Códigos

#Tx

from machine import UART, Pin

import time

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1)) # UART0 en GPIO0 (TX) y GPIO1 (RX)

led = Pin(16, Pin.OUT) # LED externo en GPIO16

while True:

uart.write('A') # Envía 'A'

print("Enviado: 'A'")

time.sleep(2)

if uart.any():

response = uart.read(1) # Espera respuesta 'B'

print(f"Recibido: {response}")

# Parpadeo del LED (3 segundos)

for \_ in range(3):

led.value(1)

time.sleep(0.5)

led.value(0)

time.sleep(0.5)

#Rx

from machine import UART, Pin

import time

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

led = Pin(16, Pin.OUT)

counter = 0

with open('recibidos.txt', 'a') as f:

while True:

if uart.any():

data = uart.read(1).decode() # Decodifica a string

print(f"Recibido: '{data}'")

led.value(1) # Enciende LED

time.sleep(5) # Espera 5 segundos

led.value(0) # Apaga LED (¡esta línea es crítica!)

uart.write('B')

print("Enviado: 'B'")

counter += 1

f.write(f"{counter}\n")

f.flush()

time.sleep(0.1) # Pequeño delay para evitar bloqueos

#Full-Duplex

from machine import UART, Pin

import time

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

led = Pin(25, Pin.OUT)

counter = 0

while True:

# Envía un carácter cada 3 segundos

uart.write('A')

print("Enviado: 'A'")

# Espera 1 segundo antes de revisar recepciones

time.sleep(1)

# Revisa si hay datos recibidos

if uart.any():

data = uart.read(1)

print(f"Recibido: {data}")

led.value(1) # Enciende LED

time.sleep(0.5)

led.value(0)

time.sleep(2) # Espera total de 3 segundos por ciclo

Gpt gp16

# TX.py — MicroPython — LED externo en GPIO16

from machine import UART, Pin

import utime

# ---------- UART ----------

BAUD = 9600

uart = UART(0, baudrate=BAUD, bits=8, parity=None, stop=1, tx=Pin(0), rx=Pin(1), timeout=0)

# ---------- LED externo ----------

led = Pin(16, Pin.OUT)

PING = b'P' # Lo que enviamos

ACK = b'Q' # Lo que esperamos de respuesta

# Tiempos

PERIODO\_ENVIO\_MS = 2000

PARPADEO\_MS = 3000

PERIODO\_BLINK\_MS = 200

ultimo\_envio = utime.ticks\_ms() - PERIODO\_ENVIO\_MS

blink\_hasta = None

proximo\_toggle = 0

estado\_led = 0

while True:

ahora = utime.ticks\_ms()

# 1) Enviar 'P' cada 2 s

if utime.ticks\_diff(ahora, ultimo\_envio) >= PERIODO\_ENVIO\_MS:

uart.write(PING)

ultimo\_envio = ahora

# 2) Leer UART sin bloquear; si llega 'Q', parpadeo 3 s

if uart.any():

dato = uart.read(1)

if dato == ACK:

blink\_hasta = utime.ticks\_add(ahora, PARPADEO\_MS)

proximo\_toggle = ahora

# Limpia el buffer por si hay bytes extra

while uart.any():

uart.read(1)

# 3) Gestionar parpadeo no bloqueante

if blink\_hasta is not None:

if utime.ticks\_diff(blink\_hasta, ahora) > 0:

if utime.ticks\_diff(ahora, proximo\_toggle) >= 0:

estado\_led ^= 1

led.value(estado\_led)

proximo\_toggle = utime.ticks\_add(ahora, PERIODO\_BLINK\_MS)

else:

led.off()

estado\_led = 0

blink\_hasta = None

utime.sleep\_ms(10)

# RX.py — MicroPython — LED externo en GPIO16

from machine import UART, Pin

import utime

# ---------- UART ----------

BAUD = 9600

uart = UART(0, baudrate=BAUD, bits=8, parity=None, stop=1, tx=Pin(0), rx=Pin(1), timeout=0)

# ---------- LED externo ----------

led = Pin(16, Pin.OUT)

led.off()

PING = b'P'

ACK = b'Q'

LED\_ON\_MS = 5000

apagar\_led\_en = None

# Inicializa contador leyendo último número del archivo (si existe)

contador = 0

try:

with open("recibidos.txt", "r") as f:

for linea in f:

pass

ultima = linea.strip() if 'linea' in locals() else ""

if ultima.isdigit():

contador = int(ultima)

except:

pass

def registra\_recepcion():

global contador

contador += 1

with open("recibidos.txt", "a") as f:

f.write(str(contador) + "\n")

while True:

ahora = utime.ticks\_ms()

# 1) Recepción

if uart.any():

dato = uart.read(1)

if dato == PING:

led.on()

apagar\_led\_en = utime.ticks\_add(ahora, LED\_ON\_MS)

uart.write(ACK)

registra\_recepcion()

# Limpia posibles bytes residuales

while uart.any():

uart.read(1)

# 2) Apagar LED a los 5 s (no bloqueante)

if apagar\_led\_en is not None and utime.ticks\_diff(apagar\_led\_en, ahora) <= 0:

led.off()

apagar\_led\_en = None

utime.sleep\_ms(10)

# full\_duplex.py — MicroPython — LED externo en GPIO16 (cargar en ambos Picos)

from machine import UART, Pin

import utime

# ---------- UART ----------

BAUD = 9600

uart = UART(0, baudrate=BAUD, bits=8, parity=None, stop=1, tx=Pin(0), rx=Pin(1), timeout=0)

# ---------- LED externo ----------

led = Pin(16, Pin.OUT)

led.off()

PING = b'P'

ACK = b'Q'

# Tiempos

PERIODO\_ENVIO\_MS = 2000

LED\_RX\_MS = 5000

LED\_ACK\_BLINK\_MS = 3000

PERIODO\_BLINK\_MS = 200

ultimo\_ping = utime.ticks\_ms() - PERIODO\_ENVIO\_MS

# Estados LED

apagar\_led\_en = None # LED fijo tras 'P' recibido

blink\_hasta = None # parpadeo tras 'Q' recibido

proximo\_toggle = 0

estado\_led = 0

# Contador de recepciones (también guarda archivo)

contador = 0

try:

with open("recibidos.txt", "r") as f:

for linea in f:

pass

ultima = linea.strip() if 'linea' in locals() else ""

if ultima.isdigit():

contador = int(ultima)

except:

pass

def registra\_recepcion():

global contador

contador += 1

try:

with open("recibidos.txt", "a") as f:

f.write(str(contador) + "\n")

except:

pass

while True:

ahora = utime.ticks\_ms()

# 1) Transmisión periódica (P) cada 2 s

if utime.ticks\_diff(ahora, ultimo\_ping) >= PERIODO\_ENVIO\_MS:

uart.write(PING)

ultimo\_ping = ahora

# 2) Recepción y reacciones

if uart.any():

dato = uart.read(1)

if dato == PING:

# Al recibir P: LED 5 s y responder Q

led.on()

apagar\_led\_en = utime.ticks\_add(ahora, LED\_RX\_MS)

uart.write(ACK)

registra\_recepcion()

elif dato == ACK:

# Al recibir Q: parpadeo 3 s

blink\_hasta = utime.ticks\_add(ahora, LED\_ACK\_BLINK\_MS)

proximo\_toggle = ahora

# Vaciar bytes residuales

while uart.any():

uart.read(1)

# 3) LED fijo por 'P'

if apagar\_led\_en is not None and utime.ticks\_diff(apagar\_led\_en, ahora) <= 0:

led.off()

apagar\_led\_en = None

# 4) Parpadeo por 'Q'

if blink\_hasta is not None:

if utime.ticks\_diff(blink\_hasta, ahora) > 0:

if utime.ticks\_diff(ahora, proximo\_toggle) >= 0:

estado\_led ^= 1

led.value(estado\_led)

proximo\_toggle = utime.ticks\_add(ahora, PERIODO\_BLINK\_MS)

else:

led.off()

estado\_led = 0

blink\_hasta = None

utime.sleep\_ms(10)

DEEP

#tx

from machine import UART, Pin

import utime

# Configuración UART - Usaremos UART0 (GPIO0=TX, GPIO1=RX)

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

# Configuración del LED externo en GPIO16

led = Pin(16, Pin.OUT)

led.off() # Aseguramos que starts apagado

# Caracteres de control

CHAR\_ENVIO = b'A' # Carácter que envía este dispositivo

CHAR\_RESPUESTA = b'B' # Carácter que espera recibir

# Temporizadores (usamos ticks\_ms para gestión no bloqueante)

intervalo\_envio = 2000 # Ms entre envíos

tiempo\_parpadeo = 3000 # Ms que debe durar el parpadeo

periodo\_parpadeo = 500 # Ms entre cada toggle del LED

ultimo\_envio = utime.ticks\_ms() - intervalo\_envio

tiempo\_fin\_parpadeo = None

proximo\_cambio\_led = 0

estado\_led\_parpadeo = False

parpadeando = False

print("Iniciando dispositivo TRANSMISOR (TX)...")

print(f"Enviando '{CHAR\_ENVIO.decode()}' cada {intervalo\_envio//1000} segundos.")

print(f"Esperando respuesta '{CHAR\_RESPUESTA.decode()}' para parpadear LED.")

while True:

tiempo\_actual = utime.ticks\_ms()

# 1. LÓGICA DE TRANSMISIÓN: Enviar 'A' cada 2 segundos

if utime.ticks\_diff(tiempo\_actual, ultimo\_envio) >= intervalo\_envio:

uart.write(CHAR\_ENVIO)

print(f"[{tiempo\_actual}] Enviado: '{CHAR\_ENVIO.decode()}'")

ultimo\_envio = tiempo\_actual

# 2. LÓGICA DE RECEPCIÓN: Buscar respuesta 'B'

if uart.any():

dato\_recibido = uart.read(1)

if dato\_recibido == CHAR\_RESPUESTA:

print(f"[{tiempo\_actual}] ¡Respuesta '{CHAR\_RESPUESTA.decode()}' recibida! Iniciando parpadeo.")

parpadeando = True

tiempo\_fin\_parpadeo = utime.ticks\_add(tiempo\_actual, tiempo\_parpadeo)

proximo\_cambio\_led = tiempo\_actual

# Limpiar buffer para evitar procesar datos residuales

while uart.any():

uart.read(1)

# 3. LÓGICA DE PARPADEO: Control no bloqueante del LED

if parpadeando:

# Comprobar si toca cambiar el estado del LED

if utime.ticks\_diff(tiempo\_actual, proximo\_cambio\_led) >= 0:

estado\_led\_parpadeo = not estado\_led\_parpadeo

led.value(estado\_led\_parpadeo)

proximo\_cambio\_led = utime.ticks\_add(tiempo\_actual, periodo\_parpadeo)

# Comprobar si el tiempo de parpadeo ha terminado

if utime.ticks\_diff(tiempo\_actual, tiempo\_fin\_parpadeo) >= 0:

led.off()

parpadeando = False

estado\_led\_parpadeo = False

print(f"[{tiempo\_actual}] Parpadeo finalizado.")

# Pequeña pausa para reducir carga de CPU

utime.sleep\_ms(10)

#RX

from machine import UART, Pin

import utime

# Configuración UART - Usaremos UART0 (GPIO0=TX, GPIO1=RX)

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

# Configuración del LED externo en GPIO16

led = Pin(16, Pin.OUT)

led.off()

# Caracteres de control

CHAR\_ESPERADO = b'A' # Carácter que espera recibir

CHAR\_RESPUESTA = b'B' # Carácter que envía como respuesta

# Tiempo que el LED permanecerá encendido (ms)

tiempo\_led\_on = 5000

# Variables de estado para el LED

tiempo\_apagado\_led = None

led\_activo = False

# Inicializar el contador leyendo el último valor del archivo

contador = 0

nombre\_archivo = "recibidos.txt"

try:

# Intentar abrir el archivo y leer la última línea

with open(nombre\_archivo, "r") as archivo:

lineas = archivo.readlines()

if lineas:

ultima\_linea = lineas[-1].strip()

if ultima\_linea.isdigit():

contador = int(ultima\_linea)

print(f"Contador inicializado desde archivo: {contador}")

except OSError:

# Si el archivo no existe, se creará luego

print("Archivo no encontrado. Iniciando contador desde 0.")

def guardar\_recepcion():

"""Función para incrementar el contador y guardarlo en el archivo"""

global contador

contador += 1

try:

with open(nombre\_archivo, "a") as archivo:

archivo.write(str(contador) + "\n")

print(f"Recepción #{contador} guardada en {nombre\_archivo}")

except OSError as e:

print(f"Error guardando en archivo: {e}")

print("Iniciando dispositivo RECEPTOR (RX)...")

print(f"Esperando carácter '{CHAR\_ESPERADO.decode()}' para encender LED y responder con '{CHAR\_RESPUESTA.decode()}'.")

while True:

tiempo\_actual = utime.ticks\_ms()

# 1. LÓGICA DE RECEPCIÓN: Buscar el carácter 'A'

if uart.any():

dato\_recibido = uart.read(1)

if dato\_recibido == CHAR\_ESPERADO:

print(f"[{tiempo\_actual}] Carácter '{CHAR\_ESPERADO.decode()}' recibido.")

# Encender LED y programar apagado

led.value(1)

led\_activo = True

tiempo\_apagado\_led = utime.ticks\_add(tiempo\_actual, tiempo\_led\_on)

# Enviar respuesta INMEDIATA

uart.write(CHAR\_RESPUESTA)

print(f"[{tiempo\_actual}] Respuesta '{CHAR\_RESPUESTA.decode()}' enviada.")

# Guardar en archivo y aumentar contador

guardar\_recepcion()

# Limpiar buffer para evitar procesar datos residuales

while uart.any():

uart.read(1)

# 2. LÓGICA DEL LED: Apagarlo tras 5 segundos

if led\_activo and utime.ticks\_diff(tiempo\_actual, tiempo\_apagado\_led) >= 0:

led.value(0)

led\_activo = False

print(f"[{tiempo\_actual}] LED apagado.")

# Pequeña pausa para reducir carga de CPU

utime.sleep\_ms(10)

# TX.py — LED externo en GPIO16 — versión mínima corregida

from machine import UART, Pin

import time

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1)) # UART0 (GP0 TX, GP1 RX)

led = Pin(16, Pin.OUT) # LED en GPIO16

led.value(0)

PING = b'A' # Lo que enviamos

ACK = b'B' # Lo que esperamos del RX

while True:

# 1) Enviar 'A' cada 2 s

uart.write(PING)

print("Enviado: 'A'")

time.sleep(2)

# 2) Revisar si llegó ACK

if uart.any():

resp = uart.read(1) # lee 1 byte

# Drenar bytes extra si hubiera (ruido, ecos, etc.)

while uart.any():

uart.read(1)

print("Recibido:", resp)

# 3) Parpadeo SOLO si es 'B'

if resp == ACK:

# Parpadeo 3 s (3 ciclos de 0.5 on + 0.5 off)

for \_ in range(3):

led.value(1); time.sleep(0.5)

led.value(0); time.sleep(0.5)

# RX.py — versión mínima corregida (LED en GP16, bloqueante con sleep)

from machine import UART, Pin

import time

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

led = Pin(16, Pin.OUT)

led.value(0)

counter = 0

with open('recibidos.txt', 'a') as f:

while True:

if uart.any():

dato = uart.read(1) # lee 1 byte

# Drenar posibles bytes residuales

while uart.any():

uart.read(1)

if dato == b'A': # SOLO si recibió 'A'

print("Recibido: 'A'")

led.value(1) # Enciende LED

uart.write(b'B') # Responde de inmediato con 'B'

print("Enviado: 'B'")

counter += 1

f.write(f"{counter}\n") # 1 línea por recepción

f.flush()

time.sleep(5) # Mantiene LED 5 s (bloquea)

led.value(0) # Apaga LED

time.sleep(0.01) # pequeño respiro