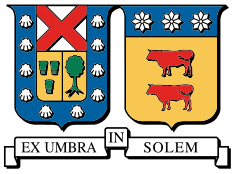

Redes de computadores

Proyecto: Visor de velocidad a traves de protocolo CAN

Luciano Muñoz 201423042-7
Benito Troncoso 201421063-9
Sebastián Neira 201473095-0

13 de Septiembre de 2019

1. Resumen

Controller Area Network (CAN) es el protocolo estándar utilizado en la industria automotriz para comunicar de manera confiable la electrónica de un vehículo. Cada subsistema de un vehículo envía constantemente o según algún requerimiento, datos de su estado actual, posibles fallas y variables de interés. En este caso se estudiará la integración de un motor eléctrico como subsistema con interfaz CAN para luego realizar la decodificación variables de interés transmitidas.

2. Introducción

Este proyecto se centra en el análisis de paquetes CAN enviados por el controlador de un motor eléctrico enviados según protocolo $SAEJ1939$. Para esto se utiliza un CAN Controller y un CAN Transciever integrado a un Microcontrolador($MSP430$) que pueda analizar los datos y desencriptar las variables de interés, en este caso RPM. Para luego implementar el envío de estos datos vía UART hacia un computador quien implementa una interfaz para la visualización de las RPM de parte del usuario final del motor.

3. Protocolo CAN

Fue un protocolo diseñado por la empresa Robert Bosch en los años 80, su objetivo es lograr la comunicación de distintas CPUs o microcontroladores. Es un protocolo robusto, de bajo costo, eficiente y flexible. Flexible en el sentido de que es modular en su implementación y es fácil extraer los errores desde cualquiera de sus nodos. Su propósito inicial y uso principalmente va ligado con la industria automotriz. CAN esta diseñado para entornos hostiles para las señales y dispositivos, donde se trabaja con sistemas en movimiento, con variaciones de temperatura, y otros problemas que provocan señales ruidosas.

3.1. Características y ventajas

Este protocolo funciona de forma multicast en torno a un bus central, por el cual se transmiten todo los mensajes a través de dos cables llamados CAN_H y CAN_L. La principal ventaja de esto es ahorrar una gran cantidad de cables y costos que se utilizarían en una red de conexiones punto a punto. Cabe mencionar que cada mensaje posee un ID especial el que permitiría saber para quien va dirigido el mensaje, de lo contrario los nodos de la red no tendría como discriminar que mensajes son dirigidos para ellos y cuales no. El protocolo CAN se implementa en la capa física y de enlace, basada en CSMA/CD+CR

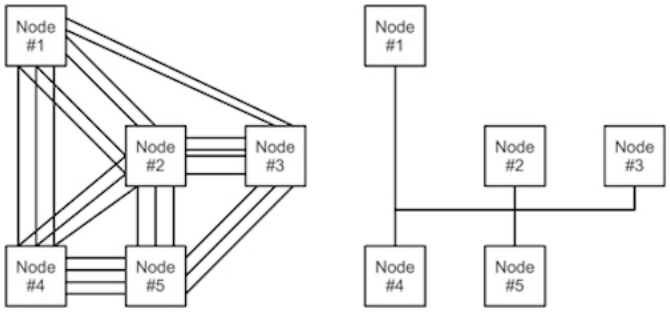

Figura 1: Red punto a punto y Red utilizando *Can Bus*

Opera a velocidades de datos de hasta 1 Megabit por segundo. Admite un máximo de 8 bytes por trama de mensaje y prioridad de mensajes, es decir, cuanto menor sea la ID del mensaje, mayor será su prioridad. Admite dos longitudes de ID de mensaje, 11 bits (estándar) y 29 bits (extendido).

No experimenta colisiones de mensajes (ya que pueden ocurrir bajo otras tecnologías en serie).

3.2. Funcionamiento

El protocolo CAN se comunica a través de dos cables como se muestra en la figura 2 donde el cable superior es CANH y el cable inferior es CANL, luego estos se conectan a los llamados nodos que básicamente son los microprocesadores de los cuales hablamos anteriormente, estos a su vez esta monitorizando alguna parte del vehículo. Cuando un nodo transmite información esta es recibido por todos los nodos lo cual convierte al protocolo en un arma muy poderosa si se trata de tener feedback entre cada parte del motor. Si se presenta un error se sigue una serie de instructivos para tratarlo.

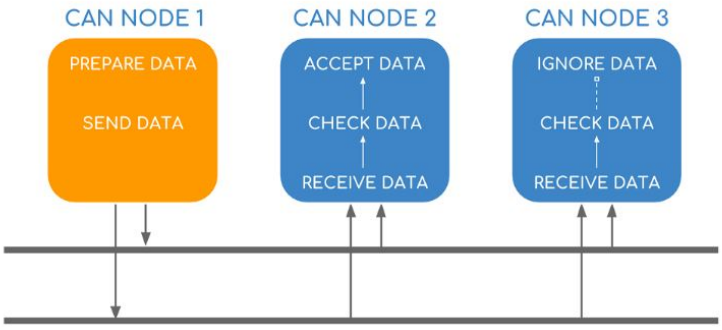

Figura 2: Nodos CAN

3.3. Bus de datos

El protocolo CAN esta compuesto por un bus de datos que tiene la forma mostrada en la figura 3. El campo SOF indica el comienzo de la transmisión de un nodo con un bit 0, luego el campo $CAN - ID$, contiene el identificador, el campo RTR indica si va a hace runa pregunta o enviara data a otro nodo, el campo $Control$ indica el largo del data, el campo $Data$ contiene la data o información del frame, el campo CRC asegura la integrada de la data, el campo ACK indica si no hubo errores en el recibimiento del data y por ultimo el campo EOF indica el final del frame.

Figura 3: Forma del frame de datos

4. SAE J1939

Aunque CAN es eficaz en automóviles y pequeñas aplicaciones integradas, CAN por sí solo no es adecuado para proyectos que requieren manejo de red y/o envío de paquetes grandes.
Por lo cual existen los protocolos de capa superior (software adicional en la parte superior de la capa física CAN) como SAE J1939 quienes fueron diseñados para proporcionar una tecnología de red mejorada que admita mensajes de longitud ilimitada y permita la gestión de la red.
SAE J1939 es un protocolo de alto nivel (HLP), es decir, es quien define una estructura sobre la cual son enviados los paquetes CAN y así permitir obtener datos legibles por un ser humano (human readable data). El protocolo está diseñado para trabajar sobre un enlace CAN 2.0 (con 29 bits de identificador) y normalmente opera a 250 [kbits/s]. Además cabe señalar que es altamente utilizado en la industria de maquinaria pesada.
Por otro lado, J1939 soporta comunicación nodo a nodo(peer-to-peer) y mensajes de difusión (broadcast communication); por último, soporta mensajes de un largo máximo de 1785 bytes.
En cuanto el proyecto en cuestión, el desarrollo consta en decodificar los mensajes CAN obtenidos desde el controlador de un motor eléctrico. El controlador utilizado es de la familia 'Kelly KLS-H' el cual permite controlar un motor brushless DC. Este driver está configurado para enviar constantemente mensajes de difusión con datos de interés a través de una interfaz CAN según J1939.

Figura 4: Controlador motor

4.1. Estructura paquetes desde el controlador

El controlador envía mensajes según la estructura especificada en la documentación entregada por el fabricante.
La estructura de paquetes del motor se puede apreciar en la figura 5, tenemos los datos de salida OUT 1 y 2 contienen la información de las RPM del motor datos de salida OUT 3 y 4, corriente y 5 y 6 voltaje 7 Y 8 contiene información de errores del motor , Cada bit indica cierto error diagnosticado e informado por el motor a través de la RED CAN

OUT	IN	ID (0x0CF11E05)						Period(ms)
controller	instrument	P	R	DP	PF	PS	SA	50
		3	0	0	241	30	05	
Data								
1	LSB of speed in RPM	Actual speed(RPM) = （（MSB*256） + LSB）, 1rpm/bit;						
2	MSB of speed in RPM	Range ： 0-6000，maps actual mechanical speed 0-6000rpm;						
3	LSB of motor current	Actual current = （（MSB*256） + LSB）/ 10, 0.1A/bit;						
4	MSB of motor current	Range：0-4000，maps actual current 0-400A;						
5	LSB of battery voltage	Actual voltage = （（MSB*256）+ LSB）/ 10, 0.1V/bit;						
6	MSB of battery voltage	Range ： 0-1800，maps actual voltage 0-180V;						
7	LSB of error code	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
		ERR 7	ERR 6	ERR 5	ERR 4	ERR 3	ERR 2	ERR 1	ERR 0
8	MSB of error code	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
		ERR 15	ERR 14	ERR 13	ERR12	ERR 11	ERR 10	ERR 9	ERR 8

Figura 5: Estructura paquetes desde el motor

En el proyecto se logró decodificar los datos de OUT 1 y 2 desde un nodo conectado a un microcontrolador quien envía los datos a través de un puerto serial a un computador , y así obtener las RPM del motor en tiempo real en pantalla.

5. Resultado de la parte practica

Implementamos en la MSP las lineas de código para la desencriptación de los datos recibidos por can desde el motor para poder interpretar las RPM . Se procesan los primeros dos datos de 8 bits del paquete recibido por el motor, ver figura 5. Luego de lograr corroborar el código se carga en la MSP y se procede a accionar el modulo principal del banco de pruebas el cual permite hacer funcionar el motor y así poder recopilar sus rpm, figura 6 , aceleramos el motor además este envía información por paquetes CAN.
Los datos de salida de la MSP están en serial uart, Es aquí donde utilizamos un pequeño modulo FTDI el cual nos permite transformar estos datos de $UART$/serial a USB/serial para ser leídos por pantalla en el computador gracias a el software $PuTTy$ que actuaria como la capa aplicación de nuestro sistema. En la figura 9 se muestra los resultados por pantalla de las revoluciones del motor en tiempo real.

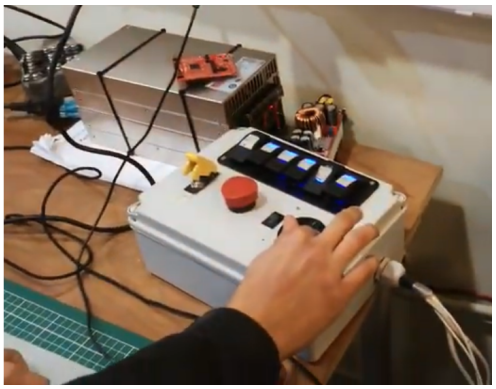

Figura 6: Modulo de accionamiento del banco de pruebas

Figura 7: Rueda del motor funcionando

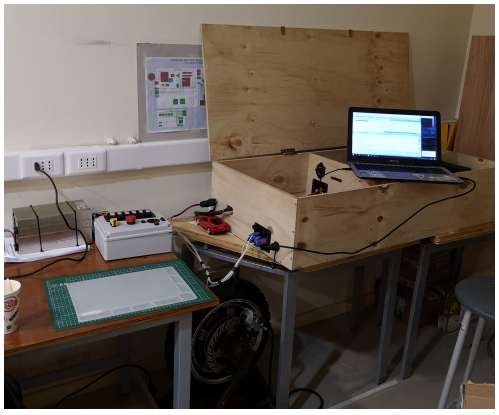

Figura 8: banco de pruebas y motor.

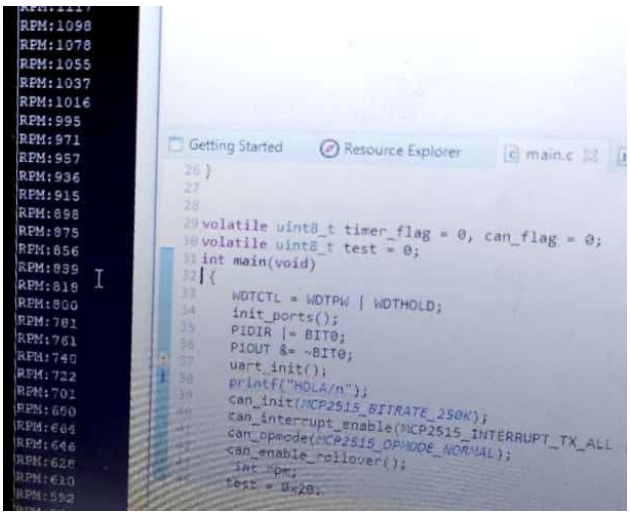

Figura 9: Resultados por USB/puTTy en computador

6. Conclusión

Después de hacer este trabajo se puede notar la importancia del protocolo CAN en el mundo, y sus diversas aplicaciones, también podemos notar lo robusto y económico que es para su aplicación. Sin embargo posee un gran falencia en la transmisión de grandes paquetes de datos la cual es abordado por el protocolo $SAEJ1939$, que como ya vimos permite incluso conectarse con internet, esto ultimo nos permitiría implementar ideas como IoT para el motor y las distintas partes del auto.

En la realización practica de este trabajo nos dimos cuenta de lo simple que es la implantación del código, aparte de lo modular ya que se pudo conectar con el computador a través de un puerto, la información se trabajo a través de una MSP, y esto fue muy útil al momento de extraer la información del motor.

Se plantea el desafío de implementar un protocolo propio, enfocado a nuestros requerimientos y mas eficiente que el estándar $SAEJ1939$. Además de tomar datos de velocidad motor, adquirir mas información útil del motor como información de las baterías, frenos, temperaturas en distintas partes del motor, y de esta forma poder tener medidas preventivas de posibles fallas dentro del auto.