# IMPLEMENTACIÓN DE PROTOCOLO DE COMUNICACIÓN PARA PRUEBAS FUNCIONALES ENTRE SISTEMA EMBEBIDO DE BAJO COSTO Y ELECTRODOMÉSTICOS

Ing. Hugo Enrique García Rico[1], Dr. Marco Antonio Garduño Ramón[2],

***Resumen*—En este artículo se expone las bases para la implementación de un protocolo de comunicación entre un sistema embebido y un electrodoméstico, esto para facilitar el monitoreo, la puesta en marcha de pruebas al sistema completo y el análisis de datos de pruebas funcionales por medio de un sistema embebido de bajo costo. Se presentan los beneficios de utilizar un microcontrolador de bajo costo y las funciones de software para evitar fallos en la comunicación. Se analiza el impacto que puede tener dentro de una planta de manufactura de electrodomésticos con las mejoras que podría presentar en la línea de pruebas funcionales.**

***Palabras clave*— protocolo de comunicación, sistema embebido, bajo costo, manufactura, pruebas funcionales.**

## Introducción

En la actualidad los equipos de prueba de tarjetas electrónicas para electrodomésticos son muy robustos y costosos, Algunas veces se necesita hacer pruebas rápidas de las principales funciones ya sea para análisis de fallas o para reparación de las mismas tarjetas electrónicas, las pruebas en la mayoría de los casos son manuales y se tiene que ingresar a modo test del aparato electrodoméstico de manera manual, de acuerdo con Neuhüttler J., Woyke I., Ganz W., Spath D (2019), el uso de datos de máquinas para mejorar el negocio de servicios ofrece nuevos potenciales de diferenciación a las empresas manufactureras.

Los equipos actuales de pruebas cuentan con una simulación de sensores, actuadores y demás hardware, para tratar de formar un sistema que funcione lo más cercano al producto final, pero no se puede asegurar la calidad del sistema completo es por eso que se requiere de pruebas funcionales antes de enviar el producto a cliente final.

En la industria de manufactura de aparatos electrodomésticos cada vez es más común encontrar fabricas inteligentes y maquinas tomando decisiones en base a resultado de pruebas, Es por esa razón que se vuelve parte importante del sistema la conectividad entre el sistema principal de los electrodomésticos y el sistema que analiza los resultados de las pruebas. Hameed B.(2011) comenta que en el panorama de la fabricación moderna, las empresas confían cada vez más en los sistemas de servicio de productos, es decir, la combinación de productos y servicios para obtener una ventaja competitiva. En un ambiente de manufactura en serie es muy importante el tiempo de producción, así que se hacen esfuerzos para hacer más eficientes y rápidas las pruebas funcionales antes de enviar el producto al cliente final. En este proyecto se expone los beneficios de utilizar un sistema embebido de bajo costo para hacer una herramienta de autodiagnóstico y monitoreo de aparatos electrodomésticos, creando un protocolo de comunicación entre el electrodoméstico y el sistema embebido.

Buscando resolver las dificultades de la aplicación en un ambiente industrial se toman los siguientes puntos como partida para analizar su funcionalidad y proponer soluciones:

- Compatibilidad de hardware: Las tarjetas electrónicas de los aparatos electrodomésticos normalmente están diseñadas para optimizar recursos por lo que debemos analizar el tipo de arquitectura que maneja.
- Sistema embebido: se necesita un sistema basado en la tarjeta electrónica, que cuente con puertos UART, que sea una tarjeta de desarrollo donde solo cargamos SW, facilitando diseño del dispositivo.
- Software: se necesita desarrollar permitiendo la comunicación y extracción de información de la memoria con la tarjeta electrónica de una secadora mediante protocolo de comunicación UART, sin afectar el performance y diseño principal del producto.
- Protocolo de Comunicación: deberá ser creado en base a comandos de transmisión y la tarjeta a probar debe identificar ese comando y responder con lo solicitado por el dispositivo de prueba. Una vez completada la comunicación el dispositivo debe analizar la respuesta de la tarjeta y los datos obtenidos.

## Descripción del Método

[1] Ing. Hugo Enrique García Rico es estudiante de maestría en Sistemas Inteligentes Multimedia en CIATEQ, Querétaro, Querétaro. hgr.garcia@gmail.com
[2] Dr. Marco Antonio Garduño Ramón es ingeniero de pruebas de robustez en Visteon de México, actualmente adscrito a la Universidad Autónoma de Querétaro como profesor investigador, Querétaro, Querétaro. mantoniogrx@gmail.com

*Marco Teórico*

Un protocolo de comunicación está formado por un conjunto de reglas y formatos de mensajes establecidas a priori para que la comunicación entre el emisor y un receptor sea posible. De acuerdo con Michael K. Powell., et al (2002) las reglas definen la forma en que deben de efectuarse las comunicaciones, incluyendo la temporización, la secuencia, la revisión y la corrección de errores. Hay 3 elementos clave que debemos considerar:

- Sintaxis (formato de los mensajes: datos + comandos).
- Semántica (significado de los comandos).
- Secuencia y temporalización (adecuado de las acciones que se toman respecto de los comandos).

La especificación de un protocolo consiste de varias partes

- Un formato preciso para los mensajes válidos
- Un conjunto de reglas de procedimiento para el intercambio de los datos
- Un conjunto de mensajes válidos que se pueden intercambiar, junto con su significado
- El servicio que provee el protocolo

*Uart: Trama de Datos*

UART es la comunicación serial asíncrona. Gadre D.V., Gupta S., et al (2018) afirma que en la comunicación serial asíncrona, el bit de inicio se envía antes que los bits de datos. Este bit indica el inicio del flujo de datos entre los dos extremos de la comunicación. Por lo tanto, indica que el extremo receptor debe prepararse para recibir los datos. Junto con los datos, se pueden transmitir bits de paridad que confirmarán que los datos se han transmitido correctamente a través del canal. Luego, después de transmitir datos y bits de paridad, se generará un bit de parada que indicará el final de la transmisión de datos. Cada byte de datos se ajustará de la manera descrita. Entonces, con cada byte de datos, también se transmiten algunos bits adicionales. Esta transmisión de bits por encima de la cabeza se rectifica cuando los datos se transmiten a través de una comunicación en serie sincrónica (ver figura.1).

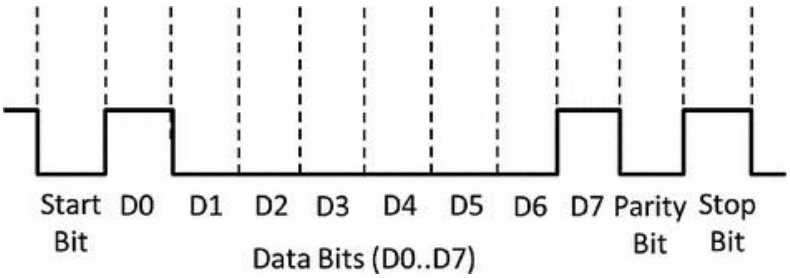


Figura 1. Muestra de trama de datos

*Uart: Habilitar Funciones*

De acuerdo con Jiménez M., Palomera R., Couvertier I. (2014), Dependiendo de la dirección de la comunicación y de si la operación se manejará mediante sondeo o mediante interrupciones, se requieren varias acciones de habilitación. Algunos UART requieren habilitar sus porciones de transmisor y / o receptor individualmente para que estén operativas, algunos habilitan todo el UART o USART. Si se desea un servicio de interrupción, se deben configurar los bits de habilitación de interrupción del transmisor y / o receptor. Para la comunicación simplex, solo se habilita el Tx o Rx correspondiente, mientras que para el funcionamiento dúplex ambos deben estar habilitados. Además, si las condiciones de error dispararan interrupciones en la operación UART, también se establecerán sus bits de habilitación.

*Uart: Operación*

Cuando se utiliza la operación de sondeo (aunque no es recomendable), el software del usuario siempre debe sondear la disponibilidad de los indicadores del transmisor y del receptor antes de cualquier operación en los búferes de datos UART. De acuerdo con Li C., Luo X., Wang H. (2015) los caracteres transmitidos por comunicación en serie se refieren generalmente a tramas de datos. Para obtener las tramas de datos correctas en el proceso de transmisión y recepción, se necesita una comunicación síncrona en la comunicación de datos. Antes de escribir un nuevo carácter para su transmisión en el búfer de salida de datos, se debe sondear el indicador TxReady para determinar la preparación. De lo contrario, podría producirse una pérdida de datos.

Al recibir mediante sondeo, la bandera RxReady debe ser consultada antes de recuperar los datos entrantes del búfer de entrada de datos para evitar la recuperación repetida de los mismos datos. Después de recibir cada carácter, se deben verificar los indicadores de error para determinar si se detectó un error en el carácter recuperado. El software del usuario decide la acción a realizar en caso de detectar un error. Cuando se habilita la operación basada en interrupciones, el software es aún más simple. La activación de la interrupción del receptor significa que se recibió un nuevo carácter y, por lo tanto, el ISR puede proceder directamente con su recuperación, sondeando los indicadores de error si no activan las interrupciones por sí mismos.

De manera análoga, un transmisor IRQ indica la disposición del transmisor para aceptar nuevos caracteres que se enviarán a través del canal y el ISR puede proceder directamente a escribir en el búfer de salida de datos. En la figura 2 muestra la topología de un canal serie full-duplex.

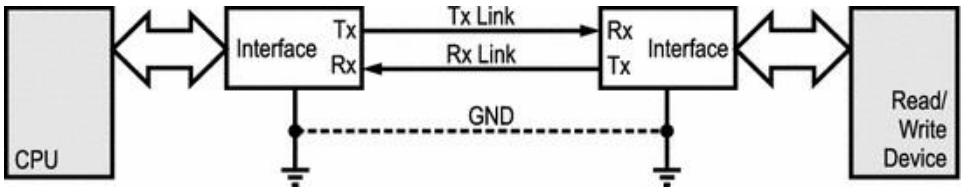


Figura 2. Muestra la topología de un canal serie full-duplex, por Jiménez M., Palomera R., Couvertier I (2014).

*Arquitectura del sistema*

En este proyecto se propuso desarrollar un sistema basado en la tarjeta RPI 3, que permita comunicarse y extraer información de la memoria de la tarjeta electrónica de una secadora mediante protocolo de comunicación UART, en la figura 3 se muestra la topología del sistema propuesto.

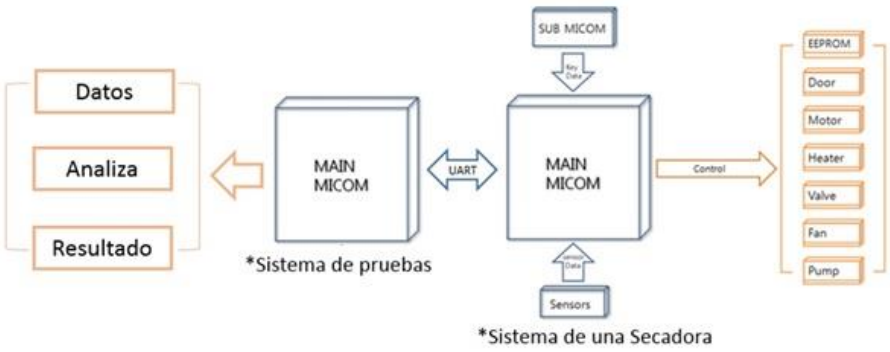


Figura 3. Topología del sistema.

*Funciones Principales*

A continuación se describen las principales funciones del protocolo de comunicación:

- Recibir datos: Si no hay error de recepción y hay espacio libre en el búfer receptor, se reciben los datos.
- Transmitir datos: Si existen datos en el búfer de envío, envíe los datos.
- Configuración UART: debe ser compatible con tasa de transferencia de 9600 bps / 8 bits DATOS / Paridad / 1 bit de parada
- Selección de Canal UART: Se debe seleccionar el canal A ya que los canales B, C, D son utilizados para el funcionamiento de la secadora.
- Estructura de paquetes: Paquetes deben cumplir con la siguiente estructura. Ver Tabla 1.

| | **Inicio** | **Tamaño datos** | **Comando** | **Datos** | **Checksum** |
|---|---|---|---|---|---|
| Hex | 0x5A | Num byte | CMD | d…d | CS |
| Byte | 1 | 1 | 1 | 0~n | 1 |

Tabla 1. Estructura de palabra de protocolo.

Inicio: Es el byte principal o identificador donde el sistema embebido y la tarjeta electrónica de la secadora podrán identificar para comenzar a enviar o recibir paquetes entre sí. Ver figura 4.

Tamaño de datos: número de bytes que serán enviados después de byte de inicio, Esto ayudara a eliminar errores o identificar perdida de bytes.

Comando: Identificador de las funciones que serán llamadas y así acceder a sensores, a datos de la memoria, iniciar pruebas o controlar algún periférico como motor, calentador, válvulas, etc.

Datos: Bytes que serán enviados como respuesta de un paquete enviado previamente y sirve para confirmar la recepción del paquete.

Checksum: es las suma XOR de todos los bytes del paquete enviado, con el propósito de identificar las pérdidas de datos entre cada interacción y si el valor de checksum no es el esperado se continua con el envio de datos hasta recibir el paquete completo.

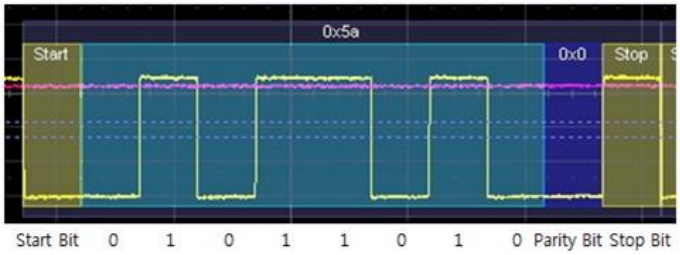


Figura 4. Ejemplo de byte en protocolo propuesto.

*Interaccion SW*

Siguiendo el diagrama de la Figura 5. Podemos observar la secuencia de interacción entre el sistema embebido y la tarjeta de control de la secadora.

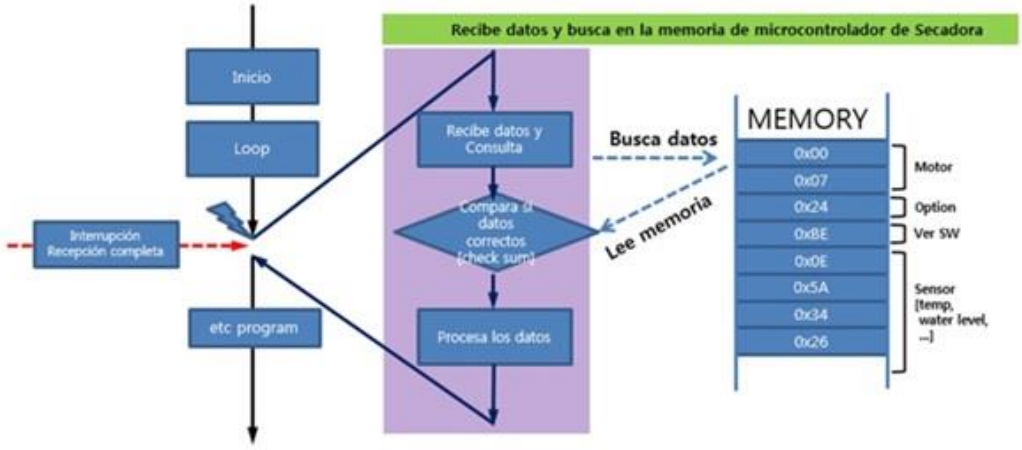


Figura 5. Diagrama de Software.

Se observa que el ciclo de sw de la tarjeta de control de secadora es interrumpido por el llamado del sistema embebido que crea una interrupción para accesar a la función deseada por el usuario, la interacción del protocolo se puede explicar en los siguientes pasos:

1.- Se identifica el primer byte que Inicio.

2.- Cuando se recibe el paquete de datos lo primero que hace el protocolo de comunicación creado y adaptado en la tarjeta de la secadora es identificar los paquetes en busca de errores, verifica el número de paquetes, si no hay errores en los paquetes continua.

3.- Identifica la función que está siendo llamada y hace la solicitud para obtener los datos o activar la función deseada.

4.- Responde con los paquetes de acuerdo al estándar que desarrollamos. Siempre hay una respuesta de la tarjeta de secadora a las peticiones que hace el sistema embebido, esto nos aseguramos que los comandos enviados por el sistema embebido fueron recibidos por la tarjeta de la secadora.

*Resultados y Discusión.*

A continuación se puede observar en la Figura 6, 7, 8 que corresponden a la configuración del puerto UART del sistema embebido, los paquetes de protocolo creado y el ejemplo de los casos para cada prueba.

```
scanner = serial.Serial(
    port = COMSCAN,
    baudrate = 9600,
    bytesize = serial.EIGHTBITS,
    parity   = serial.PARITY_NONE,
    stopbits = serial.STOPBITS_ONE)

#Abrir comuniacion con puertos seriales de escaner y set
if scanner.is_open != True: #Si la comunicacion con el puerto no esta abierta...
    i1 = True #Variable de apoyo para abrir el puerto serial
    while i1 == True: #Mientras su valor sea True
        try:
            scanner.open() #Se intenta abrir la copmunicacion del puerto
            if scanner.is_open:
                i1 = False #Y cuando se logra i1 regresa a False

        #Posibles excepciones (ERRORES) al intentar abrir el puerto
        except serial.SerialException:
            print('Port is not available')
            scanner.close()
        except serial.portNotOpenError:
            print('Attempting to use a port that is not open')
            print('End of script')
            scanner.close()
```

Figura 6. Configuración UART de sistema embebido (Rpi).

Después de configurar el puerto UART, el Rpi siempre está esperando el protocolo de inicio.

```
#Arreglos de bits para la secadora
byte1 = bytearray([0x5a, 0x03, 0x01, 0x00, 0x58]) #SMART TEST (Prender set)
byte2 = bytearray([0x5A, 0x03, 0x02, 0x00, 0x5B]) #02
byte3E = bytearray([0x5a, 0x03, 0x03, 0x00, 0x5A]) #Prueba spin para modelos electricos
byte3G = bytearray([0x5a, 0x03, 0x04, 0x00, 0x5D]) #Prueba spin para modelos de gas
byte4 = bytearray([0x5a, 0x03, 0x05, 0x00, 0x5C]) #Datos del set
byte5 = bytearray([0x5a, 0x03, 0x06, 0x00, 0x5F])
byte6 = bytearray([0x5a, 0xa3, 0x07, 0x00, 0x5E]) #Micom Writing Check
byte7 = bytearray([0x5a, 0xa3, 0x08, 0x00, 0x51]) #Comprobacion del cambio a SPN

```

Figura 7. Protocolo de comunicación.

Cada paquete de bytes está diseñado para una función específica y cada paquete tiene un checksum diferente ya que no hay dos protocolos iguales, Estos protocolos son diseñados y añadidos al software del sistema embebido y a la tarjeta electrónica de la secadora, de este modo ambas partes conocen y entienden cada protocolo.

```
    #Caso 1: Prueba 02
    if case == 1:
        print('caso 1')
        size1 = secadora.inWaiting()
        if size1 <= 0:
            secadora.write(byte2)
            Estatus.config(text='SENDING 02')
            time.sleep(0.15)
        if size1 > 0:
            data1 = secadora.read(size1)
            if str(data1.hex()) == '5a020058':
                Estatus.config(text='PERFORMANCE TEST')
                Image.config(text='Testing', bg='#FF8C00', fg='black')
                tic = time.perf_counter()
                case = 2
```

Figura 8. Caso de prueba

En la figura 9. Se puede observar el resultado de la ventana de prueba de la interface montado en la Rpi. Donde se exhiben número de serie, modelo que está siendo probado, estatus de la prueba, contador y el resultado de las pruebas de verificación de software.

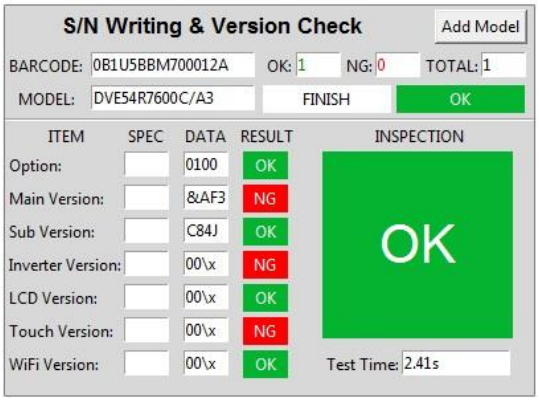


Figura 9. Interface de prueba.

*Resumen de resultados*

De acuerdo a las pruebas realizadas con 8 protocolos distintos se corrobora el funcionamiento de estas 8 pruebas, Ambas partes pudieron identificar cada protocolo y enviar la respuesta requerida en un tiempo relativamente corto de menos de ↓5s, se confirma el tiempo para un set de pruebas de 8 pruebas con un tiempo máximo de ejecución de 40s, obteniendo una diferencia de 440s con respecto a pruebas manuales accesando a modo de prueba de forma manual, este caso de estudio que corresponde a un secadora convencional.

*Conclusiones*

Los resultados demuestran la capacidad de la aplicación y el impacto que pudiera tener dentro de la industria de manufactura de aparatos electrodomésticos, optimizando pruebas funcionales con pequeñas modificaciones de código. Se concluye que el método para formar el protocolo de comunicación cumple con las expectativas esperadas en esta fase del proyecto y los resultados obtenidos son de mayor impacto que las modificaciones que se hicieron al sw principal de la tarjeta electrónica de secadora al agregar las nuevas funciones de protocolo de comunicación para esta aplicación.

*Recomendaciones*

Se exponen las siguientes recomendaciones que fueron determinadas en base a los resultados obtenidos:

1.- En las pruebas realizadas no se encontró problema de perdida de datos, pero algunas veces se presentó exceso de tiempo de recibo de datos esto debido a que algunas veces el tamaño del paquete era correcto pero el checksum no era el esperado y se mandaba la petición una vez más hasta que el checksum es el esperado, se concluye que se puede mejorar o hacer más robusto el análisis de los paquetes agregando la técnica de handshake.

2.- En una segunda fase de proyecto se recomienda agregar un byte más al frame de protocolo para agregar 8 bits que pudieran controlar los accesos a la memoria.

3.- Se recomienda fortalecer la estandarización de los protocolos desde un inicio ya que conforme se agreguen mas pruebas los protocolos crecerán y podría llegar a ser difícil identificar cada comando

## Referencias

Neuhüttler J., Woyke I., Ganz W., Spath D. “An Approach for a Quality-Based Test of Industrial Smart Service Concepts.”, In: Ahram T. (eds), vol 787. Springer, Cham, 2019.

Hameed B. “The Smart Real-Time Factory as a Product Service System.” In: Hesselbach J., Herrmann C. (eds), Springer, Berlin, Heidelberg, 2011.

Michael K. Powell. “Protocolos y modelo OSI.”, Adventure Works Annik Stahl July 2002.

Gadre D.V., Gupta S. “Universal Asynchronous Receiver and Transmitter (UART).” In: Getting Started with Tiva ARM Cortex M4 Microcontrollers. Springer, New Delhi, 2018.

Jiménez M., Palomera R., Couvertier I. “Principles of Serial Communication. In: Introduction to Embedded Systems.” Springer, New York, NY. 2014.

Li C., Luo X., Wang H. “Design of Serial Communication Module Based on Solar-Blind UV Communication System.”, In: Huang DS., Bevilacqua V., Premaratne P. (eds), vol 9225. Springer, Cham, 2015.