UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

PROYECTOS DIGITALES AVANZADOS

PRÁCTICA N°12:

MQTT

Profesor:
Pedro Rene
Cabrera

Bachiller:
Claudia Rodríguez
C.I: 27.943.668

Barcelona, febrero de 2025

1. OBJETIVOS DE LA PRÁCTICA

1.1. Investigar acerca del protocolo MQTT (búsqueda bibliográfica, roles en el protocolo, cómo establecer la comunicación, etc.).

1.2. Investigar si es necesario el uso de una biblioteca para implementar el protocolo MQTT. En caso afirmativo, instalarla y analizar qué clases contiene, cómo utilizarlas y su función en la implementación.

1.3. Revisar el material bibliográfico entregado en clase para extraer la información relevante para la práctica.

1.4. Leer sobre la herramienta Adafruit IO para comprender su funcionamiento y papel en la práctica.

1.5. Implementar el protocolo MQTT en dos Raspberry Pi Pico W, donde una actúe como **publicador** y la otra como **suscriptor**, estableciendo la comunicación a través del bróker Adafruit IO.

1.6. En el caso del publicador, debe de realizar las siguientes tareas: enviar mensajes a un feed (tema) específico, con el contenido definido por el usuario y esperar y recibir el ACK enviado por el suscriptor.

1.7. En el caso del suscriptor, debe de realizar las siguientes tareas: leer los mensajes de un feed (tema) específico y enviar un ACK al publicador, el cual consiste en el mensaje original con un añadido.

1.8. Implementar LEDs como indicadores de estado en el publicador y el suscriptor, para visualizar cuándo están enviando, recibiendo o esperando mensajes.

1.9. Utilizar la pantalla OLED SSD1306 cuando se envíen mensajes, reciban mensajes, si esta esperando por algún mensaje, etc.

2. DESARROLLO

2.1. Planteamiento del Problema

En prácticas anteriores, se ha establecido la comunicación entre dos Raspberry Pi Pico W mediante distintos métodos: alámbrico (N°6), inalámbrico (N°7) y una comunicación basada en el modelo servidor-cliente (N°11). Sin embargo, cada una de estas formas de comunicación presenta tanto ventajas como desventajas.

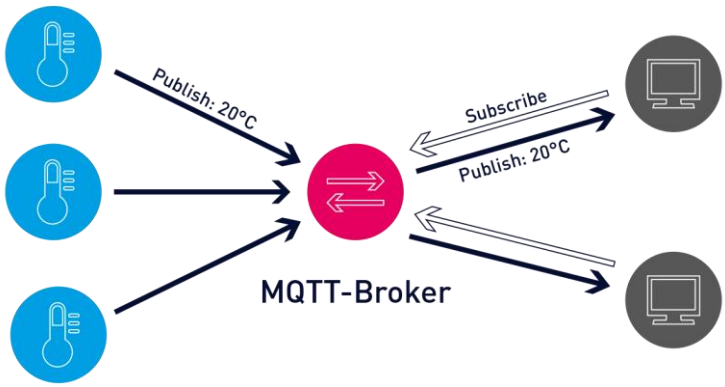
La comunicación alámbrica requiere el uso de cables, lo que implica que los dispositivos deben estar físicamente cercanos. No obstante, una de sus principales ventajas es que ofrece una mayor estabilidad en la transmisión de datos.

Por otro lado, la comunicación inalámbrica, que se implementó utilizando WiFi y Bluetooth, ofrece mayor flexibilidad y movilidad. Sin embargo, el alcance del Pico W es de aproximadamente 30 metros en espacios abiertos y puede reducirse significativamente en interiores debido a interferencias o barreras físicas.

En cuanto a la comunicación servidor-cliente, esta resulta efectiva, pero en la práctica realizada en clase, ambos dispositivos debían estar conectados a la misma red WiFi. Esto

representa una limitación cuando se desea monitorear variables ambientales en ubicaciones distantes, como en diferentes edificios o campus separados.

En cambio, el protocolo MQTT (Message Queuing Telemetry Transport o Transporte de Telemetría de Mensajes en Cola) ofrece una solución a estos inconvenientes. Esta comunicación depende de un broker para gestionar los mensajes, pero permite una mayor movilidad y la posibilidad de conectar dispositivos a través de Internet, superando así las restricciones de alcance y de estar en la misma red local. MQTT se basa en un modelo de comunicación eficiente y flexible, compuesto por los siguientes roles:



Publish (Publicador): Es el componente encargado de enviar o publicar mensajes. Estos pueden incluir datos de monitoreo, actualizaciones de estado o cualquier otra información relevante.

Subscribe (Suscriptor): Es el encargado de recibir y procesar los mensajes publicados en un determinado tema.

Broker: Actúa como intermediario entre el publicador y el suscriptor, asegurando que los mensajes sean entregados correctamente a los dispositivos suscritos.

Topic (Tema): Es el canal de comunicación donde se publican los mensajes. Los suscriptores que estén registrados en ese tema podrán recibir la información publicada.

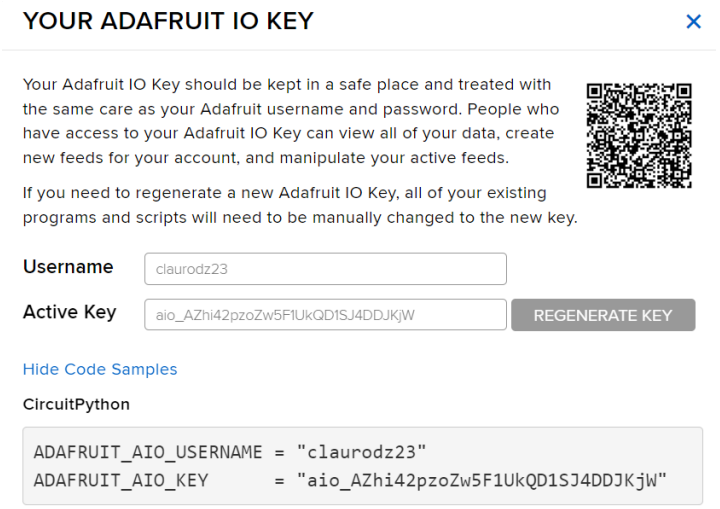
Planteado lo anterior y conociendo los roles y participantes en el protocolo, se presentan los siguientes problemas a resolver:

1- Es necesario identificar un bróker que cumpla con los requisitos de la práctica (Adafruit IO), considerando aspectos como disponibilidad, facilidad de configuración y compatibilidad con Raspberry Pi Pico W. Además, es importante documentarse sobre su funcionamiento y configuración.
2- Aunque existen ejemplos básicos de MQTT, como encender LEDs mediante mensajes publicados en un bróker, el objetivo de esta práctica es ir más allá. Se busca establecer una comunicación bidireccional entre dos Raspberry Pi Pico W, permitiendo no solo el envío y recepción de mensajes, sino también la implementación de un mecanismo de **ACK** para confirmar que los mensajes han sido correctamente recibidos. Esto garantiza una comunicación efectiva y minimiza la pérdida de datos.

2.2. Solución: La solución se dividirá en tres partes: conocimiento básico del bróker (Adafruit IO), la solución para el publicador y la solución para el suscriptor.

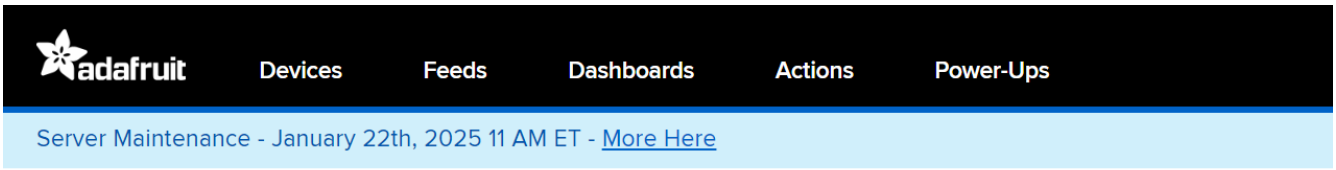
2.2.1. Bróker: El bróker utilizado será Adafruit IO, este es un servicio en la nube donde se permite almacenar, compartir datos y recuperarlos. Adafruit IO funciona con varios

microcontroladores, entre ellos la RP2040. Para poder utilizar los servicios de Adafruit IO es necesario crearse una cuenta y luego de ello se tiene que anotar la llave personal (un círculo amarillo con una llave negra) de Adafruit IO que se utilizará en el código. Debido a que estamos trabajando con microPython, se toma el siguiente código:

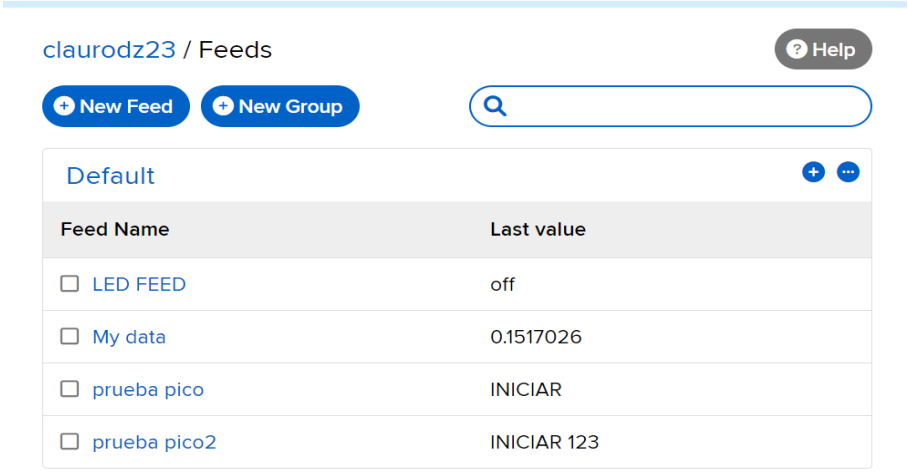


Esto se pegará en el código y con ello "sincronizaremos" nuestra cuenta de Adafruit IO, en el caso de la programadora es **"claurodz23"**.

Una vez hecho eso, se tiene que crear el **feed** con el **topic**. Para ello, hay que hacer clic en **Feeds**:

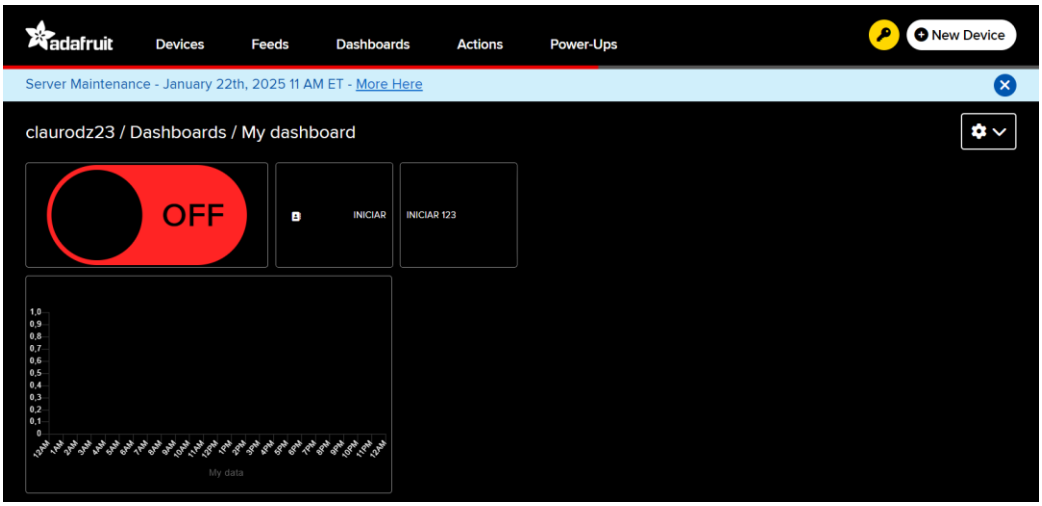


Luego se hace clic en **"New Feed"** y se le coloca tanto nombre como descripción a dicho feed. En el caso de la programadora, se crearon dos feeds, uno denominado **"prueba pico"** y **"prueba pico2"**.



Al crear los **feeds,** se le da clic y luego a **Feed Info,** esto con el objetivo de sincronizar ambos feeds en el código. Del feed **"prueba pico"** se obtiene el siguiente **endpoint:** `claurodz23/feeds/prueba-pico` y del feed "prueba pico2" se obtiene el siguiente **endpoint**: `claurodz23/feeds/prueba-pico2`.

Una vez creado ambos **feeds**, y conociendo la llave por MQTT, se tiene que crear o modificar el **Dashboard.** Se optó por crear un dashboard con que ya se estaba practicando. Entonces **Dashboard > My dashboard** y se abrirá la siguiente página:



Se hace clic en el engranaje del lado derecho > **Create new block** > Se selecciona el que es una cadena de texto de varias líneas > Se escoge con el **feed** que se desea afiliar > Next step > Por último **Create block.** Esta serie de pasos se repite para el otro feed, ya que se tiene un feed para enviar mensajes de parte del publicador al suscriptor y otro para que el suscriptor envié el ACK al publicador y este sepa que si recibió el mensaje correctamente, por lo tanto, se tienen que crear dos blocks.

2.2.2. Solución para el rol del publicador.

Lo primero a destacar es que se están reusando funciones de prácticas anteriores, como lo son crear_oled(); mostrar_oled(oled, message, n); conectar_wifi(oled) y por último se tiene la función **publicador(oled).** De esta función es que se desarrollará la solución:

1- Se definen las siguientes variables:

mqtt_host → Este es el que actuará de bróker, en este caso, sería “io.adafruit.com”

mqtt_username → De que cuenta se está tomando el feed (“claurodz23”)

mqtt_password → Es la clave que se tomo en pasos anteriores (la llave Adafruit IO personal)

mqtt_public_topic → Este representa el feed donde el publicador anunciará el mensaje (“claurodz23/feeds/prueba-pico”)

mqtt_ack_topic → Este representa el feed donde el suscriptor enviará el ACK para que el publicador se enteré de que recibió el mensaje correctamente ("claurodz23/feeds/prueba-pico2")

mqtt_client_id → Este representa un ID único en todo el universo del Adafruit IO.

2- Se inicializa la comunicación del MQTT con lo anteriormente definido y se conecta.

3- Se define una función que se llama **ack_callback(topic, message).** Esto para poder recibir el ACK correspondiente al mensaje enviado.
4- Se crea un bucle while donde el usuario es el encargado de colocar las frases que se publicarán y serán leídas por el suscriptor. Posteriormente, estas frases se publican en el topic "claurodz23/feeds/prueba-pico", esperando a que sean leídas por el suscriptor.
5- El Pico W publicador espera el ACK por parte del suscriptor. Para ello, se tiene que suscribir al feed "claurodz23/feeds/prueba-pico2"
6- Por último, recibe el ACK por parte del suscriptor, que este corresponde al mensaje original más **"123"**
7- Este bucle finalizará cuando el publicador anuncie la frase **"fin".** Para que tanto el publicador como el suscriptor se desconecten.
8- Cabe destacar que todo se visualiza tanto en la pantalla OLED como en la pantalla del computador. Esto para tener un mapeo de la información.

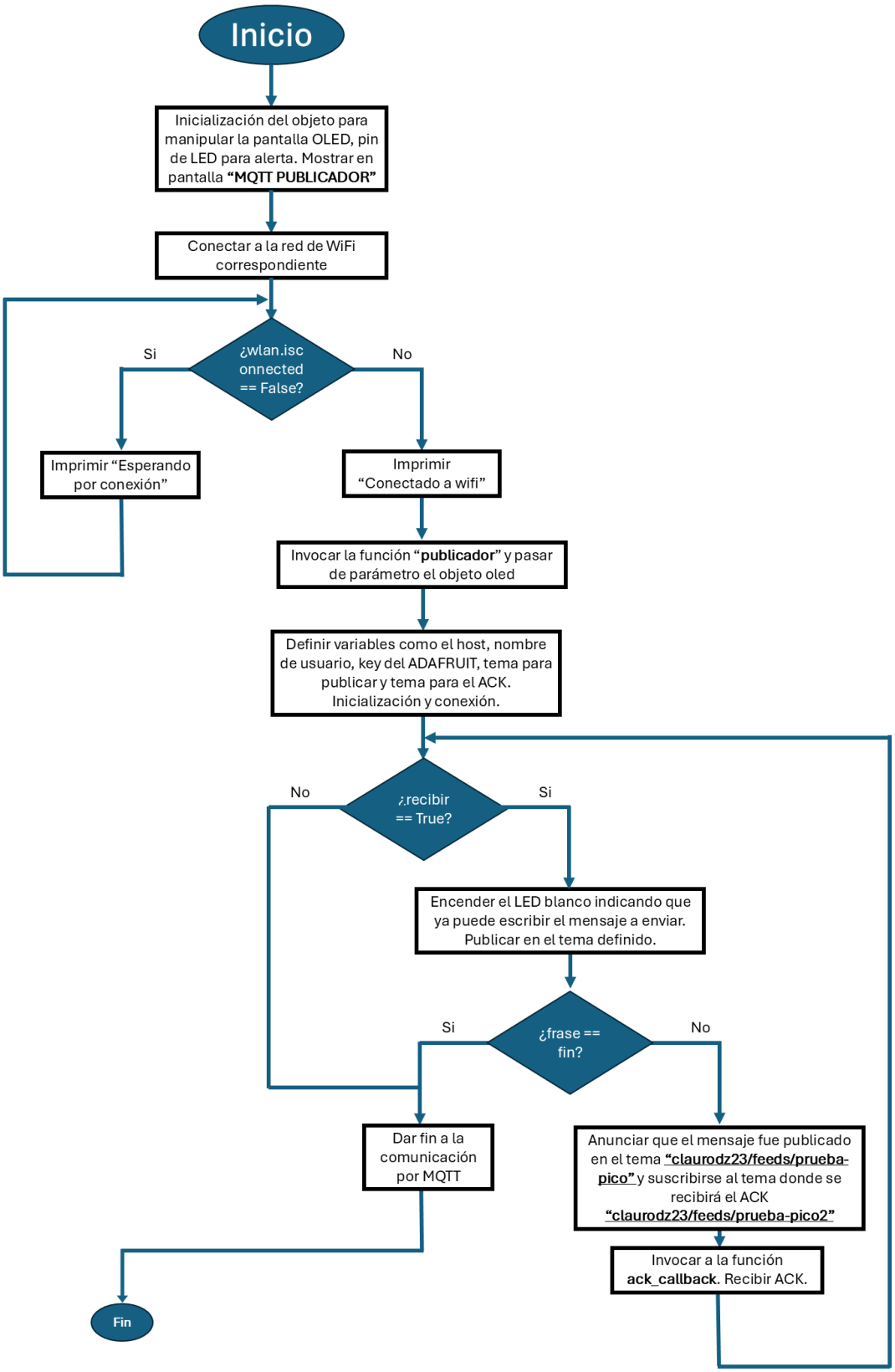
2.2.3. Solución para el rol del suscriptor

Al igual que en la solución del publicador, se estarán usando funciones ya conocidas para la programadora, que es crear_oled(), mostrar_oled(oled, message, n), conectar_wifi(oled). Ya como estas funciones están estudiadas, solo se explicará la función del suscriptor.

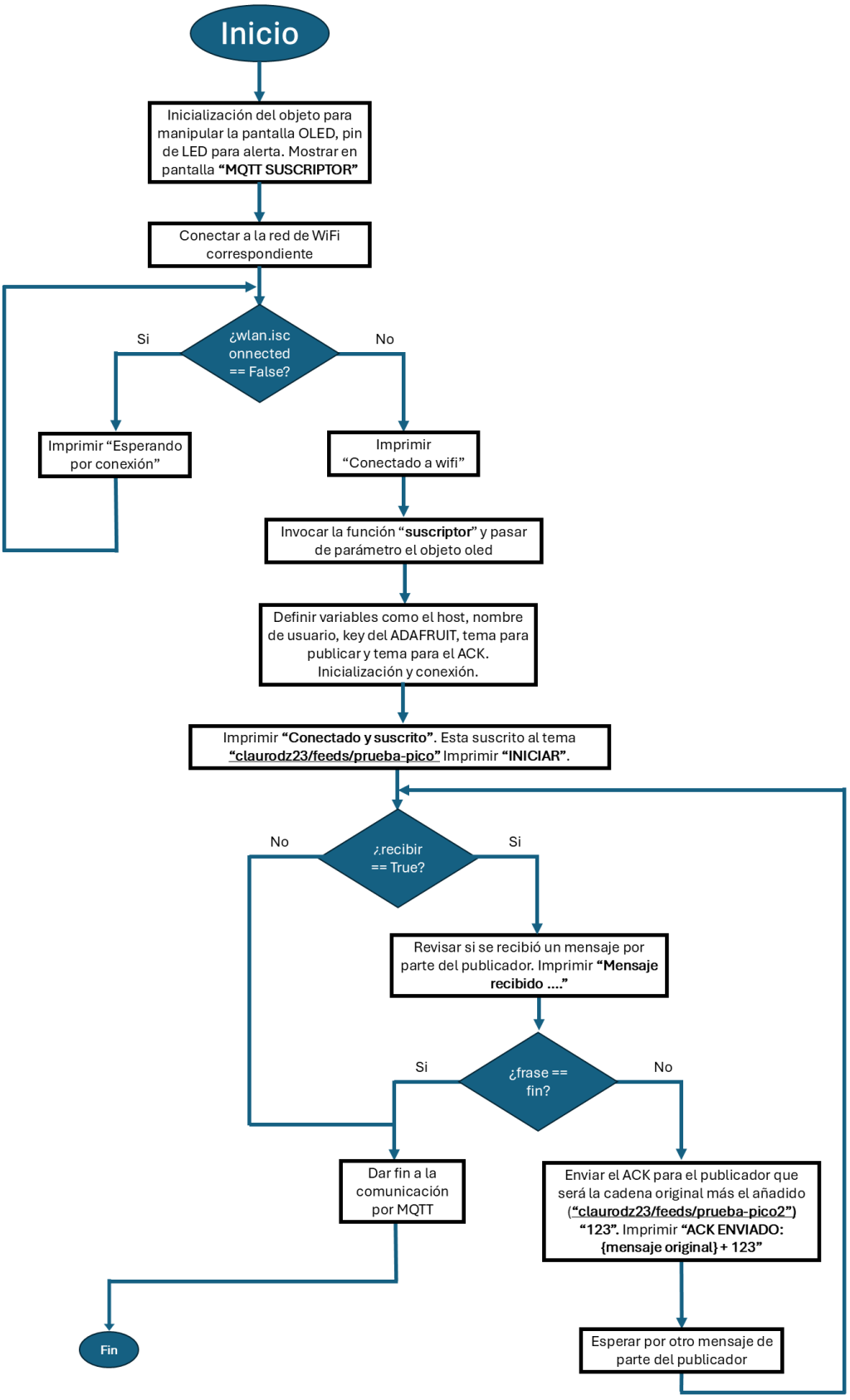
1- Al igual que en el publicador, se definen las mismas variables, a excepción de **mqtt_client_id,** que tiene que ser diferente en el universo de MQTT. En el caso del suscriptor esta es igual: "claudiaelenarodriguezdesio28686549"
2- Se inicializa la comunicación con el MQTT cliente.
3- Se suscribe al topic "claurodz23/feeds/prueba-pico", y se manda el **"INICIAR"** para dar inicio a la comunicación MQTT indicando que el suscriptor ya esta listo.
4- Se entra en un bucle while esperando a que el publicador haya anunciado algún mensaje, en el caso de que lo hizo se puede presentar dos casos, que el mensaje que haya recibido diga "fin" indicando el fin de la comunicación o que haya recibido un mensaje normal. Este mensaje normal se recibe utilizando la función **mqtt_subscription_callback** y luego se le reenviará al publicador utilizando el tema "claurodz23/feeds/prueba-pico2". El reenvío de este mensaje, en este caso, corresponde al ACK. Dicho ACK es la cadena original más "123".
5- Para realizar el envío del ACK se tiene que suscribir al tema "claurodz23/feeds/prueba-pico2", y publicar dicho mensaje para que el publicador pueda leerlo.
6- Una vez enviado el ACK, el suscriptor se queda esperando al siguiente mensaje.

2.3. Diagrama de Flujo.

2.3.1. Diagrama de Flujo: Publicador

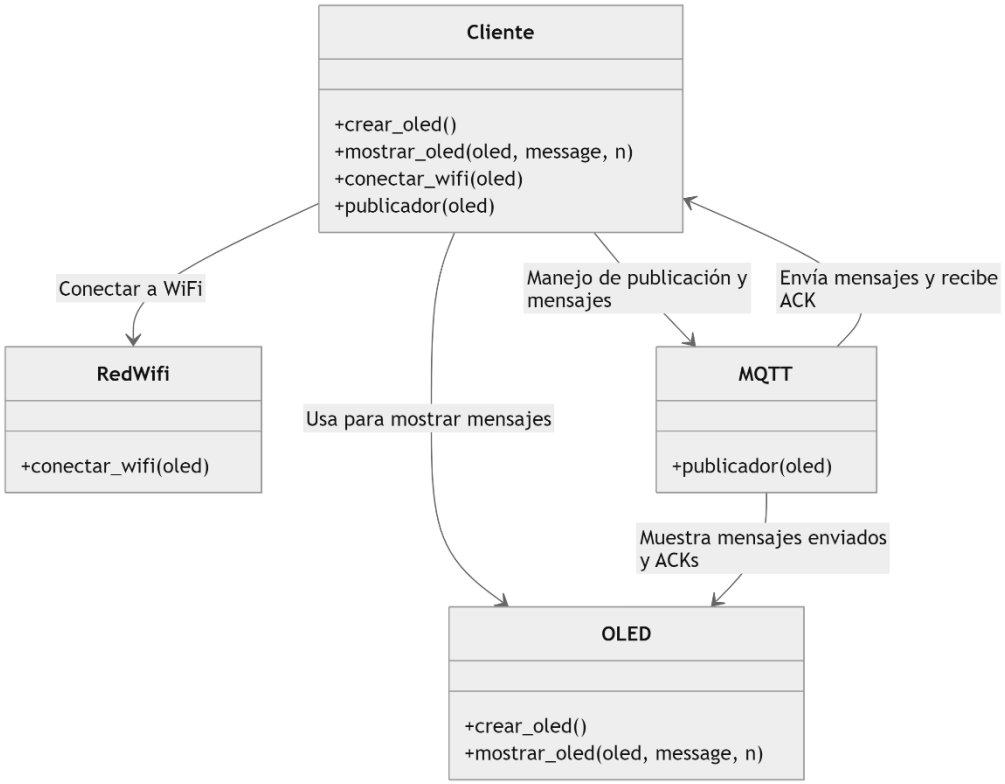


2.3.2. Diagrama de Flujo: Suscriptor

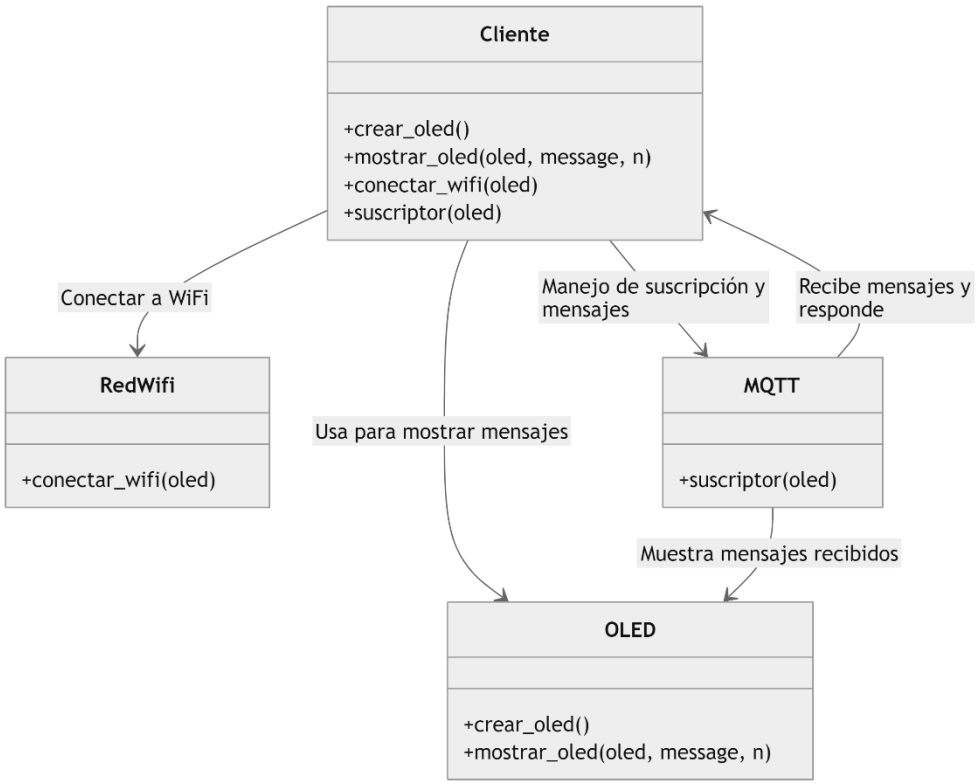


2.4. Diagrama de UML

2.4.1. Diagrama UML Publicador



2.4.2. Diagrama UML Suscriptor



3. ANEXOS

3.1. Código

3.1.1. Código del Pico W Publicador

```
import network
import time
from machine import Pin, I2C
from umqtt.simple import MQTTClient
from time import sleep
from ssd1306 import SSD1306_I2C

# -- crear objeto para manejar la pantalla oled -- #
def crear_oled():
  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=400000)
  oled = SSD1306_I2C(128,64,i2c)
  return oled

# -- funcion para mostrar mensajes en la pantalla OLED -- #
def mostrar_oled(oled, message, n):
  oled.fill(0)
  ancho_caracter = 7  # <-- tamaño en pixeles de un caracter
  max_columna = 120  # <-- long max de la pantalla
  fila = 0        # <-- primera linea
  columna = 0                # <-- primera columna
  palabras = message.split() # <-- string a lista
  for palabra in palabras:
    ancho_palabra = len(palabra) * ancho_caracter
    if columna + ancho_palabra > max_columna:
      fila += 16
      columna = 0
    if fila >= 50:
      oled.show()
      sleep(n)
      oled.fill(0)
      fila = 0
      columna = 0
    oled.text(palabra, columna, fila)
    columna = columna + 7 # --> espacio entre palabras porsia
    columna += ancho_palabra + ancho_caracter
  oled.show()
  sleep(n)

# -- conectar a wifi, modificar dependiendo de la locacion -- #
def conectar_wifi(oled):
  wifi_ssid = "clau-moto"
  wifi_password = "tata4646"
  wlan = network.WLAN(network.STA_IF)
  wlan.active(True)
  wlan.connect(wifi_ssid, wifi_password)
  while wlan.isconnected() == False:
    cadena = 'Esperando por conexion...'
    print(cadena)
    mostrar_oled(oled, cadena, 1)
    time.sleep(1)
  print(f"Conectado a wifi: {wifi_ssid}")
  cadena = f"Conectado a wifi: {wifi_ssid}"
  mostrar_oled(oled, cadena, 3)

def publicador(oled):
  led = Pin(14, Pin.OUT)
  led.value(0)

  """ DETALLES PARA AUTENTICACION REVISAR """
  # host
  mqtt_host = "io.adafruit.com"

  # mi nombre de usuario
  mqtt_username = "claurodz23"

  # key en adafruit (es en la llave que esta en la página de inicio
  mqtt_password = "aio_AZhi42pzoZw5F1UkQD1SJ4DDJKjW"
```

```
  # nombre del feed de mensaje publicador --> suscriptor
  mqtt_publish_topic = "claurodz23/feeds/prueba-pico"

  # nombre del feed para enviar ack de suscriptor --> publicador
  mqtt_ack_topic = "claurodz23/feeds/prueba-pico2"

  # id unico para hacer la comunicacion
  mqtt_client_id = "claudiaelenarodriguezdesio27943668"

  # inicializacion mqtt
  mqtt_client = MQTTClient(
      client_id=mqtt_client_id,
      server=mqtt_host,
      user=mqtt_username,
      password=mqtt_password)

  mqtt_client.connect()

  # -- funcion de callback para manejar el mensaje ACK -- #
  def ack_callback(topic, message):
    ack_message = message.decode('utf-8')
    sleep(4)
    led.value(1)
    print(f"ACK recibido: {ack_message}")
    mostrar_oled(oled, f"ACK recibido: {ack_message}", 3)
    led.value(0)

  # -- callback para recibir el ACK -- #
  mqtt_client.set_callback(ack_callback)

  # -- publicar frase publicador --> tema / prueba_pico2
  try:
    while True:
      led.value(1) # <-- indicador de poder escribir la frase
      cadena = "Ingrese una frase: "
      print(cadena)
      mostrar_oled(oled, cadena, 2)
      frase = input()
      cadena = f"Frase ingresada: {frase}"
      mostrar_oled(oled, cadena, 2.5)
      cadena = f'Frase publicada: {frase}'
      led.value(0)  # <--- indicador que ya se publico
      print(cadena)
      mostrar_oled(oled, cadena, 2)

mqtt_client.publish(mqtt_publish_topic, str(frase))
      mostrar_oled(oled, "Esperando ACK del suscriptor..", 3)
      if frase == "fin":
        cadena = "Cerrando comunicacion MQTT"
        print(cadena)
        mostrar_oled(oled, cadena, 3)
        mqtt_client.disconnect()
        break

      # esperando ack del topic prueba pico2
      print(f"Esperando ACK para: {frase}")

mqtt_client.subscribe(mqtt_ack_topic) # suscripcion
      mqtt_client.wait_msg()

      led.value(0)
      time.sleep(3)

  except Exception as e:
    print(f'Error al publicar el mensaje: {e}')
    mostrar_oled(oled, "Error al publicar mensaje", 3)

  finally:
    mqtt_client.disconnect()


if __name__ == "__main__":
  oled = crear_oled()
```

```
  oled.fill(0)
  led = Pin(14, Pin.OUT)
  led.value(0)
  oled.show()
  cadena = "MQTT PUBLICADOR"
  print(cadena)
  mostrar_oled(oled, cadena, 3)
  conectar_wifi(oled)
  sleep(10)
  publicador(oled)
```

3.1.2. Código del Pico W Suscriptor

```
import time
import network
from machine import Pin, I2C
from umqtt.simple import MQTTClient
from time import sleep
from ssd1306 import SSD1306_I2C

# -- crear objeto para manejar la pantalla oled -- #
def crear_oled():
  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=400000) ## <-- pendiente
  oled = SSD1306_I2C(128,64,i2c)
  return oled

# -- funcion para mostrar mensajes en la pantalla OLED -- #
def mostrar_oled(oled, message, n):
  oled.fill(0)
  ancho_caracter = 7 # <-- tamaño en pixeles de un caracter
  max_columna = 120  # <-- long max de la pantalla
  fila = 0       # <-- primera linea
  columna = 0    # <-- primera columna
  palabras = message.split() # <-- string a lista
  for palabra in palabras:
    ancho_palabra = len(palabra) * ancho_caracter
    if columna + ancho_palabra > max_columna:
      fila += 16
      columna = 0
    if fila >= 50:
      oled.show()
      sleep(n)
      oled.fill(0)
      fila = 0
      columna = 0
    oled.text(palabra, columna, fila)
    columna = columna + 7 # --> espacio entre palabras porsia
    columna += ancho_palabra + ancho_caracter
  oled.show()
  sleep(n)

# -- conectar a wifi, modificar dependiendo de la locacion -- #
def conectar_wifi(oled):
  wifi_ssid = "clau-moto"
  wifi_password = "tata4646"
  wlan = network.WLAN(network.STA_IF)
  wlan.active(True)
  wlan.connect(wifi_ssid, wifi_password)
  while wlan.isconnected() == False:
    cadena = 'Esperando por conexion...'
    print(cadena)
    mostrar_oled(oled, cadena, 1)
    time.sleep(1)
  print(f"Conectado a wifi: {wifi_ssid}")
  cadena = f"Conectado a wifi: {wifi_ssid}"
  mostrar_oled(oled, cadena, 3)

def suscriptor(oled):
  led = Pin(14, Pin.OUT)
  led.value(0)

  """ DETALLES PARA AUTENTICACION REVISAR """
  # host
  mqtt_host = "io.adafruit.com"

  # mi nombre/usuario en adafruit
```

```
  mqtt_username = "claurodz23"

  # mi llave de adafruit (que es la llave que esta en la esquina de mi main page)
  mqtt_password = "aio_AZhi42pzoZw5F1UkQD1SJ4DDJKjW"

  # topic para que el suscriptor lea los mensajes del publicador
  mqtt_receive_topic = "claurodz23/feeds/prueba-pico"

  # topic para que el suscriptor "publique" el ack
  mqtt_ack_topic = "claurodz23/feeds/prueba-pico2"

  # recuerda que esta tiene que ser super super super unica
  mqtt_client_id = "claudiaelenarodriguezdesio28686549"

  # inicializacion
  mqtt_client = MQTTClient(
      client_id=mqtt_client_id,
      server=mqtt_host,
      user=mqtt_username,
      password=mqtt_password)

  # funcion callback de la suscripcion
  def mqtt_subscription_callback(topic, message):
    led.value(1)
    decoded_message = message.decode('utf-8')
    cadena = f'Mensaje recibido: {decoded_message}'
    print (cadena)
    mostrar_oled(oled, cadena, 2)
    led.value(0)

    # agregado de por si quieres finalizar la comunicacion de
    # manera adecuada
    if decoded_message == "fin":
      cadena_fin = "Recibido 'fin', desconectando MQTT"
      print(cadena_fin)
      mostrar_oled(oled, cadena_fin, 3)
      mqtt_client.disconnect()
      return # salida del callback

    # envio del ack, osea, mensaje original y le añadi el 123
    ack_message = decoded_message + " 123"

    try:

mqtt_client.publish(mqtt_ack_topic, ack_message)
      print(f"ACK enviado al topic: {mqtt_ack_topic}")
      print(f"ACK enviado: {ack_message}")
      cadena = "ACK ENVIADO: " + ack_message
      mostrar_oled(oled, cadena, 3)
      cadena = "Esperando siguiente mensaje"
      print(cadena)
      mostrar_oled(oled, cadena, 3)
    except Exception as e_ack:
      print(f"Error al enviar ACK: {e_ack}")

  # antes de conectarse, le tiene que avisar al cliente para usar el callback

mqtt_client.set_callback(mqtt_subscription_callback)
  try:
    mqtt_client.connect()

    # una vez conectado, se suscribe al tema de prueba pico2

mqtt_client.subscribe(mqtt_receive_topic)
    cadena = "Conectado y suscrito"
    print(cadena)
    mostrar_oled(oled, cadena, 2)
```

```
    # esto para asegurar que el led este apagado
    led.value(0)

mqtt_client.publish(mqtt_receive_topic, "INICIAR") # limpieza del tema

    while True:
      # revisar si existen mensajes
      mqtt_client.check_msg()
      time.sleep(1) # deja ese tiempo por si acaso

  except Exception as e:
    cadena = f'Falla en el suscriptor MQTT: {e}'
    print(cadena)
    mostrar_oled(oled, cadena, 5)
  finally:
    mqtt_client.disconnect()
    cadena_desconectado = "Desconectado de MQTT"
    print(cadena_desconectado)
      mostrar_oled(oled, cadena_desconectado, 3)


if __name__ == "__main__":
  oled = crear_oled()
  oled.fill(0)
  oled.show()
  led = Pin(14, Pin.OUT)
  led.value(0)

  cadena = "MQTT SUSCRIPTOR"
  print(cadena)
  mostrar_oled(oled, cadena, 3)

  conectar_wifi(oled)
  suscriptor(oled)

  cadena_fin_programa = "Fin del programa suscriptor"
  print(cadena_fin_programa)
  mostrar_oled(oled, cadena_fin_programa, 3)
```

3.2. Bibliografía

¿Qué es Adafruit IO?. Enlace: https://www.youtube.com/watch?v=bYhRhDvTP5c&t=2s

Todo sobre el Protocolo MQTT. Enlace: https://www.youtube.com/watch?v=5_qGrvT_qww

Qué es MQTT?. Enlace: https://www.youtube.com/watch?v=RpjSwriOi9U

Instalar la libreria MQTT en Micropython en el Raspberry Pi Pico. Enlace: https://www.youtube.com/watch?v=FH6J6mzPWyU

How To Set Up MQTT With Raspberry Pi Pico W | Guide For Beginners. Enlace: https://www.youtube.com/watch?v=ybCMXqsQyDw

Raspberry Pi Pico W for IoT Project Using MicroPython and MQTT [A Complete Guide]. Enlace: https://www.youtube.com/watch?v=GQOqvvei5Do

IO – Adafruit. Enlace: https://io.adafruit.com/claurodz23/overview

Getting Started with MQTT on Raspberry Pi Pico W - Connect to the Internet of Things!
Enlace: https://core-electronics.com.au/guides/getting-started-with-mqtt-on-raspberry-pi-pico-w-connect-to-the-internet-of-things/

3.3. Diagrama circuital