VOLUMEN 37
XXX Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Desarrollo de software embebido aplicado al área automotriz

Embedded software development applied to the automotive industry

Moisés Rodríguez Zavala[1], Rosa Ivonne Peñaran Prieto[1], Ángel Francisco Cerda Rodríguez[1], Ricardo Pérez López[1], Yuriy S. Shmaliy[1], Oscar G. Ibarra-Manzano[1]

[1] División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato.
m.rodriguezzavala@ugto.mx, ri.penaranprieto@ugto.mx, af.cerdarodriguez@ugto.mx, r.perezl@ugto.mx, shmaliy@ugto.mx, ibarrao@ugto.mx

Resumen

El termistor NTC es un semiconductor ampliamente utilizado en la industria en aplicaciones de medición y control de temperatura por su bajo costo y adecuado comportamiento en su relación entre la resistencia y la temperatura. Además de requerir una sencilla configuración de componentes electrónicos para su funcionamiento, cumple satisfactoriamente con el rango de operación de temperatura requerida por la industria automotriz. En este artículo, se describe la construcción de un dispositivo que utiliza un microcontrolador de ultra baja potencia de la familia STM32L4 para resolver, utilizando programación embebida, el modelo beta del termistor y determinar la temperatura a la cual está expuesto el dispositivo, para después transmitir la lectura de la temperatura a través de una interfaz de comunicación utilizando el Bus CAN (*Controller Area Network*, Controlador de Red de Área), estándar en la industria automotriz. Finalmente, el dispositivo fue probado utilizando un segundo microcontrolador de la misma familia que, operando como controlador maestro, solicita de forma periódica las lecturas de temperatura al dispositivo esclavo, estableciendo una red de dos elementos de un Bus CAN.

Palabras clave: Microcontrolador STM32; Bus CAN; Termistor.

El software embebido en la industria automotriz

El software embebido es un programa especializado que es ejecutado en una plataforma específica para acceder, procesar y controlar datos generados por uno o varios sistemas electrónicos. Entre las plataformas más utilizadas a nivel industrial, incluyendo la automotriz, se encuentran los microcontroladores en un solo chip ARM Cortex (*Advanced RISC Machine* – Máquina avanzada RISC (*Reduced Instruction Set Computer* – Computadora con un Conjunto Reducido de Instrucciones)) (Texas Instruments Inc., 2025). Estos dispositivos incluyen características de alto rendimiento y una integración analógica avanzada que superan los límites de los microcontrolades tradicionales en aplicaciones industriales y automotrices.

Los sistemas embebidos son ampliamente utilizados en la industria automotriz, tomando prácticamente el control de total del automóvil. Desde los pequeños controles de limpiaparabrisas hasta el complejo control de un sistema de frenado antibloqueo, todo en un automóvil está bajo el control de un sistema integrado. Las unidades de control electrónico (ECU) utilizadas en un automóvil han evolucionado rápidamente en las últimas dos décadas. Cada año, los fabricantes incluyen sistemas integrados en sus automóviles para diferentes propósitos como el encendido, la seguridad o el confort. Entre todas estas funcionalidades, actualmente el sistema carga de los vehículos eléctricos requiere asegurarse que el proceso de carga del banco de baterías se realice a una temperatura segura (Thermtest Instruments, 2025). Aquí es donde un sensor de temperatura basado en un termistor encuentra una aplicación en la industria automotriz.

El termistor

Un termistor es un dispositivo electrónico que se caracteriza por presentar una gran variación de su resistividad a cambios pequeños de temperatura. Su funcionamiento se basa en la variación de la resistividad del semiconductor debido al aumento en el número de portadores de carga presentes en el dispositivo producto de un incremento de su temperatura. Los termistores se clasifican en dos tipos:

- NTC (*Negative Temperature Coefficient*). Para este tipo de termistor, un aumento de la temperatura aumentará la concentración de portadores y por lo tanto su resistividad será menor, de ahí proviene que el coeficiente sea negativo.
- PTC (*Positive Temperature Coefficient*). En este caso, el semiconductor es dopado de manera más intensa durante su fabricación, adquiriendo propiedades metálicas lo que se refleja en un coeficiente de temperatura positivo en un margen de temperatura limitado, razón por la cual su aplicación tiende a ser muy particular.

En este proyecto nos centraremos en el uso de un termistor NTC, ya que es ampliamente utilizado como un sensor de temperatura, comúnmente aplicado en la industria automotriz para medir la temperatura del refrigerante del motor, la temperatura del habitáculo, la temperatura del exterior o la temperatura del aceite del motor, o incluso en termostatos digitales en la industria del hogar.

Una configuración sencilla de aplicar un termistor como sensor de temperatura es la mostrada en la Figura 1. En este circuito, el termistor opera como una de las resistencias en una configuración de un divisor de voltaje con un resistor de referencia (R_{REF}) de un valor igual al valor resistivo del termistor a la temperatura central de operación a la cual se desea utilizar el dispositivo como sensor de temperatura. En nuestro caso, consideremos el valor nominal del termistor $R_T = 10k\Omega$ a una temperatura de 25°C.

Para estabilizar el voltaje de entrada que alimenta al divisor de voltaje, es utilizado un dispositivo electrónico conocido como voltaje de referencia (V_{REF}), el cual es polarizado por un resistor y estabilizado en la frecuencia por un par de capacitores encargados de eliminar componentes de mediana y alta frecuencia. De esta forma, el voltaje de salida estará expresado como:

$$V_o = \left(\frac{R_{REF}}{R_{REF}+R_T}\right) \cdot V_{REF} \tag{1}$$

donde V_{REF} es el voltaje de referencia que alimenta al divisor de voltaje, R_{REF} es el valor de la resistencia conectada de nodo V_o a tierra (GND) y R_T es el valor resistivo del termistor a la temperatura T(°C). Despejando R_T de (1) tenemos:

$$R_T = \left(\frac{V_{REF}}{V_o} - 1\right) \cdot R_{REF} \tag{2}$$

De esta forma, conociendo V_{REF} y R_{REF}, que permanecen idealmente constantes, y midiendo el valor de V_o, podemos determinar el valor resistivo del termistor a la temperatura T. Ahora, podemos utilizar la expresión que relaciona la resistencia de un termistor en función de la temperatura.

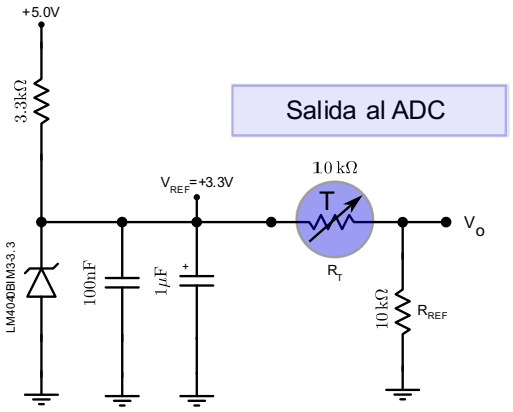

Figura 1. Configuración eléctrica de un termistor como sensor de temperatura.
Fuente: Elaboración propia.

Ecuación de Steinhart-Hart:

$$T(R_T)[^\circ C] = \frac{1}{C_0 + C_1 \cdot \ln(R_T) + C_2 \cdot \ln^3(R_T)} - 273.15 \quad (3)$$

donde los coeficientes C_0, C_1 y C_2 son especificados en la hoja de datos del termistor. Esta última ecuación, nos relaciona la resistencia calculada por (2) y la temperatura $T[^\circ C]$ que estamos buscando. En el caso de no disponer de los coeficientes C_0, C_1 y C_2, estos pueden ser determinados experimentalmente exponiendo el termistor a tres temperaturas conocidas (T_1, T_2 y T_3) y midiendo los valores resistivos que presenta el termistor ($R_{T,1}$, $R_{T,2}$ y $R_{T,3}$). Entonces, aplicando la ecuación (3) para los tres pares de resistencias y temperaturas medidas podemos encontrar los coeficientes C_0, C_1 y C_2, resolviendo el sistema de tres ecuaciones lineales:

$$\begin{bmatrix} 1 & ln(R_{T,1}) & ln^3(R_{T,1}) \\ 1 & ln(R_{T,2}) & ln^3(R_{T,2}) \\ 1 & ln(R_{T,3}) & ln^3(R_{T,3}) \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{T_1+273.15} \\ \frac{1}{T_2+273.15} \\ \frac{1}{T_3+273.15} \end{bmatrix} \quad (4)$$

2. Ecuación utilizando el parámetro $\boldsymbol{\beta}$:

La expresión que relaciona el parámetro β con dos puntos de temperatura y su valor resistivo es de la forma:

$$R_T(T) = R_0 e^{\beta\left(\frac{1}{T} - \frac{1}{T_0}\right)} \quad (5)$$

donde $R_T(T)$ es la resistencia presentada por el termistor a la temperatura T expresada en grados Kelvin, R_0 es el valor resistivo del termistor a una temperatura de referencia T_0 (normalmente 25°C o 298.15°K), β es la constante del termistor.

Ahora, considerando los parámetros del termistor a una temperatura de 25°C: $R_{T=25^\circ C} = 10k\Omega$ y $T = 298.15^\circ K$, podemos determinar la temperatura T a partir de la ecuación (5), quedando:

$$T(R_T)[^\circ C] = \frac{298.15\beta}{\beta + 298.15\, ln\left(\frac{R_T}{R_{25}}\right)} - 273.15 \quad (6)$$

esta última ecuación es la que utilizaremos para determinar la temperatura del termistor conociendo sus parámetros característicos: Resistencia nominal a 25°C y su coeficiente β.

El microcontrolador STM32L476

El microcontrolador utilizado para este proyecto pertenece a la familia de microcontroladores de ultra baja potencia STM32L476xx. Un microcontrolador basado en la arquitectura ARM Cordex-M4 de 32-bits, integrando un MCU (Unidad de microcontrolador) y un FPU (Unidad de punto flotante) capaz de ejecutar 100DMIPS (Millones de instrucciones por segundo en la referencia Dhrystone). Cuenta con hasta 1 MB de memoria flash y 128KB de SRAM. Entre sus periféricos, cuenta con hasta 114 terminales de entrada y salida, la mayoría de ellas tolerantes a 5V, reloj de tiempo real (RTC) con calendario y alarmas en hardware, puede manejar 8x40 o 4x44 segmentos de un LCD (*Liquid Cystal Display*), hasta 24 canales con sensor capacitivo para soporte de teclas táctiles o sensores rotatorios o lineales, hasta 16 temporizadores: 2 temporizadores avanzados de 16 bits para el control de motores, 2 temporizadores de 32 bits, 5 temporizadores de 16 bits de propósito general y 2 temporizadores con operación de perro guardián (WDT – *Watch-Dog Timer*), 4 filtros digitales para modulación sigma-delta, periféricos analógicos con fuentes de alimentación independientes: 3 convertidores analógico a digital – ADC de 12-bits y 5Msps, 2 convertidores digital a analógico – DAC de 12 bits, 2 amplificadores operacionales configurables y 2 comparadores de ultra baja potencia. Adicionalmente, cuenta con 20 interfaces de comunicación: USB OTG 2.0, SAI, I2C, UART, LPUART, SPI, CAN, SWPMI e IRTIM (ST Electronics, 2024).

La Figura 2 muestra una representación gráfica de la distribución de terminales del microcontrolador utilizado:

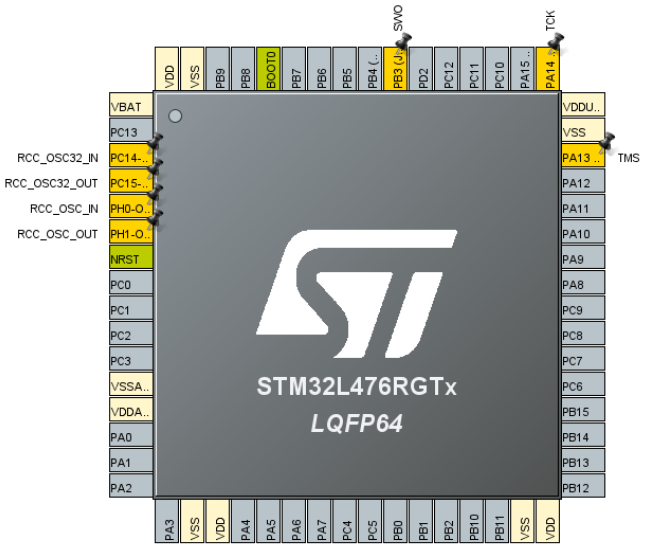

Figura 2. Distribución de terminales del microcontrolador SMT32L476RTG.
Fuente: Elaboración propia.

El Bus CAN (*Control Area Network*)

La red de comunicación CAN fue desarrollado por Bosh en los años ochenta y a partir del 2008, la organización internacional que regula las normas la establece como la única red multiplexada de comunicación en automóviles. Este protocolo de comunicación permite garantizar el acoplamiento de las redes de comunicación en el ámbito automotriz y los estándares a nivel de diagnóstico de fallas, para el funcionamiento correcto de un vehículo (Sánchez Vela, y otros, 2016).

La estructura de interconexión de la red CAN utilizada en este proyecto es mostrada en la Figura 3. En esta red, son utilizados transceptores MCP-2551 como interfaz entre el protocolo CAN manejado por el microcontrolador STM32L476 y el bus físico. Esta configuración permite la transmisión de mensajes en entornos distribuidos ofreciendo una solución eficiente a la comunicación entre múltiples microcontroladores o unidades de proceso (Martínez Requena & García Martín, 2017).

La transmisión de las señales en un bus CAN se realiza a través de dos cables trenzados. Las señales de estos cables se denominan CAN_H (CAN high) y CAN_L (CAN low) respectivamente. El bus tiene dos estados definidos: estado dominante y estado recesivo. En el estado dominante hay una diferencia de tensión entre CAN_H y CAN_L de al menos 1.5V, mientras que, en el estado recesivo, los dos cables del bus se encuentran al mismo nivel de tensión (modo de voltaje común).

El protocolo ofrece los siguientes beneficios:

- Los niveles de voltaje manejados en las dos líneas de comunicación que integran la capa física son complementarios, lo que le permite contar con una alta inmunidad al ruido.
- El protocolo CAN es un protocolo normalizado, lo que simplifica la comunicación entre subsistemas de diferentes fabricantes sobre una red común.
- El controlador maestro o anfitrión puede delegar la carga de comunicaciones a un periférico inteligente, logrando con esto, disminuir su carga computacional.
- Al tratarse de una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto.
- Se requiere de un par de cables trenzados con terminaciones de un resistor de 120Ω en cada extremo para instrumentar la capa física de la red.

- La red CAN tiene como objetivo ser robusta en la transmisión de datos y no veloz (limitada a 1Mbs).

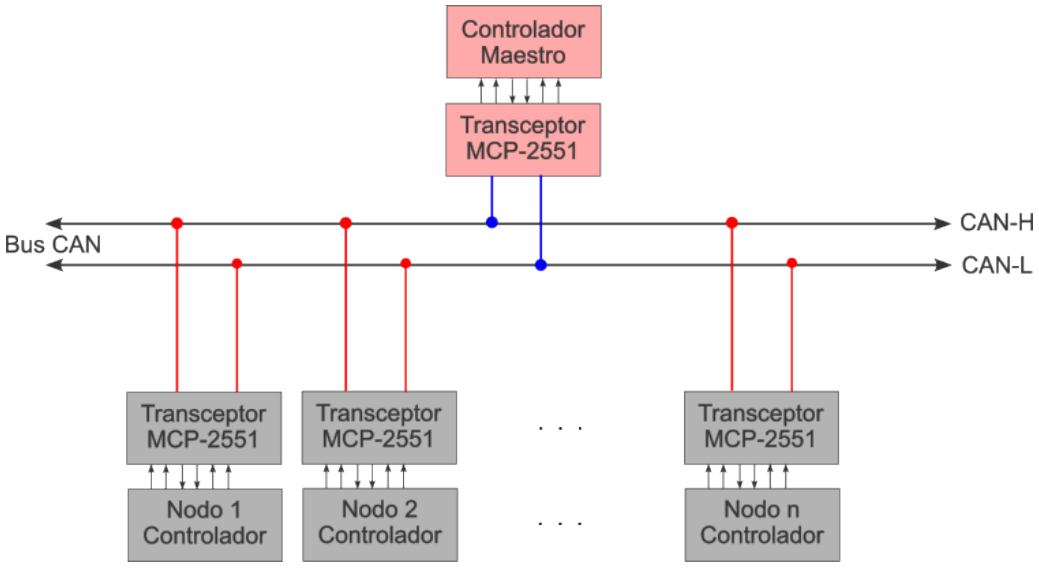

Figura 3. Realización de una red CAN utilizando un transceptor MCP-2551.
Fuente: Elaboración propia.

Configuración del microcontrolador STM32L476 (NUCLEO-L476)

Como punto inicial, debemos establecer las características de funcionamiento de nuestro sensor de temperatura. En este caso, se tendrán las siguientes características (Noviello, 2018):

- El sensor de temperatura utilizará un termistor NTC de $10k\Omega$.
- Las mediciones se realizarán a una razón de una medición por segundo.
- Se utilizará el modelo beta del termistor para determinar la temperatura.
- El ADC será operado en modo de interrupción sincronizando el inicio de conversión por el temporizador.
- La temperatura calculada será transmitida a través del bus CAN.

Configuración de operación del microcontrolador

La configuración de los tres módulos necesarios para esta aplicación es realizada mediante el STM32CubeMX (herramienta gráfica de configuración de la familia de microcontroladores STM32), la configuración es mostrada a continuación:

- El convertidor analógico a digital (ADC), configurado para operar en modo de interrupción y sincronizando el inicio de cada conversión a través del temporizador (TIM). Esta configuración es mostrada en la Figura 4.

- El temporizador (TIM3), el temporizador es configurado para que tome como fuente de reloj el reloj interno de operación PCLK1 (80MHz) (ST Electronics, 2024). De esta forma, la configuración es mostrada en la Figura 5 y la frecuencia de actualización de evento estará determinada por la ecuación:

$$F_{\text{Update Event}} = \frac{\text{PCLK1}}{(\text{PSC}+1)\cdot(\text{ARR}+1)} \quad (7)$$

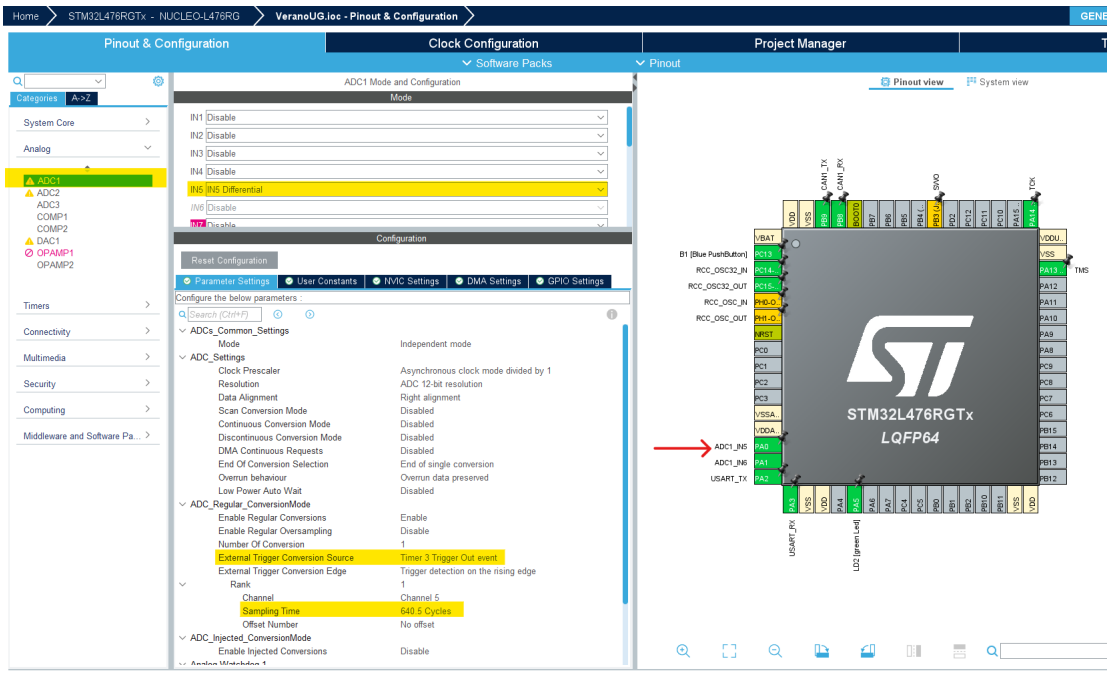

Figura 4. Configuración del ADC en modo de interrupción sincronizando el inicio de conversión a través del temporizador (TIM3). Fuente: Elaboración propia.

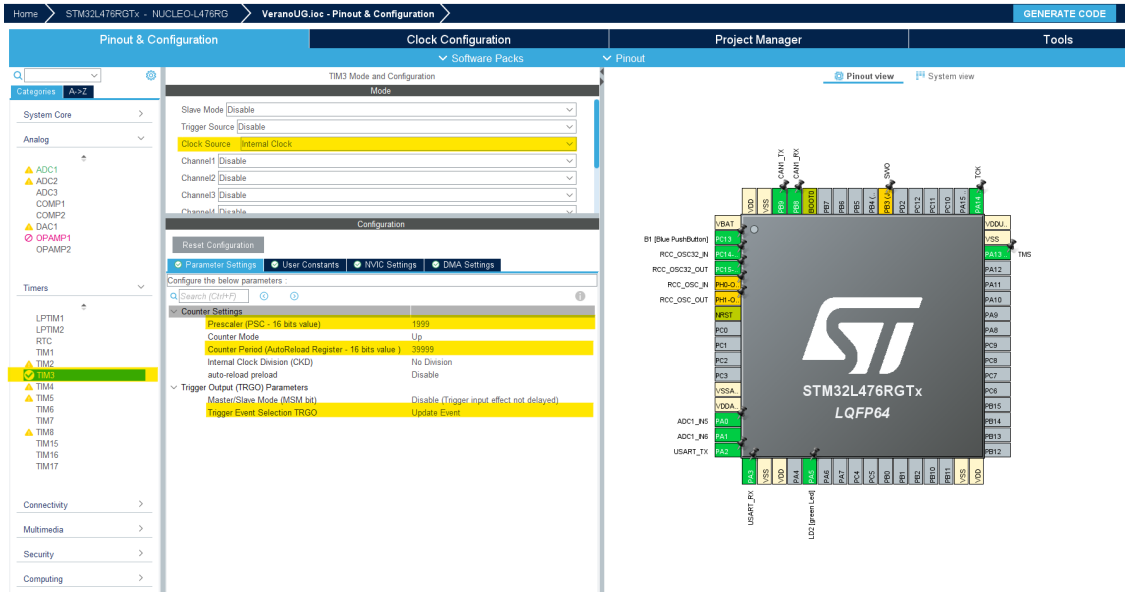

Figura 5. Configuración del temporizador (TIM3) para generar una actualización de evento cada segundo sincronizando el inicio de conversión en el ADC-CH5. Fuente: Elaboración propia.

- El puerto de comunicación CAN (CAN1) (ST Electronics, 2025). El puerto de comunicaciones CAN fue seleccionado utilizando las terminales PB8[CAN_Tx] y PB9[CAN_Rx] (ST Electronics, 2022) que serán interconectadas de las líneas Tx y Rx del transceptor MCP-2551 respectivamente. Los parámetros de configuración han sido seleccionados para obtener una frecuencia de transmisión de 500kbs. La configuración es mostrada en la Figura 6.

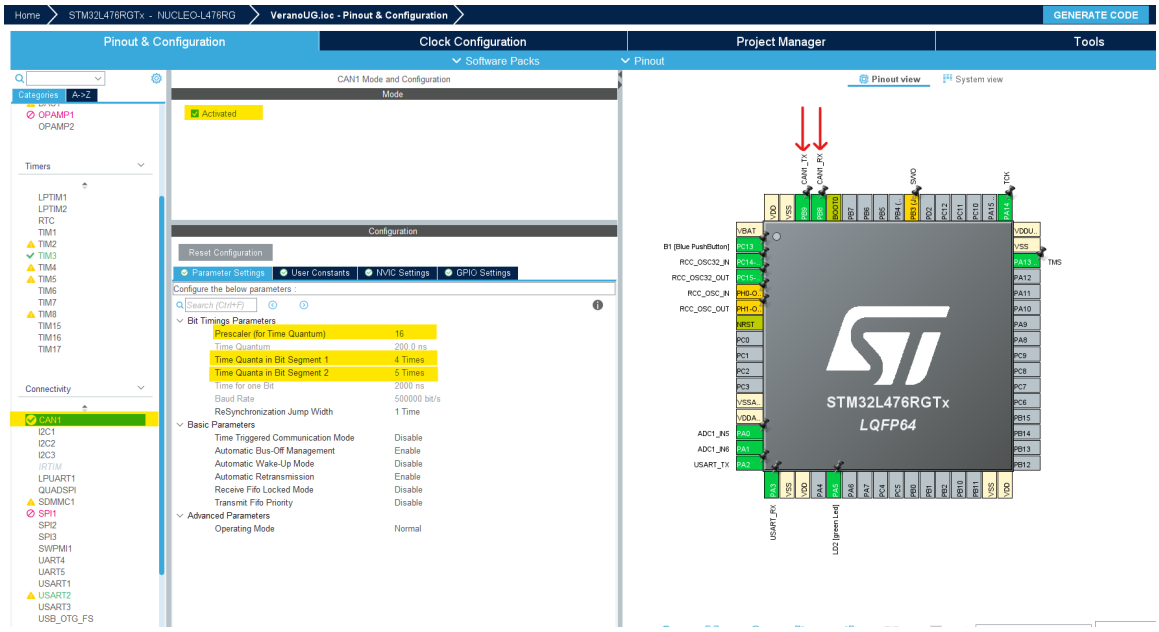

Figura 6. Configuración del puerto CAN (CAN1) para operar a una frecuencia de transmisión de 500kbs. Fuente: Elaboración propia.

Programación de la aplicación en el STM32CubeIDE

El programa fue desarrollado, compilado y enlazado utilizando la herramienta STM32CubeIDE (Entorno de desarrollo integrado de la familia de microcontroladores STM32). Esta herramienta genera el archivo ejecutable que será enviado al microcontrolador para ser almacenado en memoria no volátil y poder ejecutado como un programa embebido. El programa es mostrado en los siguientes seudocódigos:

- **Seudocódigo del programa principal:**

```
Inicio:
      // Librerías a incluir:
      Incluir: stdlib.h, stdio.h, string.h, math.h

      // Definición de constantes
      Definir:    Vref     3.3      // Voltaje de referencia
                  Rref     10000    // Resistencia utilizada en el divisor
                  RT       10000    // Resistencia nominal del termistor
                  beta     3950     // Beta del termistor

      // Definición de variables
      Definir variables:
                  ADC_data, msg_count=0, TxMailbox como uint32_t
                  Vo, Rx, T como flotante;
                  TxDataString[8], TxData[8], RxData[8]; como char
                  CAN_TxHeaderTypeDef como TxHeader
                  CAN_RxHeaderTypeDef como RxHeader

      // Declaración de funciones prototipos del HAL
      Declaración: Funciones prototipo para la atención de interrupciones:
                  HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
                  HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)

      // Sección de código de usuario 2
      Imprimir: "\ec"
      Imprimir: "Iniciando Bus CAN...\r\n"

      // Inicido del puerto CAN no. 1
      Iniciar puerto CAN:
                  HAL_CAN_Start(&hcan1)
```

```
                HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING)

    // Inicio de operación del convertidor analógico a digital (ADC)
    Calibrar ADC:          HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED)
    Iniciar ADC:           HAL_ADC_Start_IT(&hadc1)
    Iniciar temporizador: HAL_TIM_Base_Start(&htim3)

    // Ciclo infinito:
    Mientras(1){
       Si (Hay fin de conversión del ADC)
                Ejecutar: Interrupción de fin de conversión
       Fin Si
       Si (Hay petición de recepción de mensaje en el CAN)
                Ejecutar: Interrupción de recepción de mensaje en el puerto CAN
       Fin Si
    Fin Mientras
Fin
```

- **Seudocódigo de la función de atención a la petición de interrupción de fin de conversión del ADC:**

```
Interrupción de fin de conversión
Nombrar: HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
Inicio
        Si (hadc == ADC1)
                Conmutar estado del PIN: LD2
                ADC_data = valor del convertidor ADC1
                Vo = (float)(3.3*ADC_data/4095)
                Rx = (float)(Rref*(Vref/Vo - 1))
                T = (float)(298.15*beta/(beta+298.15*logf(Rx/RT)) - 273.15)

                Imprimir: "TxDataString,"%3.2f",T"
                Si (Mensaje Transmitido == OK)
                        Imprimir: "Message transmitted -> CAN\r\n"
                Si No:
                                Imprimir: "Error transmitting... \r\n"
                Fin Si
        Fin Si
Fin
```

- **Seudocódigo de la función de atención a la petición de interrupción de la recepción del mensaje en el puerto CAN:**

```
Interrupción de recepción de mensaje en el puerto CAN
Nombrar: HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
Inicio
        Imprimir: "HAL_CAN_RxFifo0MsgPendingCallback\n\r"
        Si (Mensaje Recibido == OK)
                Imprimir: "Got message %lu\n\r"+msg_count
        Fin Si
        Imprimir: "Temperatura:"+RxData
        msg_count++
Fin
```

Pruebas y resultados.

Para realizar las pruebas de funcionamiento del sensor de temperatura en una red CAN es necesario tener al menos dos nodos, uno que operará como controlador maestro y el sensor de temperatura como nodo esclavo. En una red CAN, la información que transmite cualquier nodo es escuchada por todos los nodos. De esta forma, la prueba la realizaremos programando al nodo esclavo como un transmisor que envía la temperatura detectada a través de la red CAN de forma continua a una razón de una medición por segundo, de modo que el controlador maestro la estará registrando.

La Figura 7 muestra las conexiones realizadas para operar la red CAN. En esta figura se observan dos tarjetas de desarrollo NUCLEO-L476RG que operan como nodos CAN. El controlador del nodo 1 ha sido programado para funcionar como sensor de temperatura utilizando un termistor NTC, el cual se observa en la misma figura. Una segunda tarjeta de desarrollo ha sido programada para operar como controlador maestro, registrando la temperatura que esta siendo medida y transmitida a través del bus CAN.

La Figura 8 muestra los mensajes enviados a las terminales de cada uno de los nodos. La primera terminal envía un mensaje "`Message transmitted -> CAN`" cada vez que la temperatura es determinada y enviada a través de la red. Por otro lado, el nodo maestro envía a su terminal el número de mensaje recibido y la temperatura medida.

Finalmente, una segunda prueba fue realizada utilizando la herramienta STM32Cube Monitor, para observar la variación de las variables: T (Temperatura), Rx (La resistencia del termistor) y Vo (El voltaje de salida del divisor de voltaje). Estas mediciones se muestran en la Figura 9.

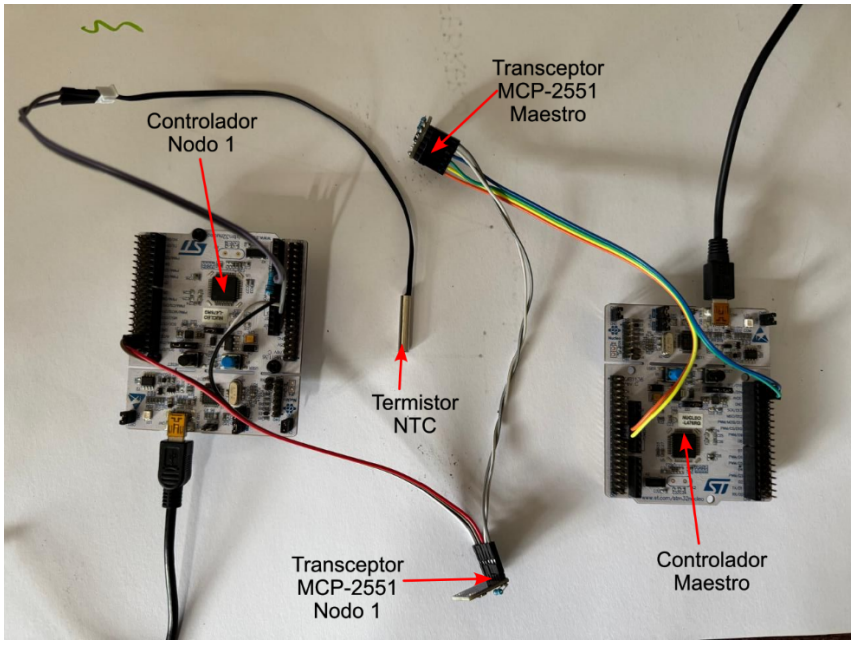

Figura 7. Realización del sensor de temperatura utilizando un termistor NTC con capacidad para establecer una conexión a un bus CAN. Fuente: Elaboración propia.

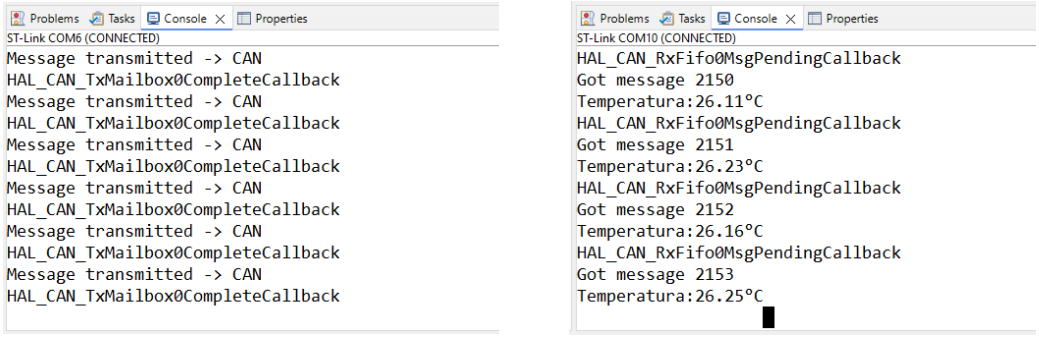

Figura 8. Lecturas observadas en el nodo esclavo y en el controlador maestro del bus CAN. Fuente: Elaboración propia.

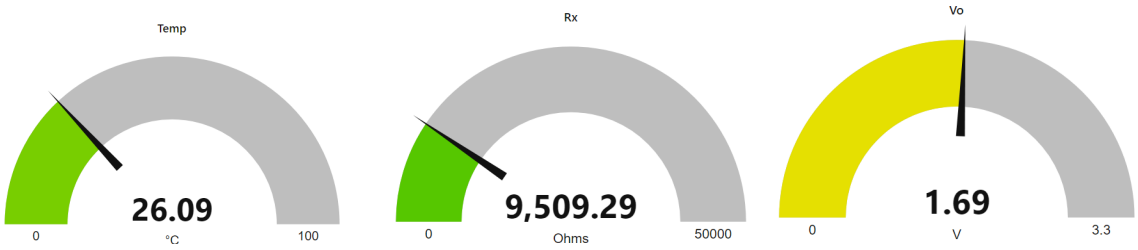

Figura 9. Lecturas de las variables de temperatura, resistencia del termistor NTC y voltaje de salida del divisor de voltaje. Fuente: Elaboración propia.

Conclusiones

En este proyecto se ha logrado realizar un sensor de temperatura basado en un termistor NTC (aplicando el modelo beta) con la capacidad de interconexión con una red de comunicación estándar tipo CAN a una velocidad de 500kbs. Las pruebas realizadas muestran un comportamiento estable y robusto en el canal de comunicación. Los resultados obtenidos dejan las bases para el desarrollo de nuevos dispositivos y sensores que puedan ser aplicados en la industria automotriz. Aplicaciones como controladores de los parámetros de los motores utilizados en vehículos eléctricos, diagnóstico del comportamiento de un automóvil, entre otras muchas aplicaciones.

Una aplicación directa del dispositivo realizado es su adaptación a los sistemas de gestión térmica de los módulos de baterías de los vehículos eléctricos, los cuales deben de permanecer dentro del rango de operación recomendado (rango de operación de un termistor) y particularmente asegurar una temperatura estable durante el proceso de carga. Salir del rango de temperatura de operación y de carga, representa un riesgo de daño permanente al sistema de baterías. Aquí es donde un sistema de medición de temperatura robusto y con capacidad de comunicación digital estándar puede ser utilizado.

Referencias

Martínez Requena, A., & García Martín, J. (2017). *Introducción a CAN bus: Descripción, ejemplos y aplicaciones de teimpo real.* Madrid, España: Universidad Politécnica de Madrid.

Noviello, C. (2018). *Mastering STM32.* Lean Publishing.

Sánchez Vela, L. G., Molano Clemente, M. J., Fabela Gallegos, M. d., Martínez Madrid, M., Hernández Jiménez, J. R., Vázquez Vega, D., & Flores Centeno, O. (2016). *Revisión documental del protocolo CAN como herramienta de comunicación y aplicación en vehículos.* Sanfandila, Qro,: Instituto Mexicano del Transporte.

ST Electronics. (2022). *AN4899 Applications note: STM32 microcontroller GPIO hardware settings and low-power consumption.* ST Electronics.

ST Electronics. (2024). *AN4013 Application note: Introduction to timers for STM32 MCUs.*

ST Electronics. (2024). *STM32L476xx.* ST Electronics.

ST Electronics. (2025). *AN3154 Application note: How to use CAN protocol in bootloader on STM32 MCUs.* ST Electronics.

Texas Instruments Inc. (24 de Octubre de 2025). *MCU Arm Cortex-R*. Obtenido de https://www.ti.com/es-mx/product-category/microcontrollers-processors/arm-based-mcus/arm-cortex-r/overview.html

Thermtest Instruments. (20 de Octubre de 2025). *Propiedades termofísicas y gestión térmica de los vehículos eléctricos*. Obtenido de Thermtest Instruments Latin America: https://thermtest.com/latinamerica/propiedades-termofisicas-y-gestion-termica-de-los-vehiculos-electricos