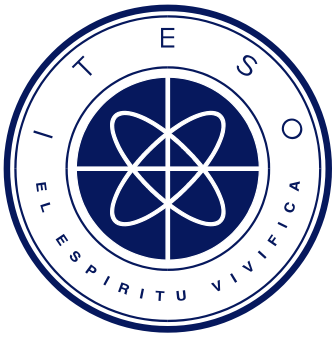
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

ESPECIALIDAD EN SISTEMAS EMBEBIDOS



REEMPLAZO DEL PROTOCOLO SENT POR LIN

Tesina para obtener el grado de:

ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presentan: José Manuel González García, José de Jesús Sepulveda Cisneros, Javier Villa Chávez, Roberto Paz Vázquez

Director: Dr. Luis Rizo Domínguez

San Pedro Tlaquepaque, Jalisco. Julio del 2018.

Abstract

Over the years, various communication protocols have been used to interconnect electronic control devices in a car; most of the time more than one communication protocol coexists within it, each one very different from the other in terms of costs, speed, hardware dependencies, etc. Our purpose is to compare 2 low cost communication protocols currently used in the automotive industry, SENT and LIN, and provide information that will help replace or choose the protocol that best suits its needs, describing the way they operate, the frame formats, and network topologies. The core part of this work is the theoretical and practical comparison of the protocols, by relying on various public domain sources, as well as the respective standards of each protocol (J2716_201001 for SENT and specification version LIN 1.3) to obtain valuable information, such as the advantages and disadvantages of each protocol. Subsequently, various metrics derived from the implementation of both protocols were obtained in the DEMO9S12XEP100 development board focusing on time of transmission, reception, and CPU usage and then, depending on the specific needs of the user, where the decision of which protocol is best will be ultimately taken by the user or developer.

Resumen

Con el paso de los años, se han utilizado diversos protocolos de comunicación para interconectar dispositivos de control electrónico en un automóvil; la mayoría de las veces más de un protocolo de comunicación convive dentro del mismo, cada uno muy diferente en términos de costos, velocidad, dependencias de hardware, etc. Nuestro propósito es comparar 2 protocolos de comunicación de bajo costo actualmente utilizados en la industria automotriz, SENT y LIN, y proporcionar información que ayudará a reemplazar o elegir el protocolo que mejor se adapte a sus necesidades, describiendo la forma en que operan, los formatos de trama y las topologías de red. La parte central de este trabajo es la comparación teórica y práctica de los protocolos, para lo cual nos basamos en varias fuentes de dominio público, así como los estándares respectivos de cada protocolo (J2716_201001 para SENT y la especificación LIN 1.3) para obtener información valiosa, como las ventajas y desventajas de cada protocolo. Posteriormente, se obtuvieron varias métricas derivadas de la implementación de ambos protocolos en tarjetas de desarrollo DEMO9S12XEP100 enfocándonos en el tiempo de transmisión, recepción y uso de la CPU y luego, dependiendo de las necesidades específicas del usuario, tomar la decisión de que protocolo es óptimo para la aplicación del usuario.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Abreviaturas/Siglas

Abreviatura /Siglas	Significado	Traducción
ADC	Analog Digital Converter	Convertidor Análogo-Digital
ASIC	Application Specific Integrated Circuit	Circuito integrado de Aplicación específica
CAN	Controller Area Network	Red Controladora de Área
CPU	Central Processing Unit	Unidad Central de Procesamiento
CRC	Cyclic Redundancy Check	Verificación de redundancia cíclica
ECU	Electronic Control Unit	Unidad de Control Electrónica
ISO	International Organization for Standarization	Organización Internacional para la Estandarización
kbps	Kilobits per second	Kilobits por segundo
LIN	Local Interconect Network	Red Local de Interconexión
PWM	Pulse Width Modulation	Modulación por ancho de pulso
RAM	Random Access Memory	Memoria de acceso aleatorio
ROM	Read-only memory	Memoria de solo lectura
SAE	Society of Automotive Engineers	Sociedad de Ingenieros de Automoción
SENT	Single Edge Nibble Trasmission	Transmisión de nibble de un solo borde
TPP	Two-way Password Protocol	Protocolo de contraseñas de 2 caminos
TTCAN	Time Triggered Controller Area Network	Red Controladora de Área basada en Tiempo
UART	Universal Asynchronous Receiver Transmitter	Transmisor-Receptor Asíncrono Universal

1. Introducción

La propuesta de este trabajo consiste en un caso de estudio que comprende el análisis comparativo de los protocolos SENT (Single Edge Nibble Trasmision, por sus siglas en inglés) y LIN (Local interconect network, por sus siglas en inglés) con el fin de establecer parámetros para la sustitución de SENT por LIN en el ámbito automotriz.

1.1. Descripción del tema

Primeramente, se analizaron los protocolos de comunicación alámbrica SENT y LIN, con el fin de realizar dos tipos de análisis, teórico y práctico.

En el análisis teórico se compararán ambos protocolos, fundamentándonos en sus respectivas especificaciones técnicas con el objetivo de obtener características comunes, así como ventajas de uno frente al otro. Algunas de los rubros a evaluar son: velocidad de transmisión, costos de materiales y desarrollo, consumo de potencia, susceptibilidad al ruido electromagnético, robustez ante fallos.

En el análisis práctico se implementarán los protocolos usando el mismo hardware, esto igualará las condiciones en ambas implementaciones y favorecerá una comparativa equitativa dado que la capacidad de procesamiento para ambos será el mismo.

1.2. Descripción del problema/necesidad detectada (problematización).

En el sector automotriz, así como en muchos otros sectores productivos, la adopción de tecnologías eficientes y económicas es de gran importancia. Esto se debe a la gran cantidad de ingeniería necesaria para diseñar y producir un solo automóvil el cual representa un costo alto Además, la adopción de tecnologías alternativas propicia tener una ventaja competitiva, con respecto a los productos que se encuentran en el mercado.

Actualmente un auto cuenta con decenas de dispositivos electrónicos de control y la tendencia sugiere que este número crecerá. Una de las áreas de oportunidad en esta industria es la optimización de la comunicación entre dichos dispositivos. El tener una comparativa entre el protocolo LIN y SENT puede brindar a los ingenieros las pautas para una elección óptima de acuerdo con las necesidades específicas de cada aplicación. Lo anterior puede significar el ahorro de millones de dólares en producción e ingeniería.

1.3. Justificación

LIN es uno de los protocolos de comunicación más usados para la interconexión entre módulos automotrices; esto se debe, entre otras cosas, a su menor costo comparado con CAN (Controller Area Network, por sus siglas en inglés) y por su relativa robustez. Otro protocolo usado, pero no tan difundido, es SENT, el cual se utiliza en la comunicación con sensores. Algunas ventajas de SENT es su resistencia a fallos, inmunidad a interferencia eléctrica y un costo más bajo en términos de desarrollo de software y hardware, ya que no requiere un transceptor para la intercomunicación; a todo esto, se le agrega que posee una velocidad de transferencia suficiente para satisfacer una gran cantidad de implementaciones que no requieren un gran caudal de datos.

Aunque LIN es muy usado en el mundo automotriz, SENT, cuya implementación puede ser mucho más económica, resulta ser mucho más atractivo cuando la comunicación bilateral no es factor importante y la velocidad de transferencia de información juega un rol importante.

1.4. Objetivos

En la investigación nos propusimos:

- Proporcionar información comparativa de características de cada protocolo con base teórica.
- Proporcionar información comparativa de características de cada protocolo con base práctica.
- Proporcionar el protocolo óptimo para cada característica analizada en los puntos anteriores.

- Con base a lo anterior, proporcionar aplicaciones recomendadas para el uso de cada protocolo

2. Antecedentes

En el presente, los automóviles modernos están compuestos de una inmensa cantidad de sistemas electrónicos; gran parte de estos requieren interconectarse entre sí para trabajar en conjunto mediante el intercambio de información. Debido a la creciente demanda de interconexión se hace cada vez más evidente la necesidad de seleccionar la tecnología apropiada para cada aplicación específica. Dicha selección viene usualmente acompañada de la comparación entre las distintas opciones existentes.

El protocolo CAN fue desarrollado en la década de los 80's para superar algunas de las deficiencias de los protocolos más usados en ese momento. Este desarrollo resultó en la producción de un grupo de estándares describiendo CAN a través de los ISO 11898-1 a ISO 11898-5 [1]. Tiempo después, la compañía Freescale (Motorola) desarrolló el protocolo LIN, que es menos costoso que CAN, así como menos complejo, por lo que es enfocado a componentes "ligeros" cuyas necesidades de comunicación son menos exigentes y de menor velocidad. A su vez SENT es destinado mayormente a interfases con sensores [2], lo cual se debe principalmente a su mayor velocidad de transferencia (mayor a LIN), una gran inmunidad a ruido eléctrico, pero conservando ciertas limitaciones que restringen su implementación a solo aplicaciones con comunicación unidireccional.

Debido a la gran cantidad de protocolos existentes, es necesario hacer una elección que se ajuste a las necesidades específicas de la aplicación a desarrollar. Uno de los aspectos a considerar para la correcta elección de un protocolo es el número de elementos a ser conectados en la misma red de comunicación. Para las redes que necesitan conectar más de 2 dispositivos; una conexión de estrella o tipo "bus", como lo es LIN es la más adecuada. La conexión punto a punto es la indicada cuando solo es necesario conectar 2 dispositivos. De acuerdo con Ullah, la capacidad de interconectar varios dispositivos en la misma red puede reducir significativamente el costo del sistema ya que menos cable será utilizado. En la industria automotriz, un número de sistemas sólo necesitan comunicación punto a punto y el utilizar LIN para este propósito es una solución más cara debido a la complejidad en su desarrollo [3]. En esta situación, los protocolos que están

orientados a conexiones punto a punto son mejores candidatos para remplazar LIN, un ejemplo de estos protocolos es SENT. Un análisis comparativo de estos protocolos puede proporcionar fundamentos para el remplazo de SENT sobre LIN.

Otras alternativas a ser utilizadas para la comunicación punto a punto son señales análogas haciendo la conversión utilizando un ADC (Analog Digital Converter) o utilizar PWM (Pulse Width Modulation). El problema es que las resoluciones que pueden ser manejadas por estos métodos son limitadas. Por lo tanto, el protocolo SENT está pensado para ser usado en aplicaciones donde una alta resolución de los datos a ser transmitidos es requerida. Como diría Srinivas, SENT tiene la intención de ser un remplazo para métodos de baja resolución como lo son usar ADC o PWM [4].

La elección informada entre protocolos requiere forzosamente un análisis comparativo que proporcione la suficiente información teórica y práctica. Existen trabajos previos que ofrecen un análisis comparativo entre distintos protocolos. Un ejemplo es el que se trata en el trabajo [1] en donde se presenta un análisis comparativo entre CAN y otros tres protocolos TTCAN, FlexRay y LIN. En este trabajo se proporcionan tablas comparativas basadas en parámetros específicos como son sincronización, manejo de errores, tipos de mensajes, costo, etc. En [2] se brinda un estudio comparativo entre CAN, TPP y Flexray, con un análisis de los protocolos más detallado con respecto al estudio [1], además de proporcionar tablas comparativas en términos de complejidad, flexibilidad y demandas de tiempo real. Determinar cuál protocolo debe ser usado para cada sistema en la industria automotriz resulta más sencillo cuando se tiene un análisis como los descritos anteriormente. En [5] se presentan algunas aplicaciones y los principales estándares de protocolos de comunicación usados en la industria automotriz. Con esta información se genera un análisis sobre cómo los protocolos presentados satisfacen de manera óptima las características de dichas aplicaciones.

Entre los protocolos antes mencionados la comparación más justa seria entre SENT y LIN, ya que por sus características son mayormente usados para aplicaciones de una menor complejidad y costo.

3. Marco Teórico

3.1. Red Local de Interconexión (LIN)

El bus de Red Local de Interconexión (LIN) fue desarrollado para crear un estándar para comunicación multiplexada de bajo costo en redes automotrices. Aunque existen otros protocolos más robustos que LIN, CAN por ejemplo, la implementación de estos otros protocolos puede llegar a ser prohibitiva en aplicaciones de categoría media o baja cuya complejidad no requiere el uso de protocolos complejos. Las redes automotrices modernas usan una combinación de LIN para aplicaciones de bajo costo principalmente en electrónicos, CAN para comunicación de tren de potencia y carrocería y el bus FlexRay para comunicaciones de datos sincronizados de alta velocidad en sistemas avanzados como suspensión activa. [6]

LIN es rentable para aplicaciones que requieren baja transferencia de datos y donde el manejo de errores no son prioridad. El protocolo LIN se basa en transferencia serial basada en UART, que son las siglas de Universal Asynchronous Receiver-Transmitter, en español: Transmisor-Receptor Asíncrono Universal, el cual se encuentra embebido en la mayoría de los microcontroladores modernos. El estándar que se basa en una comunicación serial que permite usar un transmisor/ receptor estándar serial universal asíncrono (UART), que esta embebido en la mayoría de los microcontroladores modernos. [1]

El nodo maestro es encargado de controlar la secuencia y temporalización de las tramas, lo cual se fija en una tabla específica llamada tabla de agendamiento. El maestro puede cambiar el programa según las necesidades. El dispositivo maestro contiene una tarea de maestro y además también puede contener la tarea de esclavo, a su vez el esclavo solo puede contener una tarea de esclavo. [7]

Existen varias versiones de LIN estándar. La versión 1.3 finalizó la comunicación byte-layer. Las versiones 2.0 y 2.1 añaden más especificaciones de mensajes y servicios, pero son compatibles al nivel de byte con LIN 1.3. [8]

3.1.1 Transmisión de tramas en LIN

La unidad básica de transferencia en el bus LIN es el frame que, como se mencionó anteriormente, se divide en un encabezado y una respuesta. En la figura 1 se muestra cómo está compuesta una trama de LIN. El encabezado siempre se transmite por el nodo maestro, y la respuesta que es transmitida por una tarea de esclavo y puede residir ya sea en el nodo maestro o un nodo esclavo. [9]

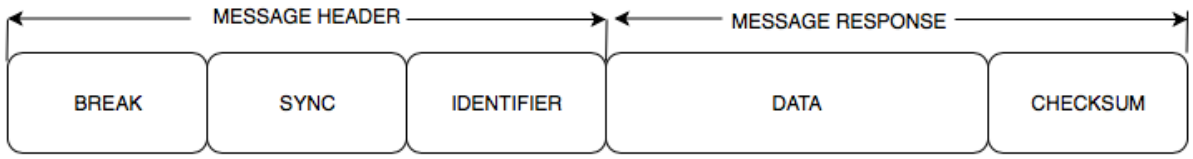


Figura 1: Trama mensaje LIN

Normalmente, la tarea de maestro consulta cada tarea de esclavo en un ciclo al transmitir un encabezado, el cual consiste en una secuencia de interrupción-sincronización-ID. Antes de comenzar la transmisión, cada esclavo es configurado para publicar datos al bus o suscribir a datos en respuesta a un ID de encabezado recibido. Una vez recibido el encabezado, cada esclavo hace la verificación de la paridad de ID y después comprueba el ID para determinar si es necesario publicar o suscribir. Si la tarea de esclavo necesita publicar una respuesta, transmite la respuesta que también contiene la suma de verificación (Checksum). Si la tarea de esclavo necesita suscribirse, lee la carga útil de los datos y el byte de la suma de verificación del bus y procede de acuerdo con la información recibida. En la figura 2 se describe gráficamente el esquema de comunicación Maestro-Esclavo. [9]

En una comunicación estándar de esclavo a maestro, el maestro transmite el identificador a la red y solamente un esclavo responde.

La comunicación de maestro a maestro se logra por una tarea de esclavo diferente en el nodo maestro. Esta tarea auto recibe todos los datos publicados al bus y responde como si fuera un nodo esclavo independiente.

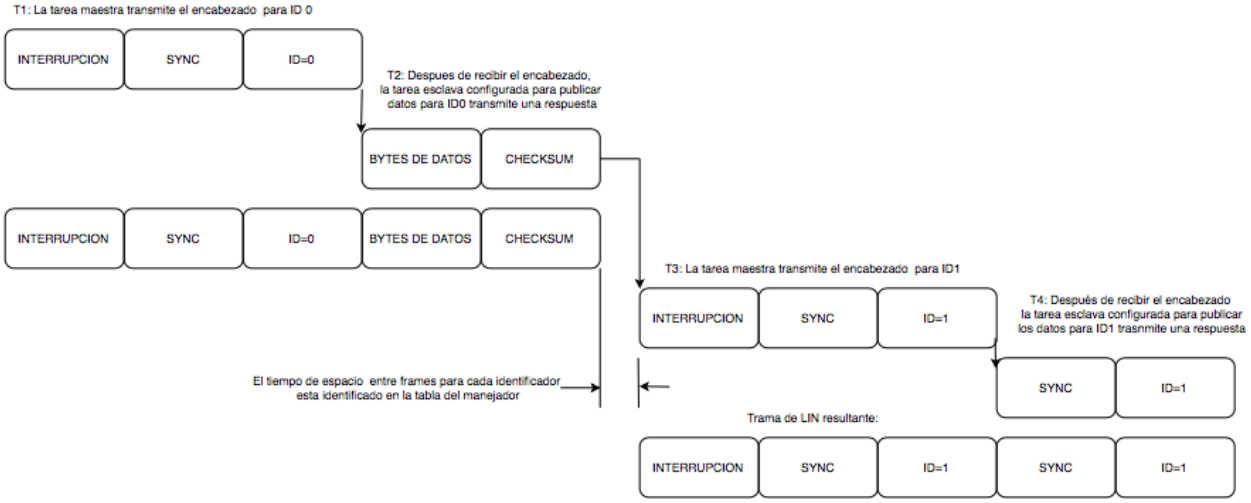


Figura 2: Comunicación Maestro-Esclavo [9]

3.1.1 Formato de la trama de LIN

Como se mencionó anteriormente, la trama de LIN se divide en dos partes: encabezado y respuesta. A continuación, se describen las subdivisiones de ambas partes. [7]

- **Break,** transmitida al principio del encabezado, comienza con la interrupción, que consta de 13 bits dominantes seguidos por un delimitador de un bit recesivo. Esto sirve como una nota de inicio del marco para todos los nodos en el bus. [9]
- **Sincronización,** está definida por el carácter 0x55. Este campo permite a los dispositivos esclavos medir el periodo de la razón de transferencia y ajustar sus razones internas para sincronizarse con el bus. [9]
- **ID,** último campo del encabezado. Proporciona identificación para cada mensaje en la red y determina el nodo en la red que recibe o responde a cada transmisión. El bus LIN proporciona un total de 64 IDs. [8]
- **Datos,** es el primer campo de la respuesta, se comprende uno a ocho bytes de carga útil de datos. [8]
- **Suma de Verificación (checksum),** el campo de la suma de verificación es transmitido por la tarea de esclavo en la respuesta. LIN define el uso de uno de dos algoritmos de suma de verificación para calcular el valor en el campo de la suma

de verificación de ocho bits. La suma de verificación clásica es calculada al sumar solamente los bytes de datos, y la suma de verificación mejorada es calculada al sumar los bytes de datos y el ID protegido. [9]

3.2. Single Edge Nibble Transmission (SENT)

SENT es un protocolo de comunicación unidireccional de punto a punto de un sensor o dispositivo de transmisión hacia un controlador o dispositivo de recepción, normalmente un ECU (Electronic Control Unit), el cual no incluye una señal de coordinación proveniente del receptor. La señal del sensor es transmitida como una serie de pulsos que contiene información que puede ser medida entre el tiempo de los flancos de bajada.

SENT es utilizado en aplicaciones donde los datos de sensores de alta resolución necesitan ser comunicados de un sensor a un ECU. Está pensado como un reemplazo para los métodos de baja resolución de un ADC de 10 bits y un PWM y como una alternativa de bajo costo comparado con CAN o LIN. La implementación asume que el sensor es inteligente y contiene un microprocesador o un circuito integrado de aplicación específica (ASIC) para crear la señal de SENT [10].

3.2.1 Características de SENT:

Las características principales del protocolo SENT que se especifican en estándar creado por la SAE [10] son:

- Protocolo unidireccional de punto a punto.
- 20% de variaciones en el reloj son permitidas por el transmisor.
- Debido al largo tiempo de subida y bajada del pulso, SENT es más robusto al ruido.
- Inmune a perturbaciones electromagnéticas [11].
- Protocolo de voltaje asíncrono que solo requiere 3 señales [12]:
    - Señal de datos (Estado bajo < 0.5 V, Estado Alto > 4.1V).
    - Señal voltaje (5 V).
    - Señal de tierra (0 V).

- Tiempo de transmisión es dependiente de los valores de los datos a ser transmitidos y la variación del reloj del transmisor.
- Entre 3 y 90 microsegundos puede durar el periodo de un tick de reloj.
- El tiempo de transmisión para el mensaje de datos más largo y la máxima variación del reloj del transmisor es menor que 1.0 milisegundos con el tiempo de tick de 3 microsegundos.

3.2.2 Descripción de la trama

La trama de un mensaje de SENT consiste en una secuencia de pulsos que es enviada repetitivamente por el módulo transmisor. Un mensaje se compone de los siguientes segmentos: pulso de sincronización o calibración, pulso de estatus o comunicación, pulso de datos, pulso de CRC (Cyclic Redundancy Check) y pulso de Pausa [3]. En la figura 3 se muestra cómo está compuesta la trama de un mensaje de SENT.

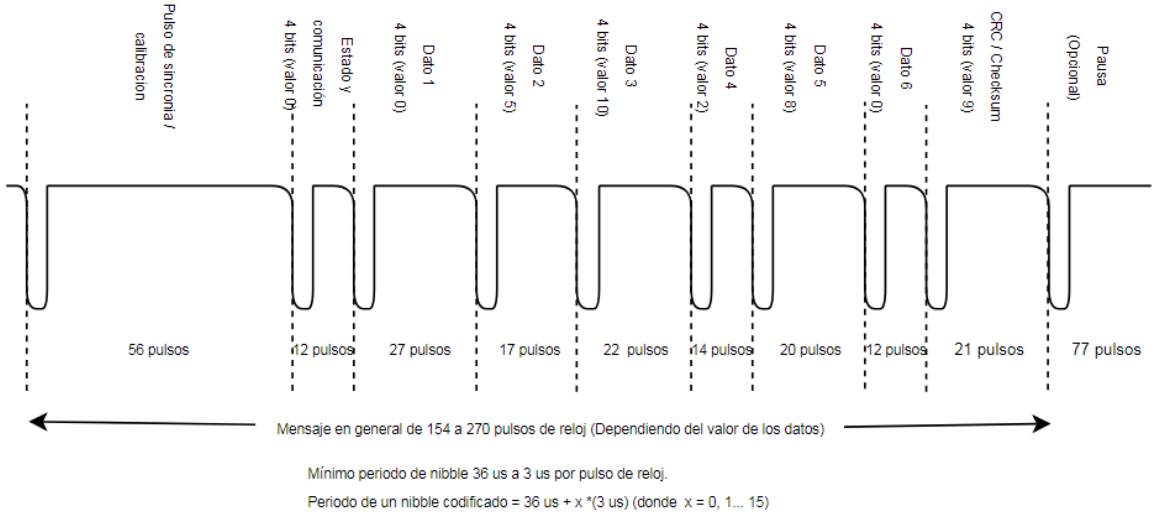


Figura 3: Trama mensaje SENT

Pulso de sincronización:

- Periodo nominal de pulso de 56 ticks de reloj.
- Al menos los primeros 5 ticks deben estar estado bajo y el resto en estado alto.

- Es usado por el receptor para volver a sincronizar con el transmisor midiendo la unidad exacta de tiempo de este pulso.

Pulso de estatus o comunicación:

- Nibble (4 bits) de 12 a 27 ticks de reloj.
- Es usado para propósitos de diagnósticos.
- Los primeros 2 bits son reservados para aplicaciones específicas como numero de parte e información de errores.
- Los 2 últimos bits pueden ser utilizados para enviar un mensaje serial de tipo corto o aumentado.

Pulso de Datos:

- Se utiliza para transmitir los datos actuales de los sensores al ECU.
- Un máximo de 6 nibbles de datos (24 bits) pueden ser transmitidos en un mensaje completo.
- Los 24 bits de datos son divididos en 2 señales, cada una está conformada por 3 nibbles.
- El ancho del pulso del nibble de datos varía de acuerdo al valor del nibble y se calcula de acuerdo a la siguiente formula: Ancho de pulso (ticks de reloj) = (12 + Valor de nibble), donde el valor puede ser entre 0 y 15.

Pulso de CRC:

- Nibble de 4 bits con una verificación de redundancia cíclica.
- Es calculado usando el polinomio x^4 + x^3 +x^2 + 1 con una semilla de valor 5.
- El cálculo solo incluye los nibbles de datos y no el nibble de estatus.
- Con este pulso es posible detectar errores de un solo bit, errores de número impares de bit consecutivos, errores de ráfaga individuales de longitud menor o igual a 4, el 87.5% de los errores de ráfaga individuales de longitud de 5 y el 93.75% de los errores de ráfaga de longitud mayor a 5.

Pulso de Pausa:

- Pulso opcional para crear transmisiones con un número constante de ticks.

- Longitud mínima de 12 ticks.
- Longitud máxima de 768 ticks.

Para la referencia de información del protocolo LIN fue utilizada la especificación técnica 1.3 [13] y para SENT fue utilizada el estándar de la SAE [10].

4. Metodología

4.1. Análisis Teórico

Para el análisis teórico se optó por empezar identificando una serie de parámetros comparativos que pueden ayudar en la selección de uno u otro protocolo, dependiendo de los requerimientos específicos de cualquier aplicación. Cada rubro es ampliamente explicado a continuación.

- **Velocidad de transmisión:** La velocidad de transmisión de datos mide el tiempo que tarda un ECU en difundir en la línea de transmisión un paquete de datos a enviar. El tiempo de transmisión se mide desde el instante en que se pone el primer bit en la línea hasta el último bit del paquete a transmitir.
- **Dirección de los datos:** Nos indica el sentido del flujo de los datos.
- **Número de nodos:** Nos indica cuántos elementos puede tener la red.
- **Detección de errores:** Nos indica los mecanismos que ofrece el protocolo para detección de errores.
- **Dependencias de hardware:** Nos indica las necesidades de un hardware especializado para el correcto funcionamiento de los protocolos.
- **Eficiencia del mensaje:** Nos indica el porcentaje de datos útiles de información de cada protocolo.
- **Control de acceso al medio:** es el conjunto de mecanismos y protocolos por los que varios "interlocutores" (dispositivos en una red), se ponen de acuerdo para compartir un medio de transmisión común.
- **Identificador de mensajes:** Se refiere a la capacidad de identificar mensajes en la línea de transmisión.
- **Capa física:** Se refiere a la comunicación en la capa más baja que es el Hardware, y a la cantidad de cables que son necesarios para establecer la conexión.
- **Longitud del mensaje:** Es la cantidad de bits máxima a ser transmitidos en un solo mensaje.

- **Power Saving:** El ahorro de energía es un tema de suma importancia en cualquier implementación electrónica, en concreto este apartado se refiere a la capacidad del protocolo para proporcionar modos de ahorro de energía que permitan bajar el consumo mientras está operando.
- **Datos útiles por cada 100ms:** Debido a que en un mensaje en ambos protocolos se encuentra información que no son datos útiles a transmitir, esta métrica nos ayuda determinar cuanta información útil es posible enviar por unidad de tiempo.
- **Consumo de corriente:** Se refiere a la cantidad de corriente que consume el hardware típico sin contar el microcontrolador, pues este consumirá dependiendo de la implementación.
- **Costo de Desarrollo:** Hace referencia al costo que ser necesita en la industria para desarrollar o comprar una implementación de los protocolos en cuestión.

4.2. Implementación LIN

Por practicidad se dividió en dos partes la implementación de LIN: transmisión y recepción, que son usadas tanto como por el maestro (ECU) el cual inicia la transmisión mediante la petición del estatus del módulo esclavo.

4.2.1 Recepción

El diagrama de estados de la figura 4 indica el proceso simplificado de recepción en el módulo esclavo. La función principal encargada de llevar a cabo la recepción es “vfnCheckCommands”, que es llamada por el agendador de tareas cada 2ms.

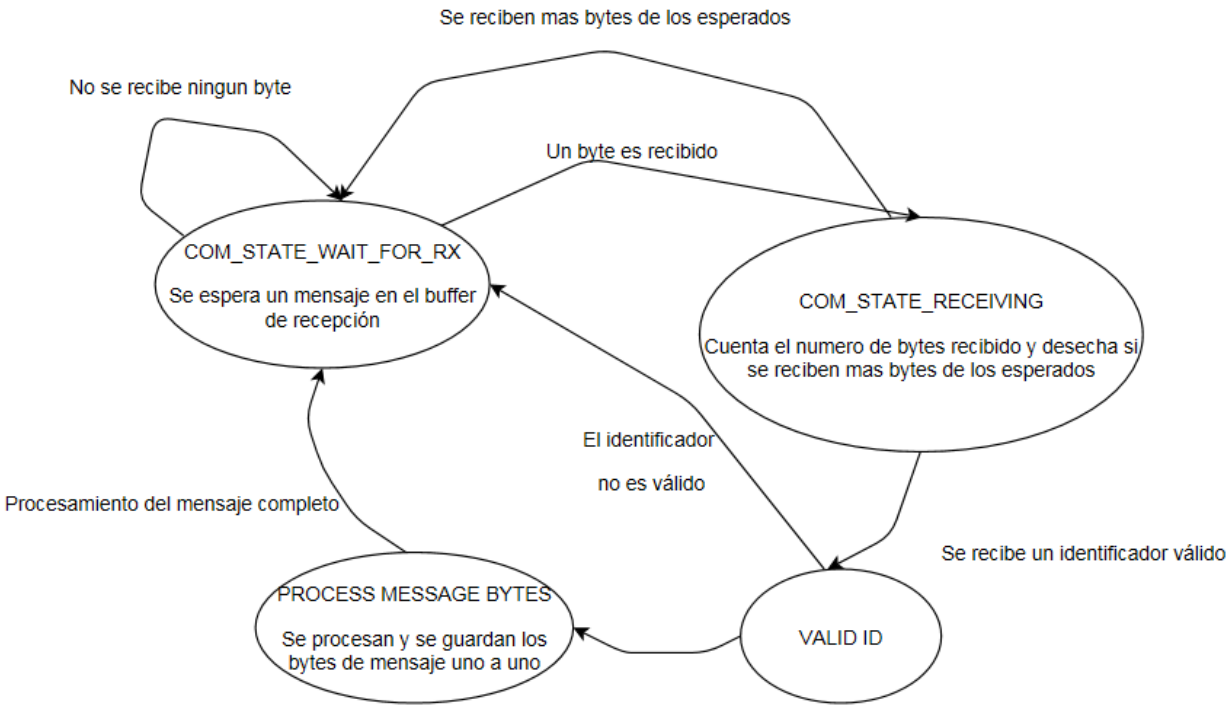


Figura 4: Diagrama de estados de recepción LIN

Diversas funciones auxiliares son utilizadas para realizar la recepción. Por simplicidad del diagrama dichas funciones no se incluyeron, pero serán explicadas a continuación:

- vfnCheckCommands:
    - Parámetros de entrada: No tiene.
    - Parámetros de salida: No tiene.
    - Descripción: Esta función se ejecuta en la tarea de 2ms. La función consiste en 2 máquinas de estados, una habilitada para el módulo del sensor y otra para el módulo maestro. Su propósito es validar cada uno de los parámetros de la trama de LIN y modificar una variable global que indica que la recepción del mensaje fue exitosa.

- vfnCOM_Init:
    - Parámetros de entrada: No tiene.
    - Parámetros de salida: No tiene.
    - Descripción: Esta función es llamada cuando el tiempo de expiración de LIN es completado y descarta toda comunicación previa.

- u8SCI_CheckRx:
    - Parámetros de entrada: Número de puerto.
    - Parámetros de salida: Número de bytes en el buffer de recepción.
    - Descripción: regresa el número de bytes en el buffer de recepción.

- vfnSCI_ClearRx:
    - Parámetros de entrada: Número de puerto.
    - Parámetros de salida: No tiene.
    - Descripción: Descarta cualquier dato recibido en el buffer de recepción y restablece las variables auxiliares a su valor por defecto.

- u8SCI_ReadRx:
    - Parámetros de entrada: Número de puerto.
    - Parámetros de salida: Valor del byte leído.
    - Descripción: regresa el valor de un byte leído del buffer de recepción.

4.2.2 Transmisión

4.2.2.1 Transmisión (ECU)

La petición de estatus del módulo esclavo se realiza cada 10ms mediante la tarea "TASKS_LIST_10MS". Esta, a su vez, dispara una serie de funciones que se encargan de "armar" el encabezado de LIN a ser enviado hacia el esclavo. A continuación, se describen las funciones usadas para la transmisión del encabezado, en el orden en el que son llamadas; adicionalmente en la Figura 5 se muestra el diagrama de secuencia de la transmisión.

- vfnTxStateMachine
    - Parámetros de entrada: N/A.
    - Parámetros de salida: N/A.

- Descripción: Esta función es encargada de emular la tabla .ldf de LIN que es usada para la sincronización de la transmisión de los mensajes a ser mandados. Dado que se plantea mandar un único mensaje, este se difunde cada 10ms. Siempre que la función vfnTxStateMachine es llamada se ejecutan las siguientes dos funciones: vfnSendHeader y vfn_TxFrame

- vfnSendHeader
  - Parámetros de entrada: Identificador de trama a ser enviada.
  - Parámetros de salida: N/A.
  - Descripción: Se encarga de comenzar la construcción del encabezado de la trama, mediante la función vfnConstructHeader, proporcionando el identificador a ser mandado.

- vfnConstructHeader
  - Parámetros de entrada: valor a ser mandado en el campo de sincronización, ID de trama a ser enviada.
  - Parámetros de salida: N/A.
  - Descripción: Se encarga de llenar el campo de sincronización y del identificador en las posiciones adecuadas de la trama en el arreglo u8TxResponse.

- vfn_TxFrame
  - Parámetros de entrada: Tamaño del buffer a ser enviado.
  - Parámetros de salida: N/A.
  - Descripción: Se encarga de comenzar la transmisión del encabezado, usando la función vfnSCI_WriteBufferTx que se encarga de la transmisión a bajo nivel.

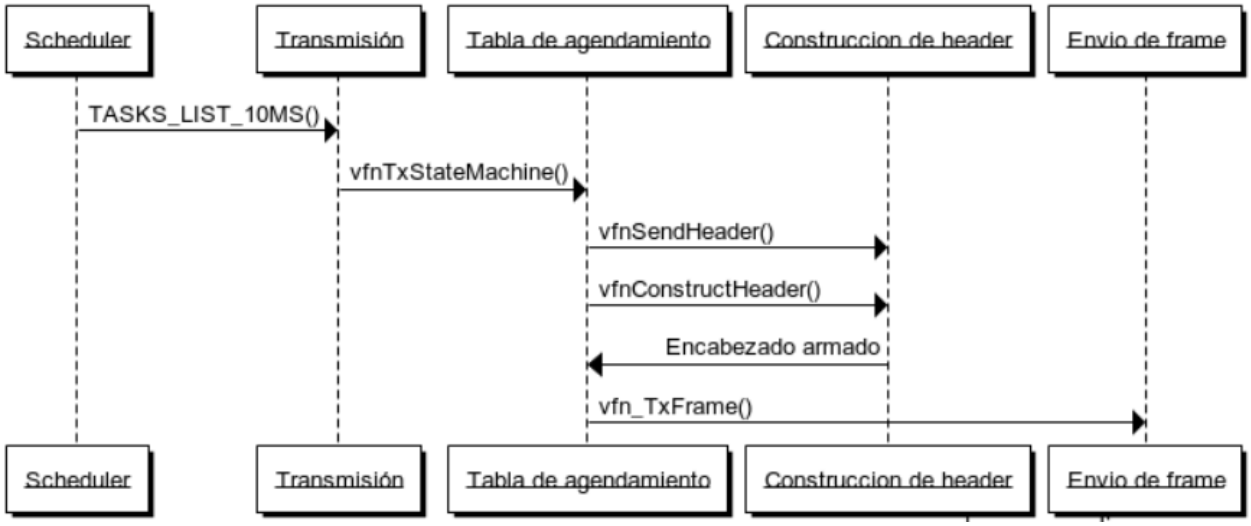


Figura 5: Secuencia de transmisión modulo maestro

4.2.2.2 Transmisión (Esclavo)

La transmisión por parte del módulo esclavo inicia cuando se ha recibido una petición de estatus válida por parte del ECU. A continuación, se presentan las funciones usadas para la transmisión de la respuesta en el orden que son ejecutadas; adicionalmente en la Figura 6 se muestra el diagrama de secuencia de la trasmisión en el módulo esclavo.

- vfnConstructResponseFrame
    - Parámetros de entrada: La trama a ser transmitida.
    - Parámetros de salida: N/A.
    - Descripción: Esta función se encarga de armar la trama a ser enviada almacenando cada uno de los bytes a ser transmitidos; para ello, primero obtiene las señales a ser transmitidas, después integra cada una de las señales en el arreglo u8TxResponse, posteriormente calcula el “checksum” y lo almacena justo después del campo de datos.

- Checksum
    - Parámetros de entrada: Tipo de trama a ser calculada, ya sea trama de respuesta o trama recibida.
    - Parámetros de salida: Checksum calculado.
    - Descripción: Esta función es encargada de calcular el “checksum” para las tramas recibidas o a ser transmitidas en una respuesta de esclavo. Una vez hecho el cálculo se proporciona como parámetro de retorno.

- vfn_TxFrame
    - Parámetros de entrada: Tamaño del buffer a ser enviado.
    - Parámetros de salida: N/A.
    - Descripción: Una vez armada la trama a ser transmitida nuevamente se usa la función vfn_TxFrame para transmitir la respuesta mediante la función vfnSCI_WriteBufferTx que se encarga de la transmisión a bajo nivel.

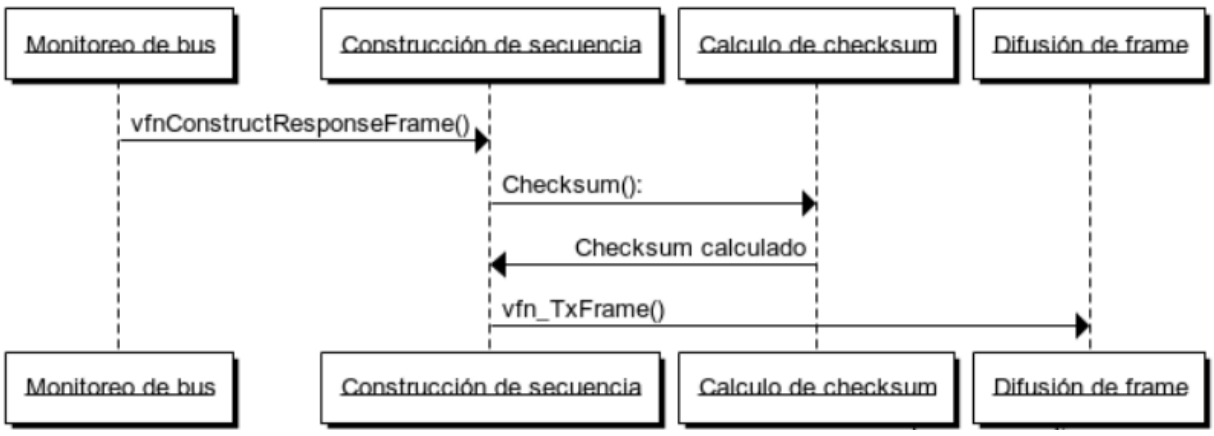


Figura 6: Secuencia de transmisión esclavo

4.3. Implementación SENT

SENT fue pensado para ser dividido por los dos nodos que componen la comunicación. El nodo que tendrá la función de sensor es el que transmite los datos al nodo que será el ECU, el cual recibe los datos.

A diferencia del protocolo LIN, cada nodo sólo desempeña una función, pues solo uno envía y solo uno recibe los datos. Esto debido, a la naturaleza unidireccional de SENT.

4.3.1 Recepción

El diagrama de la figura 7 nos muestra la secuencia para la recepción, Empezando por la inicialización del hardware, la habilitación de la interrupción en el procesador secundario. Cada vez que la interrupción de flanco de bajada es activada, la rutina de interrupción guarda el tiempo entre cada flanco de bajada, pudiendo así determinar el tiempo de cada pulso del protocolo. Una vez que todos los pulsos son guardados, se genera una interrupción de software que es ejecutada por el procesador primario. Esta rutina tiene la tarea de decodificar las muestras almacenadas y validar si el mensaje es correcto.

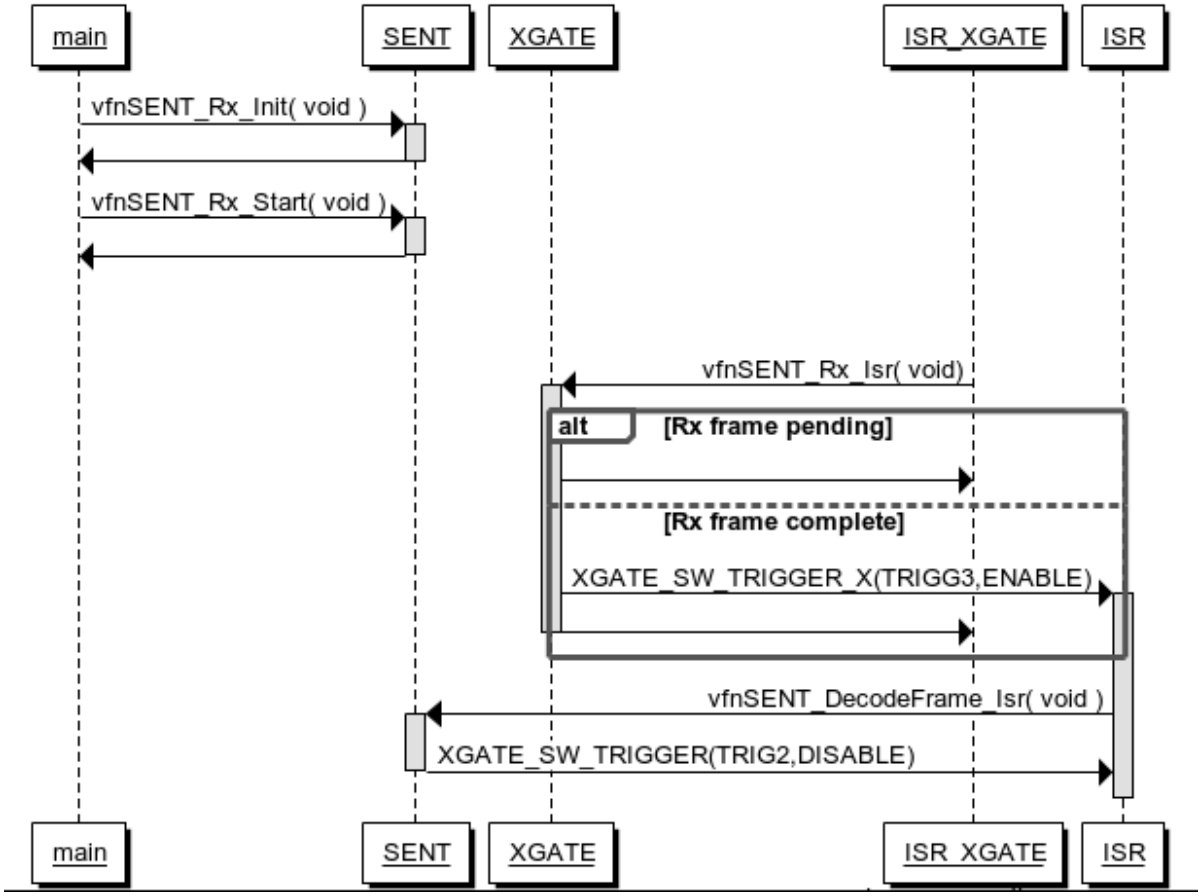


Figura 7: Secuencia de Recepción

- vfnSENT_Rx_Init:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.
  - Descripción: Esta función se ejecuta una sola vez al arrancar el microcontrolador, es la encargada de configurar los registros de los periféricos utilizados, como lo son el temporizador para medir el tiempo de cada pulso, las interrupciones a ser utilizadas, como lo es la interrupción de flanco de bajada.

- vfnSENT_Rx_Start:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.

- Descripción: Esta función se ejecuta una sola vez al arrancar el microcontrolador, es la encargada de habilitar la interrupción de flanco de bajada para así poder inicializar la recepción.

- vfnSENT_Rx_Isr:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.
  - Descripción: Esta función se ejecuta en el procesador secundario cada vez que la interrupción de flanco de bajada sucede. La primera parte de esta función es la encargada de generar el diferencial de tiempo cada vez que esta función es llamada. Después de que obtiene el diferencial de tiempo, se busca adquirir el pulso de sincronía, que tiene una duración predefinida por el estándar. Una vez que este pulso es detectado, la máquina de estados transiciona al estado para almacenar los datos de los siguientes pulsos, que contienen un nibble de datos cada uno. Para esta investigación en específico, serán enviados seis nibbles en total, de los cuales, el primero es de control, los cuatro siguientes son de datos y, el último, es de CRC. Una vez que todos los datos son almacenados, esta función genera una interrupción de software, para poder decodificar y validar los datos que han sido muestreados y almacenados. Cabe resaltar que estos datos son almacenados en una sección de memoria que es compartida entre procesadores, pues ambos necesitan acceso a esta sección de memoria, pues uno escribe en ella y otro lee.

- XGATE_SW_TRIGGER:
  - Parámetros de entrada:
    - u8TriggerNumber [0-7]: Número de la interrupción a ser llamada.
    - u8TriggerMask [0-1]: Máscara que habilita o deshabilita la interrupción.
  - Parámetros de salida: No tiene.
  - Descripción: Esta función se ejecuta cada vez que todos los datos son almacenados en el procesador secundario. Se encarga de habilitar y deshabilitar las

interrupciones de software, dependiendo del valor que tenga la máscara en el parámetro de entrada.

- vfnSENT_DecodeFrame_Isr
    - Parámetros de entrada: No tiene.
    - Parámetros de salida: No tiene.
    - Descripción: Esta función se ejecuta cada vez que la interrupción de software número 3 es activada. Es encargada de decodificar los tiempos guardados por vfnSENT_Rx_Isr y convertirlos en pulsos SENT, posteriormente en valores y, finalmente, validar los datos usando el CRC. Además, esta función deshabilita la interrupción de software que fue anteriormente habilitada en vfnSENT_Rx_Isr.

4.3.2 Transmisión

Muy similar a la recepción, la transmisión primero nos muestra la inicialización y la manera en la que se habilita la transmisión. Posteriormente nos muestra cómo funciona en sí la transmisión, la cual se basa en un temporizador que está configurado de tal forma que cada tiempo que corresponde a un pulso de SENT genera una interrupción, que decide si el pin de salida donde la señal de SENT es transmitida tiene un valor en alto o en bajo. Esta información indica el estado del pin, la cual es almacenada en un arreglo que posee una dimensión igual al máximo número de pulsos que un mensaje completo de SENT puede tener. Una vez terminado el mensaje de SENT, los valores a ser guardados en el arreglo son recalculados con los valores actuales del sensor. Y el ciclo de transmisión inicia nuevamente. Esta secuencia es mostrada en el diagrama de secuencias de la figura 8.

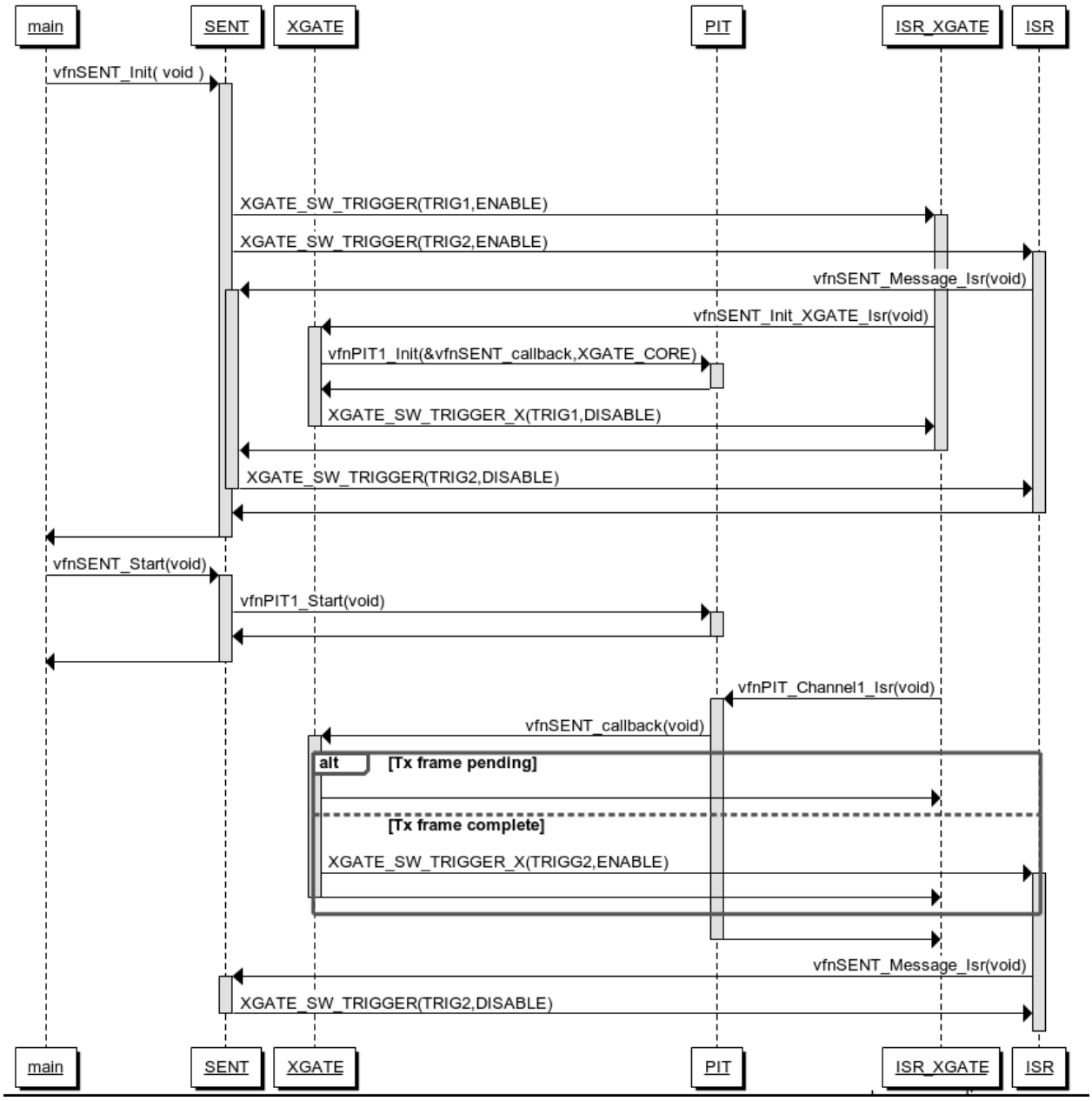


Figura 8: Secuencia de Transmisión

- vfnSENT_Init:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.

- Descripción: Esta función se ejecuta una sola vez al arrancar el microcontrolador. Es la encargada de configurar los registros de los periféricos utilizados, como lo son el temporizador para medir el tiempo periódico de cada pulso.

- vfnSENT_Start:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.
  - Descripción: Esta función se ejecuta una sola vez al arrancar el microcontrolador. Es la encargada de habilitar la interrupción de flanco de bajada para así poder inicializar la transmisión.

- vfnSENT_callback:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.
  - Descripción: Esta función se ejecuta periódicamente con una separación entre cada ejecución del tamaño de cada pulso SENT. Tiene la responsabilidad de determinar si el estado del pin de salida está en estado alto o bajo. También debe determinar si el mensaje ha sido completo para así poder actualizar los valores a transmitir; esto lo logra habilitando una interrupción de software.

- vfnSENT_Mesage_Isr:
  - Parámetros de entrada: No tiene.
  - Parámetros de salida: No tiene.
  - Descripción: Esta función se ejecuta cuando una interrupción de software es habilitada. Su tarea es actualizar el arreglo de donde se indica el estado de pin de salida. Además, esta función deshabilita la interrupción de software que causó que esta función fuera llamada.

4.4. Aplicación

Se implementó la siguiente aplicación, utilizada para obtener resultados prácticos y complementar el análisis teórico, y permitan cumplir con los objetivos definidos en la investigación.

La aplicación es un conjunto de 2 tipos de dispositivos: una botonera y un ECU, que se comunican mediante el protocolo LIN o SENT. Para ello, se utilizaron cuatro tarjetas de desarrollo DEMO9S12XEP100: una botonera y un ECU con capacidades de comunicación LIN, y una botonera y un ECU con capacidades de comunicación SENT.

El ECU es un dispositivo de control de motores conocido como DSM (Driver Seat Module). Este es capaz de mover ocho motores que conforman un asiento en ambas direcciones para obtener una posición adecuada por el ocupante en el vehículo. Para saber qué motor debe moverse y en qué dirección, normalmente es a base de botones, que pueden estar conectados directamente al ECU por medio de señales eléctricas o puede existir otro dispositivo que se encarga del procesamiento de las señales eléctricas de los botones y se comunica con el ECU por medio de un protocolo de comunicación. Este último caso es el presentado en nuestra aplicación como la botonera. La Botonera, como se menciona anteriormente, enviará por medio de LIN o SENT la información sobre qué botón está presionado de los 16 disponibles, 2 por cada dirección de motor.

Para nuestra aplicación, con el fin de obtener resultados de una forma sencilla de comparar y sin gastos extras de componentes electrónicos, se eligió simular los botones y los motores mediante mensajes CAN. Para ello, se utilizó un driver de CAN dentro la implementación de cada uno de los dispositivos (Botoneras y ECUs), y mediante un Equipo de CAN (VN1610), que es capaz de enviar y recibir mensajes de CAN por medio del software CANoe. Por medio del VN1610 se envía un mensaje que es recibido por ambas botoneras; el mensaje contiene qué botones están presionados. Cada botonera enviará la información por su respectivo protocolo de comunicación y cada ECU procesará esa información y por medio de un mensaje de CAN indicará qué motor se debe prender. En la figura 9 se muestra el Diagrama de interacción de la aplicación.

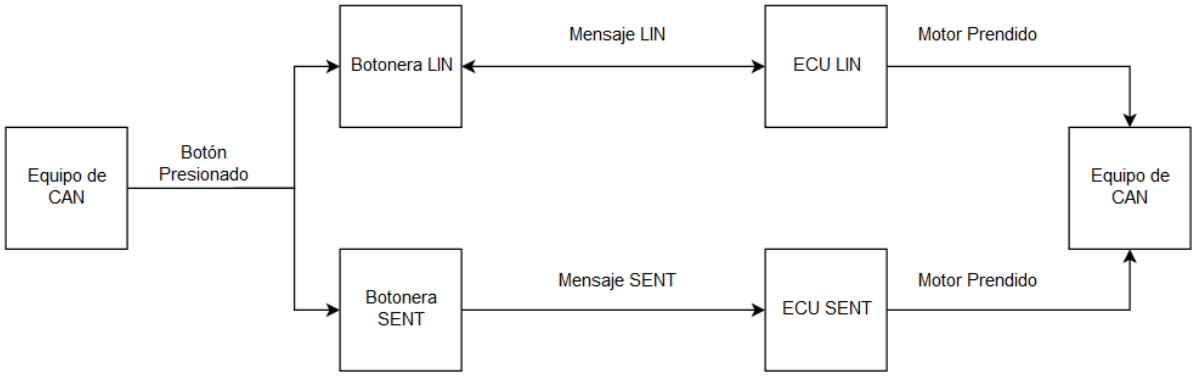


Figura 9: Diagrama de interacción de Dispositivos

4.4.1 Aplicación LIN

La botonera y el ECU se comunican mediante el protocolo LIN con una velocidad de 19200 kbps. El ECU es el nodo maestro en la red y la botonera es un nodo esclavo. El ECU es el encargado de iniciar la comunicación mediante el envío de un mensaje cada 10 milisegundos. El mensaje enviado tiene un ID 0xD3. La botonera monitorea el bus cada 2 ms para detectar el inicio de un nuevo mensaje, después valida el ID; en el caso que el ID sea el 0x3D, envía en 2 bytes los datos recibidos mediante el mensaje de CAN que indican qué botones están presionados. El ECU al recibir los datos los procesa y envía, mediante un mensaje de CAN, los motores que deberán estar encendidos. El acomodo en los 2 bytes en el mensaje LIN de los botones es el siguiente:

- Byte 0
    - Bit 0: Botón Motor 0 – Dirección hacia delante.
    - Bit 1: Botón Motor 0 – Dirección hacia atrás.
    - Bit 2: Botón Motor 1 – Dirección hacia delante.
    - Bit 3: Botón Motor 1 – Dirección hacia atrás.
    - Bit 4: Botón Motor 2 – Dirección hacia delante.
    - Bit 5: Botón Motor 2 – Dirección hacia atrás.
    - Bit 6: Botón Motor 3 – Dirección hacia delante.
    - Bit 7: Botón Motor 3 – Dirección hacia atrás.

- Byte 1
    - Bit 0: Botón Motor 4 – Dirección hacia delante.
    - Bit 1: Botón Motor 4 – Dirección hacia atrás.
    - Bit 2: Botón Motor 5 – Dirección hacia delante.
    - Bit 3: Botón Motor 5 – Dirección hacia atrás.
    - Bit 4: Botón Motor 6 – Dirección hacia delante.
    - Bit 5: Botón Motor 6 – Dirección hacia atrás.
    - Bit 6: Botón Motor 7 – Dirección hacia delante.
    - Bit 7: Botón Motor 7 – Dirección hacia atrás.

4.4.1 Aplicación SENT

La botonera y el ECU se comunican mediante el protocolo SENT con la máxima velocidad de 3 microsegundos por tick de reloj. En esta aplicación, al no existir una red basada en Maestro-Esclavo, es la botonera la que actúa como un sensor, y envía los datos a la velocidad máxima cíclicamente; como en la aplicación de LIN explicada previamente, la botonera recibe los datos mediante el mensaje de CAN que indican qué botones están presionados. El ECU detecta el pulso de sincronización y obtiene los datos de la botonera, los procesa y envía mediante un mensaje de CAN los motores que deberían estar encendidos. El acomodo en los 4 nibbles bytes en el mensaje SENT de los botones es el siguiente:

- Nibble 0
    - Bit 0: Botón Motor 0 – Dirección hacia delante.
    - Bit 1: Botón Motor 0 – Dirección hacia atrás.
    - Bit 2: Botón Motor 1 – Dirección hacia delante.
    - Bit 3: Botón Motor 1 – Dirección hacia atrás.

- Nibble 1
    - Bit 0: Botón Motor 2 – Dirección hacia delante.
    - Bit 1: Botón Motor 2 – Dirección hacia atrás.

- Bit 2: Botón Motor 3 – Dirección hacia delante.
- Bit 3: Botón Motor 3 – Dirección hacia atrás.

- Nibble 2
  - Bit 0: Botón Motor 4 – Dirección hacia delante.
  - Bit 1: Botón Motor 4 – Dirección hacia atrás.
  - Bit 2: Botón Motor 5 – Dirección hacia delante.
  - Bit 3: Botón Motor 5 – Dirección hacia atrás.

- Nibble 3
  - Bit 0: Botón Motor 6 – Dirección hacia delante.
  - Bit 1: Botón Motor 6 – Dirección hacia atrás.
  - Bit 2: Botón Motor 7 – Dirección hacia delante.
  - Bit 3: Botón Motor 7 – Dirección hacia atrás.

4.4.2 Interfaz CAN

El mensaje que se envía a la botonera tiene un ID 0x1abcdeff con un DLC de 2 bytes que representan los botones presionados.

- Byte 1
  - Bit 0: Botón Motor 0 – Dirección hacia delante.
  - Bit 1: Botón Motor 0 – Dirección hacia atrás.
  - Bit 2: Botón Motor 1 – Dirección hacia delante.
  - Bit 3: Botón Motor 1 – Dirección hacia atrás.
  - Bit 4: Botón Motor 2 – Dirección hacia delante.
  - Bit 5: Botón Motor 2 – Dirección hacia atrás.
  - Bit 6: Botón Motor 3 – Dirección hacia delante.
  - Bit 7: Botón Motor 3 – Dirección hacia atrás.

- Byte 2

- Bit 0: Botón Motor 4 – Dirección hacia delante.
- Bit 1: Botón Motor 4 – Dirección hacia atrás.
- Bit 2: Botón Motor 5 – Dirección hacia delante.
- Bit 3: Botón Motor 5 – Dirección hacia atrás.
- Bit 4: Botón Motor 6 – Dirección hacia delante.
- Bit 5: Botón Motor 6 – Dirección hacia atrás.
- Bit 6: Botón Motor 7 – Dirección hacia delante.
- Bit 7: Botón Motor 7 – Dirección hacia atrás.

El mensaje que envía el ECU tiene un ID 0x111 para LIN y 0x222 para SENT con un DLC de 2 bytes que representan el estado del motor.

- Byte 1
  - Bit 0: Estado Motor 0 – Dirección hacia delante.
  - Bit 1: Estado Motor 0 – Dirección hacia atrás.
  - Bit 2: Estado Motor 1 – Dirección hacia delante.
  - Bit 3: Estado Motor 1 – Dirección hacia atrás.
  - Bit 4: Estado Motor 2 – Dirección hacia delante.
  - Bit 5: Estado Motor 2 – Dirección hacia atrás.
  - Bit 6: Estado Motor 3 – Dirección hacia delante.
  - Bit 7: Estado Motor 3 – Dirección hacia atrás.

- Byte 2
  - Bit 0: Estado Motor 4 – Dirección hacia delante.
  - Bit 1: Estado Motor 4 – Dirección hacia atrás.
  - Bit 2: Estado Motor 5 – Dirección hacia delante.
  - Bit 3: Estado Motor 5 – Dirección hacia atrás.
  - Bit 4: Estado Motor 6 – Dirección hacia delante.
  - Bit 5: Estado Motor 6 – Dirección hacia atrás.
  - Bit 6: Estado Motor 7 – Dirección hacia delante.
  - Bit 7: Estado Motor 7 – Dirección hacia atrás.

5. Resultados y Discusión

En el presente capítulo se muestran las pruebas y resultados comparativos entre los protocolos LIN y SENT, con el fin de mostrar los puntos a favor y en contra de usar cualquiera de los dos protocolos y proporcionar una mayor información para la elección del protocolo adecuado dependiendo de la aplicación especifica.

5.1. Análisis teórico

Dado que no fue posible encontrar un análisis específico de comparación de ambos protocolos, creamos una tabla comparativa para la discusión, donde se muestran algunas características principales que pueden ser decisivas en el momento de elegir por cualquiera de los protocolos.

A continuación, la Tabla 1 muestra el resultado de la comparación y un pequeño resumen y discusión de cada uno de sus rubros.

Tabla 1: Análisis Comparativo

	LIN 1.3	SENT
Velocidad de transmisión	19.2 kbit/s @40 metros	30 kbit/s
Dirección de los datos	Bidireccional	Unidireccional
Control de acceso al medio	Maestro único	N/A
Número de nodos	2 a 16 nodos	Comunicación uno a uno
Identificador de mensajes	Identificador de 6 bits	N/A
Detección de Errores	Checksum de 8 bits	CRC de 4 bits
Capa física	Un solo cable a 12v	Un solo cable a 5v,
Dependencias de hardware	Transceptor de LIN	N/A
Longitud del mensaje	1 a 64 bits	32 bits
Eficiencia del mensaje	18% a 25%	75%
“Jitter”	40%	25%
Probabilidad de encontrar un error	Encabezado 75% Respuesta 99.6%	Errores de un solo bit - 100% Errores de número impares de bit consecutivos - 100% Errores de ráfaga individuales de longitud menor o igual a 4 – 100%

		Errores de ráfaga individuales de longitud de 5 - 87.5% Errores de ráfaga de longitud mayor a 5 - 93.75%
Inmunidad contra ruido electromagnético	No encontrado	Contacto de 8kV, 15 kV aire. C = 330 pF, R=2k ohms
Ahorro de energía	Sleep Mode	N/A
Longitud del Bus	40 metros	5 metros
Datos útiles por 100 ms	.48 kb	2.329kb - 4.329kb
Consumo de corriente	Tranceptor 40mA	50 mA Corriente promedio del Receptor sobre 1 mensaje [10]
Costo de implementación	51,400 – 64,100 UDS (1)	73,200 USD(2)

Notas:
1. El costo es por cada por proyecto
2. El costo es por la primer implementación y es reusable en distintos proyectos

Velocidad de transmisión: En la tabla se observa que SENT posee una mayor velocidad de transmisión, lo cual implica una mayor tasa de transferencia de información, decisiva para algunas aplicaciones donde la respuesta en tiempo real es decisiva.

Dirección de los datos: LIN es bidireccional ya que tanto el módulo maestro como el módulo esclavo se comunican entre sí. Por el contrario, SENT solo es unidireccional. En este caso SENT está en desventaja ya que no se puede comandar ninguna acción al nodo transmisor, por tanto, no se tiene control sobre él; por el contrario, LIN, al ser bidireccional, permite una comunicación entre los ECU interconectados.

Numero de nodos: En LIN se cuenta con hasta 16 nodos, 1 solo maestro (que también puede fungir el rol de esclavo) y hasta 15 esclavos. En SENT la comunicación es directa, es una comunicación puerto a puerto. En este rubro SENT nuevamente está en desventaja ya que permite

comunicación de dos nodos a la vez, por el contrario, en LIN hasta 15 ECU pueden fungir como esclavos.

Detección de errores: LIN nos ofrece una gama de protecciones, entre las cuales se encuentran: Detección de corto circuito del medio físico de transmisión, chequeo del campo de sincronización, monitoreo de cada bit individual para su nivel lógico, error de paridad en la trama, detección de error de datos mediante un checksum, chequeo de una trama de respuesta después de una trama encabezado y un mecanismo de detección de esclavos defectuosos [8], En el caso de SENT ofrece un CRC de 4 bits. LIN en este rubro es más robusto.

Dependencias de hardware: LIN necesita un transductor para adecuar las señales de comunicación por el contrario el protocolo de SENT no necesita un transductor. En este caso SENT resulta ganador ya que se puede implementar sin necesidad de un hardware extra al microcontrolador.

Identificador de mensajes: LIN al soportar la posibilidad de múltiples nodos esclavos bus hace uso de un identificador en su trama, SENT al ser comunicación unidireccional no lo requiere.

Capa física: En ambos protocolos un solo cable es necesario, aunque SENT usa transmisión a 5V y LIN a 12V. En este rubro ambos casos están emparejados, salvo el hecho que LIN forzosamente requiere alimentación extra de 12V.

Longitud del mensaje: LIN tiene una mayor capacidad para transmitir siendo hasta 64 bits de datos los posibles en cada trama.

Power Saving: En este rubro, solo LIN proporciona el modo de "Sleep" que permite que el ECU entre en modo de ahorro de energía y, por el contrario, SENT no proporciona una estrategia definida. Algunas aplicaciones en SENT recurren a la desactivación del módulo completamente al detener el suministro de energía con el fin de detener la transmisión. Por tanto, LIN resulta ser más robusto en temas de ahorro de energía, aunque necesita más HW y una alimentación extra de 12V.

Longitud del bus: Hasta 40 m con velocidad de 19.2 kbit/s para LIN 1.3 y 5 metros máximo para SENT.

Datos útiles por cada 100ms: La especificación 1.3 del LIN nos dicta que la velocidad máxima de transmisión es de 19.2 kbps. Esto nos da que en 100ms tendremos una transmisión de 1.92 kilo bits por cada 100ms. Si a esos 1.92 kb le aplicamos la eficiencia máxima de LIN que es de 25% nos daría que en 100 ms solo .48kb de datos útiles pueden ser transmitidos por LIN en

100ms. La longitud de un mensaje de SENT es de 6 nibbles de datos, que es el máximo que el protocolo SENT contempla, varia de entre 462 a 810 microsegundos, por lo tanto, en 100ms pueden enviarse entre 123 a 216 mensajes. Entonces en 100ms pueden enviarse entre 2.469kb a 4.329kb de datos útiles. Lo cual nos otorga que SENT nos puede proporcionar entre 5.14 a 9.02 veces más información, que la que puede otorgar LIN.

Consumo de corriente: La corriente del transceptor de LIN es 10 mA menor al consumo de corriente de un receptor en SENT. Considerando que el receptor de SENT maneja un voltaje de 5V tendríamos una potencia de 250mW en SENT y en LIN al utilizar un voltaje de 12V la potencia generada en el transceptor de LIN es de 480mW.

Costo de implementación: En la parte de LIN se utilizó el precio de una implementación provista por una empresa dedicada al ramo automotriz Vector. El costo de la implementación de LIN ronda entre los 40,000 a los 50,000 dólares; además de alrededor de 11,400 dólares en equipo de pruebas suministrado por Intrepid Control System. También debe incluirse el tiempo requerido por un ingeniero en integrar la implementación suministrada por Vector a un producto en específico dando unos 6,000 dólares por el trabajo de un mes de un ingeniero. En el protocolo SENT al no tener empresas externas que tengan una solución del protocolo en venta, se estimó que el desarrollo desde cero del protocolo sería un año de esfuerzo de un ingeniero a tiempo completo, el cual cotiza en 35 dólares la hora, esto nos da 67,200 dólares por el desarrollo de una implementación para SENT, así como desarrollar herramientas necesarias verificar su funcionamiento en futuros productos. Además de que deben incluirse los costos de integración los cuales serían similares a LIN. Estos costos indican que para un solo proyecto LIN es más costeable que SENT. Aunque también cabe señalar que los costos para más de un proyecto SENT se abarata, pues el costo del desarrollo se una implementación ya fue desarrollada y puede ser reusado sin mayor costo. En cambio, en LIN al ser proporcionado por una empresa externa, cada vez que necesites este protocolo para un nuevo proyecto, el costo debe ser pagado nuevamente.

5.2. Análisis práctico

Una serie de pruebas fueron diseñadas e implementadas para proporcionar sustentabilidad al estudio comparativo que comprende este trabajo. Todas las pruebas fueron efectuadas en ambas implementaciones SENT y LIN. A continuación, se presentan los resultados de dichas pruebas.

5.2.1 LIN (ECU)

5.2.1.1 Tiempo de transmisión-Recepción

Esta prueba consiste en medir el tiempo total necesario para transmitir un frame desde que el nodo ECU inicia la transmisión pidiendo el estatus contestando el nodo botonera, y se procesa su respuesta validando el "checksum". El resultado es 4.502ms, este tiempo se ve afectado por el tiempo de retardo dentro de la trama donde se responde el estatus, esto se observa en la Figura 10.

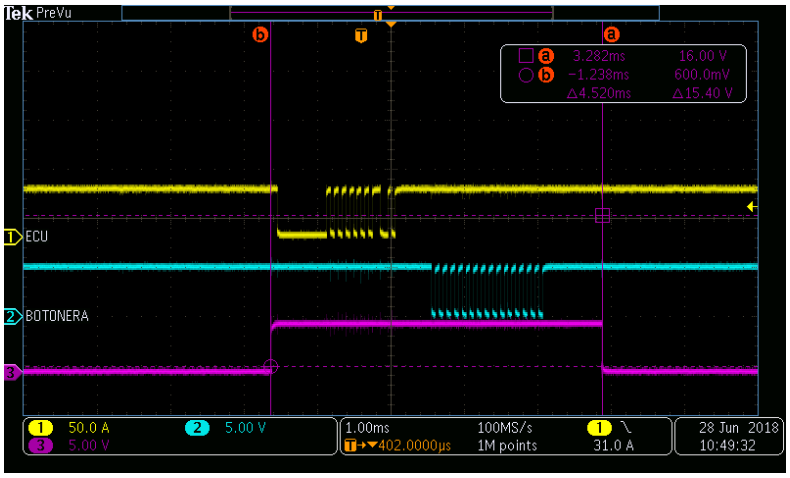


Figura 10 Tiempo transmisión recepción.

5.2.1.2 Tiempo de uso de recursos

En esta prueba se mide el tiempo de uso de recursos del procesador para transmitir una petición de estatus a la botonera, recibir la respuesta y, finalmente, procesar la misma. En esta métrica se incluyen el tiempo de procesamiento de la petición de estatus, los tiempos de poleo por parte de la recepción para recibir la respuesta de la botonera y las interrupciones necesarias para

transmitir en el bus. En la Figura 11 se muestran dichos tiempos, en donde se aprecia inicialmente un tiempo de 11.94 microsegundos (canal azul) necesarios para el procesamiento de los datos a ser transmitidos por el ECU.

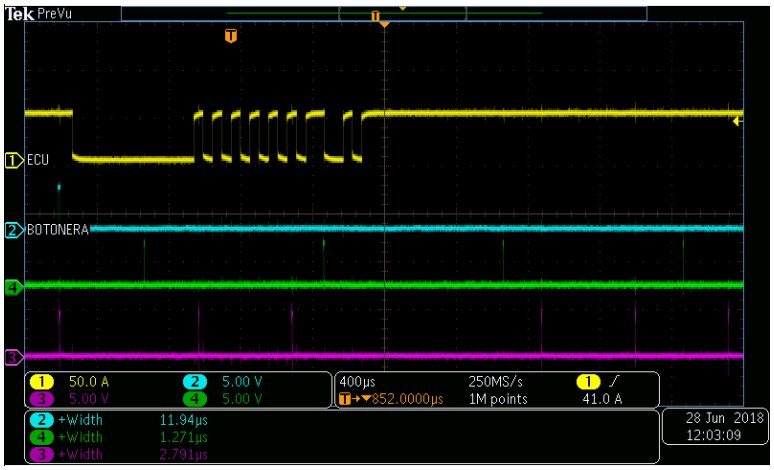


Figura 11 Uso de recursos en el ECU

En la Figura 12 se muestran los tiempos de poleo necesarios, siendo un total de 3.512 microsegundos.

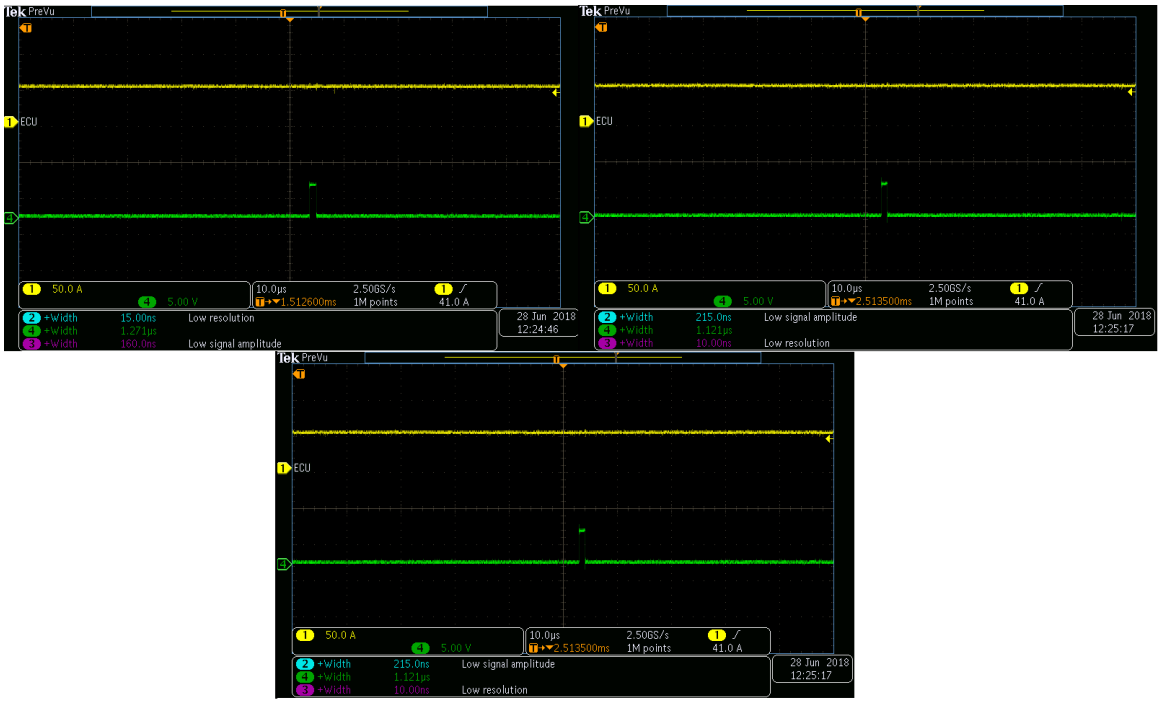


Figura 12 Tiempos de poleo

En la Figura 13 se muestran los tiempos necesarios para las interrupciones de transmisión y recepción en el nodo ECU, siendo un total de 14.988 microsegundos al sumarlos todos.

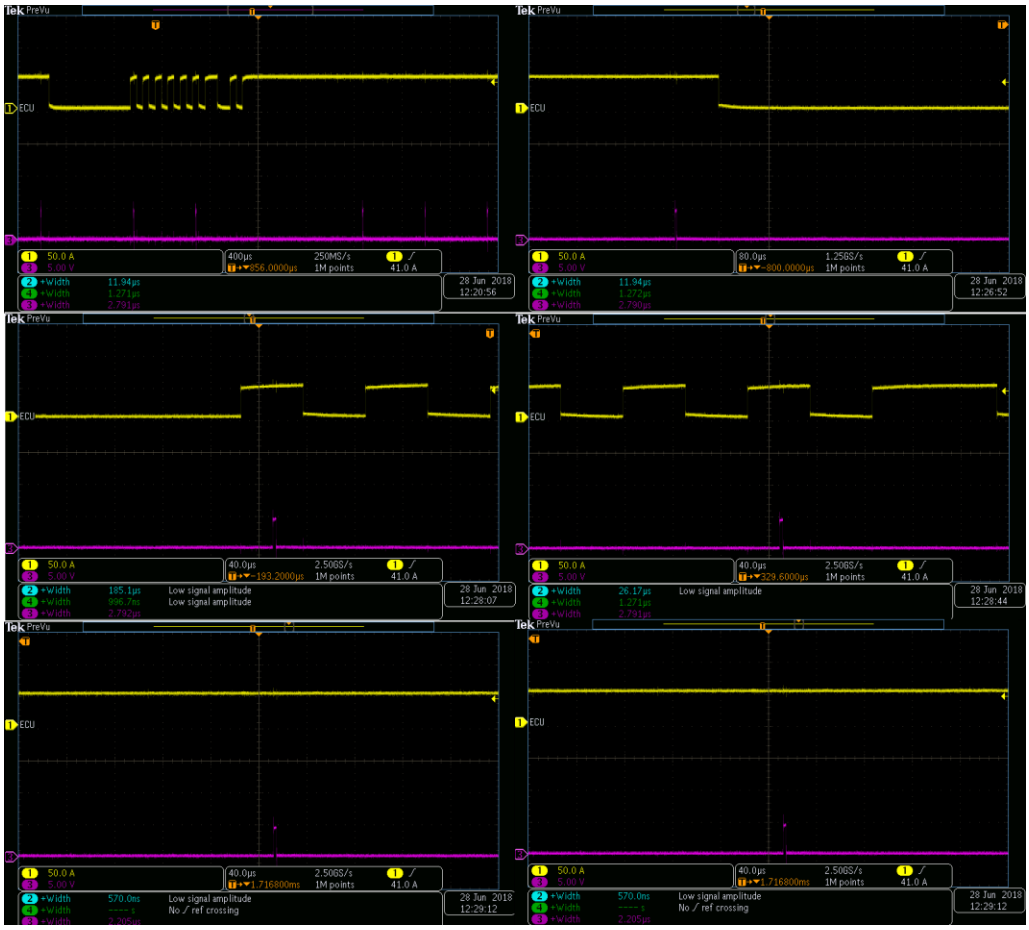


Figura 13 Tiempos de interrupción

En la implementación del ECU con el protocolo de LIN estos fueron el uso de recursos:

- ROM: 6471 bytes
- RAM: 2032 bytes

5.2.2 LIN (BOTONERA)

5.2.2.1 Tiempo de uso de recursos

En esta prueba se mide el tiempo de uso de recursos del procesador para recibir una petición de estatus del ECU, validar los identificadores y transmitir de vuelta al ECU el status de los botones. En esta métrica se incluyen el tiempo de procesamiento de la petición de estatus, los tiempos de poleo por parte de la recepción para recibir la respuesta de la botonera y las interrupciones necesarias para transmitir en el bus. En la Figura 14 se muestran dichos tiempos. Los pulsos azules representan el tiempo de CPU de la función de recepción, la línea verde representa el tiempo de transmisión de la botonera y la línea morada representa el tiempo que gastamos dentro de las interrupciones necesarias para la recepción.

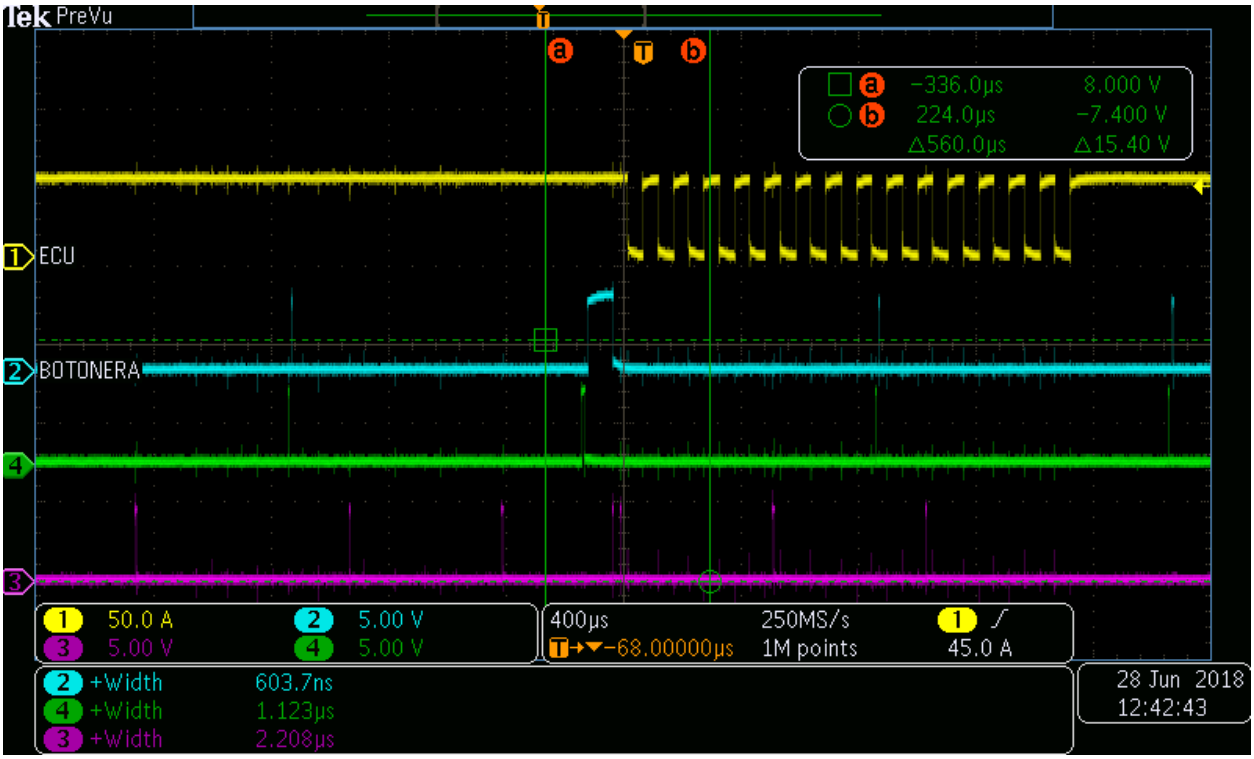


Figura 14: Tiempos de uso de CPU

En la Figuras 15 a 20 se muestran los tiempos de las interrupciones dando un total acumulado de 12.1628 microsegundos, dichas interrupciones son disparadas cuando un byte se manda o se recibe.

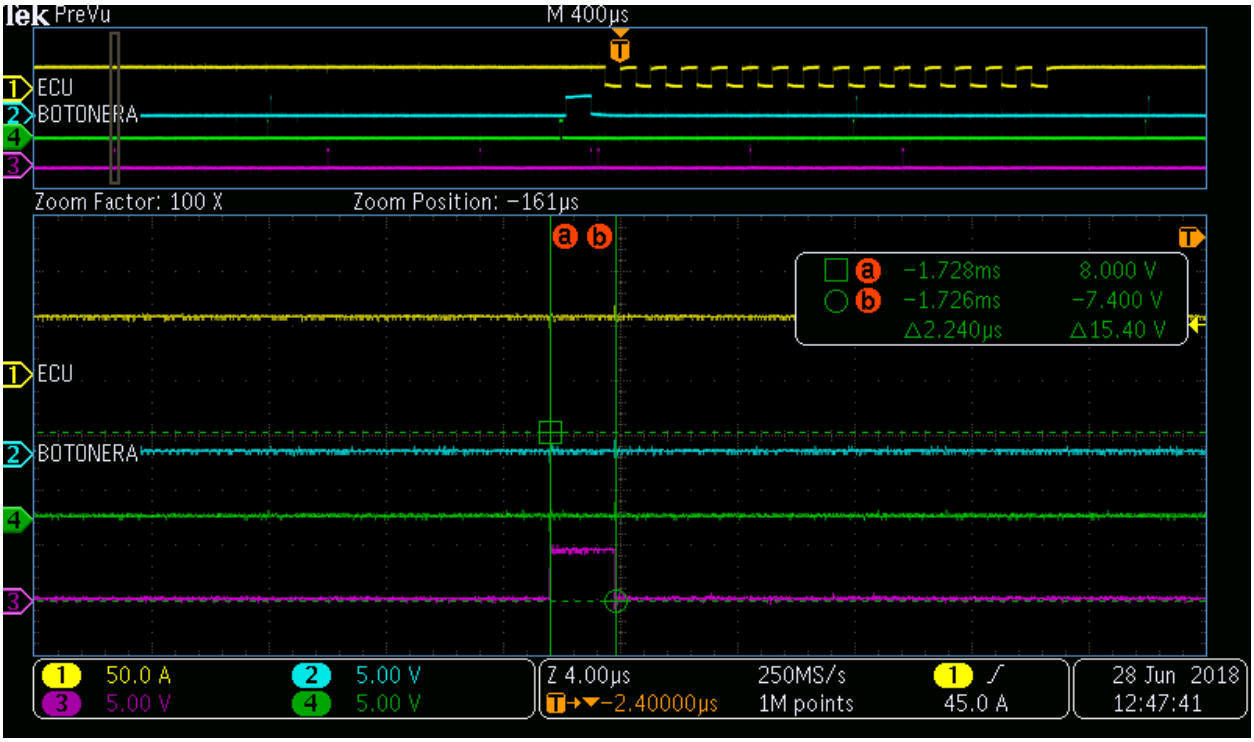


Figura 15: Tiempo de la primera interrupción

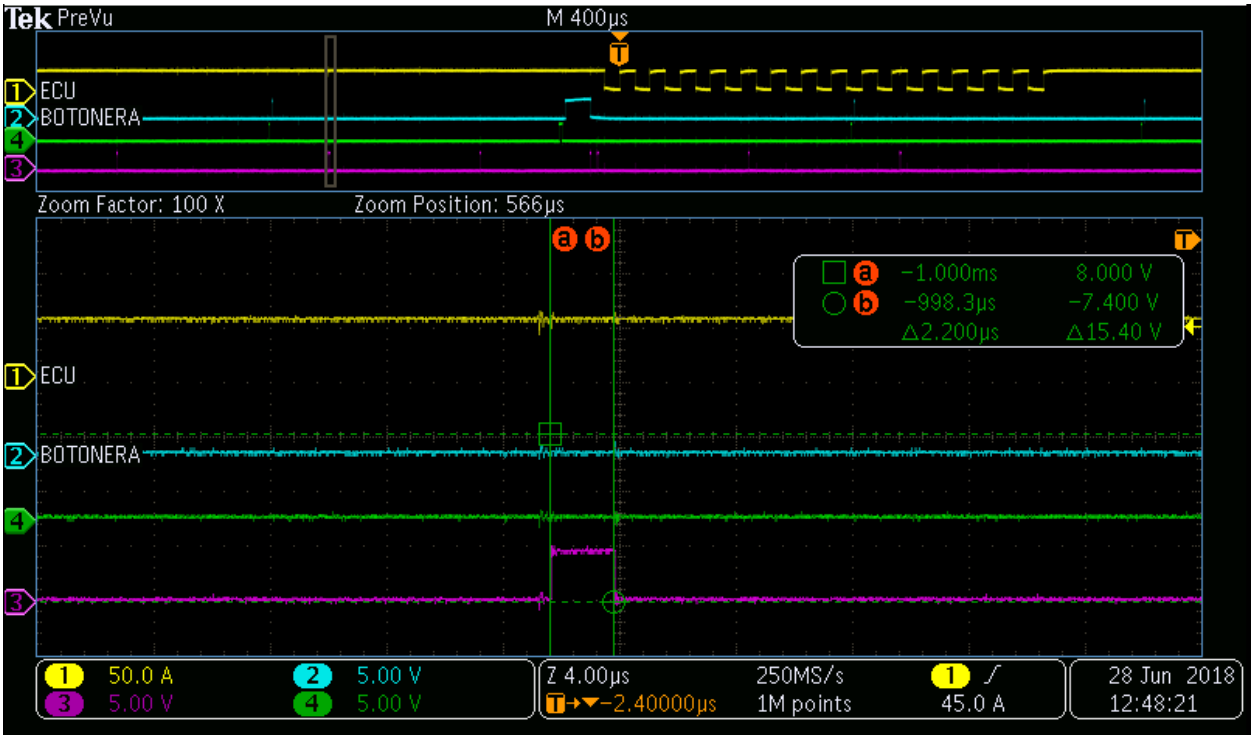


Figura 16: Tiempo de la segunda interrupción

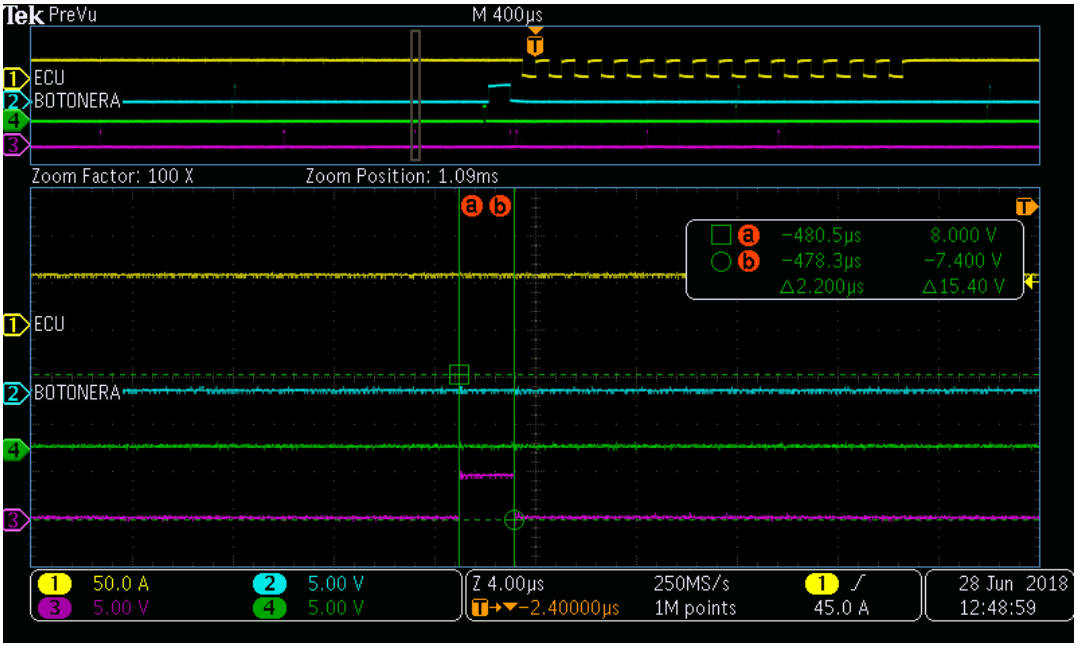


Figura 17: Tiempo de la tercera interrupción

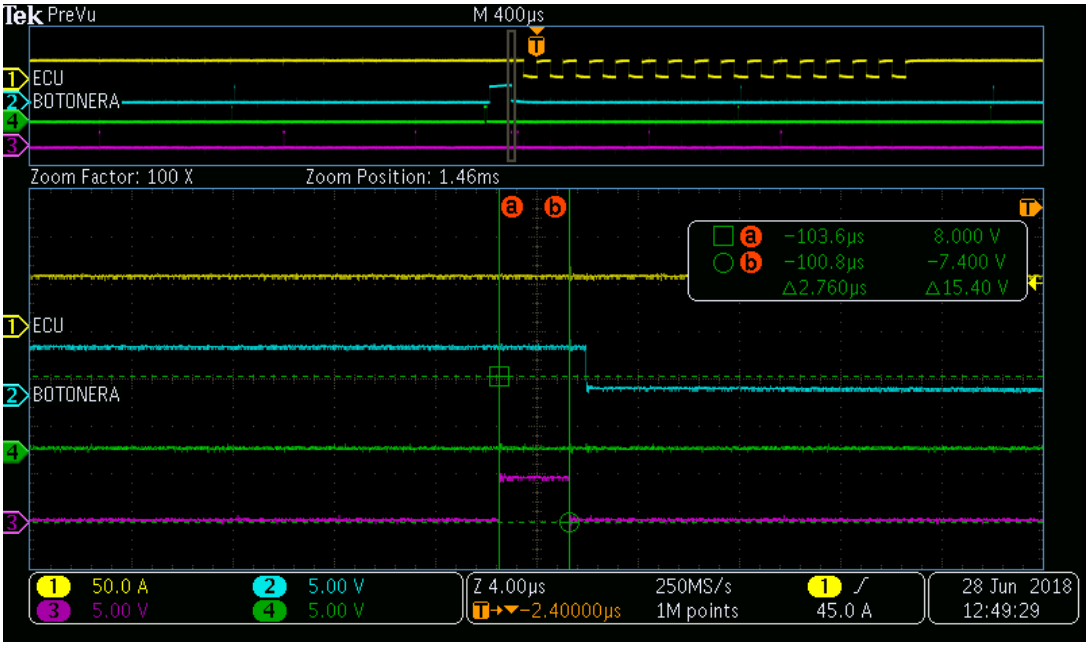


Figura 18: Tiempo de la cuarta interrupción

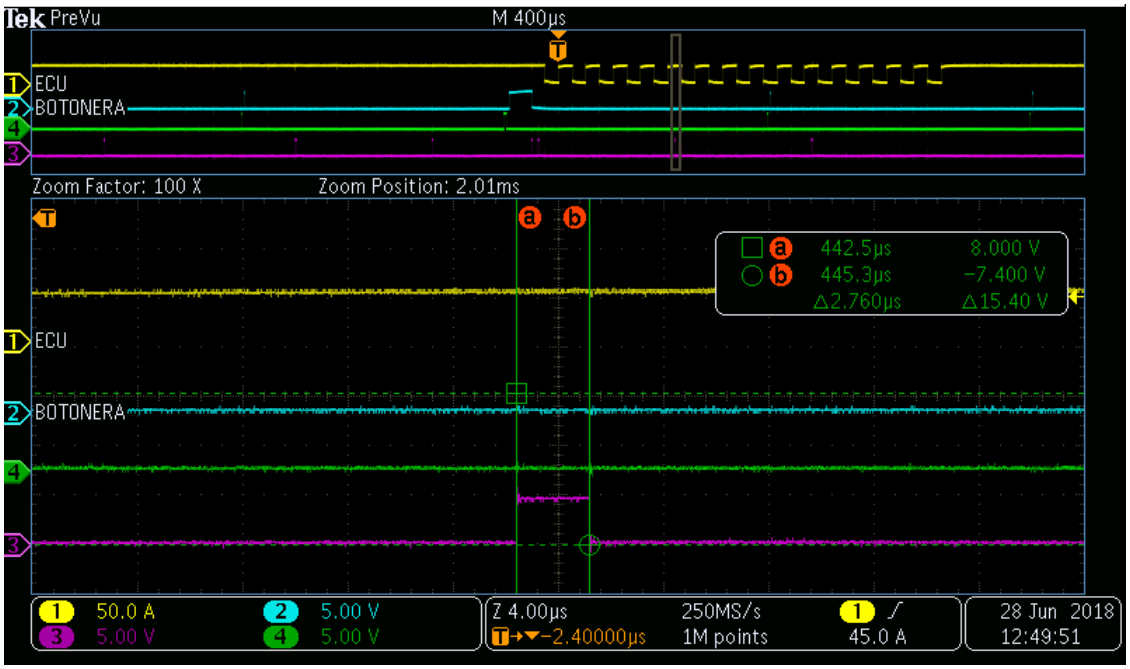


Figura 19: Tiempo de la quinta interrupción

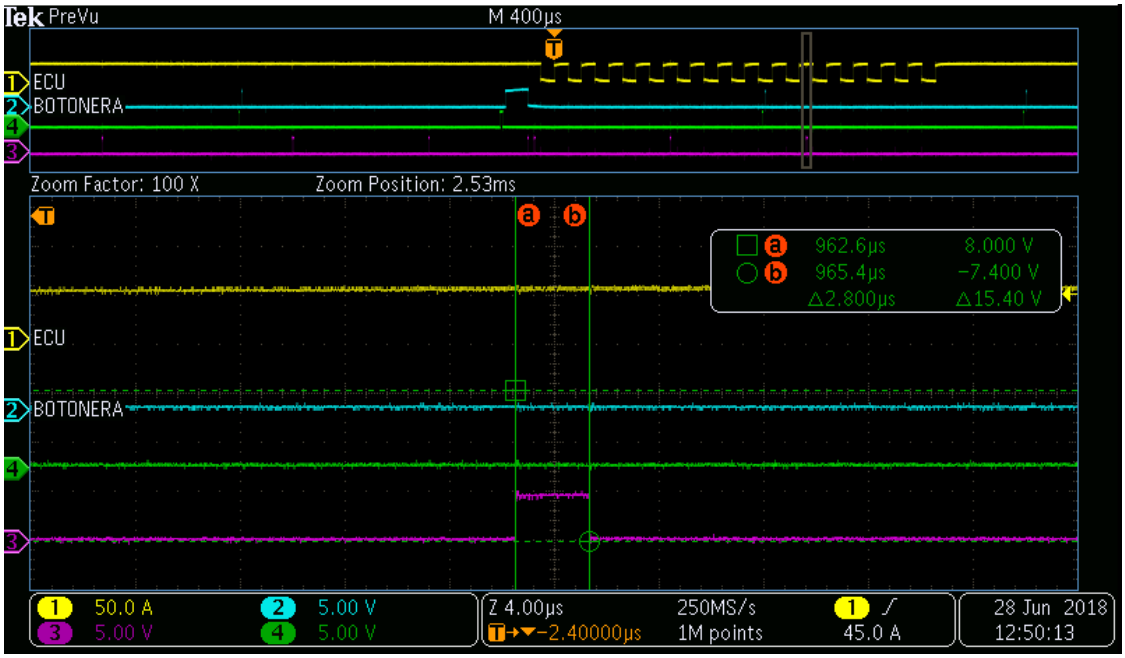


Figura 20: Tiempo de la sexta interrupción

En la Figuras 21 y 22 se muestran los tiempos del poleo de recepción dando un total acumulado de 10.2 microsegundos.

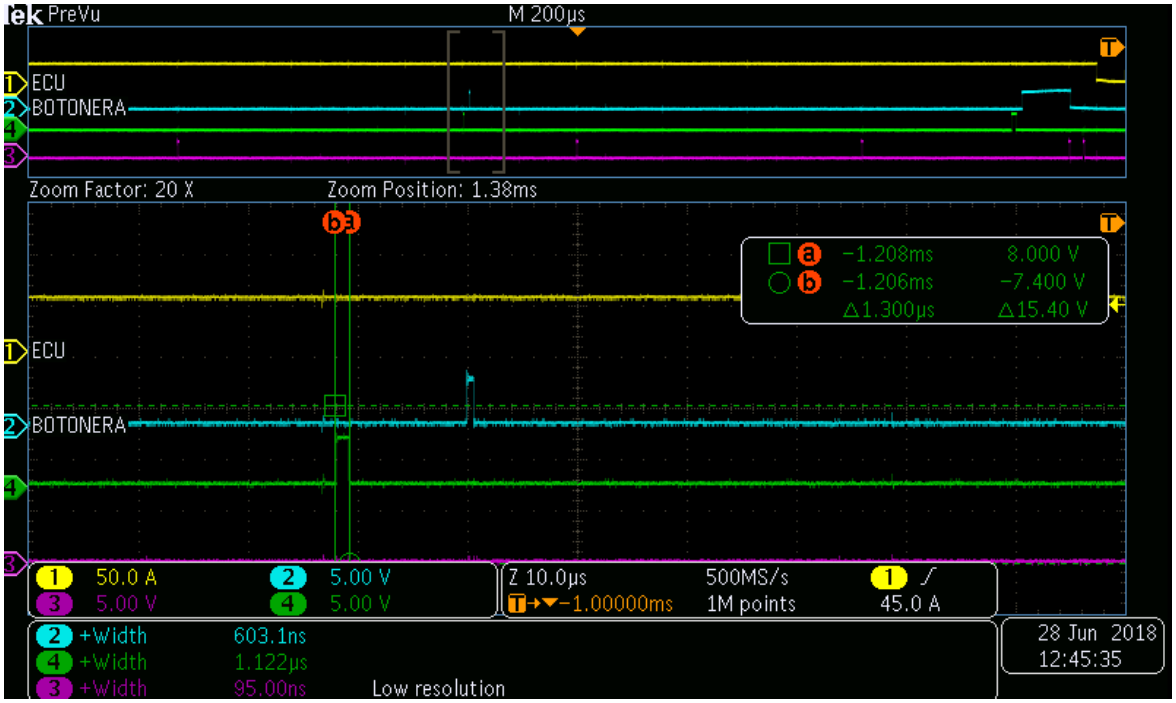


Figura 21: Tiempo poleo de recepción

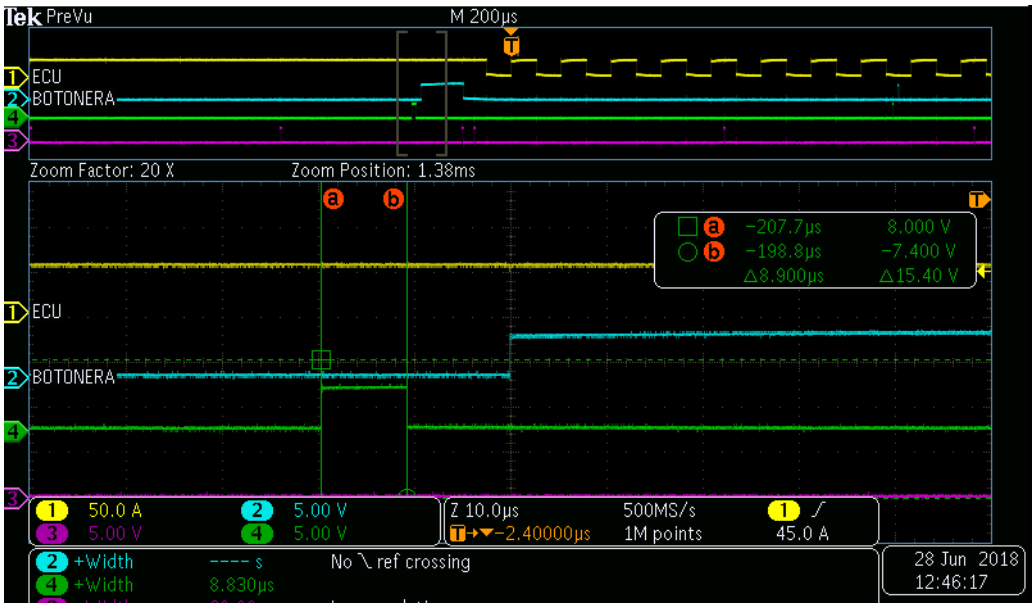


Figura 22: Tiempo poleo de recepción.

En la Figuras 23 se muestra el tiempo de CPU utilizado para la transmisión de datos hacia el ECU dando un total de 88.8 microsegundos

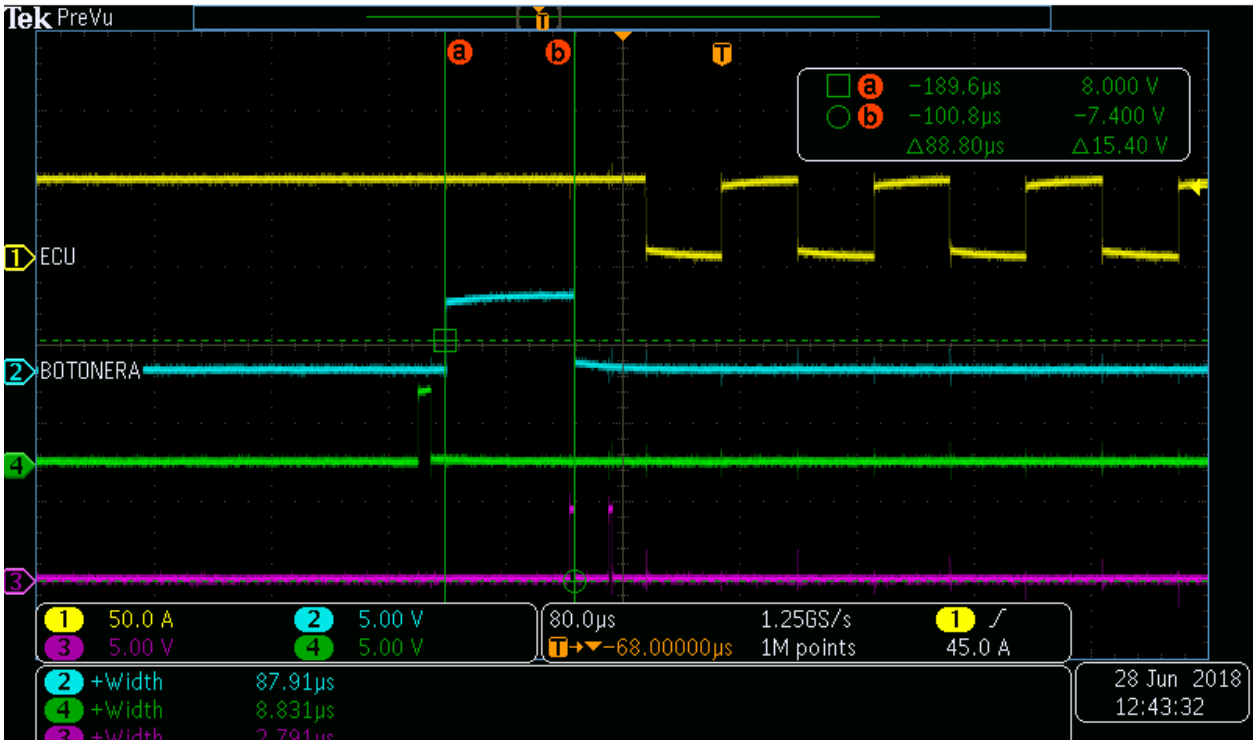


Figura 23: Tiempo de CPU para transmisión.

Tiempo total de uso de CPU para el módulo de botonera: 111.1628 microsegundos

En la implementación de la botonera con el protocolo de LIN estos fueron el uso de recursos:

- ROM: 6637 bytes
- RAM: 2022 bytes

5.2.3 SENT (Botonera)

5.2.3.1 Tiempo de Transmisión

Esta prueba consiste en medir el tiempo total necesario para transmitir un frame desde que el nodo botonera inicia la petición de transmisión, se calcula el CRC (cyclic redundancy check, por sus siglas en inglés) y, por último, completar el envío del mensaje. El resultado es variable, debido a que el valor del mensaje es determinado por los anchos de pulso, este puede variar de 384us a 654us para una trama de SENT que contiene 4 nibbles de datos, sin pulso de pausa y a un

pulso de reloj de 3us. En la figura 24 se muestra un ejemplo en donde el tiempo de transmisión es de 510 microsegundos.

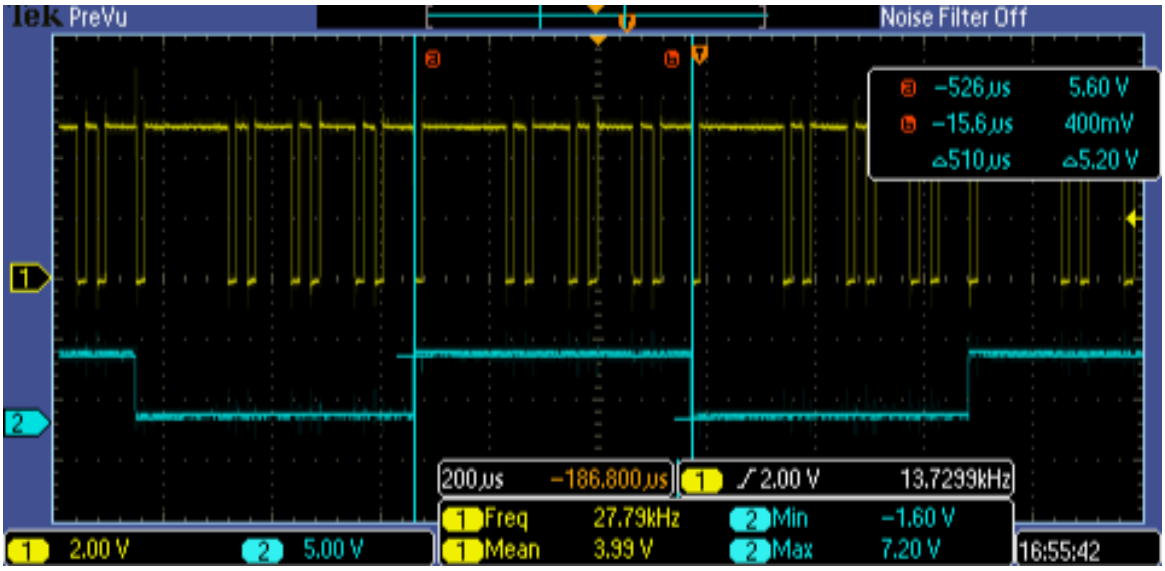


Figura 24: Tiempo de CPU para transmisión.

5.2.3.2 Tiempo de uso de recursos

En esta prueba se mide el tiempo de uso de recursos del procesador para transmitir los datos de la botonera. En esta métrica se incluyen el tiempo de procesamiento para armar un mensaje y la interrupción periódica que es necesaria para poderlo transmitir. En la Figura 25 se muestran un tiempo de 72.8 microsegundos (canal azul) necesarios para el procesamiento de los datos a ser transmitidos.

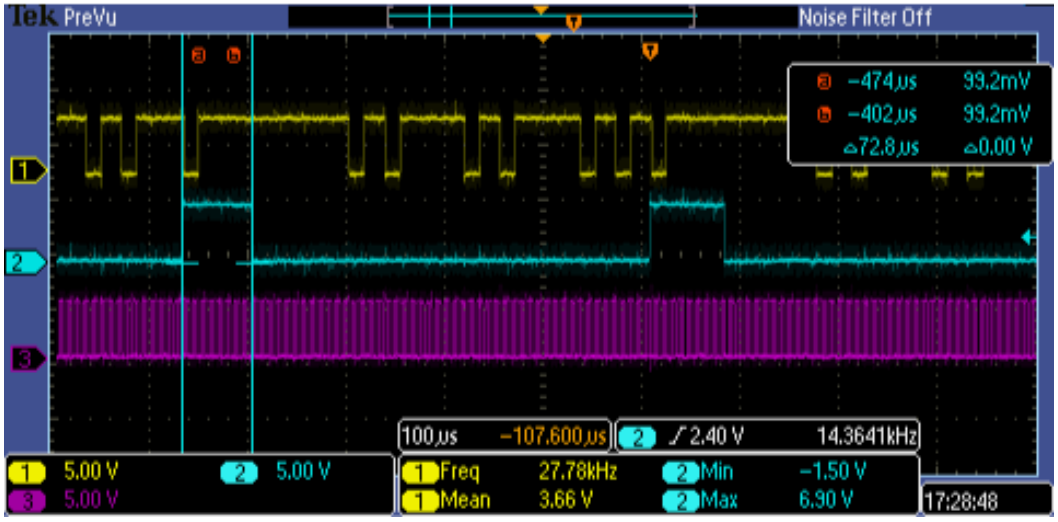


Figura 25: Tiempo de ejecución CPU para transmisión

En la Figura 26 se muestra el tiempo de ejecución de la interrupción periódica, que es ejecutada cada 3 microsegundos, la cual toma un tiempo de 800 nanosegundos por ejecución. Sumando un total de 136 microsegundos de ejecución en el contexto de interrupción para la transmisión de un mensaje.

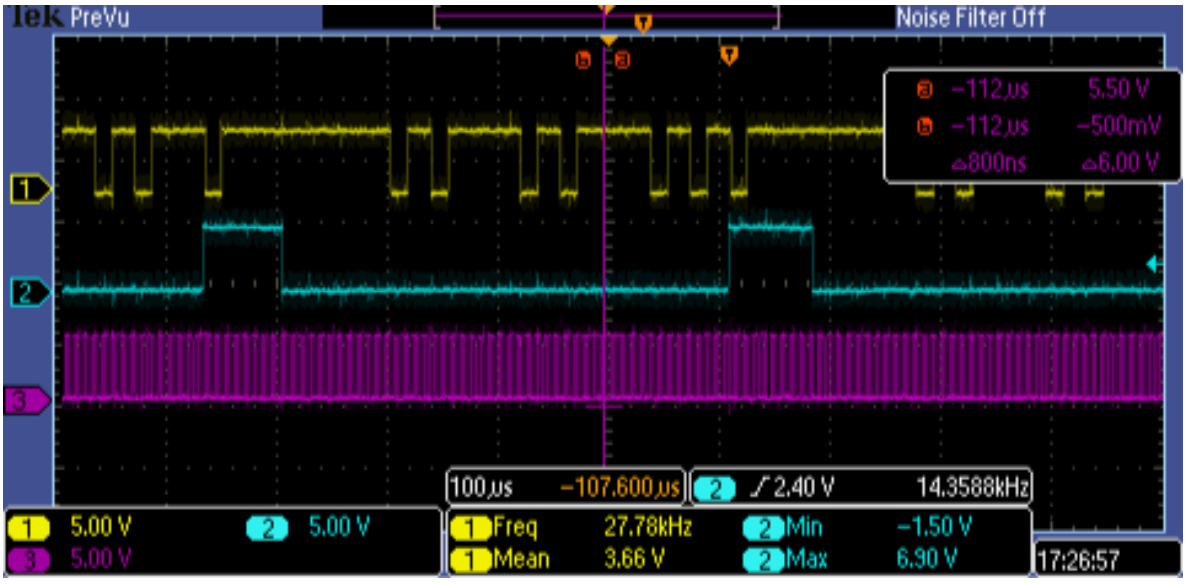


Figura 26: Tiempo ejecución de la interrupción para transmisión

En total el tiempo de ejecución para la el mensaje específico a ser mandado en SENT es de 208.8 microsegundos, pudiendo variar este de 175.2 microsegundos a 247.2 microsegundos.

En la implementación de la botonera con el protocolo SENT estos fueron el uso de recursos:

- ROM: 4801 bytes
- RAM: 1918 bytes

De los cuales la parte implementada en el XGate son los siguientes:

- ROM: 820 bytes
- RAM: 373 bytes

5.2.4 SENT (ECU)

5.2.4.1 Tiempo de Recepción

Esta prueba consiste en medir el tiempo total necesario para recibir un frame desde que el nodo ECU detecta el pulso de sincronía, se termina de recibir todos los datos, se decodifican los datos y el CRC es calculado y validado. Como se mencionó en la sección 4.1.1, el resultado es variable pues varia respecto a el ancho de pulso. En la figura 27 se muestra un ejemplo en donde el tiempo de recepción es de 335 microsegundos.

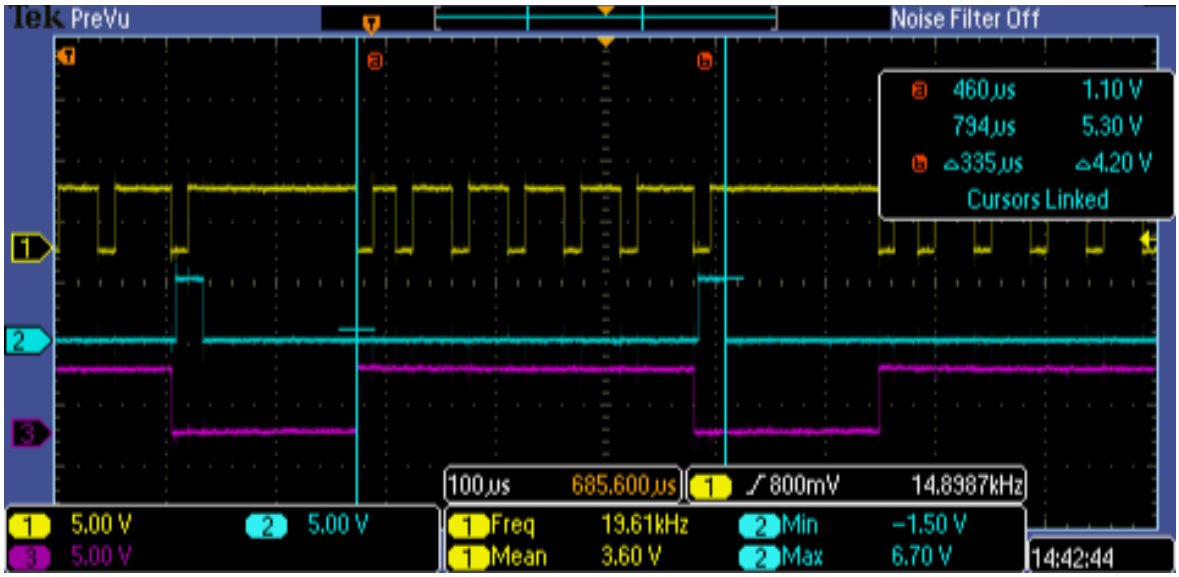


Figura 27: Tiempo para recepción

En la Figura 28 se muestran el tiempo de latencia desde que el mensaje termina de ser mandado, hasta que el mensaje termina de ser decodificado y validado, haciendo un tiempo de 30.5 microsegundos.

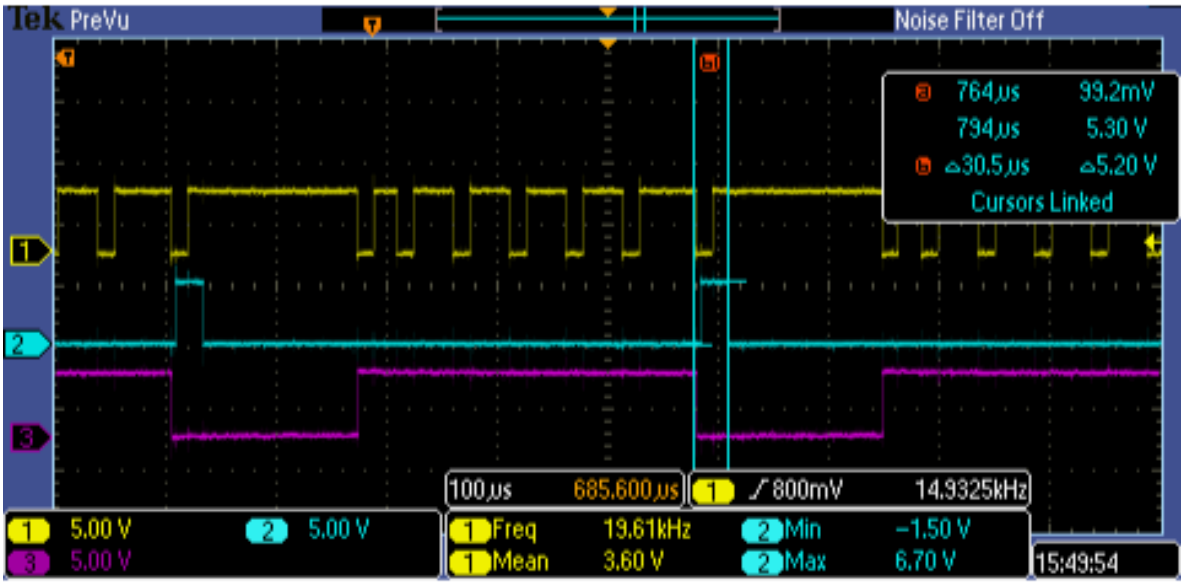


Figura 28: Tiempo de latencia en la recepción

5.2.4.2 Tiempo de uso de recursos

En esta prueba se mide el tiempo de uso de recursos del procesador para recibir los datos de la botonera. En esta métrica se incluyen el tiempo de procesamiento decodificar y validar mensaje y la interrupción de flanco de bajada que es necesaria para poderlo recibir. En la Figura 29 se muestran un tiempo de 22.7 microsegundos (canal azul) necesarios para el procesamiento de los datos a ser recibidos. También se muestran el tiempo de ejecución de la interrupción de flanco de bajada (canal morado), que es ejecutada cada que un nuevo pulso aparece, la cual toma un tiempo de entre 0.8 microsegundos a 1.391 microsegundos. Sumando un total de 6.677 microsegundos de ejecución en el contexto de interrupción para la transmisión de un mensaje.

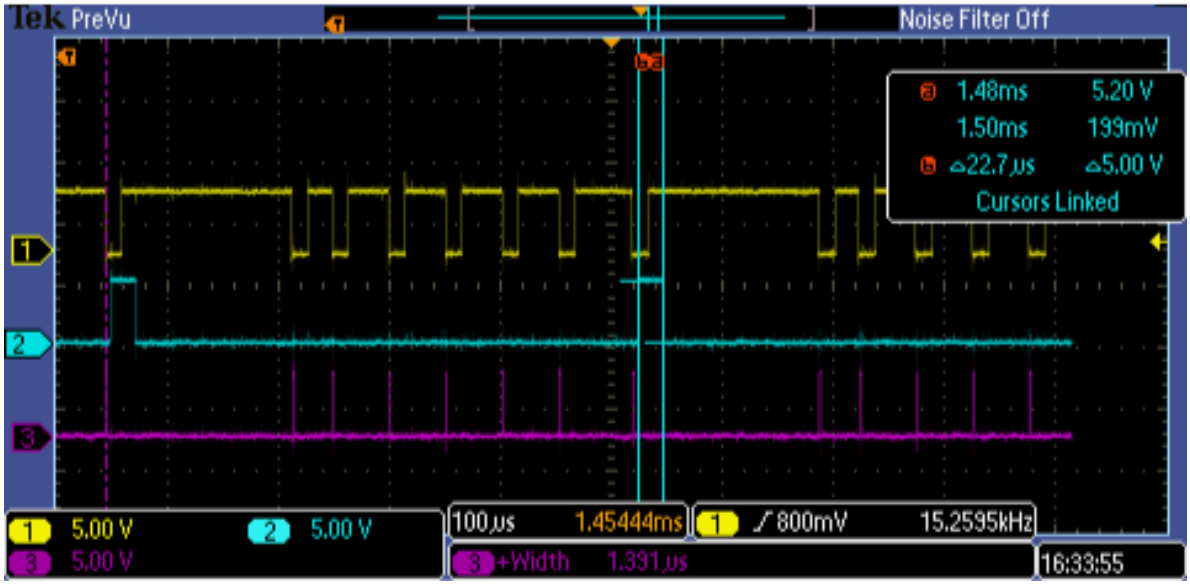


Figura 29: Tiempo de ejecución CPU para recepción

En total el tiempo de ejecución, para el mensaje específico de 4 nibbles de datos, a ser mandado en SENT es de 29.377 microsegundos.

En la implementación del ECU con el protocolo SENT estos fueron el uso de recursos:

- ROM: 4801 bytes
- RAM: 1918 bytes

De los cuales la parte implementada en el XGate son los siguientes:

- ROM: 820 bytes
- RAM: 373 bytes

5.2.5 Discusión resultados LIN y SENT

La primera y más evidente diferencia entre los resultados de LIN y SENT es que el tiempo tanto de transmisión de SENT es variable y depende de qué datos fueron transmitidos. En el peor de los casos un mensaje de SENT toma 654 microsegundos para ser enviado y un mensaje de LIN tarda 4.502 milisegundos. Esto quiere decir que el protocolo SENT es entre 6.88 a 11.72 veces más rápido mensajes que LIN. Otra clara diferencia fueron los tiempos de uso de CPU, en el protocolo LIN, el tiempo de ejecución fue de 30.452 microsegundos para el nodo de ECU y para el nodo Botonera fue de 111 microsegundos. Por su parte, el tiempo de procesamiento de SENT

para el nodo de ECU es de 29.377 microsegundos y para el nodo botonera, varia de 175.2 microsegundos a 247.2 microsegundos. Cabe señalar que, de estos tiempos, en el nodo ECU solo 22.7 microsegundos fuero usados en el CPU principal y el resto, 6.677 microsegundos en el CPU secundario también llamado Xgate. En el nodo Botonera, solamente 72.8 microsegundos fueron utilizados en el CPU principal y el resto, el cual es variable entre 102.4 microsegundos a 174.4 microsegundos fue ejecutado en el Xgate. En resumen, para el nodo de ECU con SENT es ligeramente superior siendo 3.7% más rápido que su contraparte en LIN, pero si no contamos el tiempo de ejecución del Xgate, pues sería similar a como tener un periférico dedicado para SENT, como lo es el periférico serial para LIN, resulta que SENT es 34.1% más rápido en el nodo ECU que LIN. En el caso de la Botonera, LIN es entre 97.3% a 178.4% más rápido que su similar en SENT. Pero haciendo el mismo ejercicio que en el nodo anterior y no consideramos el tiempo en el Xgate, resulta que SENT es 21.98% requiere menos recursos de CPU que LIN.

Los datos sobre el uso de recursos de la implementación de nuestras aplicaciones tanto en LIN como en SENT, nos indica que, en el caso de la botonera, la diferencia en ROM es de 1836 bytes y de RAM es de 104 bytes, y en el caso del ECU, la diferencia en ROM es de 1670 bytes y de RAM es de 114 bytes. En ambos casos el uso de recursos es menor en la implementación del protocolo SENT y también se debe tomar en cuenta que la implementación en el core Xgate podría ser lo equivalente a utilizar un periférico para la decodificación de SENT, como es en el caso de LIN, por la cual se deben descontar los recursos utilizados por el Xgate, teniendo una diferencia total de ROM de 2490 y RAM de 487 bytes en el caso del ECU y una diferencia total de ROM 2656 bytes y RAM de 477 bytes.

5.2.6 Aplicación LIN

En la aplicación de LIN se realizó la prueba mandando el mensaje de la botonera (0x1abcdeff) cada 2 milisegundos. En la figura 30 se ven los datos enviados por la botonera hasta alcanzar el valor de 65555 y en la figura 31 se ven los datos del mensaje que envía el ECU (0x111).

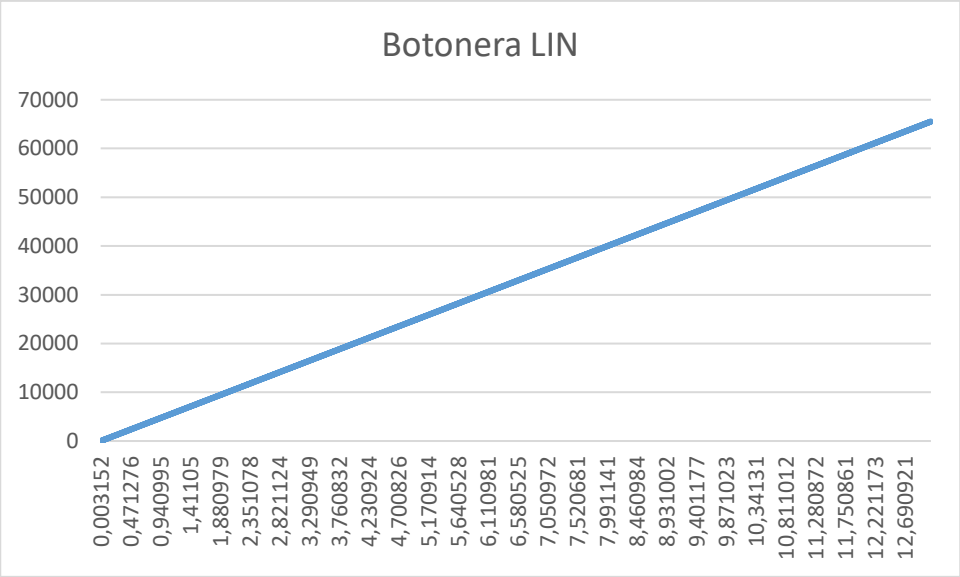


Figura 30: Datos enviados por la botonera LIN

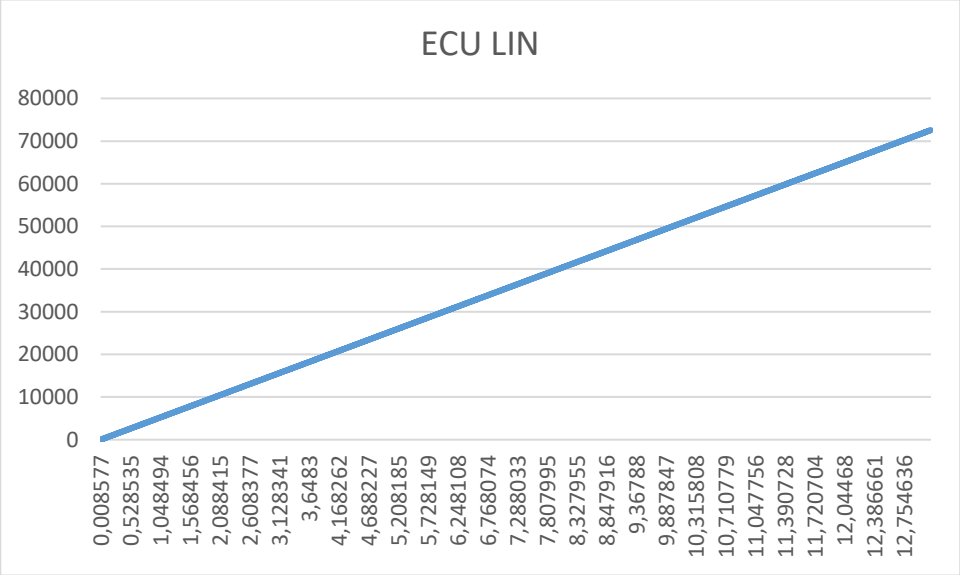


Figura 31: Datos enviados por el ECU LIN

Durante los 13 segundos se mandaron por medio de la botonera 6554 mensajes, pero solo se recibieron 3629 mensajes, esto es debido a que el mensaje de LIN es enviado cada 4 milisegundos, dado su mensaje tiene una longitud de 3.8 milisegundos. En la figura 32, se puede observar que se envían 2 mensajes de la botonera, antes de recibir el mensaje del ECU, con esto se tiene una pérdida de la mitad de los datos.

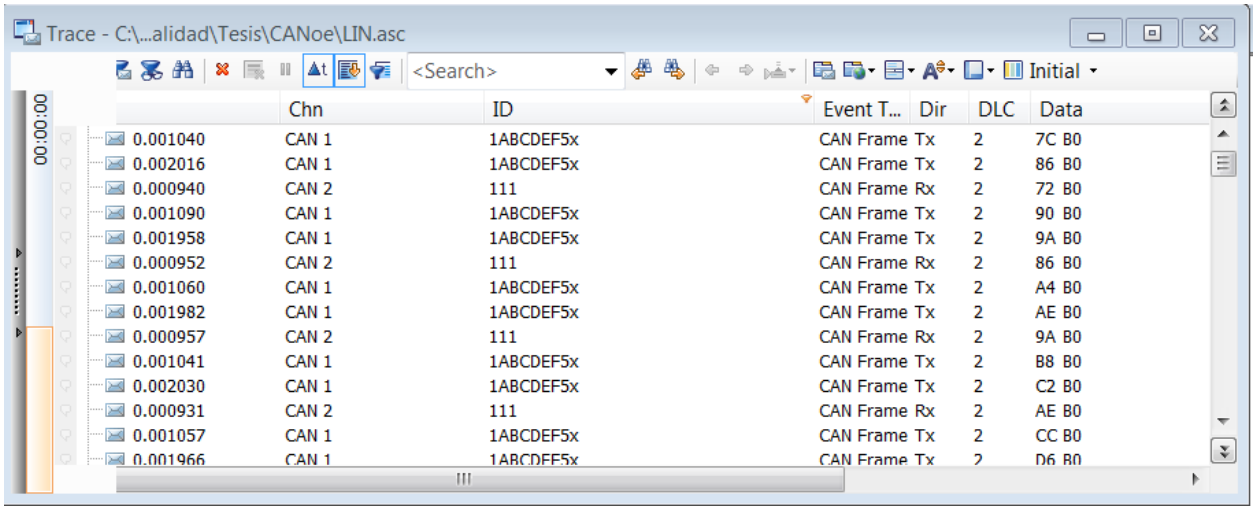


Figura 32: Mensajes de CAN en aplicación LIN

Para validar que es posible que el ECU reciba todos los datos de la botonera se realizó la misma prueba, pero ahora mandando el mensaje de la botonera cada 4 milisegundos. En la figura 33 se ven los datos enviados por la botonera hasta alcanzar el valor de 65555 y en la figura 34 se ven los datos del mensaje que envía el ECU (0x111).

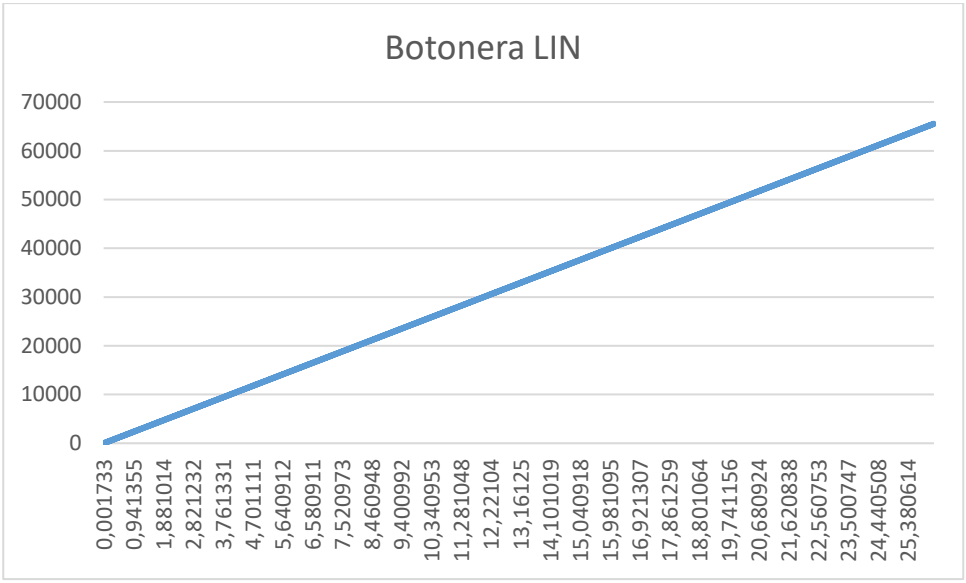


Figura 33: Datos enviados por la botonera LIN - Prueba2

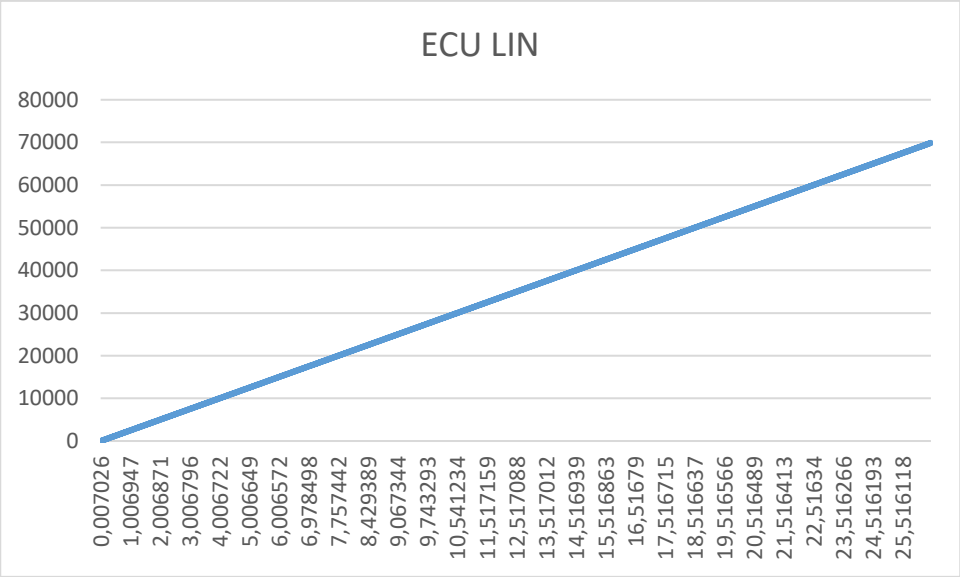


Figura 34: Datos enviados por el ECU LIN - Prueba2

Ahora todos los mensajes fueron recibidos, pero tomó un tiempo total de 26 segundos. En la figura 35, ahora se observa que, por cada mensaje enviado de la botonera existe un mensaje del ECU con el valor esperado del estado de los motores.

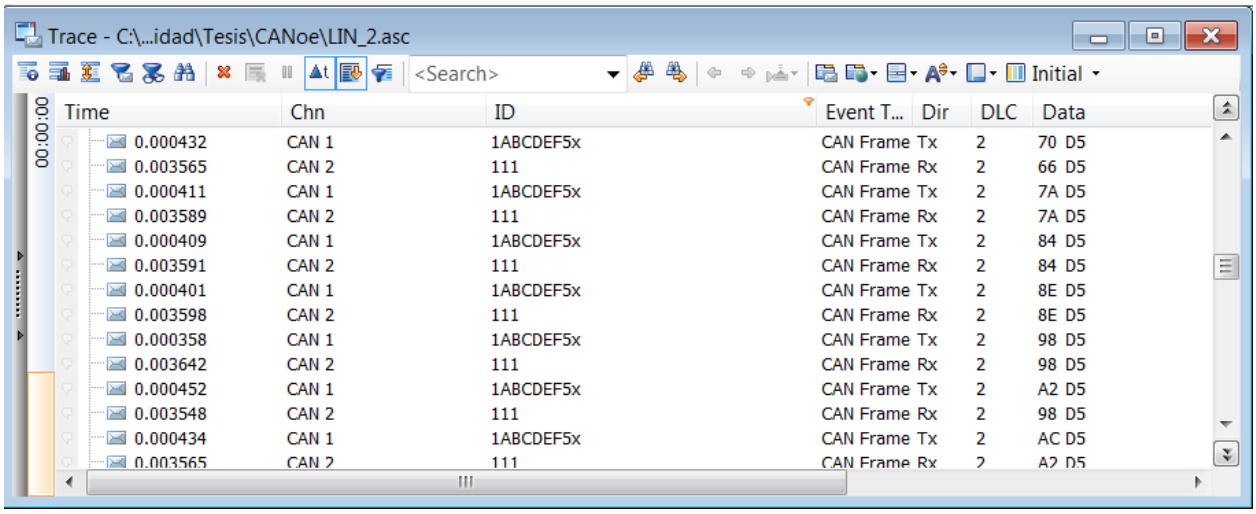


Figura 35: Mensajes de CAN en aplicación LIN - Prueba2

5.2.7 Aplicación SENT

En la aplicación de SENT se realizó la prueba mandando el mensaje de la botonera (0x1abcdeff) también cada 2 milisegundos. En la figura 36 se ven los datos enviados por la botonera hasta alcanzar el valor de 65555 y en la figura 37 se ven los datos del mensaje que envía el ECU (0x111).

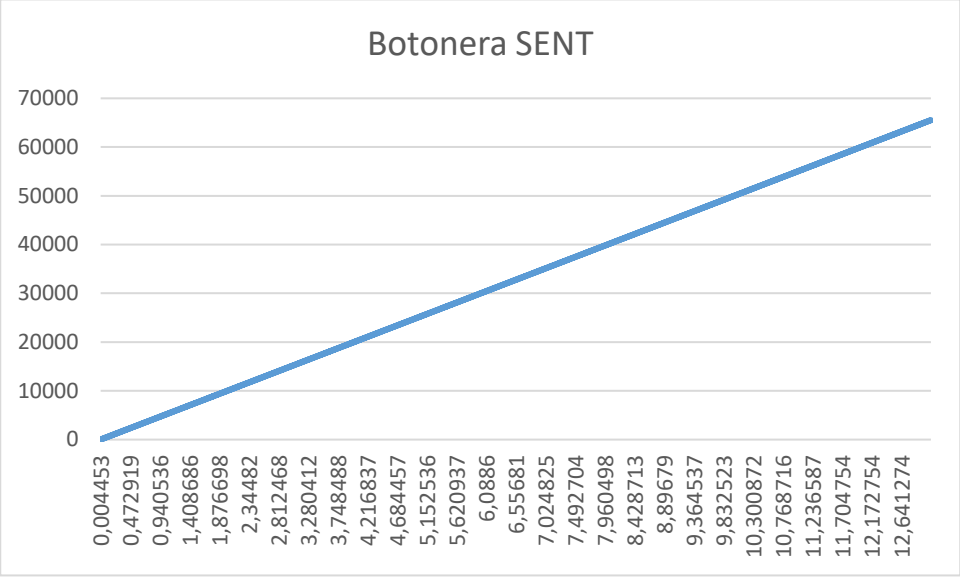


Figura 36: Datos enviados por la botonera SENT

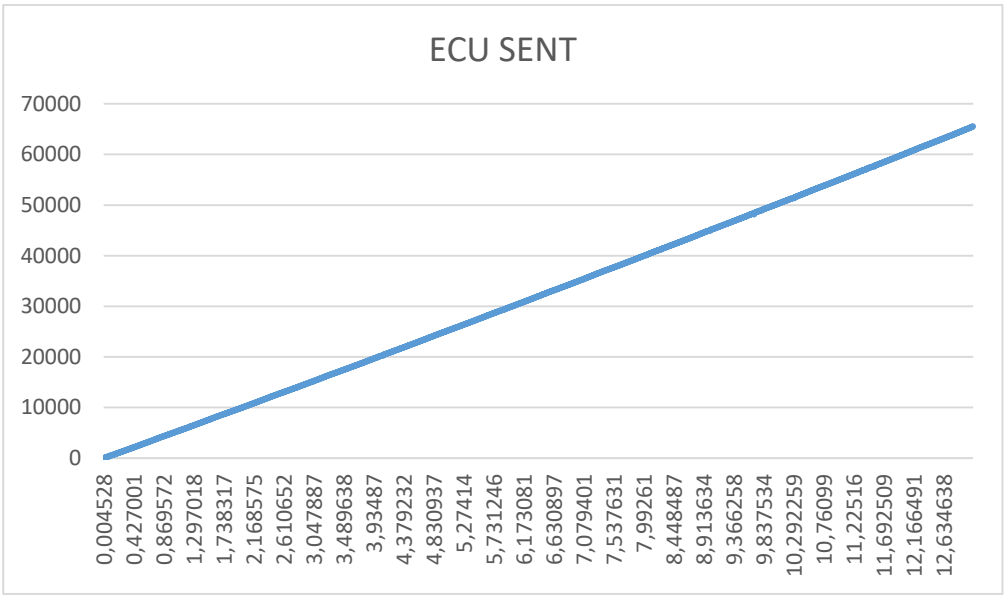


Figura 37: Datos enviados por el ECU SENT

Durante los 13 segundos se mandaron por medio de la botonera 6554 mensajes, pero se recibieron 26553 mensajes, esto es debido a que la longitud del mensaje varía entre 384 y 654 microsegundos dependiendo del valor de los datos a enviar.

En la figura 38 se puede observar por cada mensaje de la botonera el ECU logra enviar 3, 4 ó 5 mensajes con estatus de los motores.

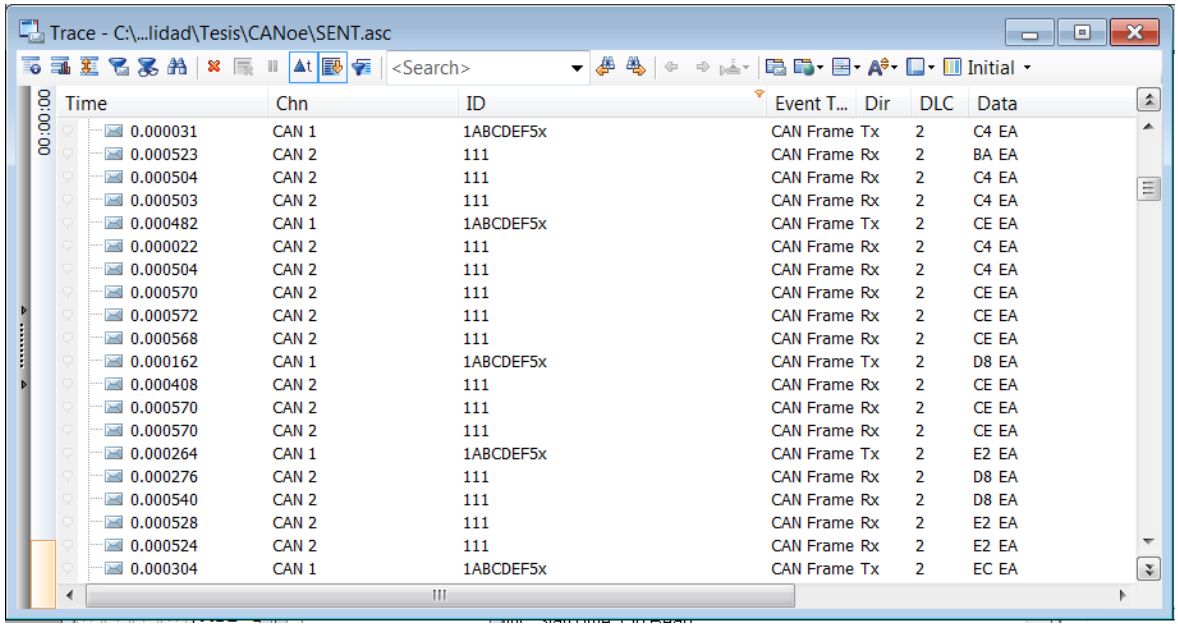


Figura 38: Mensajes de CAN en aplicación SENT

La prueba podría ser modificada para que la botonera enviara el mensaje de CAN cada 654 microsegundos y aun así no perder ningún dato, pero debido a que se implementó la lógica de la recepción de CAN cada 2 milisegundos no es posible comprobarlo prácticamente

6. Conclusiones

Tomando en cuenta las investigaciones y resultados obtenidos en el análisis teórico se puede concluir que, si bien LIN es un protocolo con una mayor versatilidad y robustez, SENT tiene la capacidad de suplir a LIN en determinadas aplicaciones, proporcionando ventajas que no son posibles de obtener con el uso de LIN.

Las aplicaciones donde SENT puede sustituir a LIN está delimitada por las siguientes características: solo 2 nodos son requeridos, la estrategia de energía tiene que ser implementada de manera separada, es decir el protocolo no soporta esta característica (Sleep Mode) por su cuenta, una transmisión a través de mensajes de dimensiones de datos no mayores a 24 bits de datos (incrementa la necesidad de segmentación de mensajes), La comunicación es exclusivamente unidireccional, la distancia máxima entre los dos nodos está limitada a 5 metros. La tolerancia a un desplazamiento en el mensaje o jitter es solo de 25%. Debido a su mayor costo de SENT y al no existir un proveedor que ofrezca una solución a la venta del protocolo SENT, es aconsejable solo ser usado este protocolo si se piensa hacerlo en más de un proyecto, de lo anterior también se puede considerar que es necesario un grupo de desarrollo más maduro para utilizar SENT, pues el trabajo requerido que se tiene que realizar es más complejo y extenso, esto debido a que LIN es posible comprar una solución desarrollada por un tercero.

Por otro lado, las ventajas que ganaría una aplicación al usar SENT serían: una mayor velocidad de transmisión que favorece la gestión de información para sistemas de respuesta en tiempo real, un manejo de voltajes mucho más bajos en el orden de 5V, lo que puede disminuir el uso del Hardware requerido al no necesitarse una fuente extra para la comunicación a 12V, además la potencia eléctrica se ve reducida y por lo tanto el consumo energético es menor. La simplificación del hardware de comunicaciones requerido al eliminar la necesidad del uso de un transceptor para efectuar la intercomunicación, que reduce el costo de la implementación en hardware (crucial en dispositivos a ser fabricados en masa) lo cual puede brindar un gran ahorro en cuanto a costo final del producto.

En las pruebas practicas se confirma que SENT es un protocolo con una transferencia de datos mucho mayor debido a que en un mismo tiempo, SENT es capaz de enviar más mensajes que los que pudo enviar LIN. Además, el tiempo de ejecución que requiere SENT en el nodo del

ECU es prácticamente igual al obtenido en LIN, en el caso del nodo Botonera, El tiempo de ejecución en SENT es entre un 97.3% a 178.4% más tiempo de ejecución. Cabe resaltar que, si se consigue un microcontrolador que posea un periférico específico para SENT, como lo tiene LIN con un periférico de UART, la carga en ambos nodos disminuye y en ambos casos logró ser menor que en sus contrapartes en LIN, reduciendo un 34.1% y 21.98% respectivamente en ECU y botonera. Por lo tanto, SENT tiene un tiempo de ejecución por mensaje que LIN.

En el caso específico de nuestra aplicación, se observa que todas las delimitantes que posee SENT son cubiertas, esto quiere decir que no existe ningún impedimento de que sea usado en nuestra aplicación específica; el único punto que podría ser sensible es si la aplicación necesitara un modo ahorrador de energía (Power Saving), pues en este caso o bien tendría que generarse una solución por hardware, energizando y des energizando la botonera cada que sea presionada, o utilizando el nibble de estado y control para dicha función, entre otras, este tema tiene que ser analizado con mayor detalle en estudios posteriores.

El hecho que LIN sea un protocolo más versátil y más antiguo que SENT evidencia que existen aplicaciones en las cuales es muy viable el poder sustituir el protocolo SENT por el protocolo LIN, pues en ciertos casos específicos esa versatilidad resolvió el problema en primera instancia, pero con la llegada de SENT estos sistemas específicos donde SENT cumple las características para poder satisfacer las necesidades sistema, sustituirán a LIN y este trabajo es una prueba en la sección practica de que es el remplazo es posible, y la sección teórica deja una guía de características que se deben analizar de cada sistema para el poder analizar si el sistema en cuestión es candidato a reemplazar LIN por SENT.

.

7. Bibliografía

[1] S. C. Talbot y S. Ren, «Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger Vehicles».

[2] P. Kamenicky, G. Vandensande y P. Quarmea, «SENT “New sensor interface" Allows data-transmission at low cost».

[3] N. Ullah, «Implementing and Analyzing Single Edge Nibble Transmission (SENT) Protocol for Automotive Applications,» 2014.

[4] S. Nakka y P. Badarinath, «Emulation of Automotive Communication protocol Single Edge Nibble Transmission(SENT) using Aurix Family of Microcontrolllers,» *International Journal of Computer Trends and Technology*, vol. 4, nº 6, 2013.

[5] B. A. Rahim y S.Krishnaveni, «Comparison of CAN, TTP and Flexray Communication Protocols,» *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, nº 4, 2014.

[6] V. I. GmbH., «Vector.com,» Vector Informatik GmbH, 03 03 2014. [En línea]. Available: https://vector.com/portal/medien/cmc/events/Webinars/2014/Vector_Webinar_IntroductionToLIN_20140930_EN.pdf. [Último acceso: 05 06 2018].

[7] M. Ruff, «Evolution of local interconnect network (LIN) solutions,» de *2003 IEEE 58th Vehicular Technology Conference*, 2003.

[8] VECTOR, «VECTOR E-LEARNING,» VECTOR, 12 09 2016. [En línea]. Available: https://elearning.vector.com/index.php?wbt_ls_kapitel_id=1599673&root=835866&seite=vl_lin_introduction_ko.

[9] «www.ni.com › White Papers,» National Instruments, 12 05 2010. [En línea]. Available: http://www.ni.com/white-paper/9733/es/. [Último acceso: 09 07 2019].

[1 S. International, «J2716 SENT - Single Edge Nibble Transmission for Automotive Application,» *Surface Vehicle Information Report ,* 2016.

[1 H. Rayo, «Design and development of a driver based on J2716 standard for data transmission on high electrical noise environments,» 2016.

[1 F. Rodrígues, «Reporte de formación complementaria en el área de sistemas embebidos y telecomunicaciones,» 2017.

[1 H.-C. v. d. Wense, «LIN Specification Package,» 2002.