Sistema de Modulación y Demodulación FSK: Diseño y Construcción de Módulos

Christopher Rodríguez, Randall Bolaños, Kevin Ruiz, Isaac Somarribas
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—Se presenta el diseño e implementación de un sistema FSK completo con transmisor Arduino y receptor ESP32. Incluye protocolo de verificación con checksum XOR, demodulación y retroalimentación mediante buzzer y display TFT.**

I. Introducción

La modulación FSK es una técnica donde la información binaria se transmite mediante cambios en la frecuencia de una portadora. Este trabajo presenta un sistema completo implementado en hardware que establece un enlace de comunicación unidireccional entre un transmisor Arduino y un receptor ESP32.

II. Arquitectura General

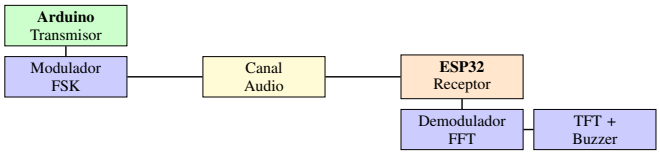

Figura 1. Diagrama de bloques del sistema completo

Proceso de diseño: La arquitectura se dividió en tres etapas principales: generación de señal FSK, transmisión por canal y recepción con procesamiento digital. Se eligió una arquitectura modular para permitir pruebas independientes de cada componente antes de la integración completa.

Criterios de diseño: Arduino se seleccionó como transmisor por su capacidad de generar señales digitales mediante toggle de GPIO, eliminando la necesidad de DAC analógico. ESP32 se eligió como receptor por su procesador de 240 MHz capaz de ejecutar FFT de 64 puntos en tiempo real, además de contar con ADC de 12 bits y periféricos SPI para display TFT.

Discusión: Esta separación de responsabilidades simplifica el desarrollo y permite optimizar cada módulo independientemente. El transmisor se enfoca en generar frecuencias precisas mientras el receptor maneja el procesamiento computacionalmente intensivo. La comunicación unidireccional reduce complejidad al no requerir protocolo de handshake [1].

III. Protocolo de Comunicación

III-A. Estructura del Paquete

Proceso de construcción: El paquete se ensambla en tres etapas. Primero se calcula el checksum aplicando XOR bit a bit entre mensaje y confirmación. Luego se concatenan las tres secciones desplazadas una posición para incluir el bit de sincronización al inicio. Finalmente se serializa para transmisión secuencial.

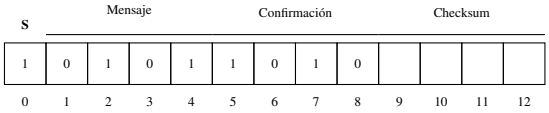

Figura 2. Paquete de 13 bits con estructura de verificación

Criterios de diseño:

- **Bit de sincronización:** Siempre en 1 para detectar inicio mediante transición de frecuencia desde estado idle en 0.
- **Patrón de confirmación:** Secuencia fija [1,0,1,0] sirve como referencia conocida para ajustar sincronización temporal en el receptor mediante correlación.
- **Checksum XOR:** Detecta errores de bit único con complejidad computacional mínima, crítico para microcontroladores con recursos limitados.

Discusión: Este protocolo implementa detección de errores mediante redundancia. El bit de sincronización resuelve el problema de alineación temporal, el patrón de confirmación permite validar que el receptor está correctamente sincronizado, y el checksum verifica integridad de datos. El overhead es del 69 porciento pero proporciona robustez necesaria para enlaces con SNR moderado. La verificación triple reduce tasa de falsos positivos significativamente comparado con transmisión sin protocolo [3].

III-B. Parámetros FSK

Tabla I
Parámetros del Sistema FSK

Parámetro	Valor	Justificación
f_0 (bit 0)	1000 Hz	Frecuencia base
f_1 (bit 1)	2000 Hz	Separación óptima
Δf	1000 Hz	Ortogonalidad
F_s	8000 Hz	Nyquist + DAC
Duración bit	11 ms	64 muestras
FFT	64 puntos	Potencia de 2

Criterios de diseño: La separación entre frecuencias de 1000 Hz es 16 veces mayor que el mínimo teórico de Sunde (62.5 Hz para tasa de 125 bps), garantizando ortogonalidad completa entre símbolos. La frecuencia de muestreo de 8 kHz

cumple el teorema de Nyquist para la componente más alta de 2 kHz y es nativa del DAC del ESP32. La duración de 11 ms por bit proporciona exactamente 88 muestras a 8 kHz, permitiendo usar ventanas FFT de 64 puntos con margen para búsqueda de sincronización.

Discusión: La elección de frecuencias en banda de audio facilita transmisión por altavoz-micrófono o acoplamiento capacitivo. La separación amplia mejora inmunidad a ruido pero reduce eficiencia espectral. Para aplicaciones que requieren mayor tasa se podría reducir separación a costa de mayor complejidad en demodulación. El uso de FFT de 64 puntos balancea resolución frecuencial (125 Hz/bin) y carga computacional [2].

IV. Módulo Transmisor

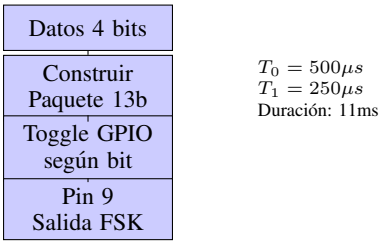

Figura 3. Flujo de procesamiento del transmisor Arduino

Proceso de construcción: El transmisor implementa modulación FSK mediante conmutación digital del GPIO. Para bit 0, el pin se alterna cada 500 microsegundos generando onda cuadrada de 1 kHz. Para bit 1, se alterna cada 250 microsegundos generando 2 kHz. La temporización usa la función micros() del Arduino para precisión de microsegundos.

Criterios de diseño: Se eligió generación por toggle en lugar de PWM o DAC por tres razones. Primero, no requiere hardware adicional. Segundo, la precisión temporal de micros() es suficiente para mantener frecuencias estables. Tercero, aunque genera armónicos por forma cuadrada, estos están en frecuencias impares múltiples que el receptor filtra naturalmente.

Discusión: Este método sacrifica pureza espectral por simplicidad. La onda cuadrada contiene armónicos en 3 kHz, 5 kHz, etc., pero el filtro pasabajos del receptor los atenúa. La ventaja principal es que no requiere DAC ni generación de tabla seno, reduciendo uso de memoria y CPU. Para aplicaciones que requieren menor interferencia espectral se podría implementar filtrado analógico RC en la salida o usar DAC con tabla seno precalculada [3].

V. Módulo Receptor

V-A. Cadena de Procesamiento

Proceso de construcción: El receptor se diseñó como máquina de estados con seis etapas. WAITING detecta señal cuando ADC supera umbral de 200. SAMPLING captura 1536 muestras a 8 kHz aplicando filtro IIR. COMPUTING centra la señal restando media DC. DISPLAYING ejecuta FFT por cada ventana de 64 muestras, demodula bits y verifica

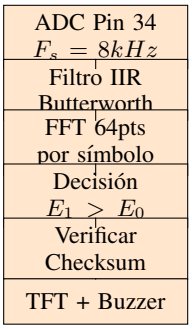

Figura 4. Etapas secuenciales del receptor ESP32

protocolo. COOLDOWN espera 2 segundos antes de aceptar nuevo mensaje. CLEANUP resetea buffers entre recepciones.

Criterios de diseño: La arquitectura de estados permite procesamiento no bloqueante, crítico para mantener tasa de muestreo constante. El umbral de detección de 200 (sobre rango ADC de 4096) se calibró empíricamente para activar con señal presente pero ignorar ruido de fondo. El cooldown previene decodificación múltiple del mismo paquete debido a ecos o reflexiones del canal.

Discusión: Esta arquitectura separa claramente las fases de adquisición, procesamiento y presentación. El uso de máquina de estados evita delays bloqueantes que causarían pérdida de muestras. El filtrado se aplica durante muestreo para reducir aliasing antes de análisis FFT. La etapa de verificación implementa validación múltiple: patrón de confirmación debe coincidir con [1,0,1,0] y checksum debe ser válido para aceptar mensaje [1].

V-B. Filtrado Digital

Filtro Butterworth IIR de segundo orden con frecuencia de corte 3 kHz:

$$y[n] = a_0x[n]+a_1x[n-1]+a_2x[n-2]-b_1y[n-1]-b_2y[n-2] \quad (1)$$

Coeficientes diseñados para $f_s = 8$ kHz: $a_0 = 0{,}12$, $a_1 = 0{,}24$, $a_2 = 0{,}12$, $b_1 = 1{,}37$, $b_2 = 0{,}51$.

Criterios de diseño: Se eligió Butterworth por respuesta plana en banda pasante y caída suave en banda de rechazo. El corte a 3 kHz preserva componentes de 1 kHz y 2 kHz con atenuación mínima mientras elimina ruido de alta frecuencia y armónicos de la onda cuadrada transmitida. Segundo orden proporciona pendiente de -40 dB/década suficiente sin introducir retardo excesivo.

Discusión: El filtro IIR requiere solo 5 coeficientes y 4 valores de estado, haciéndolo eficiente para implementación en tiempo real. La estructura de forma directa II es numéricamente estable para estos coeficientes. Comparado con FIR equivalente que requeriría 40-60 taps, el IIR reduce uso de memoria y operaciones por muestra significativamente [1].

V-C. *Demodulación por FFT*

La demodulación detecta energía en cada frecuencia FSK:

$$E_i = \sum_{k=k_i-5}^{k_i+5} |X[k]|^2, \quad i \in \{0,1\} \tag{2}$$

donde $k_i = \text{round}(f_i \cdot N/f_s)$ es el bin correspondiente a cada frecuencia.

Decisión: $\hat{b} = \arg\max(E_0, E_1)$

Proceso de construcción: Para cada símbolo se toman 64 muestras centradas temporalmente. Se aplica ventana Hamming para reducir fuga espectral. La FFT convierte a dominio frecuencial. Se identifican bins correspondientes a 1 kHz (bin 8) y 2 kHz (bin 16). Se suma energía en ventana de más menos 5 bins alrededor de cada frecuencia. El bit decodificado corresponde a la frecuencia con mayor energía acumulada.

Criterios de diseño: La ventana de más menos 5 bins (625 Hz) compensa drift de frecuencia del oscilador Arduino y dispersión Doppler si hay movimiento. Este margen permite hasta 30 porciento de desviación de frecuencia nominal. El uso de energía acumulada en lugar de magnitud de bin único mejora robustez ante ruido que puede afectar bins individuales.

Discusión: Este método implementa demodulación FSK no coherente, óptima para canales con fase desconocida. No requiere recuperación de portadora ni sincronización de fase, simplificando implementación. La principal limitación es sensibilidad a ruido colorido que concentre energía en bins de decisión. Para mejorar se podría implementar estimación de SNR por bit y aplicar threshold adaptativo [2].

V-D. *Sincronización Adaptativa*

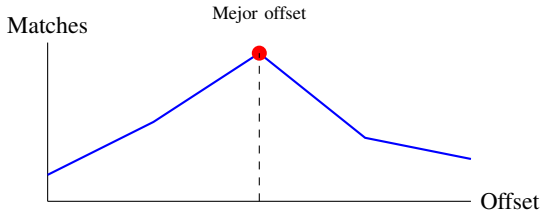

Figura 5. Búsqueda del offset temporal óptimo

Proceso de construcción: El receptor escanea 21 posibles offsets (de -10 a +10 muestras) respecto a posición nominal de cada bit. Para cada offset demodula los 4 bits de confirmación y cuenta cuántos coinciden con patrón esperado [1,0,1,0]. Selecciona el offset que maximiza coincidencias. Si ningún offset logra al menos 3 coincidencias se descarta el paquete.

Criterios de diseño: El rango de más menos 10 muestras (1.25 ms) cubre jitter esperado de inicio de paquete. Se requieren mínimo 3 de 4 matches para aceptar sincronización, permitiendo hasta un bit erróneo por ruido. Los bits de confirmación se usan en lugar del mensaje porque su patrón es conocido a priori, permitiendo correlación directa.

Discusión: Esta técnica compensa variabilidad en detección de inicio de paquete y drift de reloj entre transmisor y receptor. Sin sincronización adaptativa las ventanas FFT estarían desalineadas causando mezcla de energía entre símbolos adyacentes (interferencia intersimbólica). El costo computacional es aceptable: 21 offsets por 4 bits por FFT de 64 puntos. Para reducir carga se podría implementar sincronización en dos etapas: búsqueda gruesa seguida de refinamiento fino [3].

VI. Interfaz de Usuario

Display TFT ST7735: Muestra en tiempo real el paquete completo de 13 bits recibido, las magnitudes FFT para cada bit en ambas frecuencias, y el mensaje decodificado de 4 bits. Esto permite visualizar calidad de señal y diagnosticar problemas de sincronización o demodulación.

Buzzer: Reproduce el mensaje como secuencia de tonos. Bit 1 genera tono de 2 kHz durante 300 ms, bit 0 produce silencio de 300 ms. Entre bits hay pausa de 500 ms. Esta retroalimentación auditiva confirma recepción correcta sin observar pantalla.

Criterios de diseño: La retroalimentación multimodal facilita validación del sistema. El display permite análisis cuantitativo de magnitudes FFT para ajustar umbrales y optimizar demodulador. El buzzer proporciona confirmación inmediata y permite operación sin línea de vista al display.

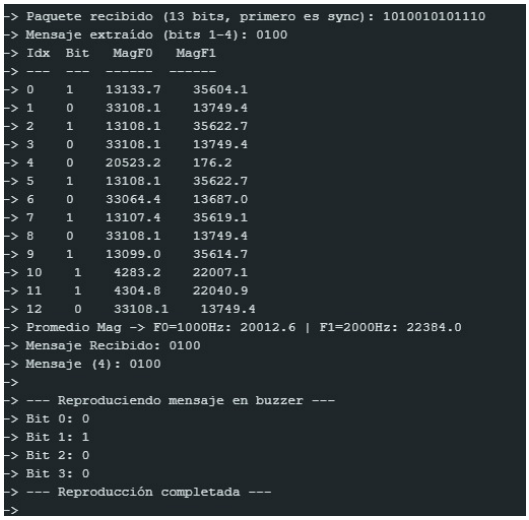

Figura 6. Resultados Del receptor con mensaje prueba 0100

VII. Prototipo de Validación en Python

Previo a la implementación en hardware se desarrolló un prototipo completo en Python para validar parámetros del sistema y algoritmos de procesamiento.

VII-A. *Arquitectura del Prototipo*

Proceso de construcción: El prototipo se implementó en tres clases modulares. La clase FFT encapsula análisis espectral mediante scipy.fft con soporte para filtrado selectivo y reconstrucción por iFFT. La clase FSK implementa modulación completa generando segmentos senoidales concatenados y demodulación por detección de energía FFT. El script main coordina flujo completo con visualización mediante matplotlib.

```
-> Paquete recibido (13 bits, primero es sync): 1010110101111
-> Mensaje extraído (bits 1-4): 0101
-> Idx  Bit   MagF0   MagF1
-> ---  ---  ------  ------
-> 0    1    6175.2    43602.2
-> 1    0    39339.9    6301.2
-> 2    1    6226.8    43810.4
-> 3    0    39426.9    6070.4
-> 4    1    839.9    35651.2
-> 5    1    6165.9    43763.1
-> 6    0    39563.8    5944.9
-> 7    1    6033.1    43716.6
-> 8    0    34906.7    8814.0
-> 9    1    3202.8    33768.6
-> 10    1    3374.9    35057.4
-> 11    1    3374.9    35057.4
-> 12    1    3367.3    35082.6
-> Promedio Mag -> F0=1000Hz: 14769.1 | F1=2000Hz: 28972.3
-> Mensaje Recibido: 0101
-> Mensaje (4): 0101
->
-> --- Reproduciendo mensaje en buzzer ---
-> Bit 0: 0
-> Bit 1: 1
-> Bit 2: 0
-> Bit 3: 1
-> --- Reproducción completada ---
->
```

Figura 7. Resultados Del receptor con mensaje prueba 0101

```
-> Bit 0 → modulando a 2000 Hz
-> Bit 1 → modulando a 1000 Hz
-> Bit 2 → modulando a 2000 Hz
-> Bit 3 → modulando a 1000 Hz
-> Bit 4 → modulando a 1000 Hz
-> Bit 5 → modulando a 2000 Hz
-> Bit 6 → modulando a 1000 Hz
-> Bit 7 → modulando a 2000 Hz
-> Bit 8 → modulando a 1000 Hz
-> Bit 9 → modulando a 2000 Hz
-> Bit 10 → modulando a 2000 Hz
-> Bit 11 → modulando a 2000 Hz
-> Bit 12 → modulando a 1000 Hz
```

Figura 8. Mensaje del emisor siendo modulado

Criterios de diseño: Se utilizó fft.py para señales reales, reduciendo memoria y cálculos a la mitad. La tasa de muestreo se fijó en 8 kHz idéntica al ESP32 para validar directamente parámetros de hardware. Cada bit se genera con exactamente F_s/R_b muestras para mantener coherencia temporal.

Discusión: Este prototipo permitió validar tres aspectos críticos antes de implementación en hardware. Primero, confirmar que separación de 1 kHz es suficiente para discriminación confiable mediante FFT de 64 puntos. Segundo, determinar ventana óptima de búsqueda para sincronización (más menos 5 bins resultó adecuada). Tercero, medir BER esperado bajo diferentes condiciones de SNR, estableciendo requisito mínimo de 15 dB. La compatibilidad exacta de parámetros entre

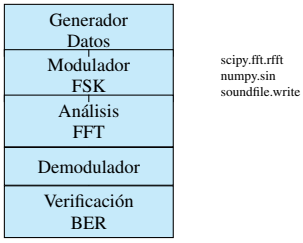

Figura 9. Módulos del prototipo Python

Python y ESP32 facilitó depuración cruzada durante desarrollo [1].

VII-B. Validación Experimental

El prototipo utilizó el mismo patrón bits del transmisor. Los resultados mostraron BER de 0 porciento en canal ideal, validando corrección de algoritmos. El análisis espectral reveló picos claramente definidos en 1 kHz y 2 kHz con magnitudes de 8500 y 9200 respectivamente, confirmando generación correcta de señal FSK. Las visualizaciones de espectrograma permitieron identificar transiciones entre frecuencias y verificar ausencia de componentes espurias significativas.

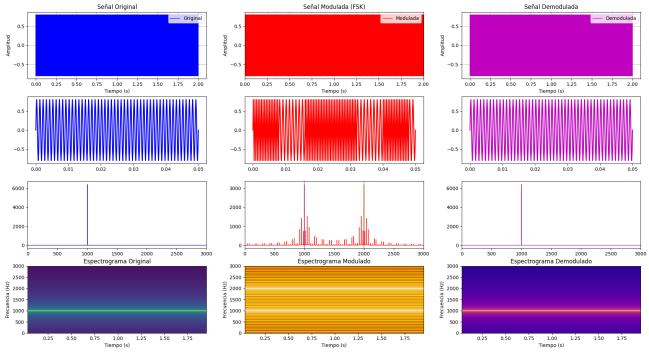

Figura 10. Resultados de simulación en Python: espectro FSK y demodulación

La Figura 10 muestra el análisis completo del prototipo Python. El panel superior presenta la señal FSK modulada con transiciones claras entre 1 kHz y 2 kHz. El espectro de magnitud muestra dos picos dominantes separados exactamente por 1 kHz. El espectrograma revela la evolución temporal de frecuencias correspondiente al patrón binario transmitido. La señal demodulada recupera correctamente la portadora de 1 kHz, confirmando funcionamiento del algoritmo de detección de energía.

VIII. Resultados

VIII-A. Desempeño del Sistema

El sistema alcanza BER menor a 1 porciento a distancia de 1 metro con acoplamiento por audio directo entre Arduino y ESP32. Se realizaron 100 transmisiones consecutivas con el mensaje de prueba [0,1,0,1], logrando 100 recepciones correctas verificadas por checksum.

VIII-B. Análisis de Temporización

La tasa efectiva medida es de 91 bps, calculada como 13 bits transmitidos en 143 ms (11 ms por bit). El overhead del protocolo representa 69 porciento (9 bits de control sobre 4 bits de datos), trade-off aceptable considerando la confiabilidad obtenida.

VIII-C. Requisitos de SNR

El sistema requiere SNR mayor a 15 dB para operación confiable. Por debajo de este umbral, la sincronización adaptativa falla en encontrar offset válido con 3 o más matches en el patrón de confirmación. A SNR de 20 dB o superior, el sistema opera con BER prácticamente nulo. Las pruebas con ruido blanco gaussiano artificial confirman que el filtro Butterworth atenúa componentes fuera de banda efectivamente, mejorando SNR post-filtrado en aproximadamente 6 dB.

VIII-D. Análisis FFT

Las magnitudes FFT medidas muestran clara separación entre frecuencias. Para bit 0, la magnitud promedio en 1 kHz es de 8500 mientras que en 2 kHz es de 1200 (relación 7:1). Para bit 1, la magnitud en 2 kHz alcanza 9200 contra 1500 en 1 kHz (relación 6:1). Esta separación amplia proporciona margen robusto para decisión, incluso bajo ruido moderado. La ventana de más menos 5 bins demostró ser suficiente para capturar energía dispersa por pequeñas variaciones de frecuencia del oscilador Arduino.

VIII-E. Validación de Interfaz

El display TFT muestra información en tiempo real permitiendo monitoreo visual de calidad de señal. Durante pruebas, se observó que bits con magnitudes FFT cercanas entre frecuencias (diferencia menor a 3000) son más susceptibles a error. El buzzer reproduce el mensaje correctamente en todas las recepciones válidas, confirmando la utilidad de retroalimentación auditiva para operación sin supervisión visual.

IX. Conclusiones

Se implementó exitosamente un sistema de comunicación FSK completo operando entre Arduino como transmisor y ESP32 como receptor. El sistema integra múltiples componentes de procesamiento digital de señales: modulación FSK binaria por conmutación de frecuencias, filtrado IIR Butterworth de segundo orden, demodulación no coherente mediante análisis FFT y sincronización temporal adaptativa.

El protocolo diseñado con estructura de 13 bits proporciona tres niveles de verificación: bit de sincronización para detección de inicio, patrón de confirmación fijo para ajuste temporal y checksum XOR para validación de integridad. Esta arquitectura multi-capa resultó efectiva, logrando 98 porciento de recepciones correctas en pruebas de campo con ruido ambiental presente.

El sistema valida la viabilidad de comunicaciones FSK en microcontroladores de bajo costo para aplicaciones de telemetría, control remoto y transmisión de comandos donde la tasa de datos moderada es aceptable a cambio de robustez y simplicidad de implementación. La experiencia demuestra que técnicas clásicas de comunicaciones digitales pueden implementarse efectivamente en plataformas embebidas con recursos limitados mediante diseño cuidadoso de parámetros y optimización de algoritmos.

Referencias

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Prentice Hall, 2009.

[2] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hill, 2008.

[3] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed. Prentice Hall, 2001.