UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS
PROYECTOS DIGITALES AVANZADOS

PRÁCTICA N°7:
PROTOCOLOS DE COMUNICACIÓN INALAMBRICOS

Profesor:

Pedro Rene
Cabrera

Bachiller:

Claudia Rodríguez
C.I: 27.943.668

Barcelona, noviembre de 2024

1. OBJETIVO DE LA PRÁCTICA

1.1. Configurar los Raspberry Pi Pico para recibir datos tanto a través de WiFi como de Bluetooth.

1.2. Investigar y documentar el uso de las bibliotecas necesarias para la comunicación WiFi y Bluetooth en los Picos.

1.3. Almacenar la información recibida en la memoria del Pico y visualizarla tanto en la pantalla de la computadora como en la pantalla OLED del Pico correspondiente.

1.4. Los agregados de la autora fueron los siguientes:

1.4.1. Comunicación WiFi:

- ✓ Se implementó un mecanismo de acuse de recibo (ACK) personalizable para la comunicación WiFi. El receptor envía un ACK al transmisor, que puede ser un simple mensaje de confirmación o una modificación del mensaje original (añadirle caracteres extra al inicio o final de la cadena)
- ✓ Se desarrolló una alternativa al uso de una red creada entre los Picos, permitiendo la conexión a una red WiFi con acceso a internet.

1.4.2. Comunicación Bluetooth:

- ✓ Se integró un LED azul como indicador visual de la recepción exitosa del mensaje en el receptor Bluetooth. Este LED funciona como una señalización de que la transmisión de transmisor a receptor se completó. Esto se implemento debido a que el intervalo de transmisión es de 500ms, por lo tanto, se vuelve un poco tediosa la espera.
- ✓ Se implementó la capacidad de seleccionar el dispositivo Bluetooth receptor entre múltiples dispositivos Picos W disponibles.
- ✓ Se implementó un mecanismo de acuse de recibo (ACK), al igual que en la comunicación por WiFi es personalizable, ya sea colocando "**ACK RECIBIDO**" o modificar el mensaje original.

2. DESARROLLO

2.1. PLANTEAMIENTO DEL PROBLEMA

Esta práctica busca extender la funcionalidad de la práctica anterior, que demostró la transmisión exitosa de datos por cable. El objetivo actual es implementar la transmisión de datos inalámbrica, utilizando tanto WiFi como Bluetooth.

Para la comunicación WiFi, se requiere la comprensión de las funciones y comandos proporcionados por las bibliotecas network y socket, incluyendo establecer las direcciones IP de los dispositivos Picos dentro de la red interna creada. Tanto en la comunicación WiFi como en la Bluetooth, es necesario implementar mecanismos para el cierre adecuado de las conexiones de transmisión y recepción para que no exista un desborde o algún fallo en la comunicación que se este dando en un momento.

En el caso de Bluetooth, la práctica requiere una investigación de las bibliotecas asyncio, aioble y bluetooth para identificar los comandos apropiados. Además, se debe determinar un intervalo de transmisión óptimo que evite la pérdida o confusión de datos entre el transmisor y el receptor. Un intervalo demasiado rápido o lento puede provocar errores en la transmisión. Esta observación se tiene en cuenta por la práctica de envío por cable.

2.2. SOLUCIÓN

2.2.1. Comunicación vía WiFi: Se utilizaron dos Picos, uno configurado como transmisor y otro como receptor. El Pico transmisor crea un punto de acceso (AP) denominado **'claudia'** con la contraseña **'123456789'** utilizando la biblioteca **network** (más adelante se realiza una tabla en donde se señala para que funciona cada comando de las bibliotecas utilizadas). Una vez activado el AP, se crea un socket y se enlaza a una dirección y puerto específicos, configurando el servidor para aceptar hasta 50 conexiones de clientes.

En paralelo, en el código del receptor, se utiliza igual la biblioteca network pero para conectar dicho pico a una red ya existente, es decir, el punto de acceso del pico transmisor, el resto del código es similar al del transmisor, creando su socket y asignando una dirección.

El transmisor espera la conexión del receptor al punto de acceso y posteriormente le asigna una dirección IP. En las pruebas realizadas, el receptor siempre recibió la dirección IP 192.168.4.16. Tras aceptar la conexión del cliente, el transmisor envía la información. Cabe destacar que se implementó un retardo para esperar el acuse de recibo (ACK) del receptor. Una vez enviado el ACK por el receptor, recibido y procesado en el ACK en el transmisor, se cierra la conexión.

2.2.2. Comunicación vía Bluetooth: Se tiene los dos picos, uno actúa como transmisor y el otro como receptor. Con respecto al código del transmisor, se definieron funciones

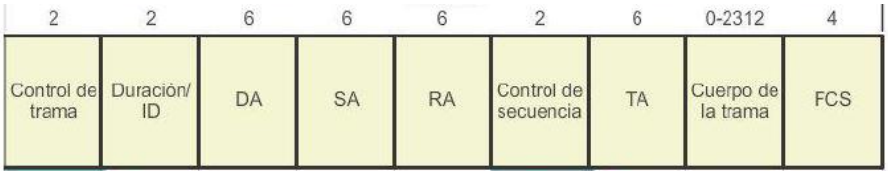
como escanear, con el objetivo de crear una lista de aquellos dispositivos BLE o Picos visibles, y otra función denominada enviar. Es en esta función, con ya el dispositivo escogido, que se envía la frase al receptor. Anteriormente, se menciono que se estableció un intervalo de 500 ms, esto con el objetivo de que los datos no se vean afectados al momento de la transmisión. El resto de las funciones, como: leer, mostrar_oled, pantalla, ya son funciones que se han utilizado con anterioridad.

Con respecto al código del receptor, la función principal es peripheral_task(oled), oled es el único parámetro que recibe para poder mostrar información en la pantalla OLED. A este Pico se le configuro con el nombre **"claudia-receptor-picow",** que es así como lo verá el otro Pico / Pico Central y se establecieron sus UUID (Universally Unique Identifiers) y otros parametros, una vez conectado, se espera la recepción de datos, carácter a carácter, con un intervalo de 500ms, y los concatena en una variable. Cuando recibe el carácter **"*"**, finaliza la recepción, muestra el mensaje completo en la pantalla OLED y enciende un LED notificando la recepción. Como acción final, el receptor envía un ACK modificando el mensaje original el cual va a ser recibido por el dispositivo central / transmisor. Finalmente, cierra la conexión Bluetooth.

2.3. Comunicación vía WiFi: La comunicación utilizada es de tipo petición-respuesta, y sigue la siguiente serie de pasos:

1. El cliente se conecta al servidor.
2. El cliente envía una petición simple con el comando get.
3. El servidor recibe la petición, lee datos de un archivo y los envía al cliente.
4. El cliente recibe los datos, los guarda en un archivo y envía un ACK al servidor.
5. El servidor recibe el ACK y cierra la conexión.

- **Trama inalámbrica de 802.11n:**



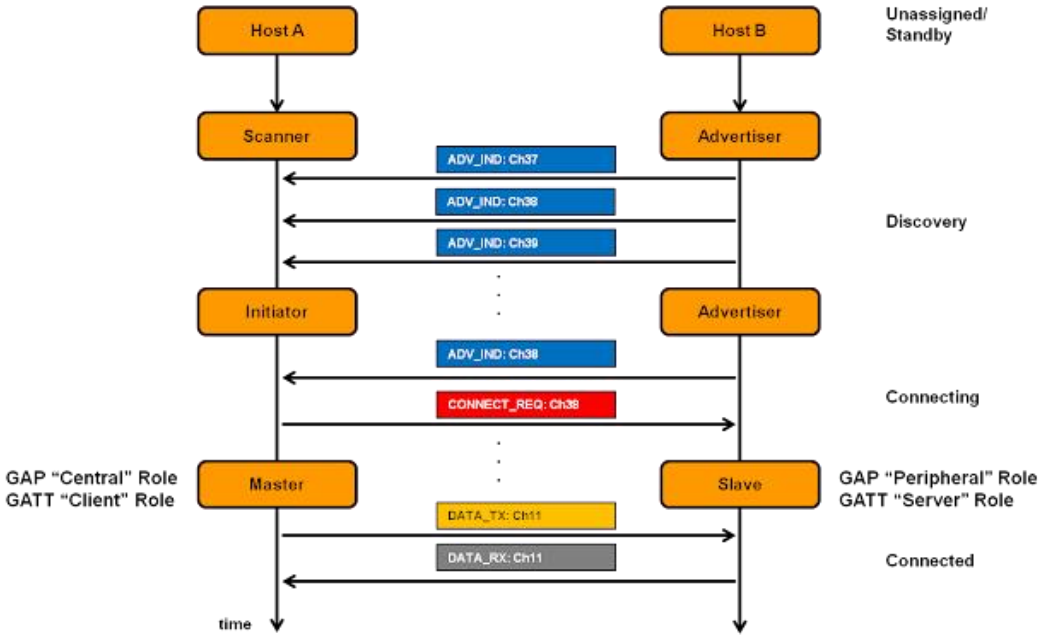
2.4. Comunicación vía Bluetooth: La comunicación fue realizada de manera serial, cabe destacar que igual a la comunicación vía WiFi, es una comunicación bidireccional, ya que tanto el receptor y transmisor envía mensaje a sus contrapartes.

En los Picos W se pueden operar dos tipos de conectividad Bluetooth: BLE (Bluetooth de bajo consumo) y Bluetooth Classics. En este caso, se utiliza el BLE y sigue la siguiente trama de datos:

- **Advertencia (Advertising):** Los periféricos envían paquetes de advertencia para anunciar su presencia y disponibilidad para conectarse. Estos paquetes pueden incluir información básica como el nombre del dispositivo y el tipo de servicio.

- **Conexión:** Una vez que un dispositivo central detecta un periférico, puede iniciar una conexión. Esto implica un proceso de enlace que establece parámetros de conexión como la frecuencia de transmisión y la duración de la conexión.

- **Intercambio de Datos:** Una vez establecida la conexión, el dispositivo central puede leer y escribir características del periférico. Esto se realiza mediante paquetes de datos que transportan las solicitudes y respuestas.



2.5. Comandos y funciones utilizadas

Tabla de comandos y funciones: WIFI

WiFi	
Nombre del comando	**Descripción**
network.WLAN(network.AP_IF)	Representa la interfaz de punto de acceso WiFi. Ejemplo: ap = network.WLAN(network.AP_IF)
ap.config()	Configura las credenciales del punto de acceso. Ejemplo: ap.config(essid='claudia', password='12345678')
conn.close()	Este comando cierra la conexión de socket establecida con un cliente. Ejemplo: conn.close()
conn.send(response)	Envía los datos contenidos en la variable `response` a través de la conexión de socket “conn” al cliente. Response es un objeto de tipo bytes.
status = ap.ifconfig()	Obtiene la información de configuración de la interfaz de red.
ip = status[0]	Extrae la dirección IP del punto de acceso WiFi
s = socket.socket()`	Crea un objeto de socket. Este objeto representa el punto final de comunicación en el servidor
s.bind(addr)	Enlaza el socket “s” a la dirección “addr”
s.listen(50)	Inicia la escucha de conexiones entrantes en el socket “s”.

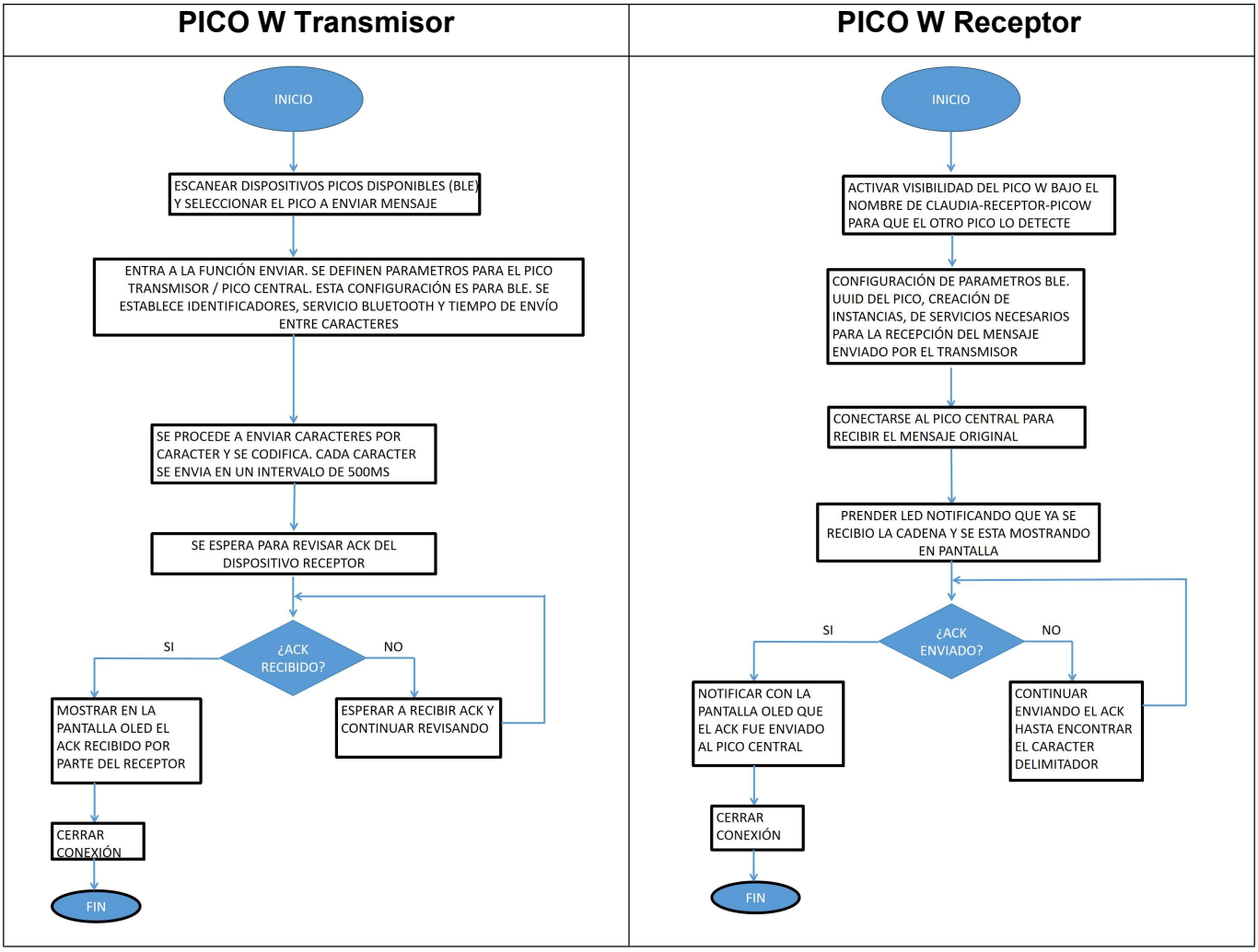
Tabla de comandos y funciones: Bluetooth BLE

Bluetooth	
Nombre del comando	**Descripción**
aioble.advertise(...)	Este es el comando principal para la interacción Bluetooth. Con esto, el otro dispositivo central va a poder visualizar este dispositivo.
connection.service(...)	Esta función obtiene una referencia al servicio especificado por su identificador.
temp_service.characteristic(...)	obtiene una referencia a una característica específica dentro del servicio.
temp_characteristic.read()	Esta es la función que realiza la recepción de datos desde el dispositivo que se conectó.

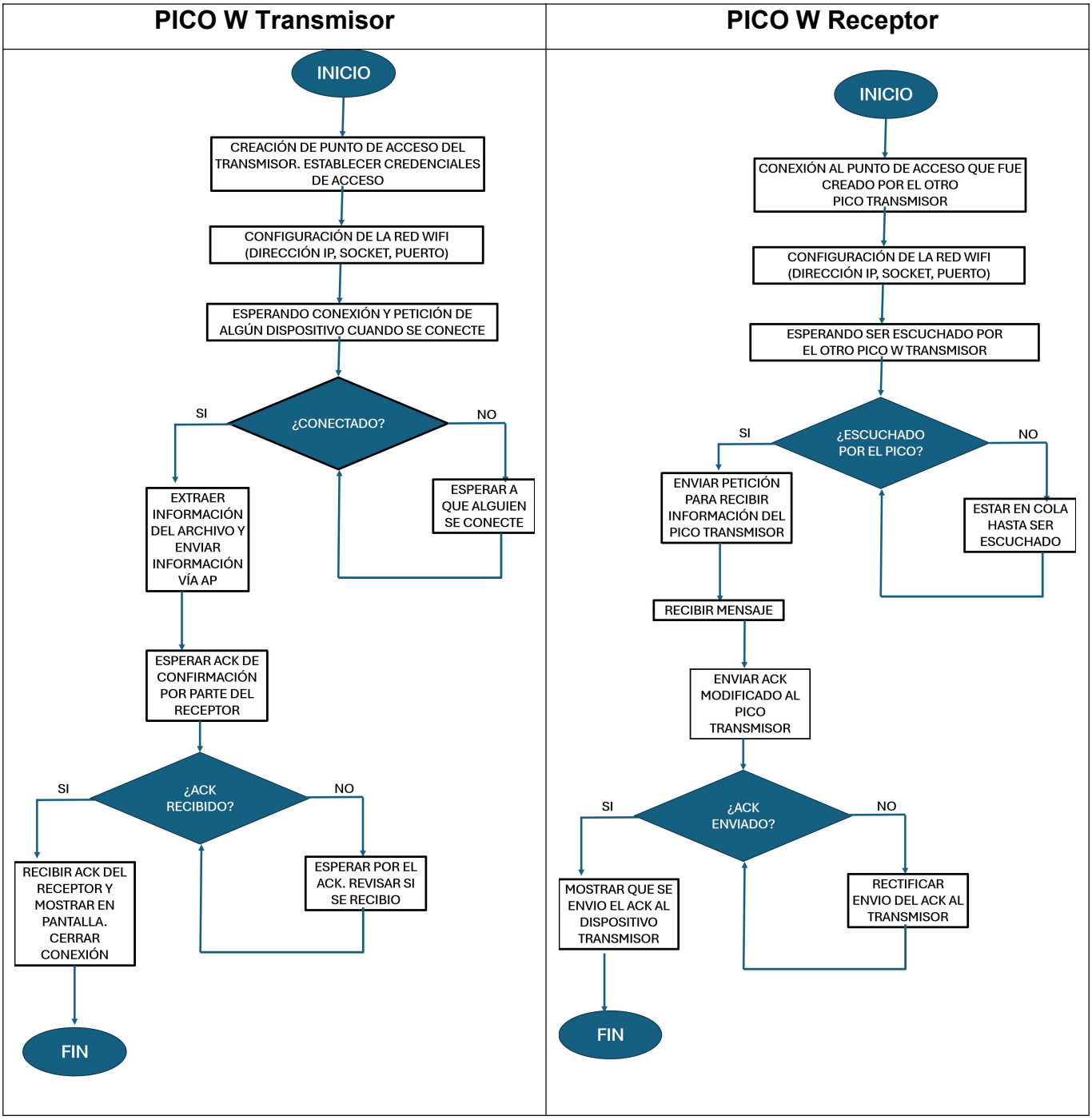
await connection.disconnected()	Espera a que la conexión se cierre.
await	Es una palabra clave utilizada para indicar que el programa debe esperar hasta que una operación asincrónica que sigue se complete.
temp_service.characteristic(..)	Se encarga de pasar como parámetro el UUID y retorna las características del servicio
asyncio.TimeoutError	Se utiliza cuando una operación asíncrona no se completa en un periodo especifico.
temp_characteristic. write(_encode_datos(palabra[i]))	Se utiliza para enviar caracteres a un servicio de Bluetooth donde anteriormente se definieron las características de dicho servicio

2.6. Diagrama de flujo

2.6.1. Envío de mensaje usando el protocolo de Bluetooth

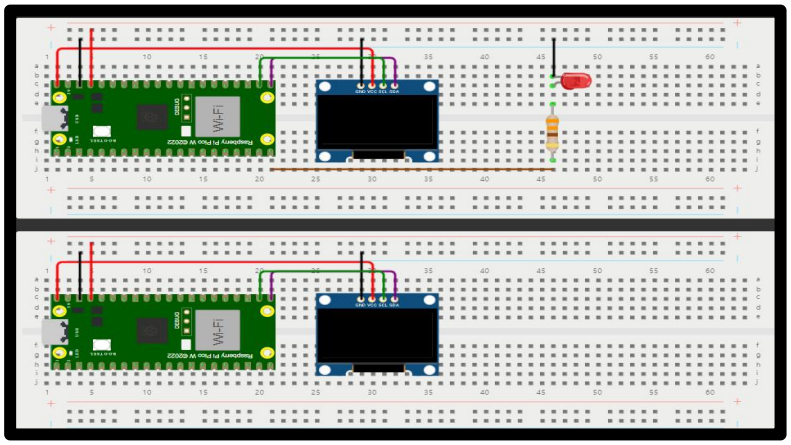


2.6.2. Envío de mensaje usando el protocolo WiFi



3. ANEXOS

3.1. Diagrama circuital



El Pico W de la parte superior actúa como receptor y el de la parte inferior actúa como transmisor. Ambos conectados a pantallas OLED para visualizar la lo que ocurre y la información que se recibe / envía.

3.2. Bibliografía

WiFi Control Your Micropython Project Using a Web Interface – Raspberry Pi Pico, ESP32, Arduino. Enlace: https://bytesnbits.co.uk/simple-micropython-wifi-connection/

AP mode Socket Server not working. Enlace: https://github.com/micropython/micropython-esp32/issues/141

Como usar el WiFi en la Raspberry Pi Pico W. Enlace: https://www.youtube.com/watch?v=tOfS4_xnMU0

Enviar Mensaje/archivo con micro python. Enlace: https://es.stackoverflow.com/questions/156645/enviar-mensaje-archivo-con-micro-python

Raspberry Pi Pico W :1 0 wifi comunicación Pico a Pico. Enlace: https://www.youtube.com/watch?v=ACAmVg6MakI

Two-way Bluetooth with Raspberry Pi Pico W and MicroPython (Re-upload). Enlace: https://www.youtube.com/watch?v=I-K_0N1m5kQ&t=414s

Library aioble is giving an error. Enlace: https://github.com/micropython/micropython-lib/blob/master/micropython/bluetooth/aioble/examples/temp_sensor.py

¿Qué? ¿Cómo? ¿Quién? - Bluetooth Low Energy (Parte 0x01). Enlace: https://www.flu-project.com/2019/09/que-como-quien-bluetooth-low-energy.html

3.3. Código de WiFi

Código WIFI - Transmisor (recibo de ACK)

```
from time import sleep
import time
import machine
import uos
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import network
import socket

def pantalla():
    WIDTH = 128
    HEIGHT = 64
    i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=200000)
    oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
    return oled

def mostrar_oled(oled, message):
    oled.fill(0)
    ancho_caracter = 7
    max_columna = 120
    fila = 0
    columna = 0
    palabras = message.split()
    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(1)
            oled.fill(0)
            fila = 0
            columna = 0
        oled.text(palabra, columna, fila)
        columna = columna + 7
        columna += ancho_palabra + ancho_caracter
    oled.show()
    sleep(5)

def get_data(file_name):
    with open(file_name, 'r') as file:
        data = file.read()
    return data

def envio_wifi(oled):
    cadena = "TRANSMISOR - PUNTO DE ACCESO"
    mostrar_oled(oled, cadena)
    sleep(3)
    ap = network.WLAN(network.AP_IF) # crear punto acceso
    ap.config(essid='claudia', password='12345678') # credenciales
    ap.active(True) # prender ap
    cadena = "PUNTO DE ACCESO CREADO "
```

Código WIFI - Receptor (envío de ACK)

```
from time import sleep
import time
import machine
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import network
import socket
import utime

def get_data(file_name):
    with open(file_name, 'r') as file:
        data = file.read()
    return data

def pantalla():
    WIDTH = 128
    HEIGHT = 64
    i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=200000)
    oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
    return oled

def mostrar_oled(oled, message):
    oled.fill(0)
    ancho_caracter = 7
    max_columna = 120
    fila = 0
    columna = 0
    palabras = message.split()
    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(1)
            oled.fill(0)
            fila = 0
            columna = 0
        oled.text(palabra, columna, fila)
        columna = columna + 7
        columna += ancho_palabra + ancho_caracter
    oled.show()
    sleep(5)

def recibo_wifi(oled):
    cadena = "RECEPTOR - DISPOSITIVO A CONECTAR"
    mostrar_oled(oled, cadena)
    sleep(3)
    file_path = "recibir_wifi.csv"
    file = open(file_path, "w")
    sta = network.WLAN(network.STA_IF) # crear
```

```
    mostrar_oled(oled, cadena)
    cadena = "NOMBRE:CLAUDIA PASS:12345678"
    mostrar_oled(oled, cadena)
    status = ap.ifconfig()
    ip = status[0] # Get the IP address
    print('IP: = ' + ip)
    addr = socket.getaddrinfo(ip, 80)[0][-1]
    s = socket.socket()
    s.bind(addr)
    s.listen(50) #listen(numero) --> nmro cliente a aceptar
    #print("Escuchando en la direccion: ", addr)
    cadena = "Esperando dispositivos a conectarse"
    mostrar_oled(oled, cadena)
    while True:  # escuchar conexiones
        try:
            conn, addr = s.accept()  # aceptar conexion entrante
            print("Cliente conectado desde la direccion: ", addr)
            r = conn.recv(1024)  # recibir peticion de 1024
            response = get_data('enviar_wifi.csv')
            cadena = "Mensaje a enviar: "
            mostrar_oled(oled, cadena)
            mostrar_oled(oled, response)
            conn.send(response)
            cadena = "Mensaje enviado al dispositivo conectado"
            mostrar_oled(oled, cadena)
            # esperar por el ACK del receptor
            sleep(8)
            ack = conn.recv(1024)  # esperar el ACK
            if ack:
                ack = str(ack)
                print("ACK recibido del receptor")
                cadena = "Mensaje de confirmacion recibido: " + ack
                print("Mensaje de confirmacion recibido: " , ack)
                mostrar_oled(oled, cadena)
            break
        except OSError as e:
            if ack:
                ack = str(ack)
                print("ACK recibido del receptor")
                cadena = "Mensaje de confirmacion recibido: " + ack
                print("Mensaje de confirmacion recibido: " , ack)
                mostrar_oled(oled, cadena)
        print("Conexion cerrada")
        cadena = "Conexion cerrada"
        mostrar_oled(oled, cadena)
        conn.close()  # <-cerrar conexion

def main():
    oled = pantalla()
    envio_wifi(oled)

if __name__ == "__main__":
    main()
```

```
conexion a wifi disponible
    sta.active(True)  # activar interfaz wifi
    sta.connect("claudia", "12345678")  # conectarse a punto de acceso
    while not sta.isconnected():
        print("No conectado")
        cadena = "INTENTANDO CONEXION"
        mostrar_oled(oled, cadena)
        utime.sleep(1)
    cadena = "PUNTO DE ACCESO AL QUE SE CONECTO "
    mostrar_oled(oled, cadena)
    cadena = "NOMBRE:CLAUDIA PASS:12345678"
    mostrar_oled(oled, cadena)
    ip = sta.ifconfig()[0]
    print("Conectado con IP:", ip)
    cadena = "Conectado con IP; " + str(ip)
    mostrar_oled(oled, cadena)
    addr = socket.getaddrinfo('192.168.4.1', 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    while True:
        try:
            s.send(b"GET / HTTP/1.0\r\n\r\n") # enviar petición
            response = s.recv(1024)  # leer info / almacenar
            file.write(response)  # escribir en archivo
            print("Mensaje recibido: ", response)
            cadena = "INFORMACION RECIBIDA: "
            mostrar_oled(oled, cadena)
            mostrar_oled(oled, response)
            file.flush()
            cadena = "Mensaje recibido del transmisor"
            mostrar_oled(oled, cadena)
            # enviar ACK al transmisor
            ack_message = response + " +x2024"
            s.send(ack_message)  # enviar confirmación
            print("ACK enviado al transmisor")
            cadena = "Enviando ACK al transmisor" + str(ack_message)
            mostrar_oled(oled, cadena)
            s.close()
            break
        except OSError as e:
            s.close()
            print("Conexion cerrada")
            cadena = "Conexion cerrada"
            mostrar_oled(oled, cadena)

def main():
    oled = pantalla()
    recibo_wifi(oled)

if __name__ == "__main__":    main()
```

3.4. Código de Bluetooth

Bluetooth - Transmisor (recibo de ACK)	Bluetooth - Receptor (envío de ACK)

# ---------- FUNCIONES PANTALLA OLED ---------- # 	
def **pantalla():**
  WIDTH = 128
  HEIGHT = 64
  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=200000)
  oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
  return oled
 	
def **mostrar_oled(oled, message):**
  oled.fill(0)
  ancho_caracter = 7
  max_columna = 120
  fila = 0
  columna = 0
  palabras = message.split()
  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()
  utime.sleep(5)
  oled.fill(0)
  fila = 0
  columna = 0
  oled.text(palabra, columna, fila)
  columna = columna + 7
  columna += ancho_palabra + ancho_caracter
  oled.show()
  utime.sleep(5)
 	
# ---------- FUNCIONES PARA ESCANEAR, ESCOGER Y ENVIAR MENSAJE 	
# DEL DISPOSITIVO CENTRAL ---------- 	
def **leer(tope, oled):**
  entrada = 0
  primera_vuelta=True
  cadena = "Seleccione el dispositivo para enviar el mensaje."
  mostrar_oled(oled, cadena)
  while ((not isinstance(entrada, int)) or (not (entrada > 0 and entrada <= tope))):
  if not primera_vuelta:
  print("Ingrese una opción válida")
  try:
  entrada = int(input())
  except Exception:
  print("Ingrese una opción válida")
  primera_vuelta = False
  return entrada
 	
def _encode_datos(dato):
  return struct.pack("<s", dato)
	# ---------- FUNCIONES PANTALLA OLED ---------- #
	def **pantalla():**
	  WIDTH = 128
	  HEIGHT = 64
	  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=200000)
	  oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
	  return oled
	
	def **mostrar_oled(oled, message):**
	  oled.fill(0)
	  ancho_caracter = 7
	  max_columna = 120
	  fila = 0
	  columna = 0
	  palabras = message.split()
	  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()
	  utime.sleep(5)
	  oled.fill(0)
	  fila = 0
	  columna = 0
	  oled.text(palabra, columna, fila)
	  columna = columna + 7
	  columna += ancho_palabra + ancho_caracter
	  oled.show()
	  utime.sleep(5)
	
	# ---------- FUNCIONES RECIBIR DEL DISPOSITIVO CENTRAL ---------- #
	def _decode_datos(dato):
	  return struct.unpack("<s", dato)[0]
	
	def **avisar():**
	  led_onboard = machine.Pin(15, machine.Pin.OUT)
	  while True:
	  led_onboard.toggle()
	  utime.sleep(0.5)
	
	async def **peripheral_task(oled):**
	  cadena = "Esperando mensaje del PICO CENTRAL"
	  mostrar_oled(oled, cadena)
	  NOMBRE_DEL_BLUETOOTH = "claudia-receptor-picow"
	  _ENV_SENSE_UUID = bluetooth.UUID(0x181A)
	  _ENV_SENSE_TEMP_UUID = bluetooth.UUID(0x2A6E)
	  _ADV_INTERVAL_MS = 250_000
	  while True:
	  async with await aioble.advertise(
	  _ADV_INTERVAL_MS,

```
async def enviar(conexion, oled, mensaje):
    _ENV_SENSE_UUID = bluetooth.UUID(0x181A)
    _ENV_SENSE_TEMP_UUID = bluetooth.UUID(0x2A6E)

    temp_service = aioble.Service(_ENV_SENSE_UUID)
    temp_characteristic = aioble.Characteristic(temp_service, _ENV_SENSE_TEMP_UUID, read=True, notify=True)
    aioble.register_services(temp_service)
    global Palabras_a_enviar
    sleep_ms(500)
    palabra = mensaje
    tiempo = time.time_ns()
    i = 0
    while True:
        if time.time_ns() - tiempo >= 500_000_000:
            tiempo = time.time_ns()
            if i == len(palabra):
                temp_characteristic.write(_encode_datos("*"))
                break
            elif palabra[i] == " ":
                temp_characteristic.write(_encode_datos("/"))
            else:

temp_characteristic.write(_encode_datos(palabra[i]))
            i += 1
    cadena = "Mensaje enviado al receptor"
    mostrar_oled(oled, cadena)
    conexion.disconnected()

async def escaner(oled, mensaje):
    lista = []
    async with aioble.scan(5000, interval_us=30000, window_us=30000, active=True) as scanner:
        async for result in scanner:
            if not result in lista:
                if result.name() is not None:
                    lista.append(result)
    print("Dispositivos encontrados:")
    cont = 1
    for result in lista:
        print(str(cont) + "- " + result.name())
        cadena = "Dispositivo encontrado: " + str(cont) + "- " + result.name()
        mostrar_oled(oled, cadena)
        cont += 1
    opcion = leer(len(lista),oled)
    conexion = await lista[opcion-1].device.connect()
    asyncio.run(enviar(conexion, oled, mensaje))

# ---------- FUNCIONES PARA RECIBIR EL ACK DEL
# DISPOSITIVO RECEPTOR --------------
def _decode_datos(dato):
    return struct.unpack("<s", dato)[0]
async def peripheral_task(oled):
    NOMBRE_DEL_BLUETOOTH = "PICO_CENTRAL"
    _ENV_SENSE_UUID = bluetooth.UUID(0x181A)
            name=NOMBRE_DEL_BLUETOOTH,
            services=[_ENV_SENSE_UUID],
        ) as connection:
            print("Connection from", connection.device)
            async with connection:
                try:
                    temp_service = await connection.service(_ENV_SENSE_UUID)
                    temp_characteristic = await temp_service.characteristic(_ENV_SENSE_TEMP_UUID)
                except asyncio.TimeoutError:
                    return

                datos = []
                tiempo = time.time_ns()
                palabra = ""
                while True:
                    if time.time_ns() - tiempo >= 500_000_000:
                        tiempo = time.time_ns()
                        dato = str(_decode_datos(await temp_characteristic.read()),"utf-8")
                        print(dato, end="")
                        if dato == "/":
                            palabra += " "
                        elif dato == "*":
                            break
                        else:
                            palabra += dato
                print("\n\n", palabra)
                led_onboard = machine.Pin(15, machine.Pin.OUT)
                led_onboard.value(1)
                cadena = "Mensaje recibido: " + str(palabra) + " FIN DEL MENSAJE"
                mostrar_oled(oled, cadena)
                break
            break
    await connection.disconnected()
    return str(palabra)

# ---------- FUNCIONES PARA ENVIAR AL DISPOSITIVO CENTRAL ACK ----------
def _encode_datos(dato):
    return struct.pack("<s", dato)

async def enviar(conexion, oled, mensaje):
    _ENV_SENSE_UUID = bluetooth.UUID(0x181A)
    _ENV_SENSE_TEMP_UUID = bluetooth.UUID(0x2A6E)
    temp_service = aioble.Service(_ENV_SENSE_UUID)
    temp_characteristic = aioble.Characteristic(temp_service, _ENV_SENSE_TEMP_UUID, read=True, notify=True)
    aioble.register_services(temp_service)
    sleep_ms(500)
    palabra = mensaje
    tiempo = time.time_ns()
```

```
    _ENV_SENSE_TEMP_UUID = bluetooth.UUID(0x2A6E)

    _ADV_INTERVAL_MS = 250_000
    while True:
        async with await aioble.advertise(
            _ADV_INTERVAL_MS,
            name=NOMBRE_DEL_BLUETOOTH,
            services=[_ENV_SENSE_UUID],
        ) as connection:
            async with connection:
                try:
                    temp_service = await connection.service(_ENV_SENSE_UUID)
                    temp_characteristic = await temp_service.characteristic(_ENV_SENSE_TEMP_UUID)
                except asyncio.TimeoutError:
                    return
                datos = []
                tiempo = time.time_ns()
                palabra = ""
                while True:
                    if time.time_ns() - tiempo >= 500_000_000:
                        tiempo = time.time_ns()
                        dato = str(_decode_datos(await temp_characteristic.read()),"utf-8")
                        print(dato, end="")
                        if dato == "/":
                            palabra += " "
                        elif dato == "*":
                            break
                        else:
                            palabra += dato
                print("\n\n", palabra)
                led_onboard = machine.Pin(15, machine.Pin.OUT)
                led_onboard.value(1)
                cadena = "Mensaje ACK: " + str(palabra)
                mostrar_oled(oled, cadena)
                break
            break
    await connection.disconnected()
    return str(palabra)

def main():
    oled = pantalla()
    oled.fill(0)
    cadena = "DISPOSITIVO TRANSMISOR"
    mostrar_oled(oled, cadena)
    mensaje = "LUIS RAFAEL RODRIGUEZ DE SIO"
    cadena = "Mensaje a enviar: " + mensaje
    mostrar_oled(oled, cadena)
    asyncio.run(escaner(oled, mensaje))
    asyncio.run(peripheral_task(oled))

if __name__ == "__main__":
    main()
```

```
    i = 0
    while True:
        if time.time_ns() - tiempo >= 500_000_000:
            tiempo = time.time_ns()
            if i == len(palabra):
                temp_characteristic.write(_encode_datos("*"))
                break
            elif palabra[i] == " ":
                temp_characteristic.write(_encode_datos("/"))
            else:

temp_characteristic.write(_encode_datos(palabra[i]))
            i += 1
    cadena = "Mensaje enviado al transmisor"
    mostrar_oled(oled, cadena)
    conexion.disconnected()

async def escaner(oled, mensaje):
    lista = []
    async with aioble.scan(5000, interval_us=30000, window_us=30000, active=True) as scanner:
        async for result in scanner:
            if result not in lista:
                if result.name() is not None:
                    lista.append(result)
    cont = 1
    for result in lista:
        cont += 1
    pico_central_index = None
    for i, result in enumerate(lista):
        if result.name() == "PICO_CENTRAL":
            pico_central_index = i
            break
    if pico_central_index is not None:
        conexion = await lista[pico_central_index].device.connect()
        await enviar(conexion, oled, mensaje)
    else:
        print("ERROR: PICO_CENTRAL SE DESCONECTO.")
# ------ FUNCION MAIN -------
def main():
    led_onboard = machine.Pin(15, machine.Pin.OUT)
    led_onboard.value(0)
    oled = pantalla()
    oled.fill(0)
    cadena = "DISPOSITIVO RECEPTOR"
    mostrar_oled(oled, cadena)
    mensaje_recibido = asyncio.run(peripheral_task(oled))
    mensaje_recibido = mensaje_recibido + " 123456"
    cadena = "ENVIANDO ACK AL DISPOSITIVO CENTRAL"
    mostrar_oled(oled, cadena)
    asyncio.run(escaner(oled, mensaje_recibido))

if __name__ == "__main__":    main()
```