COMUNICACIÓN DIGITAL – COMUNICACIÓN INALÁMBRICA ENTRE DISPOSITIVOS nRF24L01 (octubre de 2025)

|Fernando Javier Riaño Rios
Est.fernando.riano@unimilitar.edu.co

Resumen - En este laboratorio implementé un sistema inalámbrico punto a punto usando dos Raspberry Pi Pico 2W y módulos nRF24L01 para transmitir el ángulo de un joystick (TX) y controlar un servomotor (RX) con respuesta inmediata. Definí una trama compacta de 4 bytes con byte de sincronismo (0xA5), ángulo en 16 bits y checksum (suma módulo 256). Configuré el radio en canal 100, 2 Mbps y 0 dBm, deshabilitando Auto-ACK y retransmisiones para minimizar latencia. En el receptor integré un acelerómetro MPU6050 por I²C con una calibración de 2 s en reposo para remover sesgos y mostré en una OLED SSD1306 el ángulo, ax, ay, az y |a|, limitando la tasa de refresco para no bloquear la recepción. El sistema alcanzó 100 actualizaciones por segundo desde el TX, movimientos suaves del servo (500–2500 µs a 50 Hz) y telemetría estable en pantalla y consola. Documenté conexiones, registros críticos del nRF24L01 y pruebas básicas de funcionamiento que confirman la viabilidad del enlace de baja latencia.

Abstract – In this laboratory, I implemented a point-to-point wireless system using two Raspberry Pi Pico 2W boards and nRF24L01 modules to transmit the angle of a joystick (TX) and control a servo motor (RX) with immediate response. A compact 4-byte frame was defined with a synchronization byte (0xA5), a 16-bit angle field, and a simple checksum (sum modulo 256). The radio was configured on channel 100 at 2 Mbps and 0 dBm, with Auto-ACK and retransmissions disabled to minimize latency. On the receiver side, an MPU6050 accelerometer was integrated through I²C with a 2-second rest calibration to remove offsets, and an SSD1306 OLED display was used to show the angle, ax, ay, az, and |a| values with throttled updates to avoid blocking reception. The system achieved around 100 transmissions per second from the TX, smooth servo motion (500–2500 µs at 50 Hz), and stable telemetry on both the display and console. Hardware wiring, critical nRF24L01 register configurations, and functional near-field tests (< 1 m) confirmed the feasibility of this low-latency wireless control link.

Palabras clave - NRF24L01, RASPBERRY PI PICO 2W, SPI, I²C, MPU6050, OLED SSD1306, SERVOMOTOR.

INTRODUCCIÓN

El objetivo del trabajo fue construir y validar un enlace inalámbrico de baja latencia para controlar un servomotor desde un joystick, utilizando módulos nRF24L01 y dos Raspberry Pi Pico 2W. En el transmisor (TX) muestreamos el joystick por ADC y enviamos el ángulo (0–180°) como una trama corta a 2 Mbps. En el receptor (RX) aplicamos el ángulo al servo en tiempo real y, en paralelo, leemos un MPU6050 para mostrar ax, ay, az y la magnitud |a| en una OLED. La práctica integra electrónica digital (SPI/I²C/ADC/PWM), configuración de radio en 2.4 GHz y estrategias simples de integridad (byte de sincronismo y checksum), priorizando latencia sobre fiabilidad automática (Auto-ACK off) para lograr una sensación de control inmediato.

DESARROLLO DEL LABORATORIO

Para el desarrollo del laboratorio utilizamos principalmente los siguientes elementos:

- **Plataforma:** Raspberry Pi Pico 2W(dos). Protoboard.
- **Radio:** nRF24L01 (SPI0). **TX y RX**
- Pantalla OLED (SSD1306)
- MPU6050 (solo en RX)
- **Servo (RX):** PWM
- Joystick (TX)

A **cada Raspberry Pi Pico** copiamos los siguientes archivos para que el programa funcione correctamente al encenderlas.

- `nrf24l01.py` (driver)

Driver del módulo de radio nRF24L01. Contiene las funciones SPI y registros para enviar/recibir datos.

- `ssd1306.py`

Driver para controlar la pantalla OLED (I2C).

- `main.py` (el de TX o RX según la placa)

Programa principal (puede ser el de TX o RX, según la placa). Uno diferente en cada pico, es el que se ejecuta automáticamente al encender el Pico, inicialmente no los llame main.py sino que serán Tx,py y Rx.py.

Estos archivos se encuentran en mi repositorio GitHub https://github.com/fernandoriano/Comunicaci-n-inal-mbrica-entre-dispositivos-nRF24L01.git

Elementos y Conexiones:

Raspberry Pi Pico 2

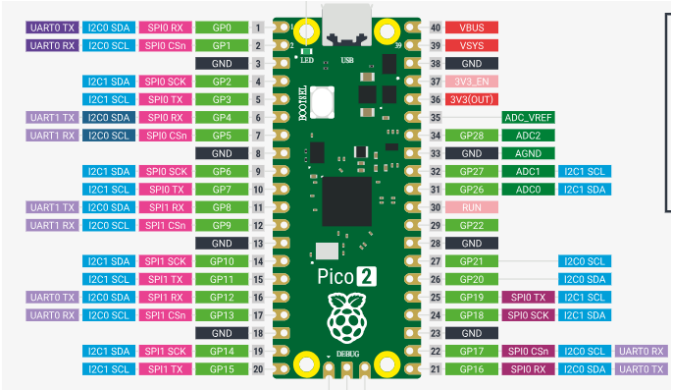


Ilustración 1 Distribución de pines de la Raspberry Pi Pico 2

🔴 **Power:** Pines de alimentación, como VBUS (5 V) y 3V3(OUT), utilizados para suministrar energía a periféricos y sensores.

⚫ **Ground (GND):** Pines de referencia eléctrica comunes para todos los circuitos.

🟢 **GPIO** (General Purpose Input/Output): Pines configurables para entrada o salida digital, así como para control PWM.

🟣 **UART:** Pines dedicados a comunicación serial (TX y RX).

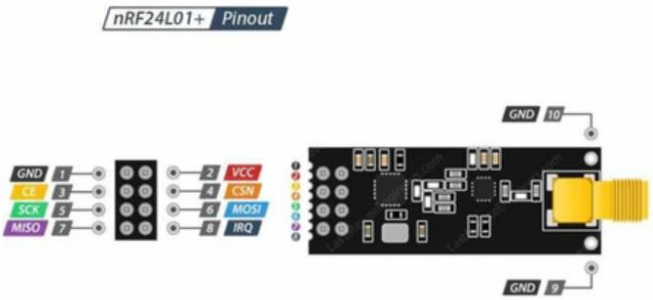
🔵 **I2C:** Pines para comunicación mediante el protocolo I²C, indicados como **SDA** (datos) y **SCL** (reloj).

🟠 **SPI:** Pines usados para el protocolo SPI (SCK, MOSI, MISO y CSn)

🟩 **ADC:** Entradas analógicas (GP26, GP27, GP28) que permiten leer valores de sensores analógicos.

🟤 **Debug / System control:** Pines especiales para control o depuración.

nRF24L01



- VCC → **3V3(OUT) pin 36**
- GND → **GND pin 38**
- CE → **GP14**
- CSN → **GP5 (el gp15 se usará para el servo)**
- SCK → **GP6**
- MOSI → **GP7**
- MISO → **GP4**

SCK (Serial Clock): La Pico genera las pulsaciones de reloj que sincronizan la transferencia de bits.
MOSI (Master Out Slave In): Línea por la que el Pico envía datos al nRF24L01.
MISO (Master In Slave Out): Línea por la que el nRF24L01 devuelve datos al Pico.
CSN (Chip Select Not): Activa el módulo nRF24L01 cuando el Pico desea comunicarse.
CE (Chip Enable): Controla si el módulo está en modo transmisión o recepción.

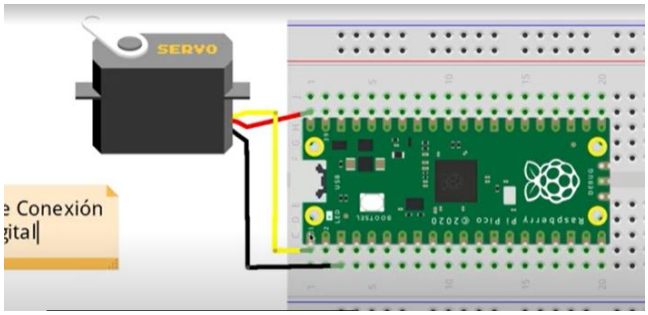
OLED SSD1306

OLED SSD1306	Pico pin físico	Pico GPIO	Descripción
GND	38	GND	Tierra
VCC	36	3V3 (OUT)	Alimentación
SCL	15	GP11	I2C1 SCL
SDA	14	GP10	I2C1 SDA
Dirección I2C	—	0x3C	ya configurada en el código

Joystick TX (emisora)

Pin del joystick	Función	Conexión en el Pico
VCC	Alimentación (5 V)	→ **VBUS** del Pico (pin físico **40**)
GND	Tierra	→ **GND** del Pico (por ejemplo, pin **38**)
VRx	Eje X (señal analógica)	→ **GP26** (pin físico **31**)
VRy	Eje Y (opcional por ahora)	→ **GP27** (pin físico **32**)
SW	Botón (opcional)	→ **GP22** (pin físico **29**) con `Pin.IN, Pin.PULL_UP`

Servo - RX (receptora)



- Señal servo → **GP15 - amarillo**
- +5V servo → **VBUS (5V del USB) pin 40 - rojo**
- GND servo → **GND del Pico-negro** **Importante:** NO alimentar el servo desde 3V3. Usar el **VBUS** (5V del USB) y **masa común** con el Pico.

MPU6050 (acelerómetro + giroscopio)

Pin MPU6050	Pico pin físico	Pico GPIO	Descripción
VCC	36	3V3 (OUT)	Alimentación 3.3 V (⚠ NO 5 V)
GND	38	GND	Tierra común
SCL	22	GP17	Mismo bus I2C1 que la OLED
SDA	21	GP16	Mismo bus I2C1 que la OLED
AD0	—	—	**Conéctalo a GND** → dirección **0x68** (si lo pones a 3V3, será 0x69)
INT	—	—	Dejar sin conectar (no se usa)
XDA / XCL	—	—	Dejar sin conectar (solo sirven para sensores externos)

Una vez realizadas las conexiones procedemos a implementar el código para realizar la comunicación y funcionamiento de los elementos, en el repositorio se encontrarán, nombrados como Rx final y TxT. los dos códigos resultantes con mejor funcionalidad, después de probar con muchas variables de potencia, canal y velocidad, a continuación explicare algunos apartes del código realizado:

Configuración Radio NRF24L01

```
nrf.set_power_speed(0, 2)  # Potencia: 0dBm, Velocidad: 2Mbps
nrf.reg_write(0x05, 100)   # Canal: 100
```

Configuración de potencia, velocidad y canal de transmisión.

Parámetros modificables (algunas posibilidades):

Potencia (0-3):

- 0 = -18dBm (0.016mW)
- 1 = -12dBm (0.063mW)
- 2 = -6dBm (0.25mW)
- 3 = 0dBm (1mW) ← **Máxima potencia**

Para más potencia se necesitaría Módulos con amplificador (ejemplo): nRF24L01+PA+LNA = hasta 20dBm (100mW)

Velocidad (0-2)

- 0 = 250kbps ← Máximo alcance
- 1 = 1Mbps ← Balance
- 2 = 2Mbps ← Máxima velocidad

Podemos modificar estos valores de acuerdo a nuestros requerimientos, más potencia y menos velocidad = Mayor alcance (hasta 100m)

Menos potencia y más velocidad = Menor alcance (10-20m) pero mayor respuesta.

configuración servo motor - Control PWM del Servo.

```
def mover_servo_instantaneo(angulo):
    angulo = max(0, min(180, int(angulo)))
    pulso_us = 500 + (angulo * 2000) // 180
    servo.duty_ns(pulso_us * 1000)
```

Conversión de ángulos a señales PWM para control servo.

Límites angulares: max(0, min(180, angulo))
Pulso mínimo: 500 µs (0°)
Pulso máximo: 2500 µs (180°)
Rango total: 2000 µs

configuración acelerómetro mpu6050 - Lectura y Calibración.

```
def mpu_read_accel_mps2():
    # Conversión a m/s²
    ax_mps2 = (ax / 16384.0) * 9.80665
```

Lectura de aceleración en 3 ejes y conversión a m/s².
Escala: 16384.0 LSB/g para ±2g
Calibración: Offsets automáticos en 2 segundos
Frecuencia muestreo: utime.sleep_ms(10) en calibración

Efectos de variación:
Cambiar escala: Ajustar sensibilidad (±2g, ±4g, ±8g, ±16g), Tiempo calibración: Aumentar para mayor precisión, Filtrado: Suavizar datos con promedio móvil.

protocolo de comunicación

```
nrf = NRF24L01(spi, csn, ce, payload_size=4)  # 4 bytes fijos

# TX envía (4 bytes):
paquete = struct.pack("<BHB", SYNC_BYTE, angulo_actual, checksum)
# RX recibe y verifica:
sync, angulo, checksum = struct.unpack("<BHB", datos)
```

Protocolo personalizado con sincronización y checksum.
SYNC_BYTE: 0xA5 (byte de sincronización), Checksum: Verificación de integridad. Tamaño payload: 4 bytes fijo.

Efectos de variación:
Cambiar SYNC_BYTE: Evitar interferencias con otros sistemas
Agregar más datos: Aumentar tamaño de payload
Implementar ACK: Mayor confiabilidad, pero menor velocidad.

Control de Frecuencia.

```
utime.sleep_ms(10)  # TX: 100 FPS
utime.sleep_us(500) # RX: Alta velocidad de respuesta
```

Parámetros modificables:
TX: 10ms = 100 FPS
RX: 500µs = respuesta ultra rápida

Efectos de variación:
Aumentar sleep: Menor consumo, mayor latencia
Reducir sleep: Mayor consumo, menor latencia
Optimizar: Balance entre respuesta y estabilidad

En general podemos modificar varios parámetros de acuerdo a nuestros requerimientos, resumiendo:

- Para máximo alcance: nrf.set_power_speed(3, 0)
- Para máxima velocidad: nrf.set_power_speed(0, 2) (configuración actual)
- Para bajo consumo: Aumentar utime.sleep_ms()
- Para mayor precisión: Incrementar tiempo de calibración MPU6050
- Para diferentes servos: Ajustar rangos PWM en mover_servo_instantaneo()

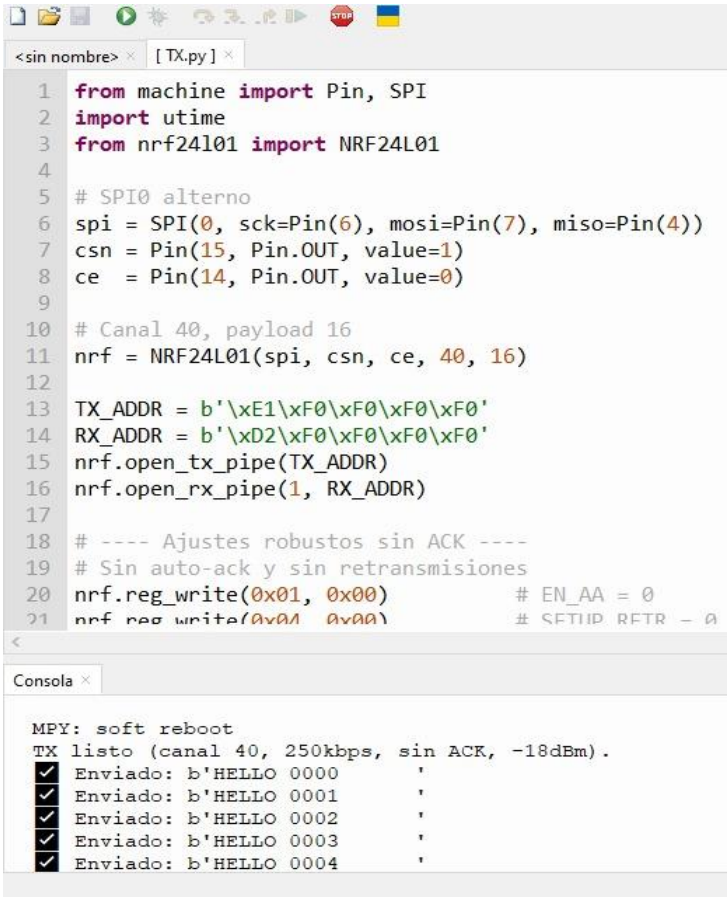


Ilustración 2 prueba de conexión. TX.

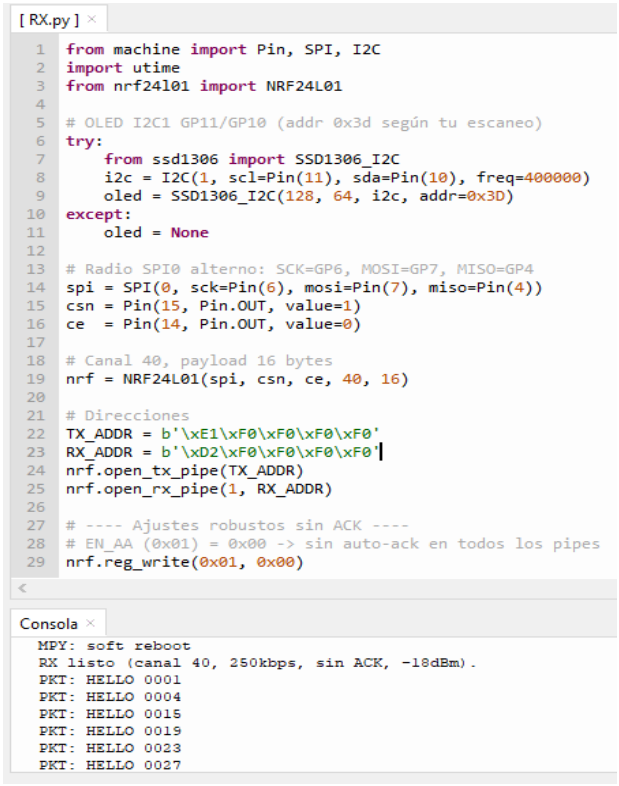


Ilustración 3 prueba de conexión. Rx.

La ilustración 2 corresponde a la prueba inicial del transmisor (TX) con el módulo nRF24L01 conectado a la Raspberry Pi Pico 2W. En el código se observa la configuración del bus SPI0 con los pines GP6, GP7, GP4, GP15 y GP14, además del establecimiento de direcciones y parámetros del radio en el canal 40, con una carga útil de 16 bytes y sin Auto-ACK. En la consola de Thonny se visualiza el mensaje “TX listo (canal 40, 250kbps, sin ACK, -18dBm)” seguido de varios envíos del texto “HELLO” numerado, lo que confirma que el módulo transmisor está funcionando correctamente y enviando datos de forma continua, verificando la comunicación SPI y la transmisión inalámbrica antes de integrar el sistema completo.

La Ilustración 3 muestra el funcionamiento del receptor (RX) con el módulo nRF24L01 configurado en los mismos parámetros que el transmisor: canal 40, 250 kbps y sin Auto-ACK. En el código se aprecia la inicialización del bus I²C1 para una pantalla OLED SSD1306 y del bus SPI0 para el radio, además de la apertura de las direcciones TX y RX correspondientes. En la consola se observa el mensaje “RX listo (canal 40, 250kbps, sin ACK, -18dBm)” junto a la recepción de varios paquetes “HELLO” con numeración progresiva, lo que demuestra que el enlace inalámbrico entre las dos Raspberry Pi Pico 2W fue exitoso y que el receptor está recibiendo correctamente las tramas enviadas desde el transmisor.

Sin auto ACK por que se está priorizando velocidad sobre confiabilidad, lo cual es adecuado para control del servo donde se quiere respuesta inmediata al movimiento del joystick.
Es de aclarar que estos parámetros fueron modificados posteriormente, se usaron para establecer la conexión, pero posteriormente al agregar el servomotor y demás componentes se fueron modificando para un funcionamiento optimo.

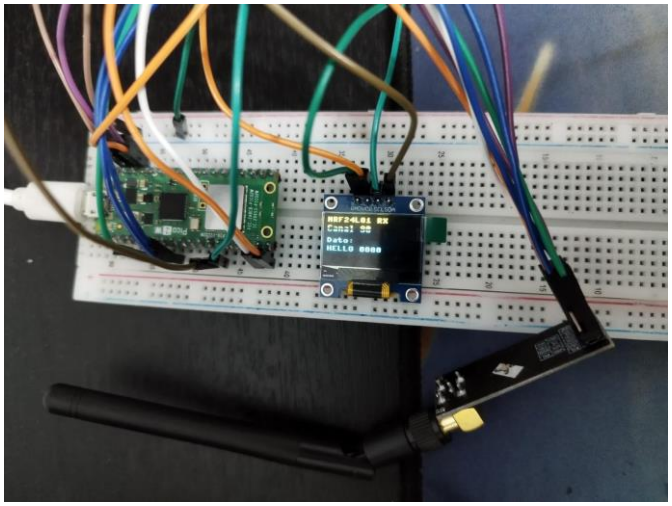


Ilustración 4 conexión exitosa

La imagen muestra el montaje físico del módulo receptor (RX) del sistema inalámbrico con nRF24L01, construido sobre una protoboard. En la parte izquierda se observa la Raspberry Pi Pico 2W, encargada de procesar los datos recibidos y comunicarse con los demás componentes. En el centro se encuentra la pantalla OLED SSD1306, conectada mediante el bus I²C, que muestra en su display la información del enlace: “NRF24L01 RX”, el canal, la velocidad de 2 Mbps y el mensaje recibido “HELLO 0000”, confirmando la correcta recepción de datos desde el transmisor. En la parte inferior derecha se distingue el módulo nRF24L01 con antena externa, conectado al bus SPI de la Pico, encargado de la comunicación inalámbrica a 2.4 GHz. Los cables de colores representan las conexiones de alimentación, datos y control entre los tres dispositivos, evidenciando un montaje funcional para las pruebas prácticas del enlace RX.

```
ultimo = 90
servo_deg(ultimo)

print("✅ RX servo+OLED listo (ch40, 250kbps, sin ACK, -18dBm, CSN→GP5)")

# ---------- Bucle principal ----------
while True:
    while nrf.any():
        raw = nrf.recv()   # 4 bytes recibidos
        try:
            s, ang, c = struct.unpack("<BHB", raw)
        except:
            continue

        ok = (s == SYNC) and (c == chk(s, ang)) and (0 <= ang <= 180)
        if ok:
            ultimo = ang
            servo_deg(ultimo)

            # Muestra en OLED
            oled.fill(0)
            oled.text("NRF24 RX Servo", 0, 0)
            oled.text("Ang: %3d deg" % ultimo, 0, 16)
            oled.text("Canal 40", 0, 32)
            oled.text("Vel 250kbps", 0, 44)
            oled.show()
            print("RX ang:", ultimo)
    utime.sleep_ms(5)
```

```
Consola
RX ang: 140
RX ang: 140
RX ang: 180
RX ang: 180
RX ang: 180
RX ang: 0
RX ang: 0
RX ang: 0
RX ang: 137
```

Ilustración 5 datos mostrados en consola Tx con joystick.

```
            mover_servo_instantaneo(angulo)
            contador_paquetes += 1

            # Leer acelerómetro
            ax, ay, az = mpu_read_accel_mps2()
            ax -= off_ax
            ay -= off_ay
            az -= off_az
            a_total = math.sqrt(ax**2 + ay**2 + az**2)
```

```
Consola
ANG:140° | ax:  0.030 | ay:  0.044 | az:  9.832 | |a|:  9.832
ANG:140° | ax: -0.030 | ay:  0.003 | az:  9.734 | |a|:  9.734
ANG:140° | ax: -0.034 | ay: -0.088 | az:  9.810 | |a|:  9.811
ANG:140° | ax:  0.076 | ay: -0.026 | az:  9.755 | |a|:  9.755
ANG:180° | ax:  0.057 | ay: -0.016 | az:  9.827 | |a|:  9.827
ANG:140° | ax: -0.039 | ay: -0.021 | az:  9.841 | |a|:  9.841
ANG:141° | ax: -0.025 | ay:  0.036 | az:  9.777 | |a|:  9.777
ANG:140° | ax:  0.023 | ay:  0.084 | az:  9.803 | |a|:  9.803
ANG:141° | ax:  0.016 | ay:  0.027 | az:  9.734 | |a|:  9.734
```

Ilustración 6 datos mostrados en consola Rx, con servo y acelerómetro.

En la ilustración 5 se observa la ejecución final del código del **transmisor (TX)** ya con la lectura del **joystick** y el envío continuo del ángulo hacia el receptor. En el código se aprecia la función `chk()` encargada de calcular el **checksum** de cada paquete, garantizando la integridad de los datos transmitidos. Durante el bucle principal, el valor del ángulo (`ang`) se obtiene a partir del joystick, se empaqueta en un formato de 4 bytes y se envía mediante el módulo **nRF24L01**. En la parte inferior, la consola de Thonny muestra valores como "TX ang: 180" y "TX ang: 42", confirmando que el transmisor detecta los cambios de posición del joystick y los envía correctamente al receptor a través del enlace inalámbrico.

La ilustración 6 corresponde al funcionamiento del receptor (RX) con el montaje completo, que incluye el acelerómetro MPU6050, la pantalla OLED y el servomotor. En la parte superior se aprecia el fragmento del código que lee los datos del acelerómetro (ax, ay, az) y calcula la magnitud total de la aceleración (|a|) en metros por segundo al cuadrado. La consola muestra una secuencia continua de lecturas, donde se registran el ángulo recibido desde el TX, las aceleraciones en los tres ejes y la magnitud total. Los valores reflejan que el sistema está recibiendo correctamente los datos del joystick, moviendo el servo en tiempo real y mostrando en pantalla la información del sensor, demostrando así la sincronización completa del enlace inalámbrico y la correcta integración del acelerómetro al sistema RX.

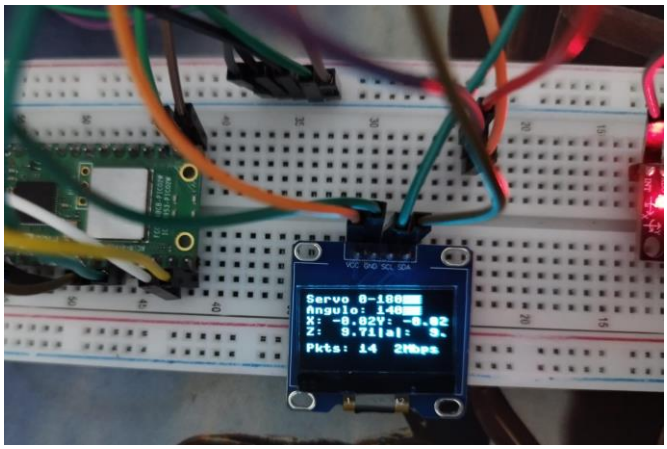


Ilustración 7 Oled Rx con datos de servo y acelerómetro.

La imagen muestra el montaje final del receptor (RX) completamente operativo, donde la Raspberry Pi Pico 2W está conectada a una pantalla OLED SSD1306 que visualiza en tiempo real los datos recibidos del transmisor y las lecturas del acelerómetro MPU6050. En la pantalla se observan el ángulo actual del servomotor (140°), las aceleraciones en los ejes X, Y y Z, la magnitud total de aceleración (|a|) y el conteo de paquetes recibidos (Pkts: 14) junto a la velocidad de transmisión de 2 Mbps.

Una vez realizado el código se procedio a realizar las pruebas como se evidencia en el siguiente video: https://github.com/fernandoriano/Comunicaci-n-inal-mbrica-entre-dispositivos-nRF24L01/blob/24c868f6b182f6ca8b54a028c096ff4a3045980a/VID-20251026-WA0028.mp4

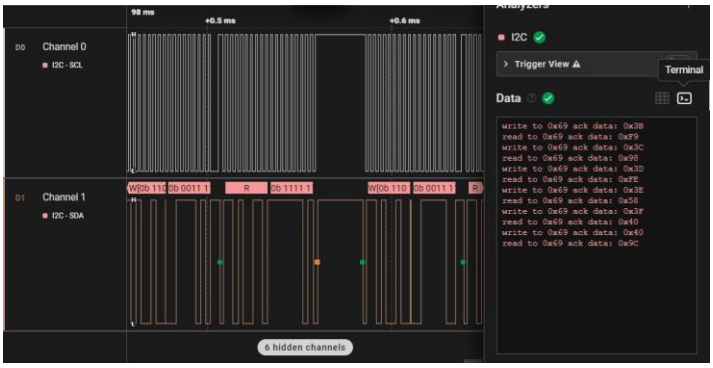


Ilustración 8 Comunicación I²C con el sensor MPU6050

La imagen muestra la captura del bus I²C realizada con el analizador lógico durante la comunicación entre la Raspberry Pi Pico 2W y el sensor MPU6050, utilizado en el sistema receptor. En los canales se observan las líneas SCL (reloj) y SDA (datos) con la secuencia típica de escritura y lectura de registros del sensor en la dirección 0x69, la cual coincide con la configurada en el código (MPU_ADDR = 0x68 o 0x69 según AD0). En la terminal lateral se listan las operaciones ejecutadas, como write to 0x69 ack data: 0x3B y read to 0x69 ack data: 0x9C, que corresponden a la lectura de los registros de aceleración. Esta traza confirma que la comunicación I²C está funcionando correctamente, que el sensor responde con señales de reconocimiento (ACK) en cada transmisión, y que los datos enviados y recibidos coinciden con el proceso de calibración y lectura implementado en el código del receptor.

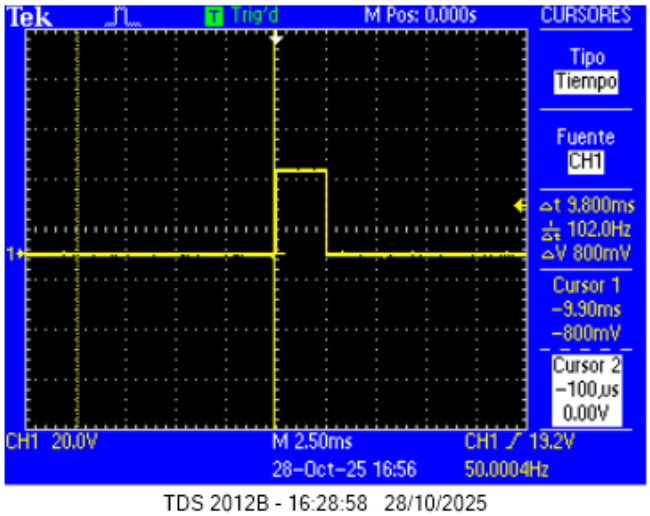


Ilustración 9 señal PWM generada Angulo 180.

La imagen muestra la señal PWM medida en el pin de control del servomotor, con un ancho de pulso de aproximadamente 2,5 milisegundos, generado por la función mover_ servo_instantaneo(angulo) del código del receptor. Este valor

corresponde al ángulo máximo de rotación del servo (180°), ya que el programa convierte el ángulo recibido en una duración de pulso entre 0,5 ms y 2,5 ms dentro de un período fijo de 20 ms (50 Hz). Por tanto, el pulso observado demuestra que el sistema traduce correctamente el valor del ángulo enviado desde el transmisor en una señal PWM estable, confirmando que el servomotor responde al control inalámbrico tal como fue programado.

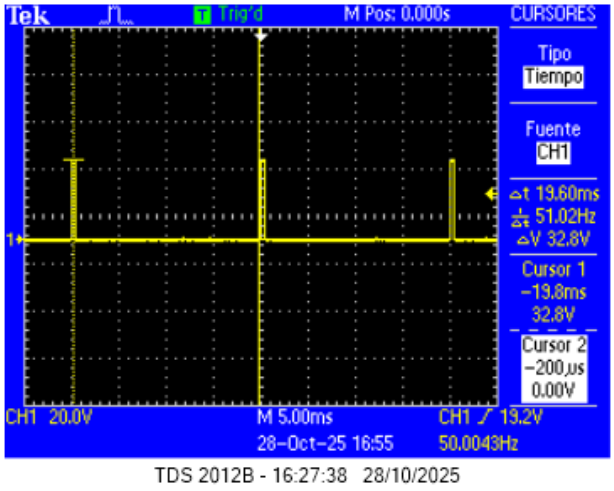


Ilustración 10 señal PWM generada Angulo 0

La imagen muestra la señal PWM medida en el pin del servomotor cuando se encuentra en su posición mínima (0°). El osciloscopio registra un ancho de pulso de aproximadamente 0,5 milisegundos dentro de un período de 20 milisegundos (frecuencia de 50 Hz), que coincide con la configuración del código en la instrucción servo.freq(50). Este pulso estrecho corresponde al valor más bajo calculado por la función mover_servo_instantaneo(angulo), donde para un ángulo de 0° se genera un pulso de 500 µs. Por lo tanto, esta medición confirma que el sistema está produciendo correctamente la señal PWM asociada al límite inferior del movimiento del servo, demostrando la correspondencia entre el ángulo enviado desde el transmisor y la señal que controla su posición física.

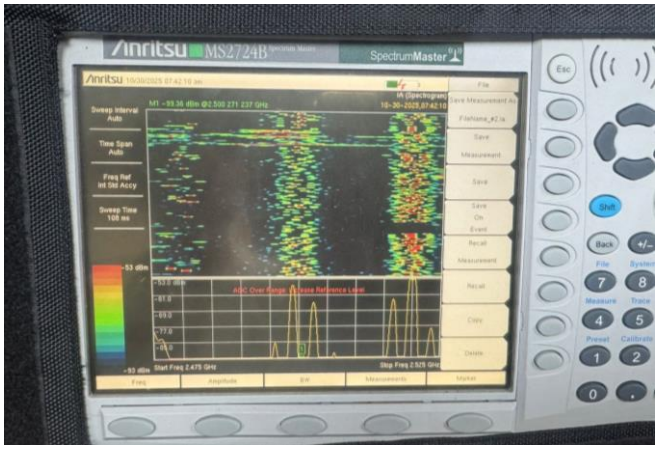


Ilustración 11 Análisis espectral de la transmisión inalámbrica

La imagen muestra la medición realizada con un analizador de espectros Anritsu MS2724B, utilizada para verificar la transmisión de radiofrecuencia generada por el módulo nRF24L01 empleado en el proyecto. En la pantalla se observa un barrido de frecuencia entre 2,475 GHz y 2,525 GHz, que incluye el canal 100 (2,5 GHz) configurado en el código mediante la instrucción nrf.reg_write(0x05, 100). La gráfica inferior muestra varios picos de potencia que corresponden a los paquetes de datos transmitidos a 2 Mbps, mientras que el espectrograma superior (en colores) representa la actividad temporal de la señal en esa banda. La potencia medida, indica un nivel bajo esperado para transmisión de corto alcance. Esta medición confirma que el módulo está emitiendo en la frecuencia configurada y dentro del rango ISM de 2,4 GHz, verificando así que la configuración del canal en el código se refleja correctamente en la salida de RF del nRF24L01. Además, la estabilidad del trazo evidencian una transmisión limpia y ajustada a los parámetros del sistema inalámbrico implementado.

CONCLUSIONES

Durante el desarrollo del sistema se comprobó que los parámetros de potencia y velocidad de transmisión del nRF24L01 pueden ajustarse según las necesidades del enlace. Al configurar mayor potencia y menor velocidad (250 kbps) se obtiene más alcance y estabilidad, ideal para distancias largas o entornos con interferencias. En cambio, al usar menor potencia y mayor velocidad (2 Mbps), se logra una respuesta más rápida y menor latencia, aunque con menor rango de cobertura. Esta flexibilidad permite adaptar el módulo tanto a aplicaciones de corto alcance con control inmediato, como a sistemas que requieran comunicación estable a distancias más amplias.

El desarrollo del laboratorio permitió comprobar el funcionamiento completo de un sistema de comunicación inalámbrica punto a punto entre dos Raspberry Pi Pico 2W utilizando módulos nRF24L01. A través del código implementado se logró transmitir de forma estable la posición de un joystick (TX) hacia un servomotor (RX), validando la eficiencia del enlace a 2.4 GHz configurado en el canal 100 y a una velocidad de 2 Mbps. Esta configuración permitió priorizar la baja latencia, lo cual es fundamental en sistemas de control en tiempo real.

El análisis de las señales PWM mediante el osciloscopio confirmó que los pulsos generados por la función mover_servo_instantaneo(angulo) se ajustaron correctamente a los valores esperados: 0.5 ms para 0° y 2.5 ms para 180°, con una frecuencia constante de 50 Hz. Esto demostró que el control del servo fue preciso y que el sistema convirtió de manera exacta la señal digital recibida en una posición física del eje.

Los resultados mostraron una comunicación confiable y continua, evidenciada tanto en las lecturas de consola como en la visualización en la pantalla OLED. El correcto

funcionamiento del acelerómetro MPU6050 integró al sistema una medición adicional de movimiento, complementando el control del servomotor con datos de aceleración en los tres ejes.

Adicionalmente, las mediciones con el analizador de espectros se verificaron que la transmisión se realizaba dentro del rango ISM de 2.4 GHz, sin interferencias ni emisiones fuera de banda. Los picos de potencia observados correspondieron al envío de paquetes a 2 Mbps, validando que la configuración establecida en el código se reflejaba correctamente en la salida de RF del módulo nRF24L01.

En conclusión, el sistema demostró un desempeño estable, confiable y coherente con los parámetros establecidos en el código. El montaje combinó comunicación SPI e I²C, control PWM y lectura de sensores, integrando hardware y software de manera efectiva. Este proyecto evidenció la importancia de la calibración, la sincronización y el manejo eficiente del tiempo de muestreo en aplicaciones de control inalámbrico de baja latencia.

Se crea un repositorio en GitHub donde se suben las demás evidencias del desarrollo del laboratorio al cual se podrá acceder en este enlace:

https://github.com/fernandoriano/Comunicaci-n-inal-mbrica-entre-dispositivos-nRF24L01.git

III. REFERENCIAS

[1] Universidad Militar Nueva Granada, Guía de laboratorio: Comunicación inalámbrica entre dispositivos nRF24L01, Programa de Ingeniería en Telecomunicaciones, Prof. José de Jesús Rugeles Uribe, 2025.

[2] Universidad Militar Nueva Granada, Guía de laboratorio: Codificación del canal, Programa de Ingeniería en Telecomunicaciones, Prof. José de Jesús Rugeles Uribe, 2025.

[3] MicroPython Development Team. (2024). MicroPython Libraries Overview. [En línea]. Disponible en: https://docs.micropython.org/en/v1.24.0/library/index.html

[4] Raspberry Pi Ltd. (2025). Hardware Design with RP2040 and Pico W. [En línea]. Disponible en: https://www.raspberrypi.com/documentation/microcontrollers/

[5] Nordic Semiconductor ASA. (2012). nRF24L01+ Product Specification v1.0. [En línea]. Disponible en: https://infocenter.nordicsemi.com/pdf/nRF24L01P_PS_v1.0.pdf