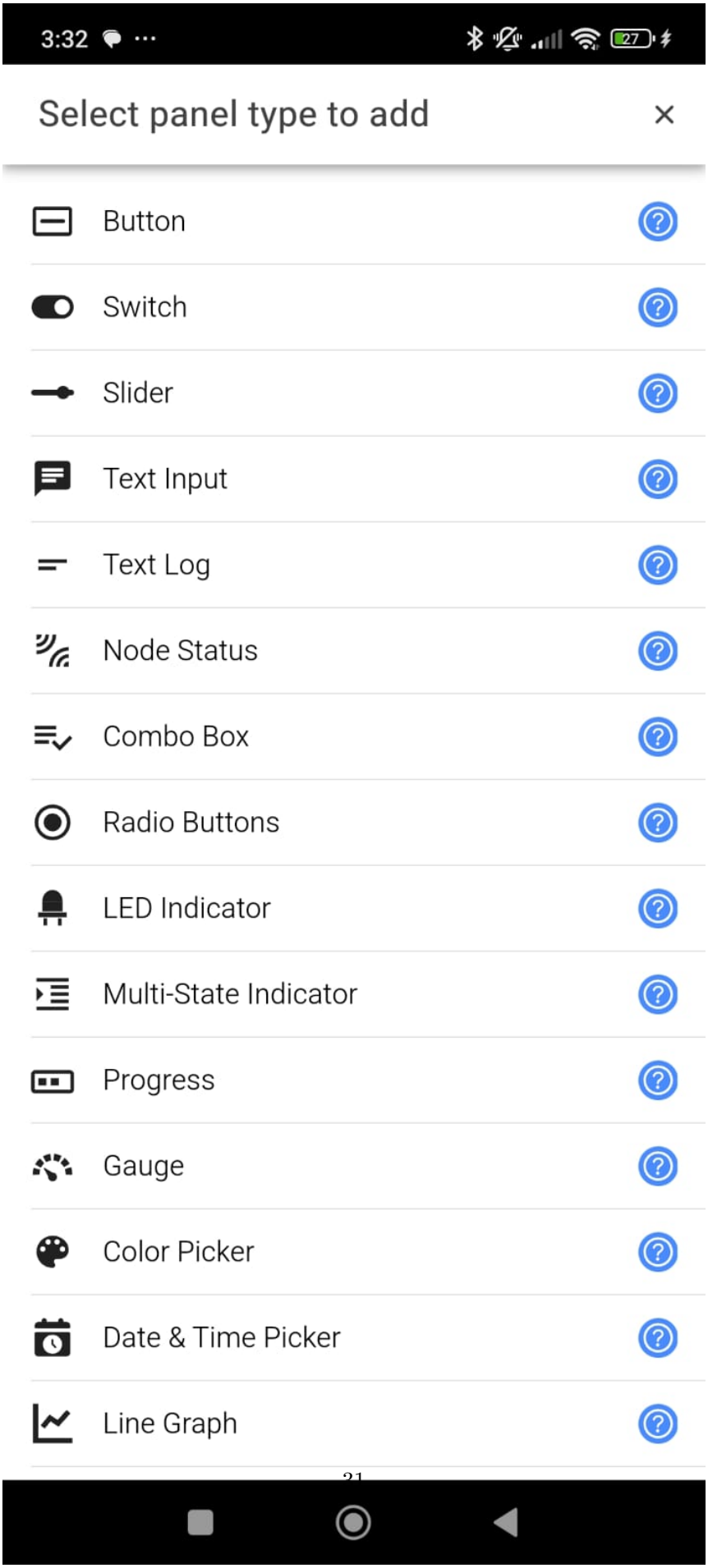
Cuando el simbolo es color naranja, significa que se ha establecido la conexión. Ahora entramos al dashboard


3:27
Connections
oracle 20240709-1306

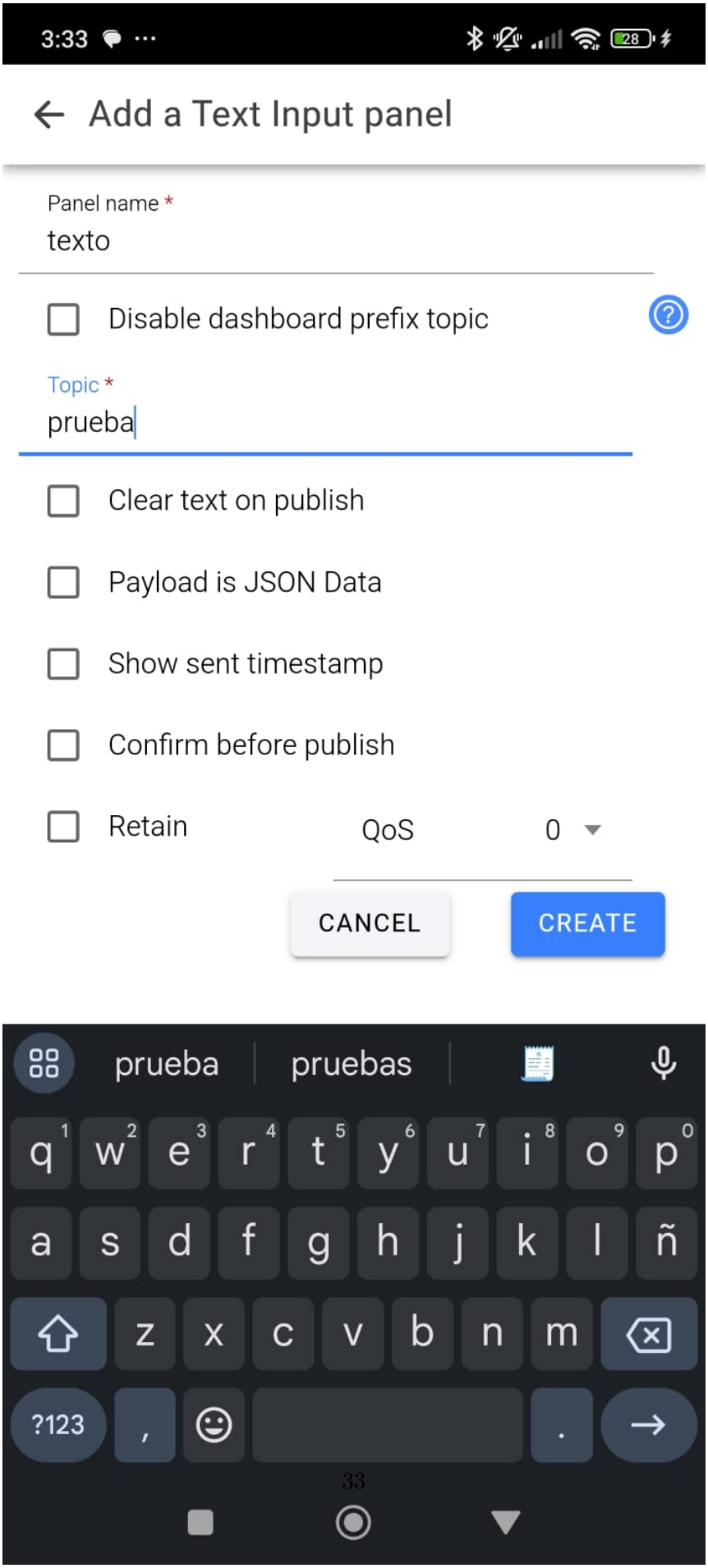
Agregar un panel


3:28
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

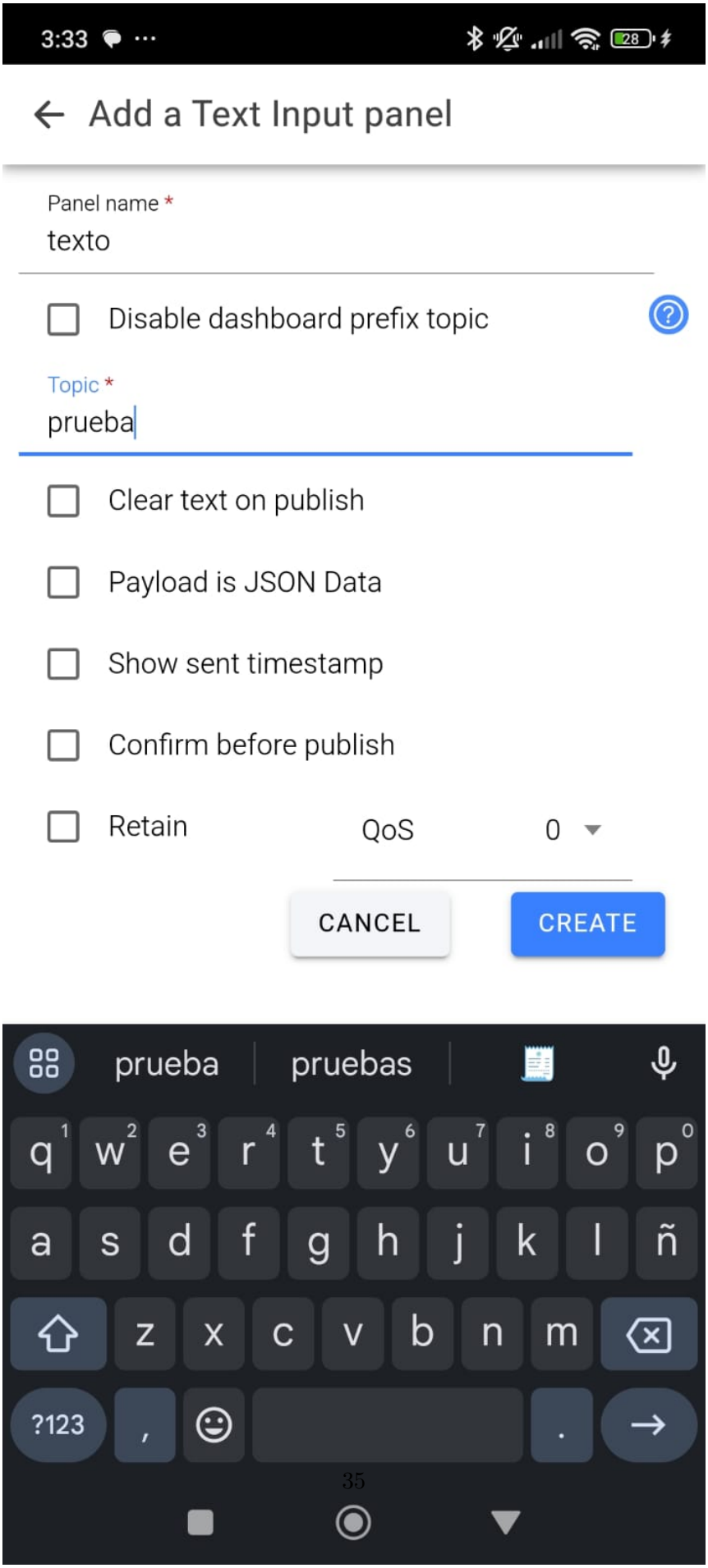

3:32
Select panel type to add
Button
Switch
Slider
Text Input
Text Log
Node Status
Combo Box
Radio Buttons
LED Indicator
Multi-State Indicator
Progress
Gauge
Color Picker
Date & Time Picker
Line Graph

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


3:33
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
prueba
pruebas

Y se selecciona el botón de **CREATE**

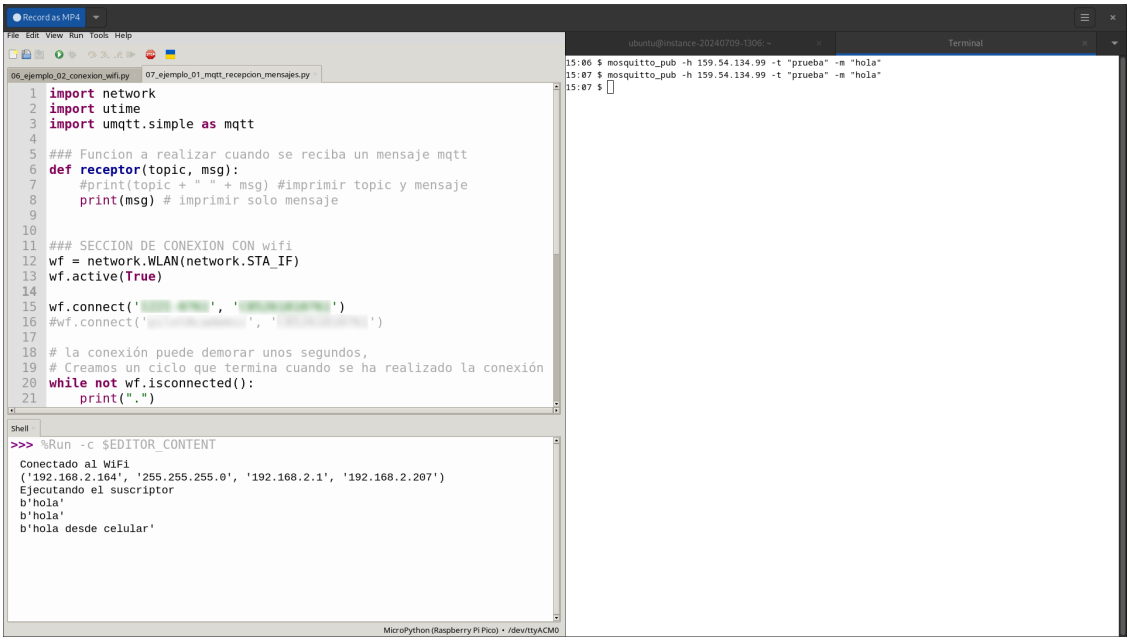
Escribimos el mensaje **hola desde celular** y lo enviamos


3:33
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
prueba
pruebas

Escribimos el mensaje a enviar


3:36
PanelPrueba
texto
hola desde celular

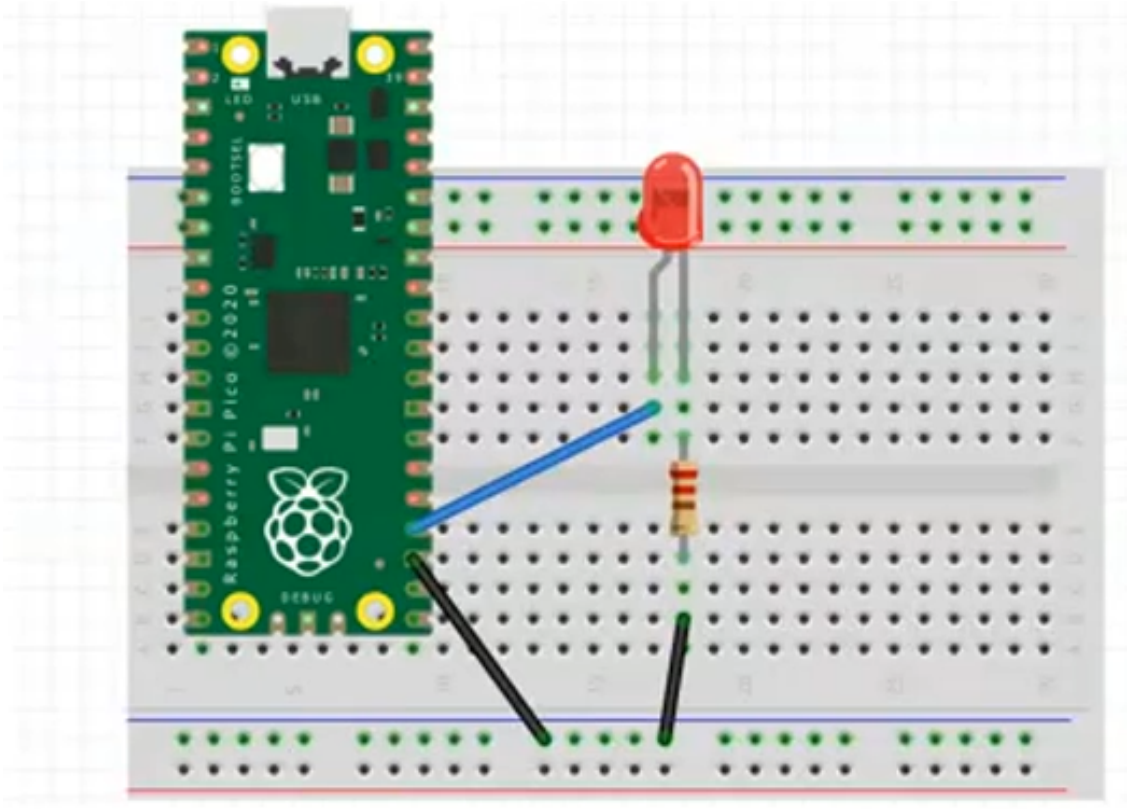
Y se verifica que haya sido recibido en la Raspberry Pi Pico W



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 boton ”Encender” y publicar B con el botón ”Apagar”



07_ejemplo_02_mqtt_encender_led.py

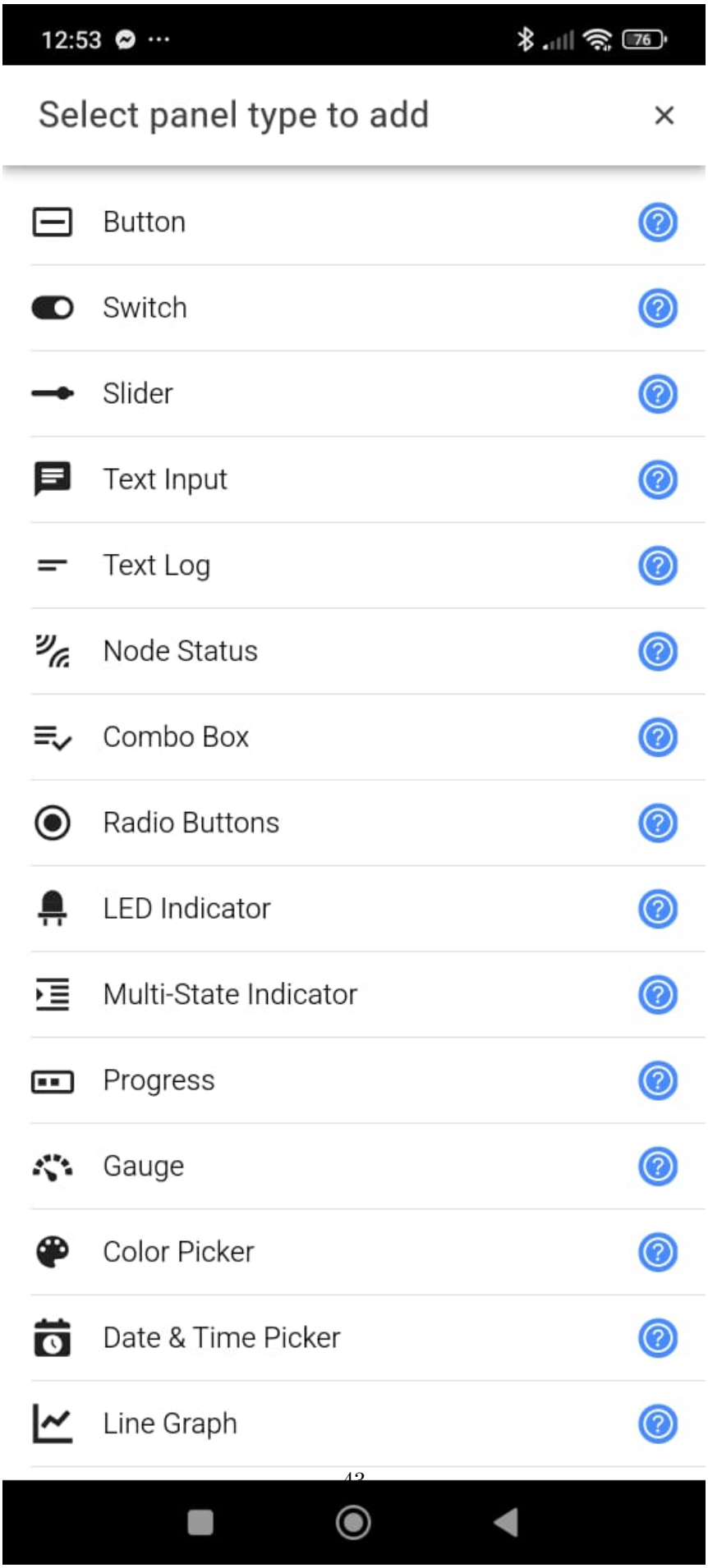
```
import network
import utime
import umqtt.simple as mqtt
from machine import Pin


### 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")
#definimos 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 mensaje")
while True:
    c.wait_msg()
```

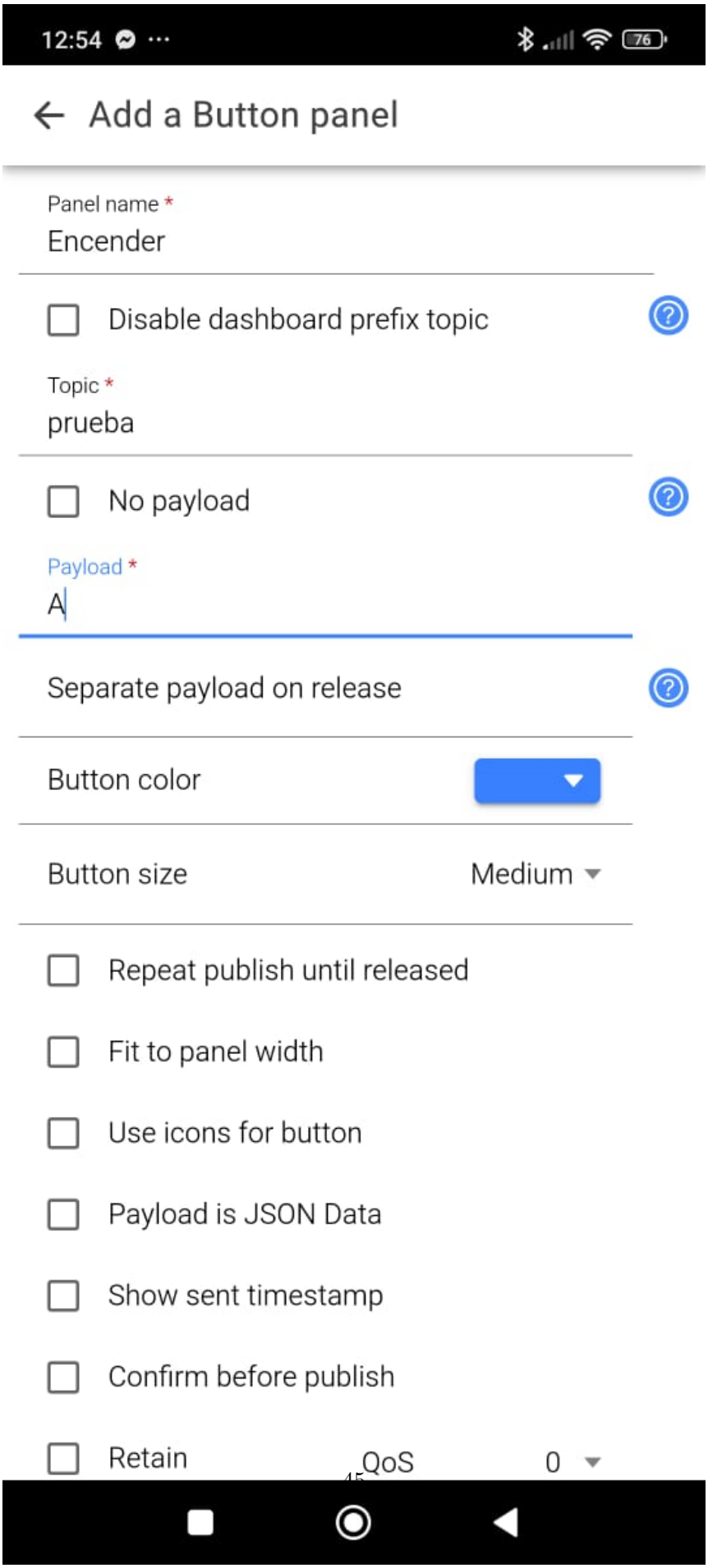
Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón +


12:53
PanelPrueba
texto
hola desde celular

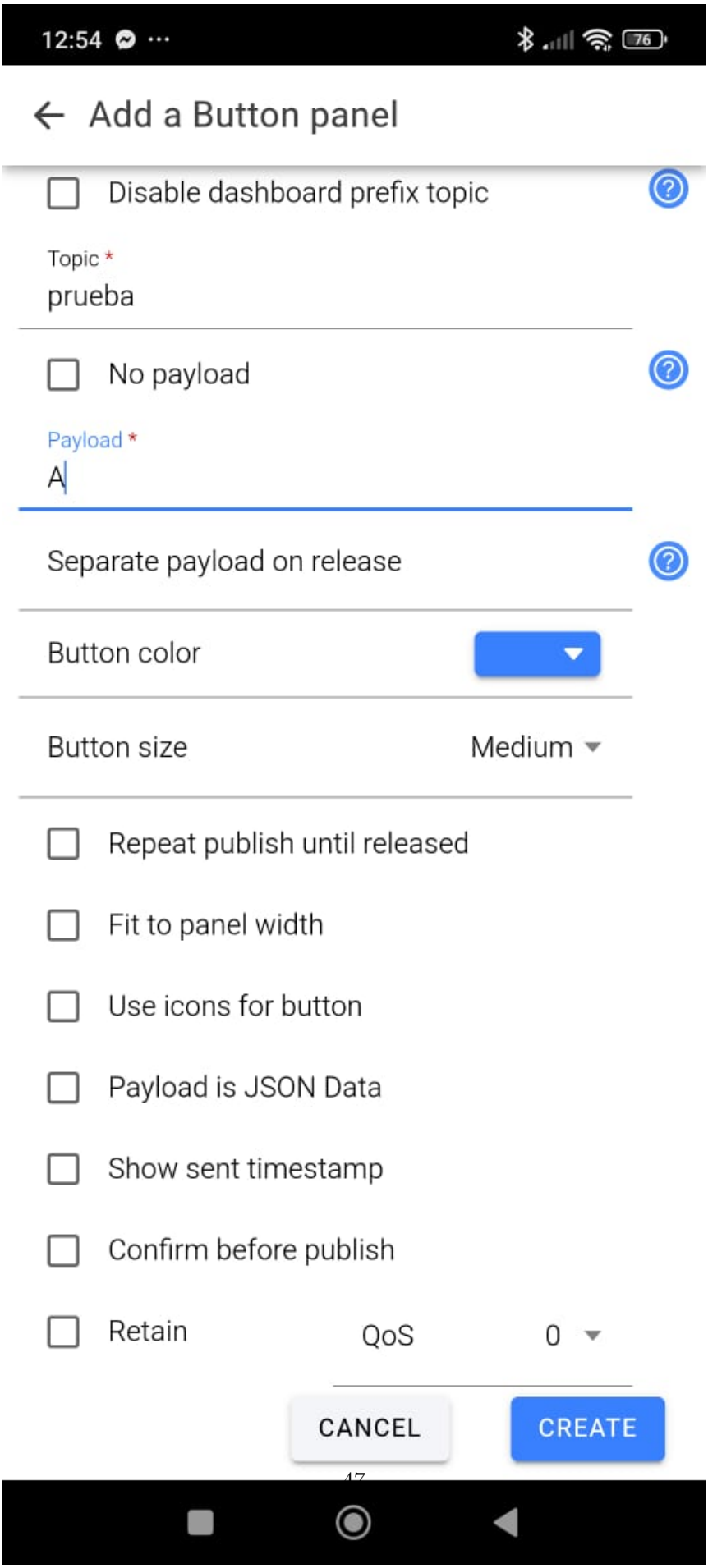
Selección **Button**


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**,

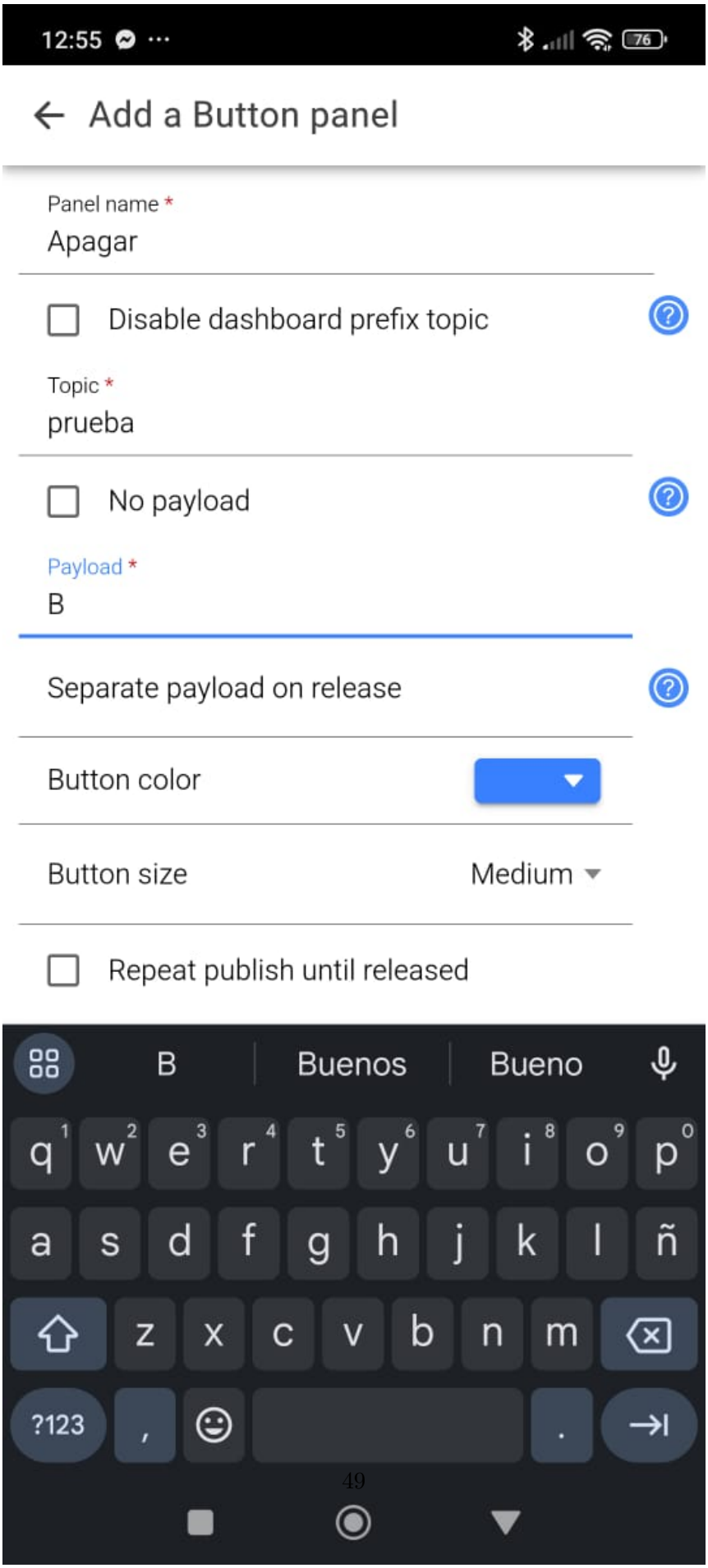

12:54
Add a Button panel
Panel name *
Encender
Disable dashboard prefix topic
Topic *
prueba
No payload
Payload *
A
Separate payload on release
Button color
Button size
Medium
Repeat publish until released
Fit to panel width
Use icons for button
Payload is JSON Data
Show sent timestamp
Confirm before publish
Retain
QoS
0

todos los demas atributos se dejan como estan, y posteriormente dar clic en **Aceptar**

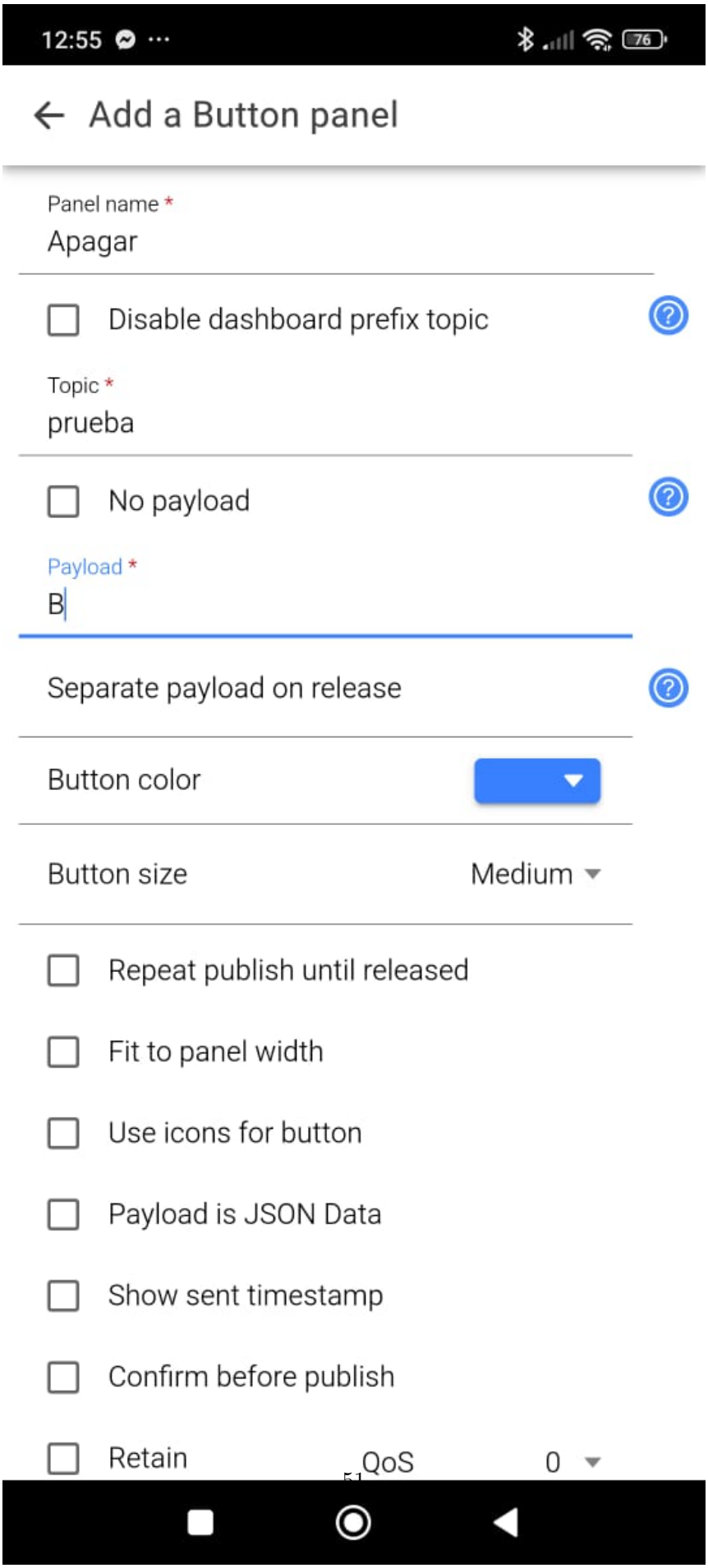

12:54
Add a Button panel
Disable dashboard prefix topic
Topic *
prueba
No payload
Payload *
A
Separate payload on release
Button color
Button size
Medium
Repeat publish until released
Fit to panel width
Use icons for button
Payload is JSON Data
Show sent timestamp
Confirm before publish
Retain
QoS
0
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
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
B
Buenos
Bueno

todos los demas atributos se dejan como estan, 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

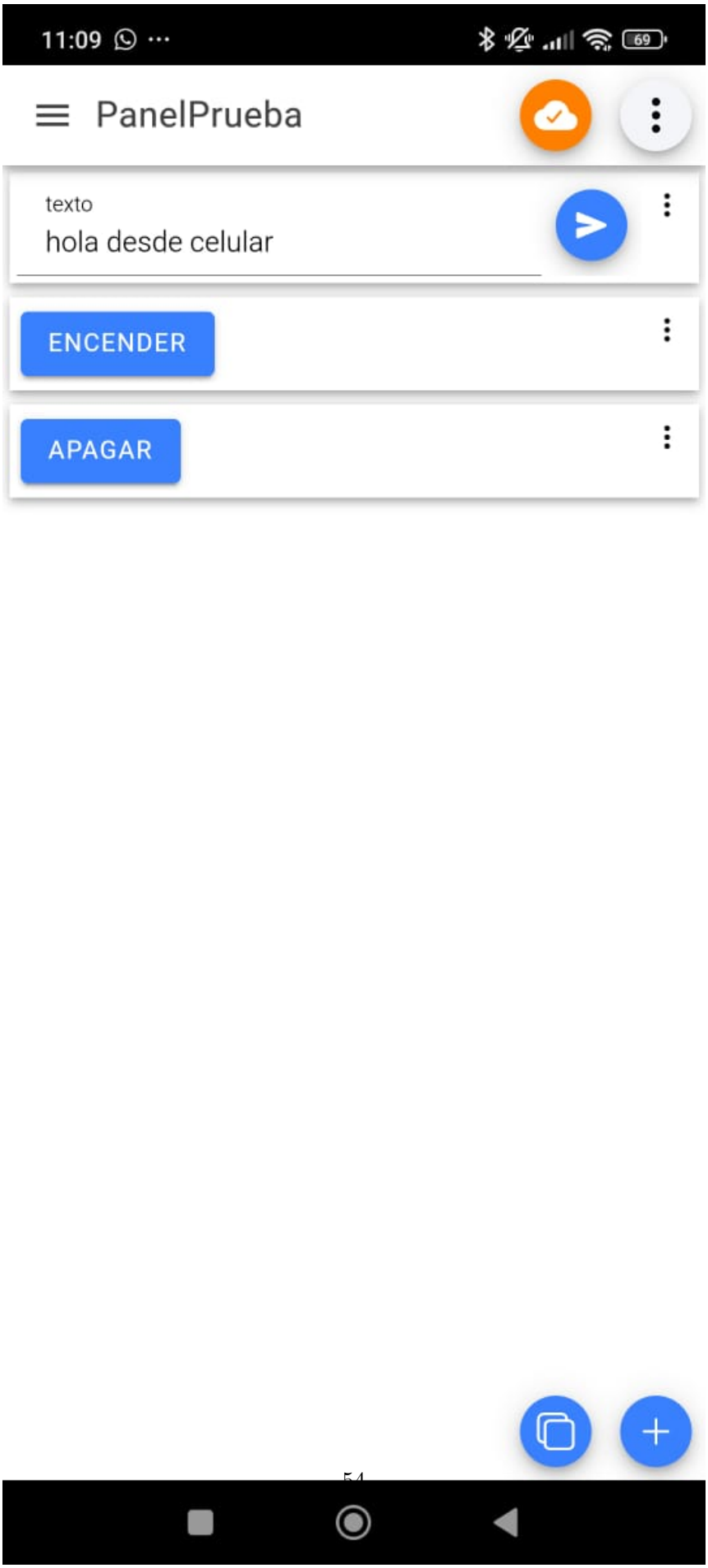
Hacer un programa en el Raspberry Pi Poco W que pida un mensaje por consola, si el mensaje es ”z”, terminar el programa, si es otro mensaje publicarlo en el servidor MQTT. En la App adicionar un control ”Text Log” que se suscriba al mismo topic del Raspberry Pi Pico W, monitorear los mensajes que publica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

07_ejemplo_03_mqtt_publicador_mensajes.py

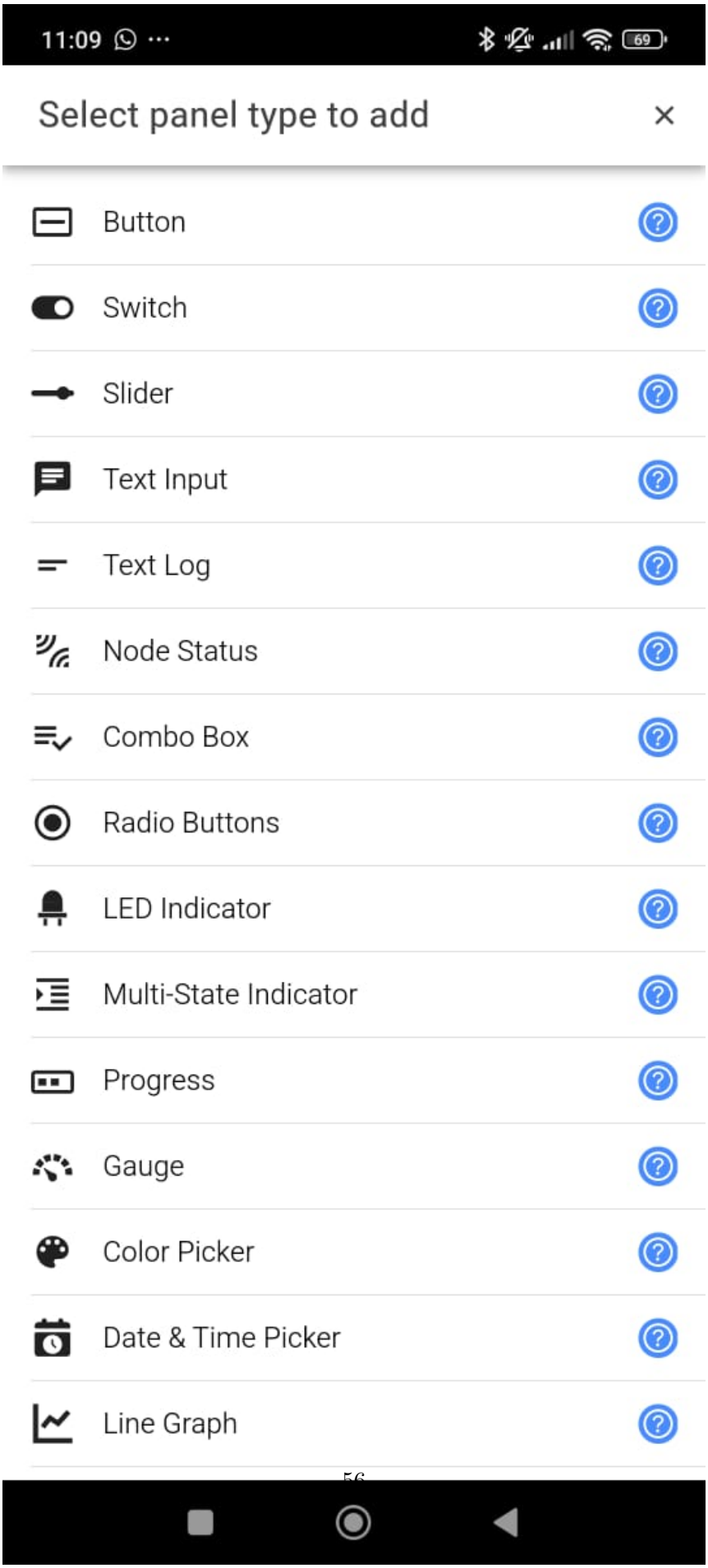
```
import network
import utime
import umqtt.simple as mqtt



### 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')
```

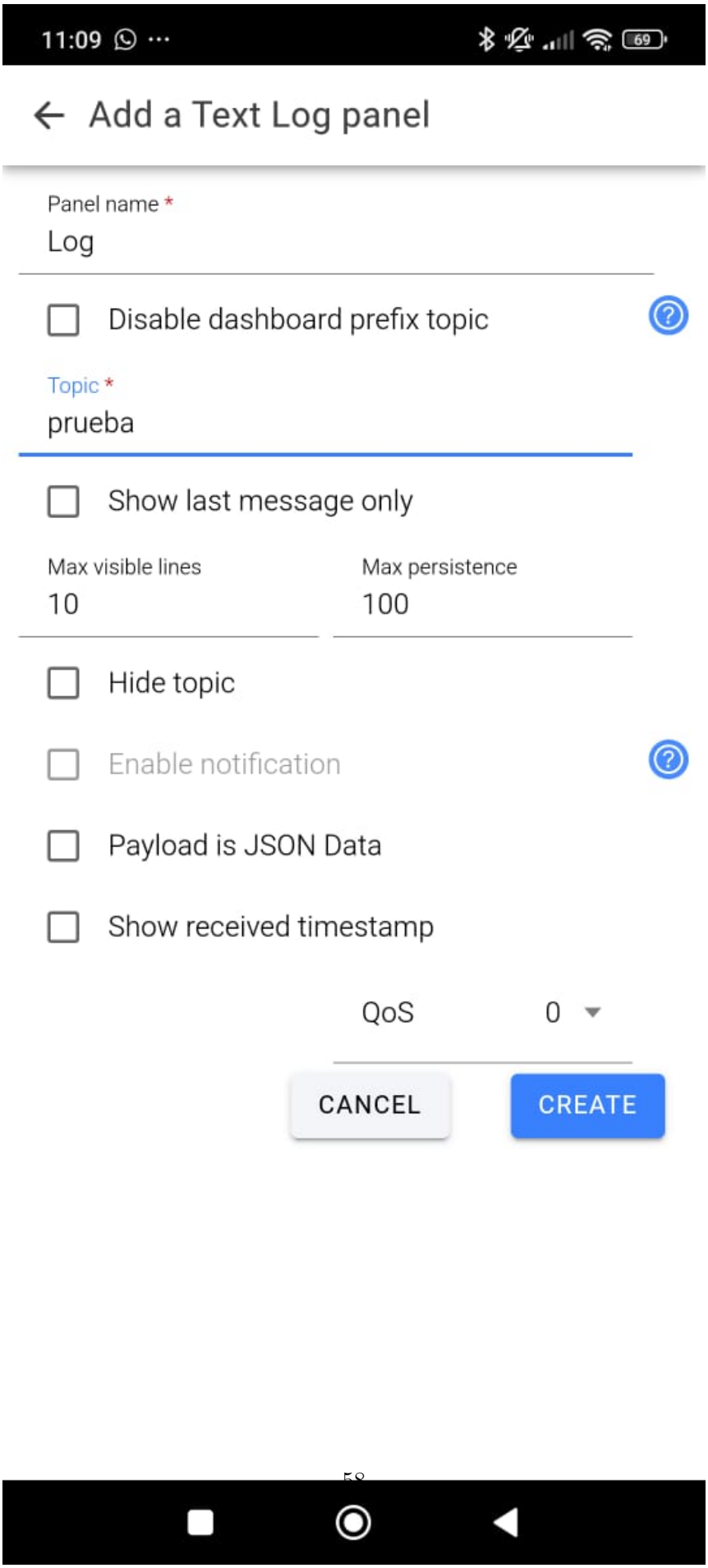
Agregar un nuevo panel en la aplicación **IoT MQTT Panel** dando click en el botón +


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

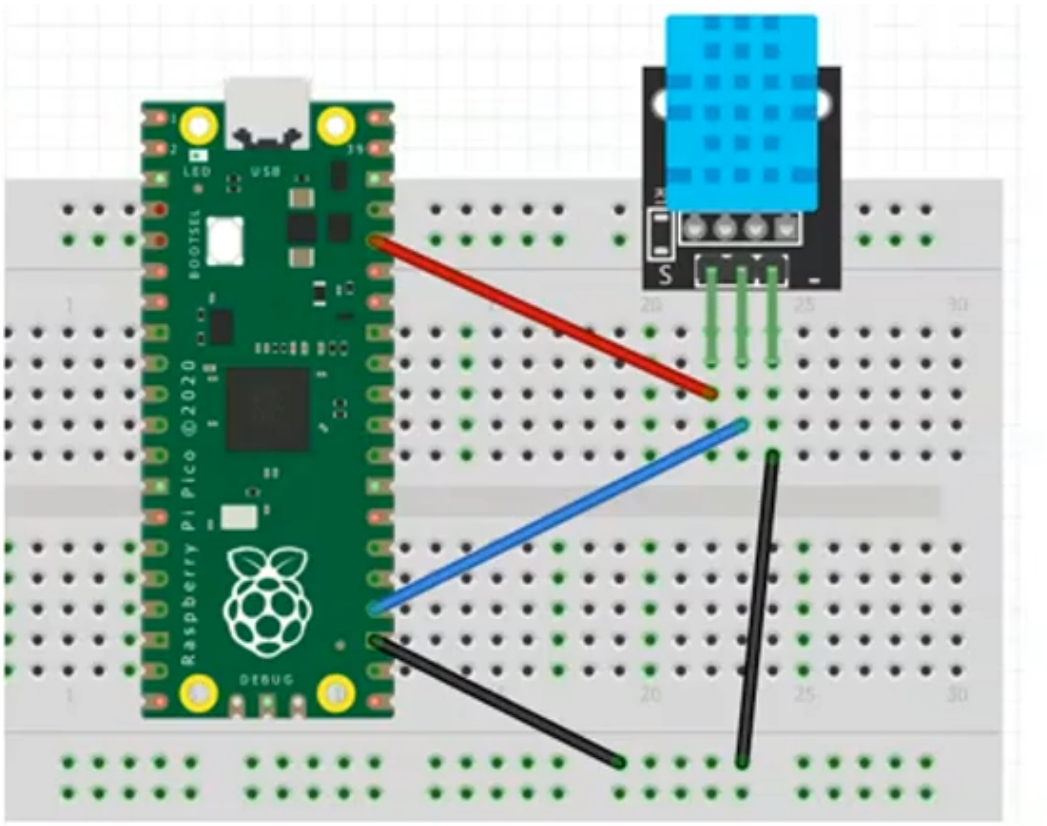
Configurar el panel con el atributo Panel Name **Log**, el atributo Topic **prueba** y dar click en el botón **Create**


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')
```

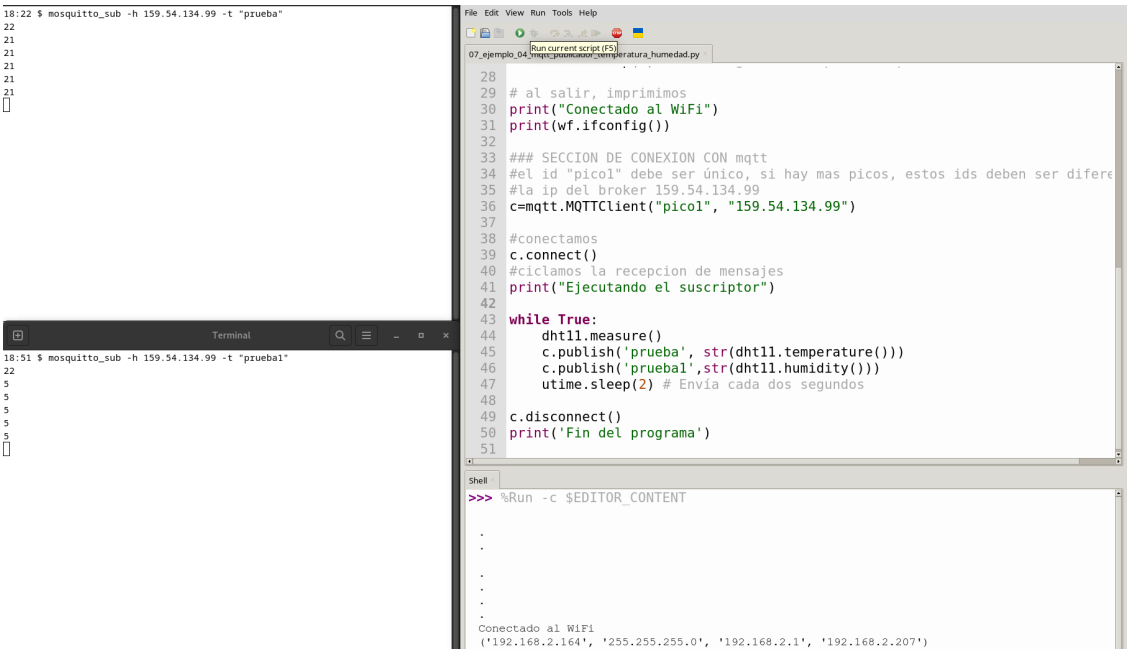
Ejecutamos el programa

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:

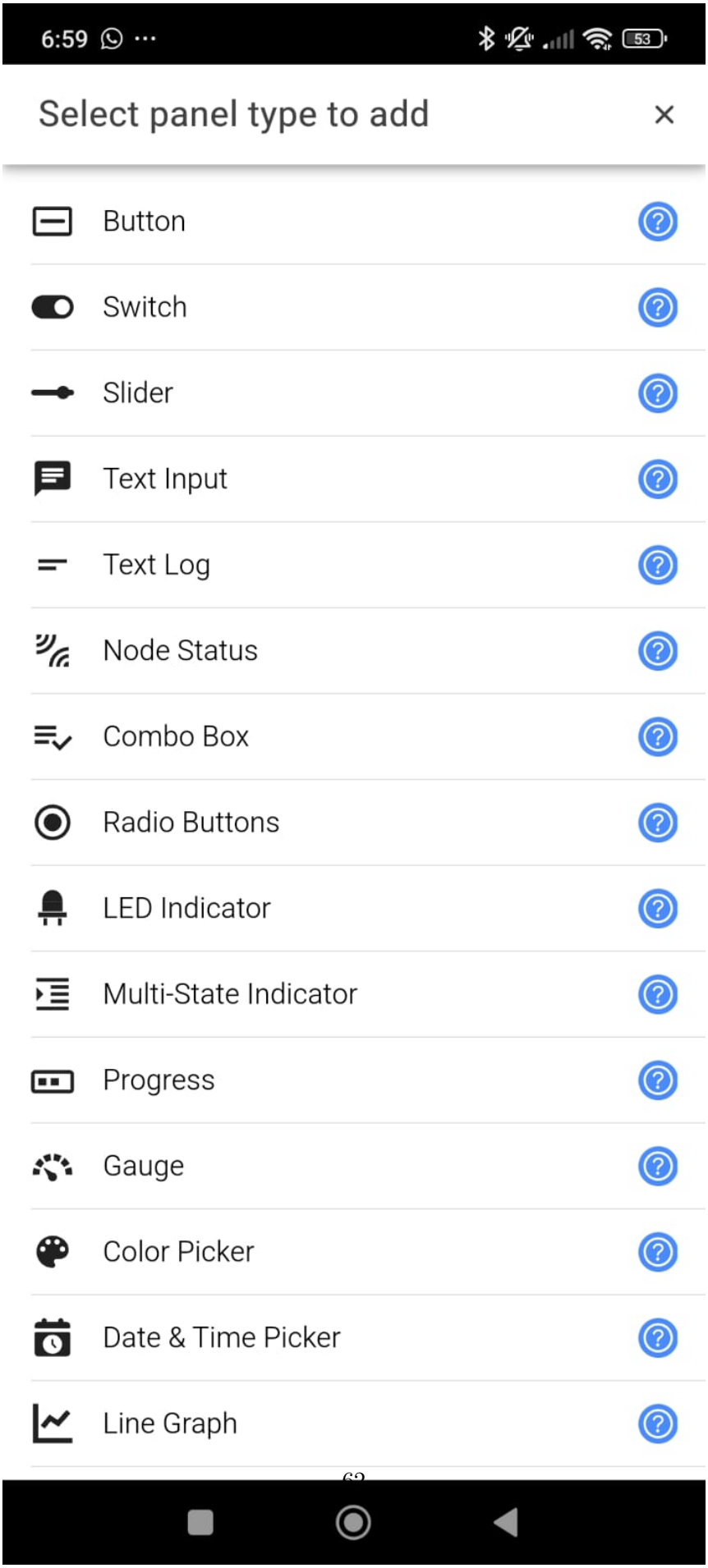
```
mosquitto_sub -h 159.54.134.99 -t "prueba"
```



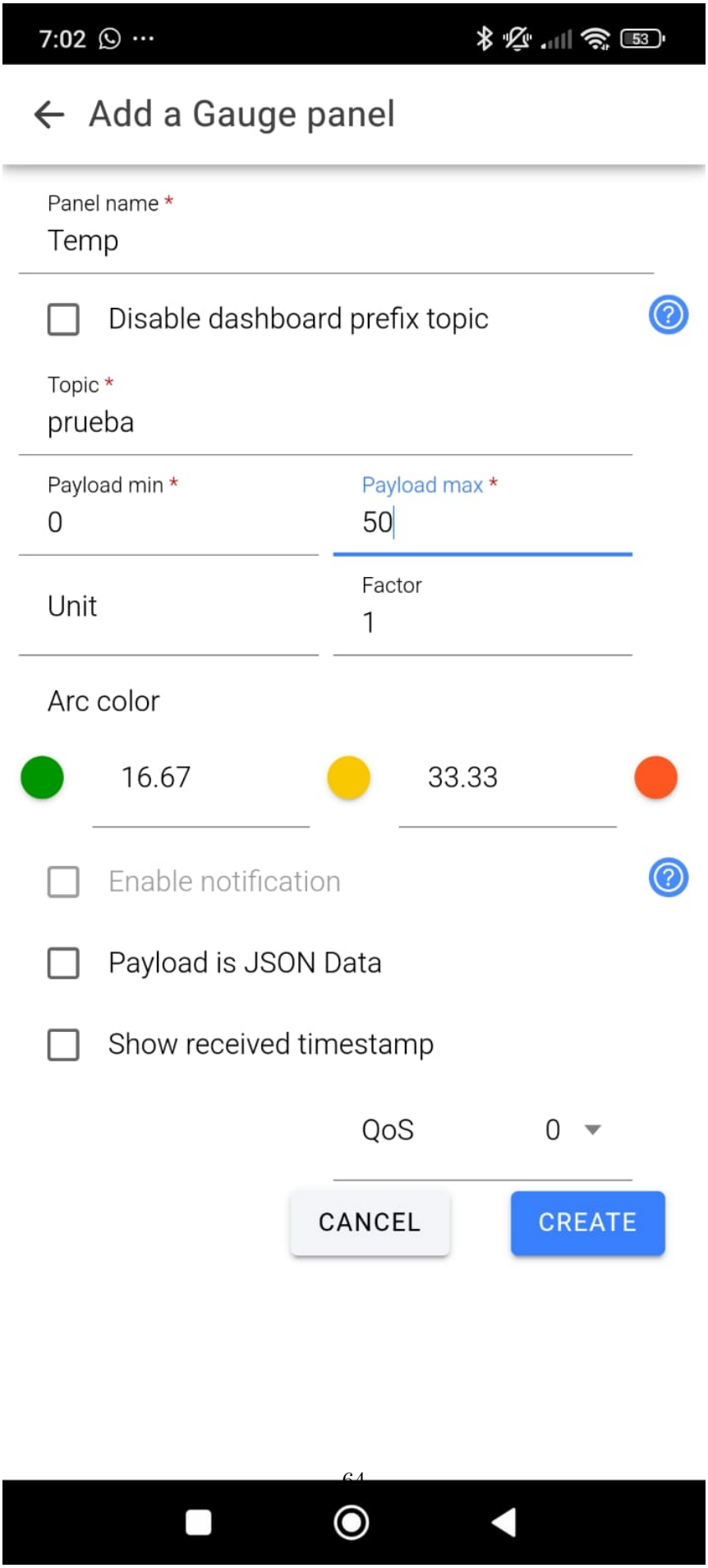
0.9.1 Eliminar el Text log panel creado en el ejemplo anterior

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

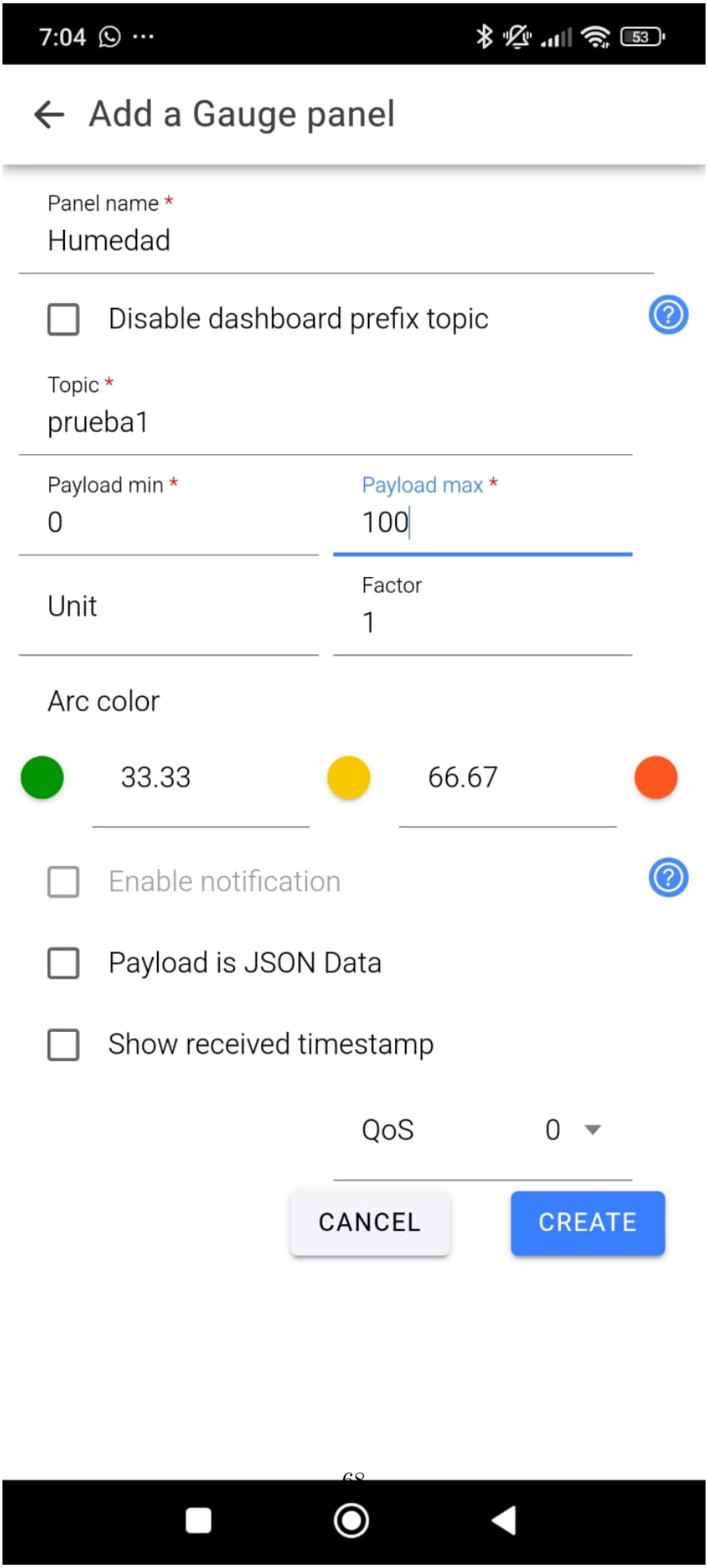
Configurar el panel con el atributo Panel Name **Temp**, el atributo Topic **prueba**, el atributo Payload min **0** y el atributo Payload max **50**, dar click en el botón **Create**


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

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
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

Your browser does not support the video tag.

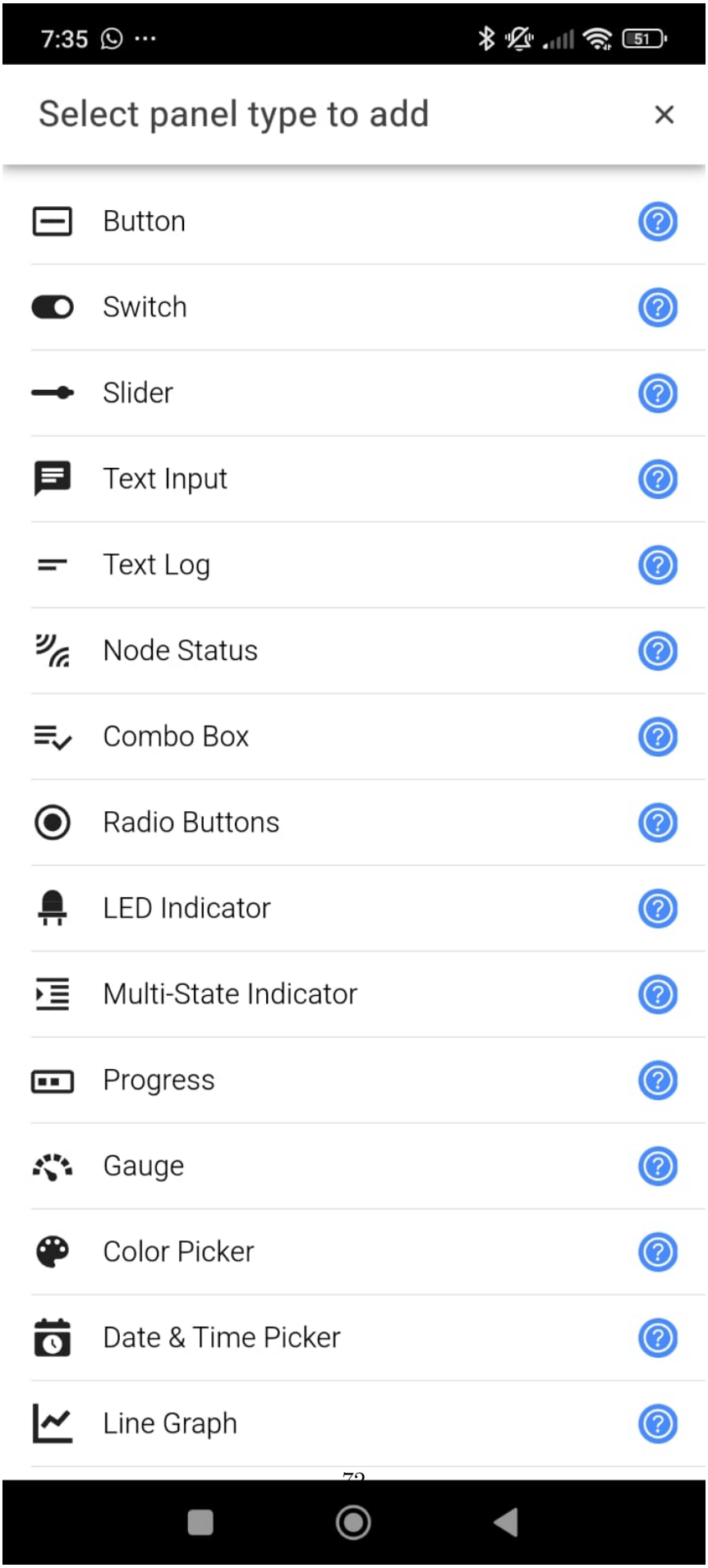
0.10 Ejemplo 05 MQTT

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.

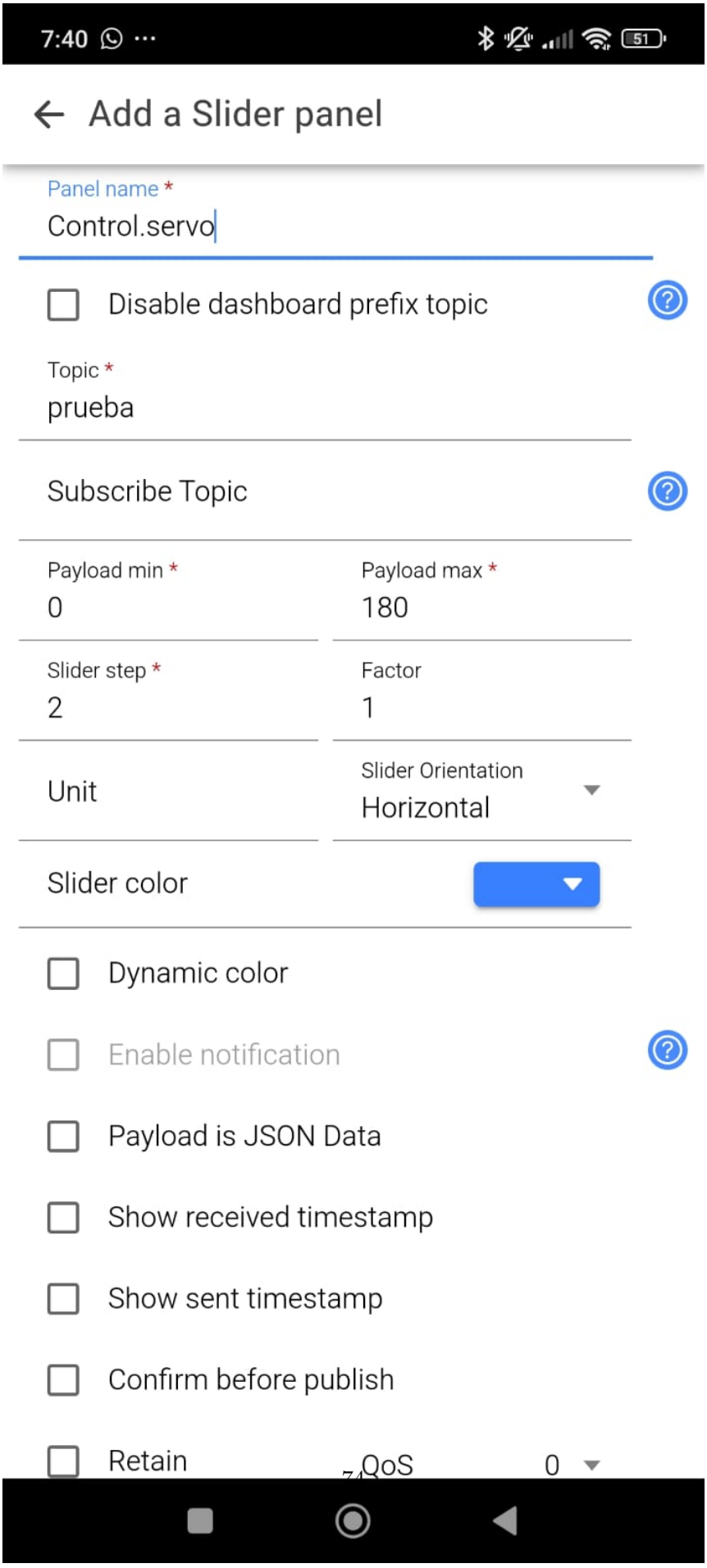
0.10.1 Ahora crearemos primero la aplicación en el App del celular

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

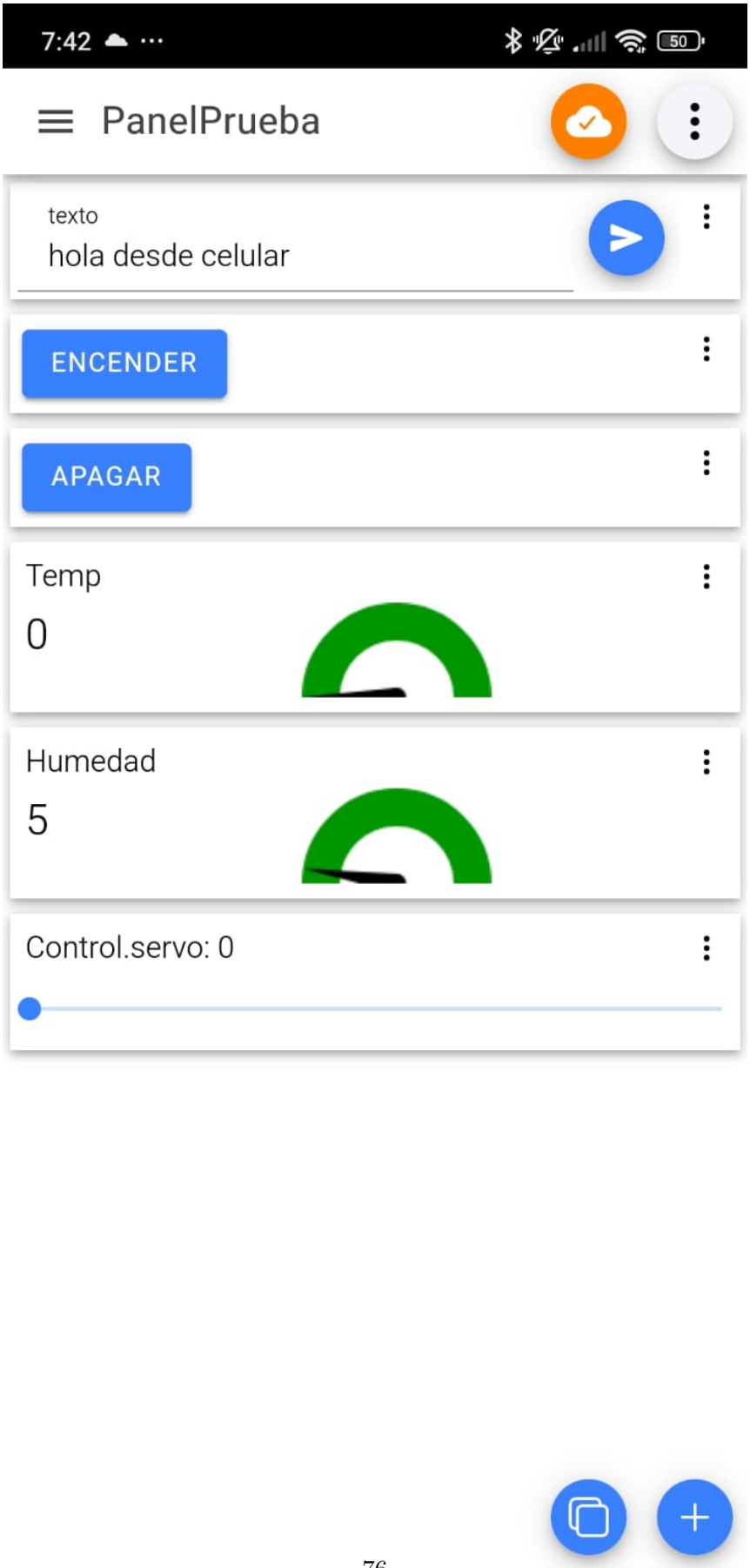
Seleccionar la opción **Slider**


7:35
Select panel type to add
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)

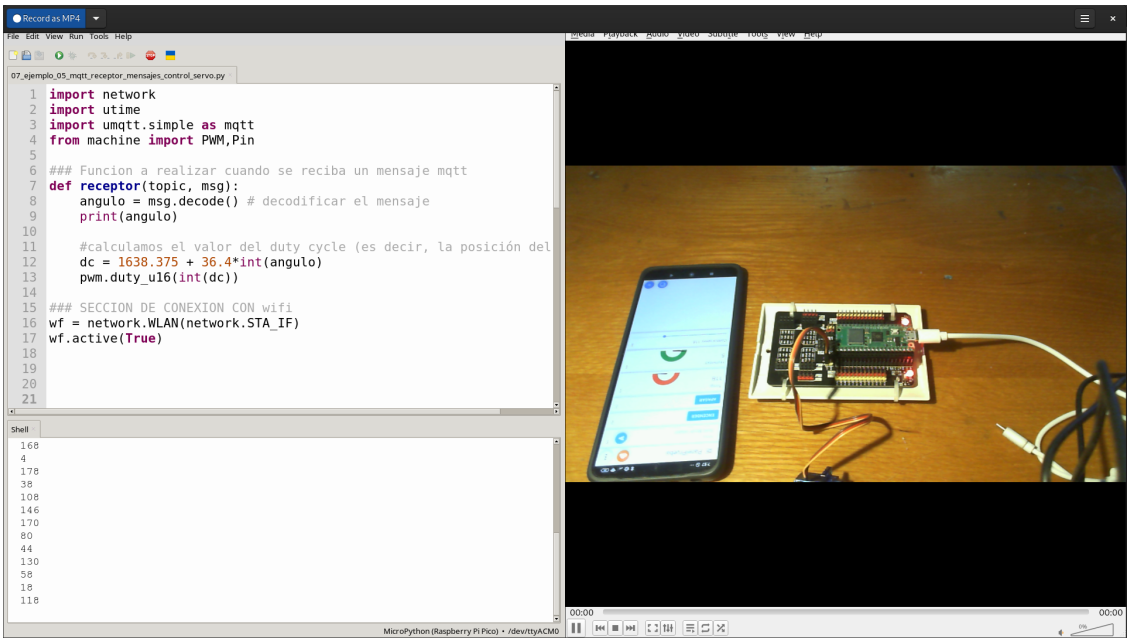

7:42
PanelPrueba
texto
hola desde celular
ENCENDER
APAGAR
Temp
0
Humedad
5
Control.servo: 0

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=mqtt.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
######

#definimos 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.