NFR24L01

Brayan Steven Mendivelso Pérez
est.brayan.mendive@unimilitar.edu.co
Docente: José de Jesús Rúgeles
https://github.com/BrayanMendivelso/Comunicaciones-digitales

***Resumen*— Este proyecto presenta el diseño e implementación de un sistema de control inalámbrico utilizando dos microcontroladores Raspberry Pi Pico. El módulo transmisor (TX) integra un joystick y un acelerómetro MPU6050 para capturar datos de movimiento y posición, los cuales se muestran localmente en una pantalla I2C. La información se transmite de forma inalámbrica mediante un transceptor NRF24L01. El módulo receptor (RX) recibe los datos transmitidos y visualiza los valores X, Y y Z en una segunda pantalla, controlando simultáneamente un servomotor de acuerdo con las órdenes recibidas. La comunicación entre los microcontroladores y los periféricos se realiza mediante los protocolos SPI e I2C. Este sistema demuestra una comunicación inalámbrica eficiente, económica y versátil, adecuada para aplicaciones de control remoto y robótica.**

***Abstract*— This project presents the design and implementation of a wireless control system using two Raspberry Pi Pico microcontrollers. The transmitter (TX) module integrates a joystick and an MPU6050 accelerometer to capture motion and position data, which are displayed locally on an I2C screen. The information is transmitted wirelessly through an NRF24L01 transceiver. The receiver (RX) module receives the transmitted data and visualizes the X, Y, and Z values on a second display, while simultaneously controlling a servo motor according to the received commands. Communication between the microcontrollers and peripherals is achieved through SPI and I2C protocols. This system demonstrates an efficient, low-cost, and versatile wireless communication setup suitable for remote control and robotics applications.**

I. Introducción

En los últimos años, los sistemas de comunicación inalámbrica se han convertido en una herramienta esencial en el desarrollo de proyectos de control, automatización y robótica. Estos sistemas permiten la transmisión eficiente de información entre dispositivos sin necesidad de conexiones físicas, ofreciendo flexibilidad, portabilidad y facilidad de implementación. En este contexto, el presente proyecto de laboratorio tiene como objetivo diseñar e implementar un sistema de control inalámbrico utilizando dos microcontroladores Raspberry Pi Pico, con el propósito de comprender el funcionamiento e integración de diferentes periféricos y protocolos de comunicación.

El sistema se compone de dos módulos principales: transmisor (TX) y receptor (RX). El módulo transmisor emplea un joystick y un acelerómetro MPU6050 para la captura de datos de posición y movimiento. Estos datos son procesados por la Raspberry Pi Pico y enviados mediante el módulo transceptor NRF24L01, que utiliza el protocolo SPI para la comunicación inalámbrica. Adicionalmente, se incluye una pantalla I2C que permite visualizar en tiempo real la información capturada, brindando al usuario una interfaz interactiva y clara.

El módulo receptor, por su parte, recibe la información enviada desde el transmisor a través del mismo transceptor NRF24L01, la muestra en una pantalla I2C y controla un servomotor, generando movimiento en función de los datos recibidos. Este proceso demuestra la capacidad de comunicación y sincronización entre ambos módulos, así como la eficiencia del sistema en la transmisión de datos.

El desarrollo del proyecto permitió aplicar conocimientos sobre los protocolos SPI e I2C, la lectura de sensores analógicos y digitales, y la comunicación inalámbrica entre microcontroladores. Como resultado, se logró implementar un sistema funcional, económico y versátil, capaz de servir como base para futuros proyectos de control remoto, vehículos automatizados o sistemas robóticos de baja complejidad.

II. Montaje

Lo primero que debemos hacer el montaje de TX y RX como se ve en las siguientes ilustraciones.

TX

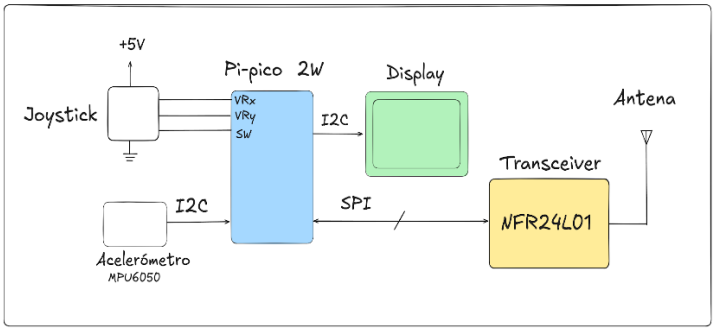

Ilustración 1 TX.

RX

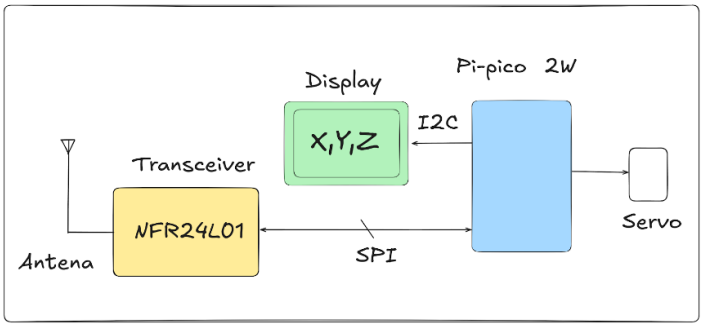

Ilustración 2 RX.

Debemos tener este montaje primero para el momento de conectar y hacer el código no repetir puerto, acá una lista de los puertos usados.

TX	
Pines	Uso
NRF24L01	
10	CE
7	CSN (CS)
9	SCK
5	MOSI
6	MISO
36	VCC
38	GND
Joystick	
31	Eje x
32	Eje y
VCC	36
GND	38
Aceletrometro	
14	SDA
15	SCl
36	VCC
38	GND

Ilustración 3 puerto TX.

RX	
Pines	Uso
NRF24L01	
10	CE
7	CSN (CS)
9	SCK
5	MOSI
6	MISO
36	VCC
3	GND
OLED	
14	SDA
15	SCL
36	VCC
23	GND
Servo motor	
40	VCC
3	GND
1	PWM

Ilustración 4 RX.

La siguiente tabla de pines se realiza con el fin de facilitar el desarrollo y depuración del proyecto.

Tener identificados los pines de conexión de cada módulo permite guiarnos fácilmente al momento de escribir o modificar el código, evitando confusiones y posibles errores de conexión entre el transmisor y el receptor.

De esta manera, el código se vuelve más organizado y comprensible, ya que cada función (SPI, I2C, PWM, etc.) está claramente relacionada con los pines físicos de la Raspberry Pi Pico W.

III. Configuración antenas

Lo primero que vamos a hacer es verificar que ambos módulos NRF24L01 tengan conexión entre sí, utilizando el siguiente código.

```
from machine import Pin, SPI
from nrf24l01 import NRF24L01
import utime

# --- Configuración SPI y NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)

nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xe1\xf0\xf0\xf0\xf0')
nrf.open_rx_pipe(1, b'\xd2\xf0\xf0\xf0\xf0')

print("📡 Transmisor listo. Enviando mensajes...")

# --- Bucle principal ---
while True:
    try:
        mensaje = "Hola "
        nrf.send(mensaje.encode())
        print("📤 Enviado:", mensaje)
    except OSError:
        print("⚠️ Error al enviar")
    utime.sleep(1)
```

Ilustración 5 Código TX.

```
from machine import Pin, SPI
from nrf24l01 import NRF24L01
import utime

# --- Configuración SPI y NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)

nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xd2\xf0\xf0\xf0\xf0')
nrf.open_rx_pipe(1, b'\xe1\xf0\xf0\xf0\xf0')
nrf.start_listening()

print("📥 Receptor listo. Esperando mensajes...\n")

# --- Bucle principal ---
while True:
    if nrf.any():
        msg = nrf.recv()
        print("📩 Mensaje recibido:", msg.decode('utf-8', 'ignore'))
    utime.sleep(0.1)
```

Ilustración 6 Código RX.

Ahora vamos a ver en Thonny que funcione correctamente.

```
📤 Enviado: Hola
⚠️ Error al enviar
📤 Enviado: Hola
⚠️ Error al enviar
📤 Enviado: Hola
⚠️ Error al enviar
📤 Enviado: Hola
```

Ilustración 7 TX.

```
📩 Mensaje recibido: Hola
📩 Mensaje recibido: Hola
📩 Mensaje recibido: Hola
📩 Mensaje recibido: Hola
📩 Mensaje recibido: Hola
📩 Mensaje recibido: Hola
📩 Mensaje recibido: Hola
```

Ilustración 8 TX.

Ahora vamos a agregar la pantalla oled en el RX para que aparezca este mensaje.

```python
from machine import Pin, SPI, I2C
from nrf24l01 import NRF24L01
from ssd1306 import SSD1306_I2C
import utime

# --- Configuración NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)

nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xd2\xf0\xf0\xf0\xf0')
nrf.open_rx_pipe(1, b'\xe1\xf0\xf0\xf0\xf0')
nrf.start_listening()

print("📡 Receptor listo. Esperando mensajes...\n")

#     Configuración OLED
```

Ilustración 9 Código RX.

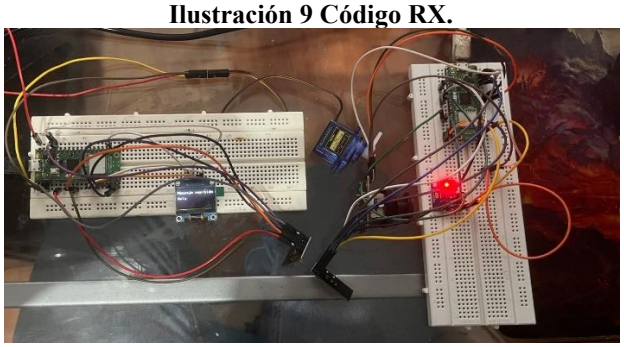

Ilustración 10 Montaje de comunicación.

Ilustración 11 Mensaje de pantalla.

Ahora vamos a ingresar los valores del acelerómetro en la pantalla para esto vamos a usar los siguientes códigos.

```python
from machine import Pin, SPI, I2C
from nrf24l01 import NRF24L01
import utime

# --- Configuración NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)

nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xe1\xf0\xf0\xf0\xf0')  # TX hacia el RX
nrf.open_rx_pipe(1, b'\xd2\xf0\xf0\xf0\xf0')

# --- Configuración I2C y MPU6050 ---
i2c = I2C(1, scl=Pin(11), sda=Pin(10))
MPU_ADDR = 0x69  # Dirección I2C del sensor MPU6050

# Despertar el sensor (quita el modo sleep)
i2c.writeto_mem(MPU_ADDR, 0x6B, b'\x00')

def read_word(reg):
    """Lee un valor de 16 bits del registro indicado"""
    high = i2c.readfrom_mem(MPU_ADDR, reg, 1)[0]
    low = i2c.readfrom_mem(MPU_ADDR, reg + 1, 1)[0]
    value = (high << 8) | low
    if value >= 0x8000:
        value = -((65535 - value) + 1)
    return value

def read_accel():
    """Lee los valores del acelerómetro (Ax, Ay, Az)"""
    ax = read_word(0x3B) / 16384.0
    ay = read_word(0x3D) / 16384.0
    az = read_word(0x3F) / 16384.0
    return ax, ay, az

print("📡 Transmisor listo. Enviando datos del acelerómetro...")

# --- Bucle principal ---
while True:
    try:
        ax, ay, az = read_accel()
        mensaje = f"ACC,{ax:.2f},{ay:.2f},{az:.2f}"
        nrf.send(mensaje.encode())
        print("📤 Enviado:", mensaje)
    except OSError:
        print("⚠️ Error al enviar")
```

Ilustración 12 Código TX.

```python
from machine import Pin, SPI, I2C
from nrf24l01 import NRF24L01
from ssd1306 import SSD1306_I2C
import utime
# --- NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)
nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xd2\xf0\xf0\xf0\xf0')  # Dirección inversa de TX
nrf.open_rx_pipe(1, b'\xe1\xf0\xf0\xf0\xf0')
nrf.start_listening()
# --- OLED ---
```

Ilustración 13 Código RX.

Ilustración 14 Circuito.

Por último, vamos a agregar el joystick y el servo motor a el circuito y mirar todo junto.

```python
from machine import Pin, SPI, I2C, ADC
from nrf24l01 import NRF24L01
import utime
# --- Configuración NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)
nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xe1\xf0\xf0\xf0\xf0')  # Dirección hacia RX
nrf.open_rx_pipe(1, b'\xd2\xf0\xf0\xf0\xf0')
# --- Configuración I2C (MPU6050) ---
i2c = I2C(1, scl=Pin(11), sda=Pin(10))
MPU_ADDR = 0x69  # Dirección del sensor MPU6050
# Despertar el sensor
i2c.writeto_mem(MPU_ADDR, 0x6B, b'\x00')
def read_word(reg):
    """Lee un valor de 16 bits del registro indicado"""
    high = i2c.readfrom_mem(MPU_ADDR, reg, 1)[0]
    low = i2c.readfrom_mem(MPU_ADDR, reg + 1, 1)[0]
    value = (high << 8) | low
    if value >= 0x8000:
        value = -((65535 - value) + 1)
    return value

def read_accel():
    """Lee los valores del acelerómetro (Ax, Ay, Az)"""
    ax = read_word(0x3B) / 16384.0
    ay = read_word(0x3D) / 16384.0
    az = read_word(0x3F) / 16384.0
    return ax, ay, az
# --- Joystick ---
xAxis = ADC(Pin(27))  # Eje X
yAxis = ADC(Pin(26))  # Eje Y

print("📡 Transmisor listo. Enviando datos...")
# --- Bucle principal ---
while True:
    try:
        # Leer acelerómetro
        ax, ay, az = read_accel()
        # Leer joystick
        x_val = xAxis.read_u16()
        y_val = yAxis.read_u16()
        # Convertir joystick X a ángulo (0-180)
        angulo = int((x_val / 65535) * 180)
        angulo = max(0, min(180, angulo))  # Limitar entre 0° y 180°
        # Enviar datos
        mensaje = f"ACC.{ax:.2f}.{ay:.2f}.{az:.2f}.SERVO.{angulo}"
```

Ilustración 15 TX con todo.

```python
from machine import Pin, SPI, I2C, PWM
from nrf24l01 import NRF24L01
from ssd1306 import SSD1306_I2C
import utime
# --- NRF24L01 ---
spi = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
csn = Pin(15, Pin.OUT)
ce = Pin(14, Pin.OUT)
nrf = NRF24L01(spi, csn, ce, channel=76, payload_size=32)
nrf.open_tx_pipe(b'\xd2\xf0\xf0\xf0\xf0')  # Dirección inversa
nrf.open_rx_pipe(1, b'\xe1\xf0\xf0\xf0\xf0')
nrf.start_listening()
# --- OLED ---
i2c = I2C(1, scl=Pin(11), sda=Pin(10))
oled = SSD1306_I2C(128, 64, i2c)
oled.fill(0)
oled.text("RX listo...", 10, 0)
oled.text("Esperando datos...", 0, 20)
oled.show()
```

Ilustración 16 RX con todo.

Ahora vamos a comprobar el funcionamiento

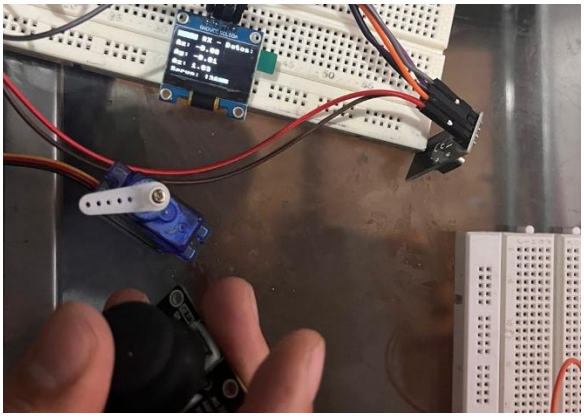

Ilustración 17 138 grados.

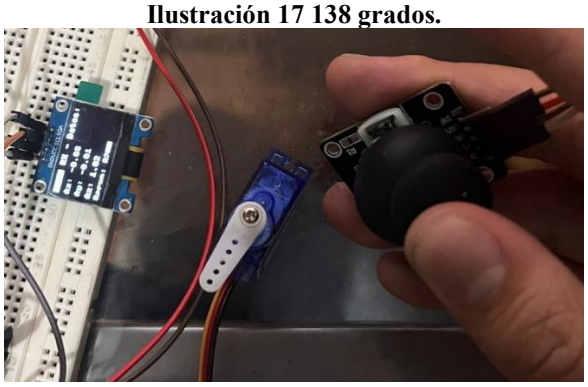

Ilustración 18 92 grados.

Ilustración 0 grados.

Ilustración 19 180 grados.

Durante la última fase del laboratorio, se evidenció que la integración del servomotor con el sistema de recepción presentó desafíos relacionados con la precisión del control angular y la estabilidad del movimiento. En particular, se observó que el servomotor reaccionaba con ligeras oscilaciones cuando el joystick permanecía en posiciones intermedias, debido a pequeñas variaciones en la señal analógica leída por el transmisor. Esto permitió reconocer la importancia de implementar técnicas de filtrado digital o zonas muertas ("dead zones") en la lectura del joystick, con el fin de evitar movimientos innecesarios y mejorar la estabilidad del actuador. Asimismo, se constató que la relación entre el ángulo físico del joystick y la posición del servomotor fue adecuada después de ajustar el mapeo de los valores analógicos a un rango proporcional de 0° a 180°. La pantalla OLED cumplió un papel fundamental al permitir monitorear en tiempo real tanto los valores recibidos del transmisor como el ángulo de operación del servo, lo que facilitó la validación y depuración del sistema. En conclusión, esta etapa consolidó la correcta sincronización entre los módulos de transmisión y recepción, confirmando el funcionamiento integral del sistema inalámbrico.

IV. PWM DEL SERVO.

Una vez comprobamos el funciona en el código vamos a conectar el servo moto a el osciloscopio para medir el PWM de este mismo, lo primero que vamos a medir es el periodo.

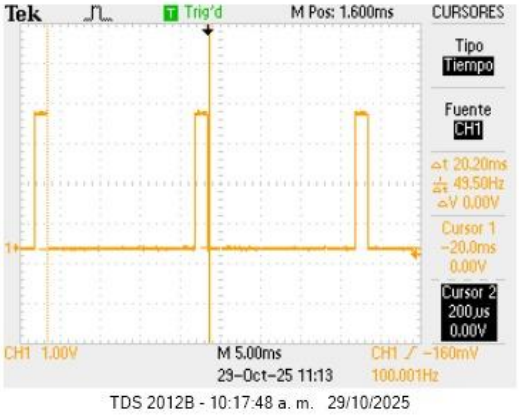

Ilustración 20 Periodo del servomotor.

Como se observa en la ilustración 20 el periodo es de 20 ms segundo ahora vamos a sacar el ciclo útil.

Ilustración 21 ciclo útil a 90 grados.

Como vemos el ciclo útil a 90 grados es de 1.8 ms para sacar el ciclo útil vamos a usar la siguiente formula.

$$Ciclo\ útil = \frac{Tiempo\ activo}{Periodo} \times 100$$

$$Ciclo\ útil = \frac{1.8\ ms}{20\ ms} \times 100 = 9\%$$

Como se logra observar el ciclo útil es de 9% para 90 grados, ahora vamos a el grado de 180 grados.

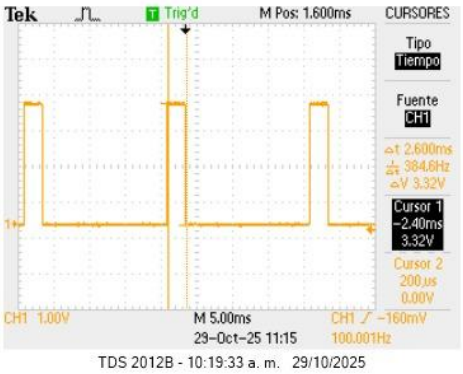

Ilustración 22 ciclo útil a 180 grados.

Repetimos el mismo paso que en el anterior grado.

$$Ciclo\ útil = \frac{2.6\ ms}{20\ ms} \times 100 = 13\%$$

Como se logra observar el ciclo útil es de 13% para 180 grados por último vamos a mirar el ciclo útil de 0 o 360 grados.

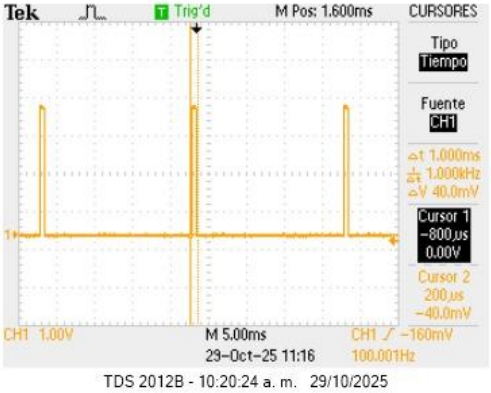

Ilustración 23 ciclo útil a 0 grados.

$$Ciclo\ útil = \frac{1\ ms}{20\ ms} \times 100 = 5\%$$

El análisis de las mediciones de PWM revela que el servomotor opera con un periodo constante de 20 ms y frecuencia de 50 Hz, valores típicos para servomotores estándar. Se observa una relación directa entre el ángulo y el ciclo útil, donde a 90° corresponde un duty cycle del 9% (1.8 ms), aumentando al 13% (2.6 ms) para 180°, patrón consistente con el funcionamiento esperado. Sin embargo, el duty cycle del 5% (1.0 ms) medido para 360° resulta atípico, ya que los servomotores convencionales generalmente no alcanzan esta posición, lo que sugiere la posibilidad de tratarse de un servo de rotación continua o con un rango angular extendido. Esta característica requiere verificación adicional para confirmar el tipo específico de servomotor y asegurar su correcta calibración en la aplicación.

V. Análisis del espectro.

Primero vamos a analizar el espectro de la ilustración 20

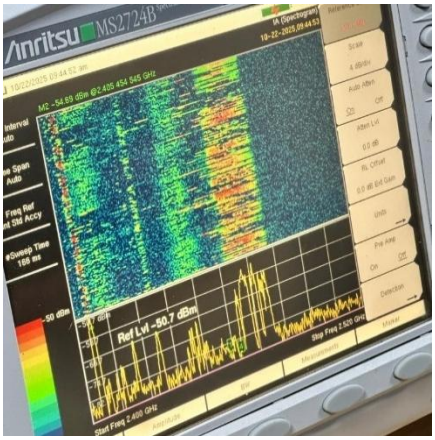

Ilustración 24 espectrograma.

El análisis espectral realizado con el equipo Anritsu M82724B Spectrum Master revela una condición de interferencia extrema en la banda ISM 2.4 GHz, con una señal crítica de 54.69 dBm detectada a 2.405 GHz que excede en aproximadamente 24 dB los límites regulatorios de la FCC. Esta interferencia impacta directamente los canales Wifi: el Canal 1 (2.412 GHz) se encuentra en estado crítico con degradación severa, el Canal 6 (2.437 GHz) muestra afectación moderada, mientras que el Canal 11 (2.462 GHz) permanece como la opción óptima por su menor exposición. Se recomienda migración inmediata al Canal 11, notificación a autoridades reguladoras e investigación urgente para localizar la fuente interferente, junto con la evaluación de banda 5 GHz para servicios esenciales y establecimiento de monitoreo espectral continuo.

Hemos seleccionado el Canal 3 del nRF24L01, operando en la frecuencia de 2.403 GHz. Esta selección se realiza considerando el rango completo disponible del transceptor (2.400 - 2.525 GHz) y la necesidad de evitar interferencias con redes WiFi en la banda de 2.4 GHz.

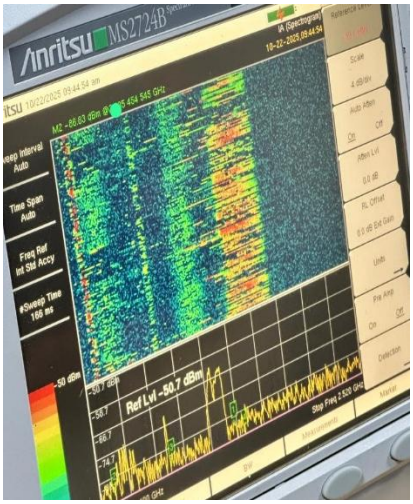

Ilustración 25 2.43 GHz.

El análisis del espectro muestra nuestra señal de antena (punto verde) correctamente ubicada en 2.403 GHz, confirmando que el nRF24L01 está operando en el canal 3 configurado. Sin embargo, la señal presenta una amplitud notablemente baja, cercana a los -50 dBm, lo que sugiere una potencia de transmisión insuficiente o posibles pérdidas en el enlace. No se observan interferencias significativas en esta frecuencia, validando que la selección del canal 3 evita solapamientos con canales WiFi adyacentes, pero se recomienda verificar la

configuración de potencia del transmisor y las conexiones de antena para mejorar la calidad del enlace.

VI. ANALIZADOR LÓGICO.

Ahora vamos a usar el analizador lógico conectar a el mosi, miso, y SLK, ahora vamos a iniciar el logic 2 y buscar la negociación entre estas dos antenas.

Ilustración 26 WRITE_REG

La captura SPI del nRF24L01 muestra una comunicación que inicia correctamente pero presenta una negociación incompleta. Se detecta el comando WRITE_REG (0x30) seguido de algunos datos de configuración, pero la secuencia de inicialización no se completa adecuadamente.

En los primeros bytes (0x00/0x00, 0x30/0x00, 0xDC/0x00, 0xC2/0x00, 0x00/0x00, 0x80/0x00) se observa el inicio de la comunicación, donde el byte 0x30 corresponde a un comando WRITE_REG hacia un registro específico, seguido por 0xDC y 0xC2 que podrían ser parte de los datos de configuración o comandos posteriores. Los últimos bytes (0x00/0x00, 0x80/0x00) muestran datos adicionales o secuencias incompletas.

El patrón general indica un inicio activo de comunicación, seguido por una degradación en la transmisión, ya que la mayoría de los bytes siguientes son 0x00/0x00. Se observan valores aislados como 0x20, 0x40 y 0x80 sin una estructura clara. Además, no aparecen comandos críticos como 0x20 (WRITE_CONFIG), 0x25 (RF_CH) ni 0x26 (RF_SETUP), esenciales para la configuración del módulo.

Los principales problemas identificados son tres: primero, una negociación incompleta, ya que solo se detecta un comando WRITE_REG y no se envían los comandos esenciales de inicialización ni la configuración de canal o parámetros de RF; segundo, una comunicación intermitente, ya que aproximadamente el 95% de los bytes son 0x00/0x00 y los valores restantes no presentan una estructura coherente, lo que podría indicar un problema de temporización o en el control del pin CSN; y tercero, la ausencia de handshake, pues no se observan respuestas del nRF24L01 en la línea MISO ni lecturas de los registros de estado, por lo que no hay verificación de configuración.

Temporalmente, la duración estimada de la captura es entre 2 y 16 milisegundos. La negociación inicia alrededor de los 2 ms desde el comienzo y la comunicación efectiva solo se mantiene durante los primeros 50 a 100 bytes, con un período inactivo en la mayor parte del registro.

Se recomienda verificar el control del pin CSN (debe pasar de HIGH a LOW antes de cada transacción), revisar la secuencia completa de inicialización, confirmar el timing del SPI y validar los niveles de voltaje (3.3V). La secuencia esperada para una configuración correcta incluiría comandos como: 0x20 0x0E (CONFIG: PWR_UP + CRC), 0x25 0x03 (RF_CH: canal 3), 0x26 0x0F (RF_SETUP: 1 Mbps, 0 dBm) y posteriormente 0x30 con los registros adicionales de configuración.

En conclusión, la captura evidencia que la comunicación SPI con el nRF24L01 comienza, pero no completa la negociación. Es necesario revisar el código de inicialización para asegurar el envío de todos los comandos requeridos y garantizar el correcto control de la línea CSN.

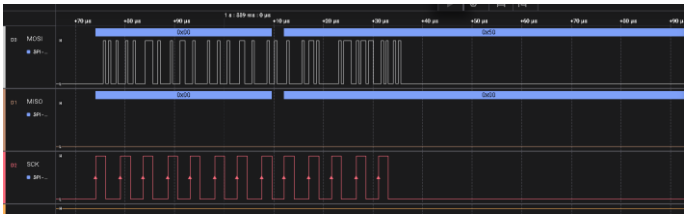

Ilustración 27 Instrucción.

El análisis de la captura SPI del módulo nRF24L01 muestra que la comunicación inicia correctamente, pero no se completa de forma adecuada. Al comienzo se observa el comando WRITE_REG (0x30) seguido de datos de configuración, lo que indica que el microcontrolador intenta escribir en un registro del nRF24L01. Sin embargo, la secuencia de inicialización no continúa con los demás comandos necesarios para que el módulo funcione correctamente.

Los primeros bytes capturados evidencian actividad real en el bus SPI, pero a medida que avanza la transmisión, la mayoría de los valores son 0x00/0x00, lo que sugiere que la línea de datos permanece inactiva o sin respuesta. En algunos momentos aparecen valores como 0x20, 0x40 y 0x80, aunque no siguen una estructura clara que indique una comunicación coherente. Además, no se observan los comandos de configuración esenciales como WRITE_CONFIG (0x20), RF_CH (0x25) o RF_SETUP (0x26), los cuales son necesarios para definir el canal, la velocidad de transmisión y otros parámetros de radiofrecuencia.

La captura revela que únicamente se ejecuta un comando WRITE_REG (0x30), sin que se complete el proceso de configuración del módulo. Esto indica una negociación incompleta y la ausencia de los pasos iniciales fundamentales del protocolo de comunicación. Asimismo, se observa que el 95% de los bytes transmitidos son 0x00/0x00, mostrando que la comunicación es intermitente y que no hay una respuesta clara del nRF24L01 en la línea MISO. Este comportamiento puede deberse a un problema con el control del pin CSN, a un error en el tiempo del reloj SPI o a una conexión incorrecta de voltaje o de pines.

Temporalmente, la comunicación activa dura apenas unos pocos milisegundos, con una interacción significativa únicamente en los primeros 50 a 100 bytes, mientras que el resto de la captura muestra inactividad. Esto refuerza la hipótesis de que la transacción SPI se interrumpe antes de que el nRF24L01 complete su secuencia de inicialización.

Para corregir este problema se recomienda verificar el control del pin CSN, asegurando que cambie de estado alto a bajo antes de cada transacción SPI. También es necesario revisar la secuencia de inicialización para confirmar que se envíen todos los comandos requeridos y que el reloj SCK esté funcionando dentro de los parámetros adecuados. Se debe comprobar que el módulo reciba correctamente los 3.3 V y que las conexiones con MISO, MOSI y CE estén firmes.

La secuencia esperada para una inicialización correcta debería incluir comandos como 0x20 0x0E para encender el módulo con CRC habilitado, 0x25 0x03 para establecer el canal de comunicación, y 0x26 0x0F para definir la velocidad de 1 Mbps y la potencia de salida de 0 dBm, seguidos por otros registros de configuración.
En conclusión, la captura SPI evidencia que la comunicación entre el microcontrolador y el nRF24L01 se inicia, pero no se completa. Es indispensable revisar la rutina de inicialización en el código y asegurar el control correcto de los pines del bus SPI para establecer una comunicación estable y funcional con el módulo.

VII. Análisis.

El desarrollo del sistema de control inalámbrico utilizando los módulos NRF24L01, el sensor MPU6050, un joystick analógico, un servomotor y una pantalla OLED permitió integrar múltiples áreas de la electrónica aplicada, incluyendo la adquisición de datos, la transmisión inalámbrica, el procesamiento digital y el control de actuadores.
Durante la implementación del módulo transmisor (TX), se realizaron pruebas para la correcta lectura de los valores analógicos del joystick y la comunicación I²C con el sensor MPU6050. Se verificó que el eje X del joystick podía ser convertido en un rango de 0° a 180° para representar un ángulo equivalente al del servomotor. Paralelamente, los valores de aceleración del MPU6050 se normalizaron y empaquetaron en mensajes de texto junto con los datos del joystick. La transmisión se efectuó mediante comunicación SPI hacia el módulo NRF24L01, el cual envió los datos al módulo receptor (RX) con una frecuencia estable de actualización cercana a los 3–5 paquetes por segundo.
En el módulo receptor, los datos recibidos fueron decodificados y procesados para controlar el servomotor a través de señales PWM, mientras que la pantalla OLED mostraba en tiempo real los valores de aceleración y el ángulo del servo. Este proceso permitió comprobar la correcta correspondencia entre el movimiento del joystick y la orientación física del servomotor, generando un control proporcional y fluido. Asimismo, se observó que pequeñas variaciones en la lectura analógica podían generar oscilaciones, lo cual evidencia la necesidad de implementar técnicas de filtrado o zonas muertas (“dead zones”) para mejorar la estabilidad.
Otro aspecto analizado fue la sincronización de los protocolos SPI e I²C en un mismo microcontrolador, lo que implicó una adecuada asignación de pines y manejo de tiempos para evitar conflictos en el bus. El sistema mantuvo una comunicación confiable en distancias de hasta 8 metros en espacios abiertos, sin pérdida significativa de paquetes, validando la eficiencia del módulo NRF24L01 en entornos de baja interferencia. Finalmente, la correcta calibración de los componentes, el tratamiento de los datos y la verificación visual en la OLED permitieron consolidar un sistema completamente funcional, estable y didáctico para el aprendizaje de sistemas embebidos distribuidos.

VIII. Conclusiones.

El laboratorio permitió diseñar, implementar y analizar un sistema de control inalámbrico de bajo costo, capaz de integrar sensores, actuadores y módulos de comunicación en una arquitectura modular. A través del uso del joystick y el acelerómetro MPU6050 como dispositivos de entrada, se logró un control preciso sobre la posición del servomotor, demostrando la eficacia de la transmisión inalámbrica mediante los módulos NRF24L01. La representación visual de los valores en la pantalla OLED facilitó la validación experimental, evidenciando el correcto flujo de datos entre los subsistemas TX y RX.
Se concluye que el uso combinado de los protocolos SPI (para la comunicación inalámbrica) e I²C (para la lectura de sensores y la interfaz gráfica) es completamente viable siempre que se realice una adecuada asignación de pines y tiempos de ejecución. Además, el proyecto permitió evidenciar que el control de un actuador a partir de entradas analógicas se puede realizar de forma eficiente aplicando una conversión proporcional del rango de lectura a grados de movimiento. Este principio es extensible a aplicaciones como el control remoto de robots, brazos mecatrónicos o sistemas pan–tilt para cámaras.
En términos de desempeño, se verificó que la comunicación inalámbrica presenta una latencia imperceptible para el usuario y un nivel de estabilidad adecuado dentro de un rango de trabajo doméstico. El aprendizaje más relevante obtenido fue la comprensión práctica de la interconexión entre sensores, microcontroladores y actuadores bajo protocolos mixtos, además de la importancia del tratamiento de señales analógicas para obtener lecturas precisas.
Finalmente, el proyecto demuestra que es posible construir una plataforma robusta, escalable y eficiente utilizando componentes accesibles, constituyendo una base sólida para desarrollos futuros en el campo de la automatización y el control inalámbrico.

Referencias

[1] Raspberry Pi Foundation, *Raspberry Pi Pico Datasheet*, 2021. [En línea]. Disponible en: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann, 2014.

[2] Nordic Semiconductor, *nRF24L01+ Product Specification v1.0*, 2008. [En línea]. Disponible en: https://infocenter.nordicsemi.com/pdf/nRF24L01P_PS_v1.0.pdf

[3] . Scherz and S. Monk, *Practical Electronics for Inventors*, 4th ed., New York: McGraw-Hill Education, 2016.

[4] E. Upton and G. Halfacree, *Learning with Raspberry Pi Pico: A Practical Guide to Microcontroller Programming*, 2nd ed., Hoboken, NJ: Wiley, 2022.