CAN BUS COMMUNICATION IN MOTORSPORT

**Presentación**

Mi nombre es Jose María Fernández León, soy un exmiembro del ya extinto Formula UC3M y actual MAD Formula Team perteneciente a la Universidad Carlos III de Madrid. Trabajé durante 2 años completos en dicho equipo, un primer año dedicado a tiempo parcial entre el departamento de Marketing como encargado de los documentos audiovisuales y el departamento de Electrónica y Control donde, durante esos dos años y más en especial durante ese segundo año me dedique de forma integra a la investigación y desarrollo del protocolo Can Bus de comunicación orientado al mundo del motor.

Formula UC3M es una asociación universitaria que desarrolla y fabrica un coche de competición bajo una normativa específica para poder competir con otras universidades a lo largo y ancho del planeta en Formula Student. En dicho equipo se realiza el proceso íntegro desde el momento de la investigación, pasando por la gestión hasta la fabricación de un coche de competición funcionando así como un equipo profesional de dicho ámbito.

**Algunas otras opciones estudiadas**

**Antes de poder entrar en materia acerca de la comunicación CAN Bus es necesario observar otras posibles opciones que aquí venimos a nombrar de manera breve.**

***Bus LIN***: Se trata de un sistema de comunicación que trabaja como subsistema de can bus utilizado para comunicar un sistema central, conectado a la línea de CAN, con los sistemas esclavos dentro de una subred de comunicación convencional. Se descartó debido a que nos pareció más barato y, para las prestaciones que necesitamos, no era necesario un sistema mayor que el de un Arduino haciendo las labores de dicho sistema central.

**Descripción general**

**Finalidad del proyecto: *Implementación del sistema CAN-BUS para los sensores***

**Qué es el sistema CAN-BUS**

El sistema de Can Bus es un protocolo de comunicación entre CPUs o sistemas centrales de proceso que busca la comunicación entre los diferentes elementos electrónicos del coche, incluyendo los sensores.

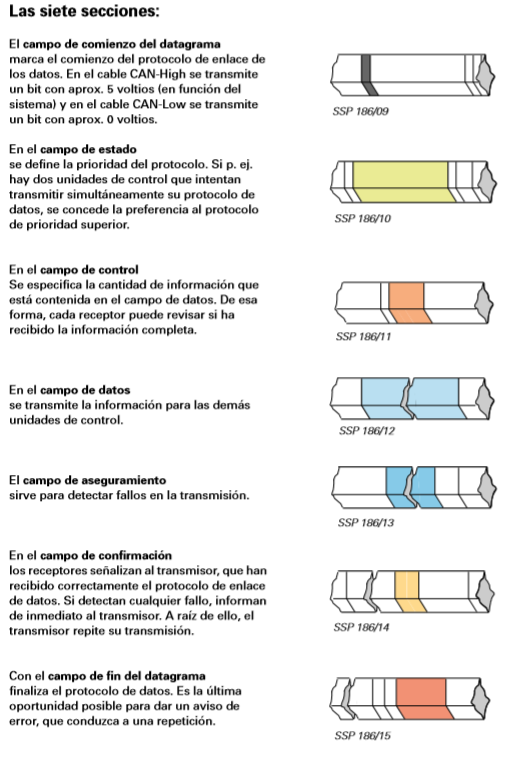
El sistema se basa en una comunicación binaria mediante dos canales (Can High- Can Low) que basan su comunicación en la diferencia de voltaje, estos dos canales son complementarios, es decir, juntos siempre sumarán 5V.

La idea principal de la comunicación es un ecosistema en la que todos tienen la posibilidad de enviar/recibir información de todos. De esta manera, consideraremos a cada uno de los clientes del ecosistema, de aquí en adelante lo llamaremos línea, como nodos dentro de esta. Cada uno de los nodos es responsable de decidir si la información que circula por la línea le es útil o no, por un sistema de identificadores únicos (ID). Dicha ID será proporcionada antes de realizar la descarga en la línea por el nodo transmisor.

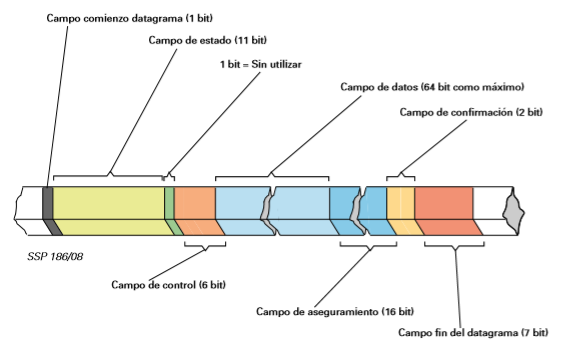
El sistema incluye un sistema jerárquico de envíos en el cuál el mensaje cuya ID sea menor será el que realice el envío mientras que el resto de los sistemas tendrá que esperar su turno.

Para evitar la pérdida de paquetes por la línea, el sistema incluye un sistema de reenvío que estudiaremos más adelante.

La codificación del mensaje es muy similar a la de un paquete IP (Imagen 1 y 2)



(Imagen 1 arriba - Imagen 2 abajo)



**Puntos fuertes**

Gestión de los paquetes - Los envíos se realizan de manera sencilla y corta, al disponer de 8 bits para el envío.

Sencillez de interpretación - Debido a que hemos decidido utilizar la identificación de paquetes mediante el contenido y no a través de la ID (Esto es posible porque todos nuestros sensores comprenden valores entre 0 y 1240, aunque podríamos llegar a comprender un rango de entre 0 a 99999 con una sencilla implementación).

Rapidez en el envío - Al tener unos tramos de cable de comunicación relativamente cortos, podemos llegar a enviar información a 1Mbps, siendo este el máximo del protocolo. Tras las pruebas realizadas podemos concluir que el tiempo mínimo, sin ningún tipo de interferencia entre tramas, es de 0.6s/trama. Es necesario considerar que cada trama comprende todos los sensores a utilizar por conjunto de rueda o nodo. Recibiendo así 1 dato de cada sensor del total de los conjuntos de rueda o nodos cada 2.8 segundos. Podríamos aumentar la velocidad de recepción de datos de ciertos sensores implementando una estructura diferente si fuera necesaria.

***Pruebas ya realizadas con éxito***

* Recepción de datos de ejemplo de hasta 3 nodos diferentes - Funcional y escalable
* Recepción de datos de sensores (4 sensores por nodo) de hasta 2 nodos diferentes - Funcional y escalable
* Protocolo de recuperación del sistema tras pérdida - La pérdida de algún sensor no acarrea la parada del sistema
* Protocolo de inicio del sistema - Comprobado y descrito más adelante, se podría incluir señales luminosas para confirmar el funcionamiento correcto del sistema

***Pruebas a realizar si procede la implementación***

* Calibrado de los tiempos de envío para una mayor eficiencia
* Comprobación de las interferencias electromagnéticas que se puedan producir.

***Protocolo de inicio del sistema***

1. Inicializar los nodos emisores
2. Inicializar el nodo receptor
3. Comprobar que la señal luminosa está encendida
4. Inicializar el sistema de guardado de datos (SD)

**Material necesario**

Dado que todavía no sabemos con total seguridad cuál es el número de nodos a implementar ( a gusto de la dirección técnica) incluiremos aquí los materiales necesarios para cada nodo y el sistema general

* 1xNodo Arduino nano
* 1xNodo Transcriptor MCP 2515
* 1xNodo PCB tipo SOCKET para colocar arduino y transcriptor
* 1xNodo Caja en impresión 3D para encapsular la PCB
* Cable can bus.
* 1 Arduino Mega en receptor (opcional)

**Conexiones necesarias**

*Transcriptores*

* SCK - Pin 13
* MISO - Pin 12
* MOSI - Pin 11
* CS - Pin 10
* GND - Pin GND
* VCC - Pin VCC

Sensores de temperatura (En Paralelo)

* SCL - Pin 3
* SDA - Pin 2
* VCC - Pin VCC
* GND - Pin GND

Sensores de amortiguadores

* Data - Pin A 6 (Válido algún otro)
* GND - Pin GND
* VCC - Pin VCC

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Proceso | Tiempo de Ejecución |
|  | Lectura de sensores de temperatura x4 | 100ms |
|  | Envío completo V1.0 | 631ms |
|  | Envío completo V1.1 (Paquetes sin espacios) | 582ms |
|  | Envío completo V1.2 (Delay de 470ms total) | 511ms |
|  | Envío completo V1.3 (Cambio en el sistema de reconocimiento de paquetes) | 470ms |

**Resolución primer desarrollo**

Tras ver los datos recibidos en el primer desarrollo de nuestro sistema concluimos que no nos proporciona la velocidad suficiente para implementarlo en el coche actual.

Volviendo al paso inicial de documentación podemos observar que alguno de los problemas podría ser la librería que llevamos utilizando a lo largo de todo el proceso.

Tras cambiar a otra librería de implementación similar pero procesado de paquetes diferentes observamos una notoria diferencia, esta vez sí consiguiendo una velocidad muy competente.

Por tanto, podemos concluir que el problema se hallaba en la librería utilizada hasta el momento y, tras unas cuantas investigaciones, concretamos que el problema de dicha librería era el sistema de creación de los paquetes, debido a que en la librería de la primera versión del proyecto se atribuía de manera dinámica el tamaño del paquete, aumentando así en gran medida el tamaño del paquete (1000 veces mayor).

En esta nueva versión se realiza una atribución de tamaño estático a todos los paquetes, consiguiendo una velocidad de procesamiento ínfima y adecuada para el proyecto que nos compete.