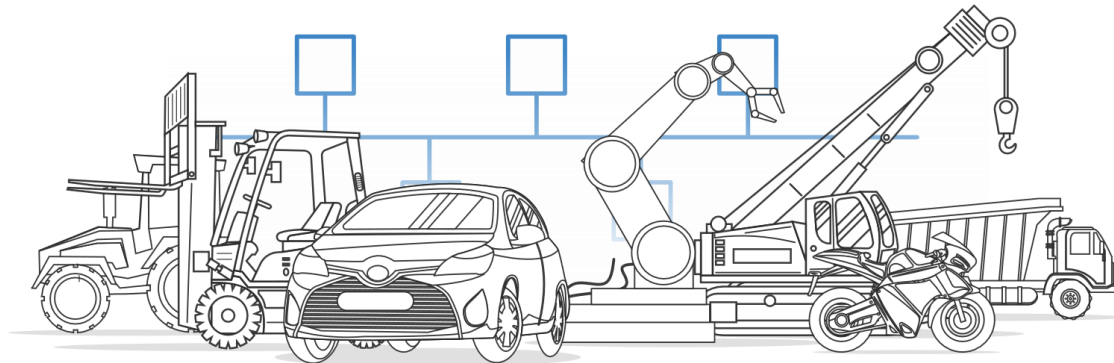


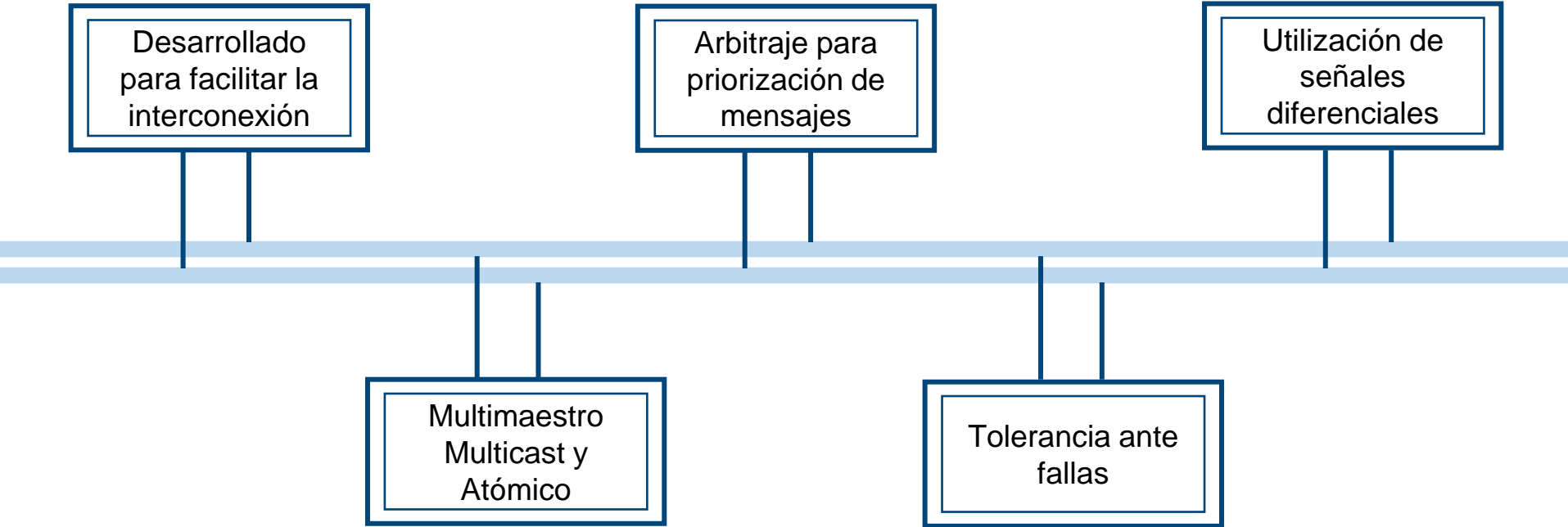
Controller Area Network (CAN)

**Kendell Calderón Láscarez,
Kaleb Granados Acuña &
Lorenzo Sancho Fallas**

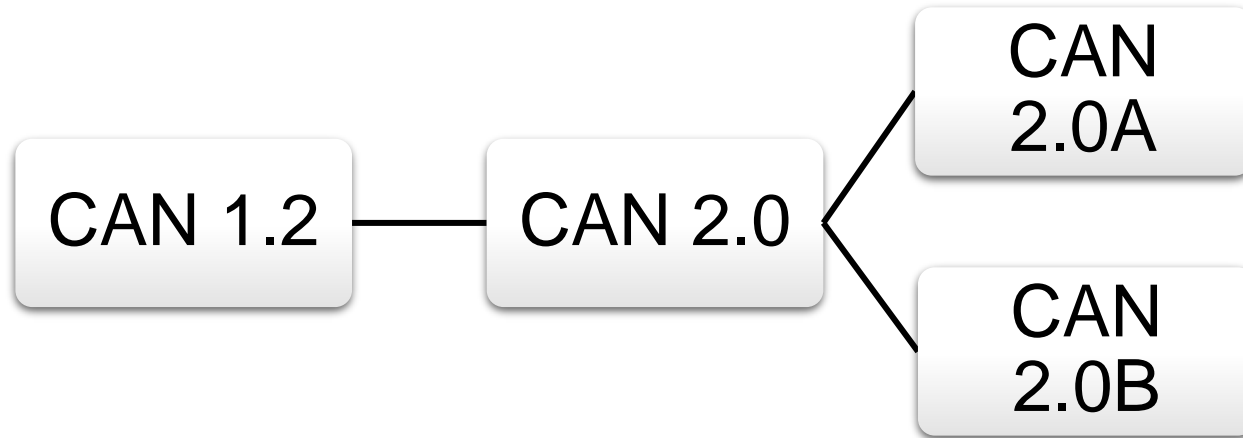
Instituto Tecnológico de Costa Rica
MT8001 - Teoría de Comunicación y Procesamiento de Señales
24 de Octubre del 2024



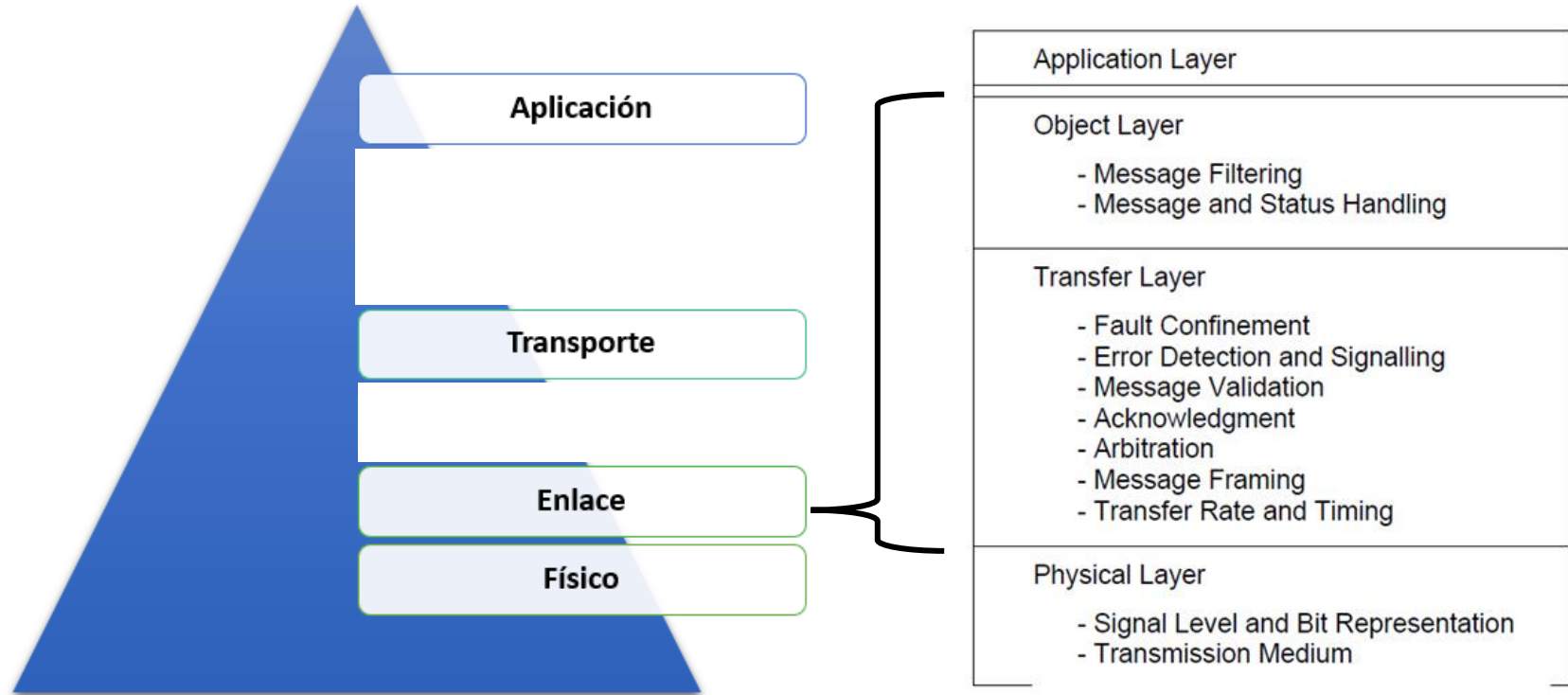
¿Qué es CAN?



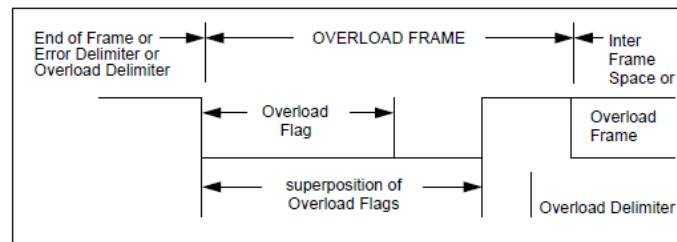
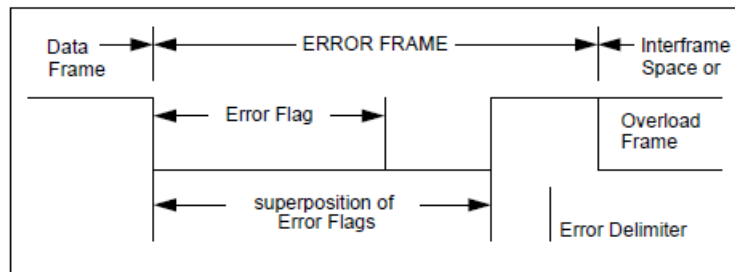
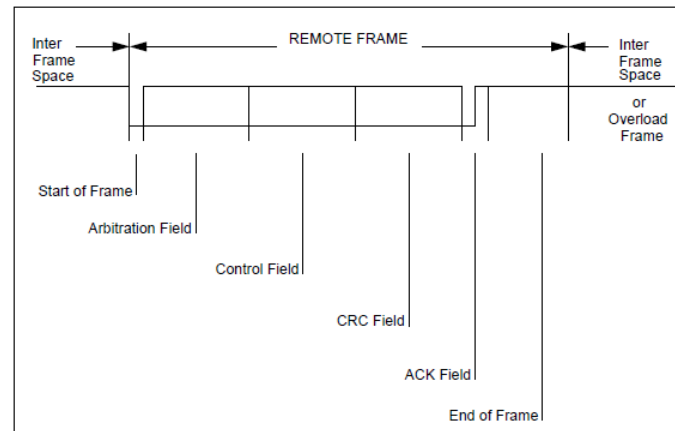
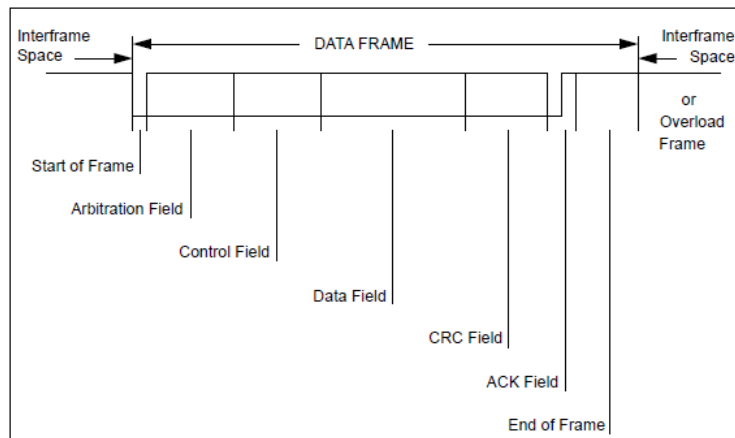
Versiones



CAN 1.2/2.0A

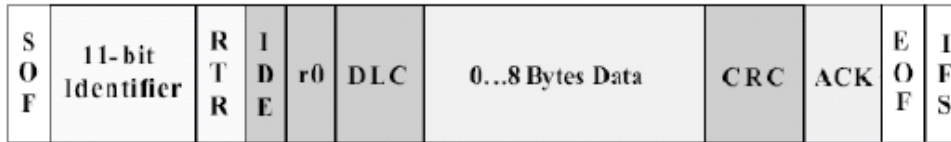


CAN 1.2/2.0A

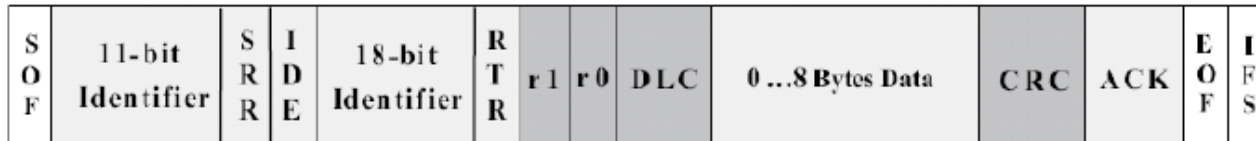


Diferencia CAN 1.2/ 2.0B

- Tamaño identificador
- Compatibilidad
- Arbitraje
- Estructura de las tramas



(a)



(b)

Diferencia CAN 1.2/ 2.0B

Capa Física:

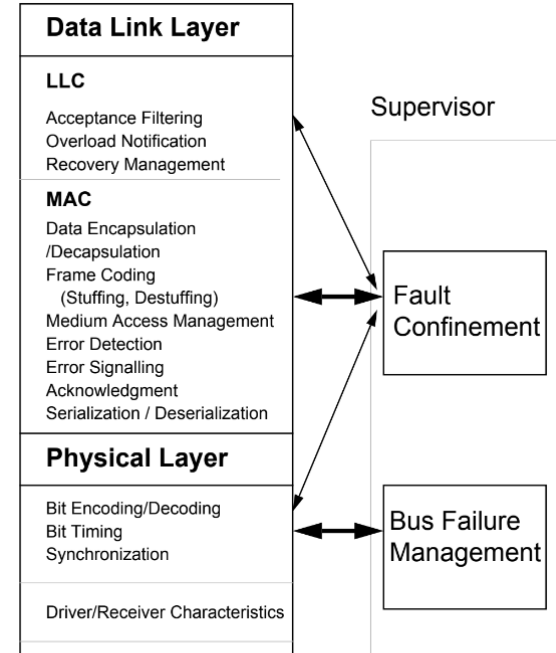
- Codificación
- Sincronización
- Bit timing

MAC:

- Mayor cantidad componentes

LLC:

- Mismos componentes que 1.2.

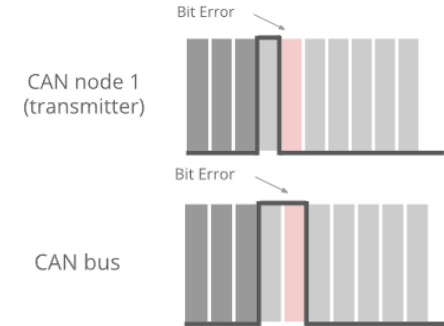


LLC = Logical Link Control
MAC = Medium Access Control

Manejo de errores

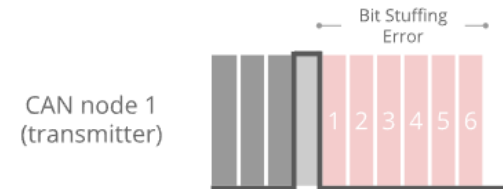
Bit error:

- Los nodos deben leer su transmisión
- Ocurre una lectura diferente del valor transmitido



Bit stuffing:

- Regla de los 5 bits consecutivos
- Ocurre si el 6 no es el complemento



Manejo de errores

Form error:

- SOF, EOF, ACK, CRC (delimitadores) tienen niveles establecidos
- Un nivel inválido de estos genera un error de formato

ACK error:

- Nodos receptores deben confirmar recibido poniendo ACK dominante
- Si un nodo no coloca el ACK en dominante, se produce el error

CRC error:

- Los transmisores envían el CRC asociado al mensaje
- Los receptores deben calcular su propio CRC
- Si la comparación de ambos difiere, se levanta un error de CRC

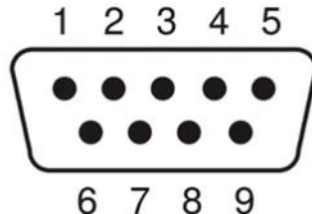
Puertos Físicos Utilizados

Puertos comunes:

- MiniDin, DB15 y DB9
- Complejidad en mantenimiento de sistemas.

CAN in Automation:

- CIA promociona y estandariza tecnologías basadas en CAN.
- CANopen gestiona las complejidades de CAN mediante un protocolo alto nivel.



Estandarización puerto D_sub de 9 pines:

Pin N°	Descripción
Pin 1	No conectado (Reservado)
Pin 2	CAN_L (línea baja del bus CAN)
Pin 3	CAN_GND (Tierra del bus CAN)
Pin 4	No conectado (Reservado)
Pin 5	CAN_SHLD (Pantalla del bus, opcional)
Pin 6	GND (Tierra del dispositivo)
Pin 7	CAN_H (línea alta del bus CAN)
Pin 8	No conectado (Reservado)
Pin 9	No conectado (Reservado)

Puertos Físicos Utilizados

Comunicación diferencial:

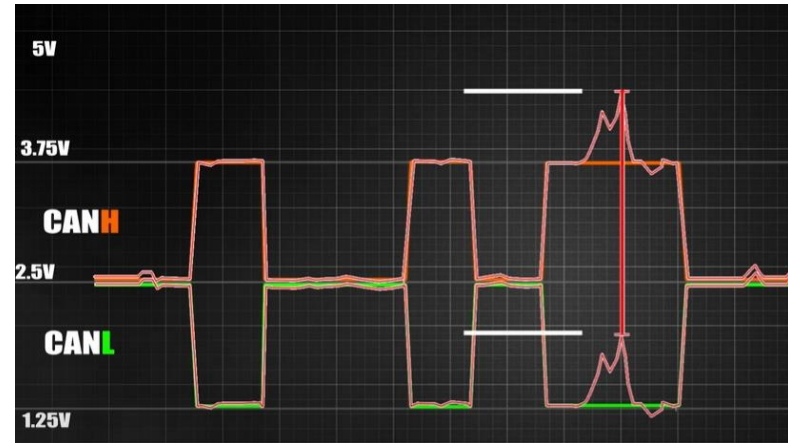
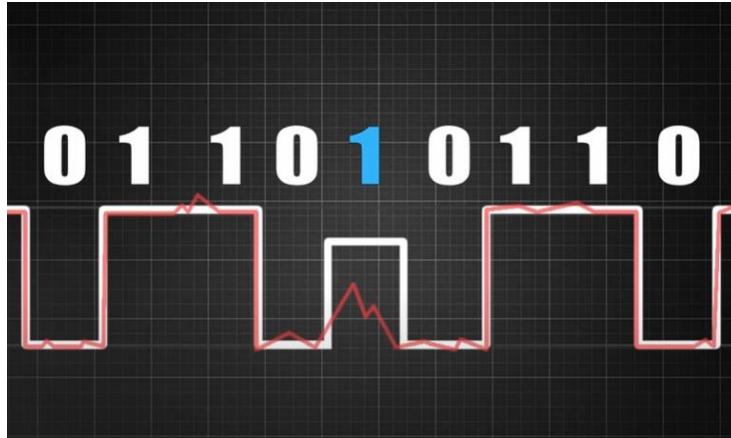
- Cables CAN H y CAN L, para transmitir señales entre nodos.
- Diferencia entre SPI y I2C, una sola línea.

Ventajas:

- Robustez
- Menos susceptible a interferencias

CANH y CANL

- Lectura del diferencial
- Líneas trenzadas con resistencias de terminación
- Dominancia en 0 (2.5V), recesividad en 1 (0V)
- La dominancia relacionada al overwrite.

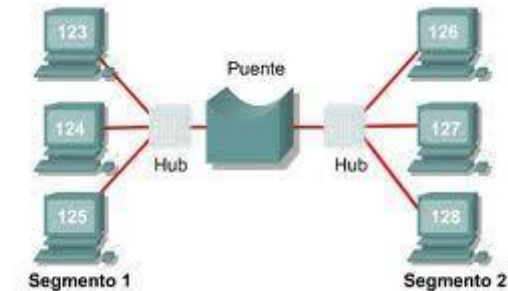
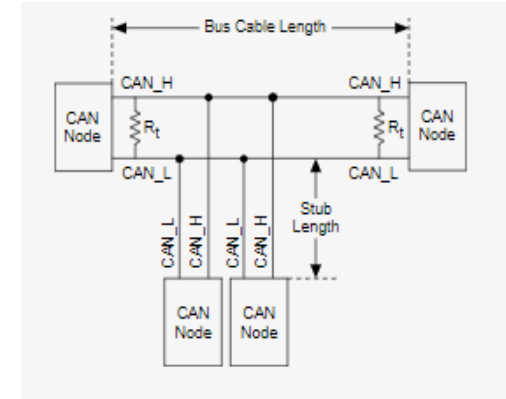
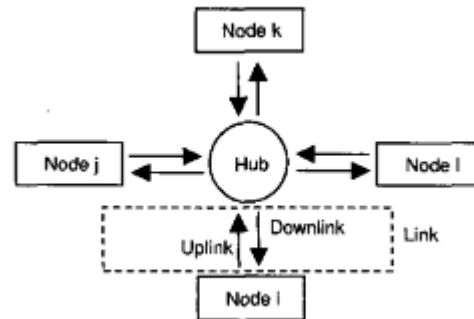


Topologías CAN

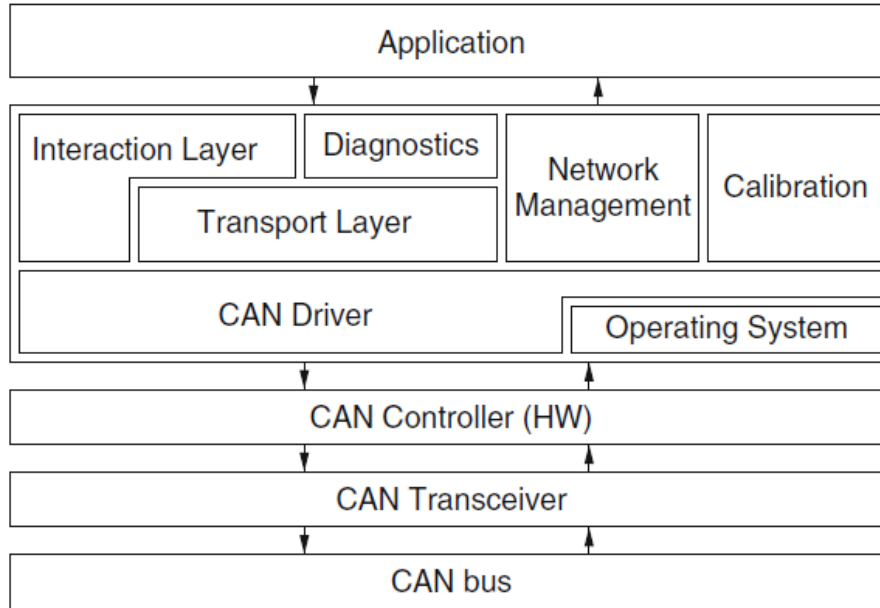
- Topologías nacen de CANOpen
- The General Architecture of CANOpen Network
- Resistencia de terminación

Topologías Comunes:

- Bus lineal
- Estrella
- De puente



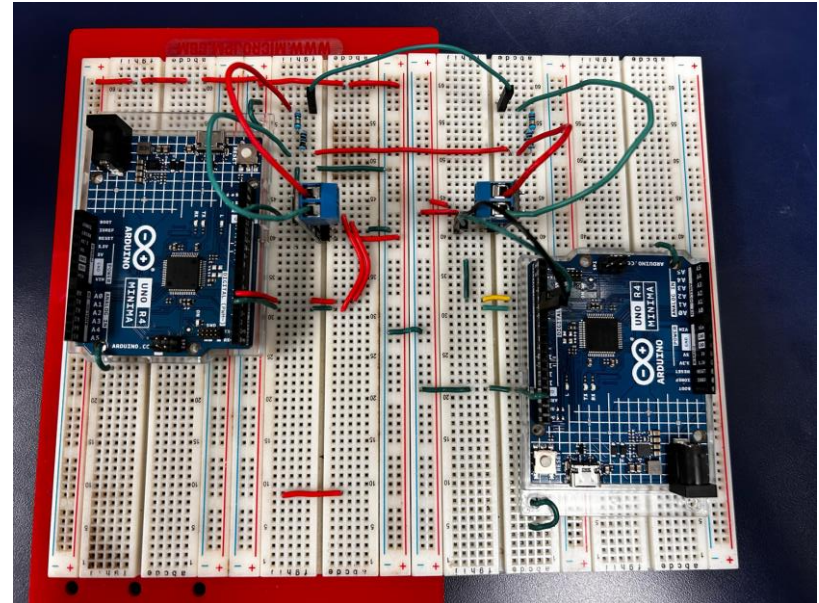
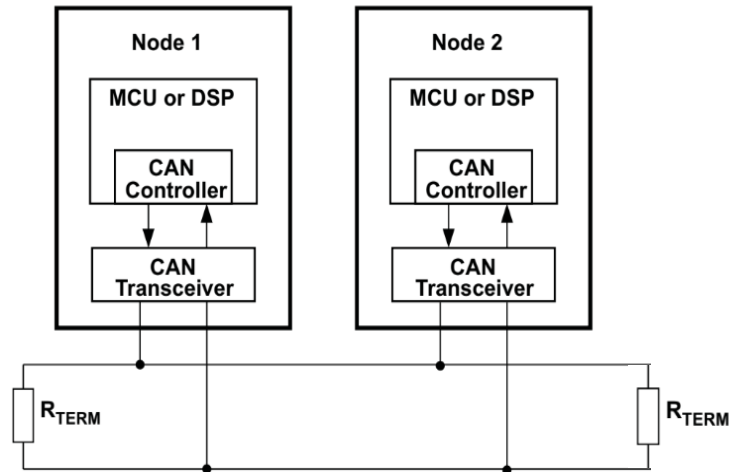
Arquitecturas en CAN



- CANcentrate
- FlexCAN
- FTTCAN
- N-server CAN
- Clock sync CAN
- CAN- RT-TOP
- ...

Implementación topología básica

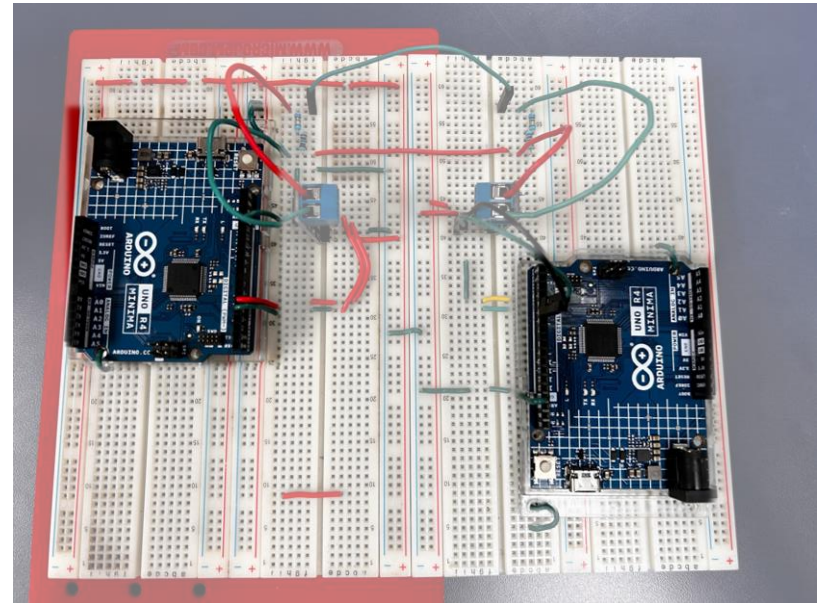
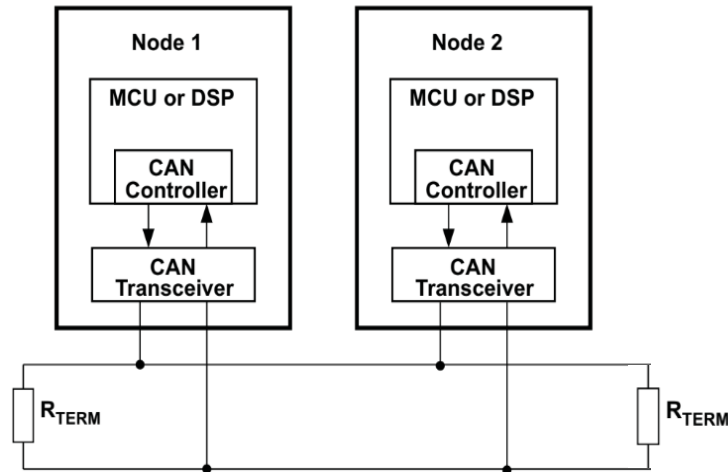
- ECU: UNO R4 MINIMA
- Transceptor: CAN SN65HVD230.



Implementación topología básica

- Implementación con UNO R4 MINIMA
- Transceptores CAN SN65HVD230.

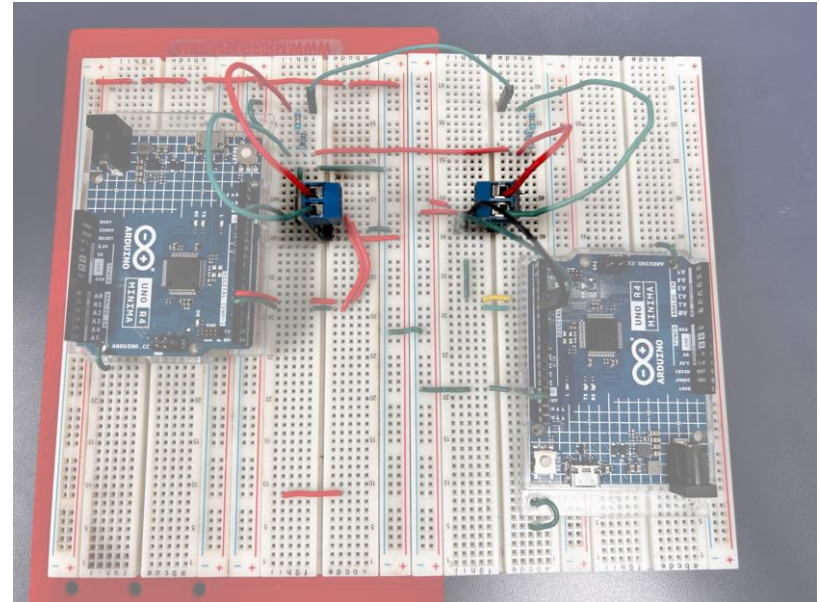
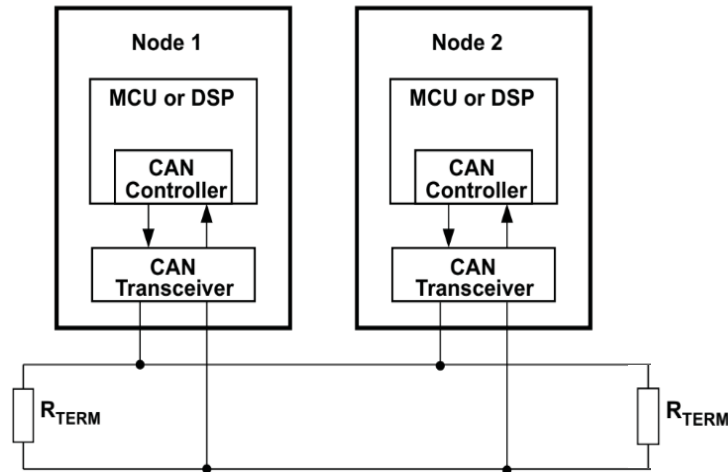
MCU + CAN Controller



Implementación topología básica

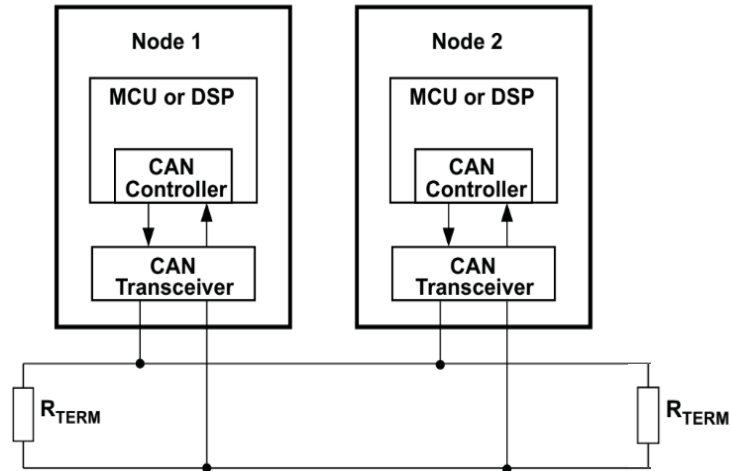
- Implementación con UNO R4 MINIMA
- Transceptores CAN SN65HVD230.

CAN Transceiver

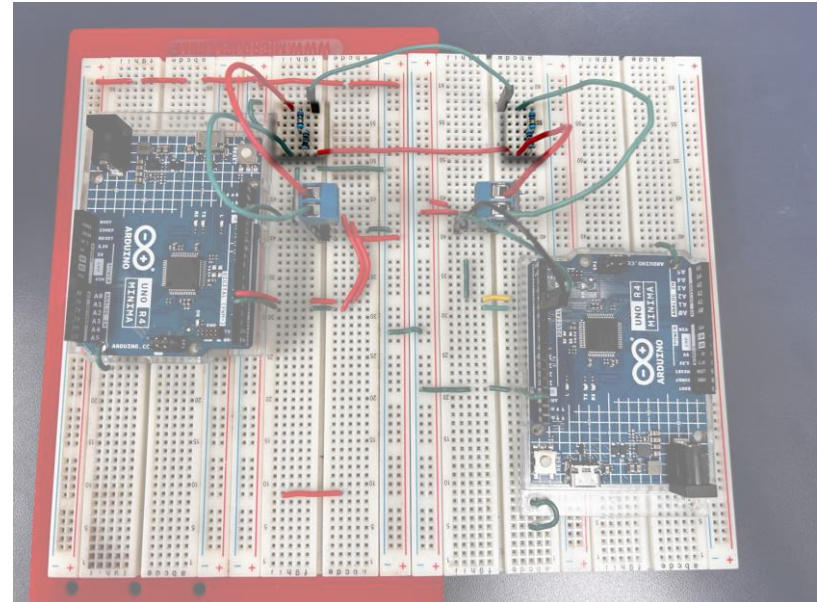


Implementación topología básica

- Implementación con UNO R4 MINIMA
- Transceptores CAN SN65HVD230.

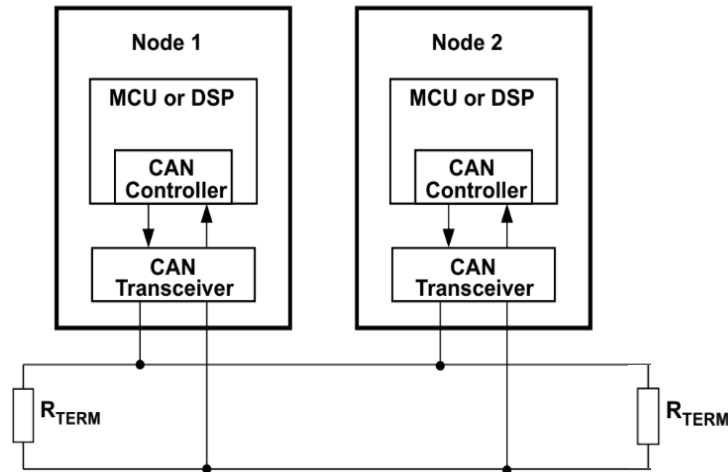


Resistencias Terminación

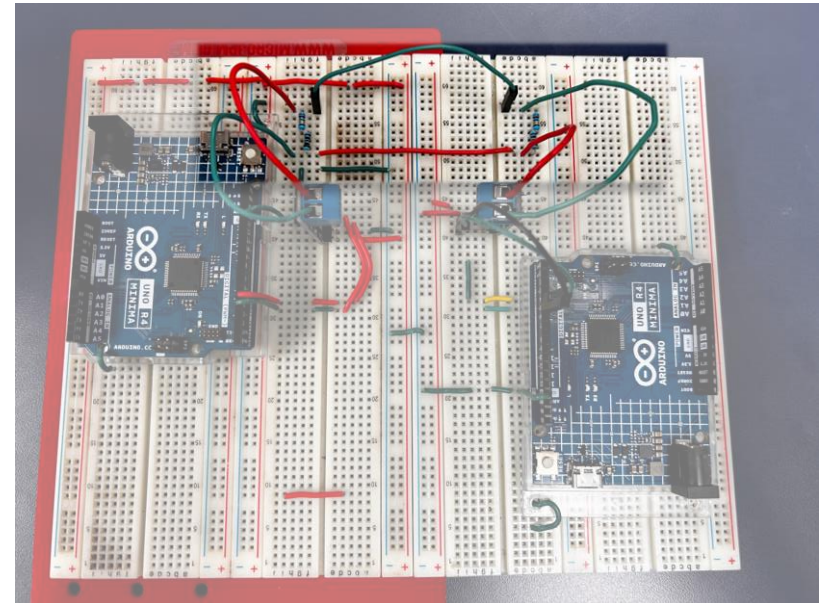


Implementación topología básica

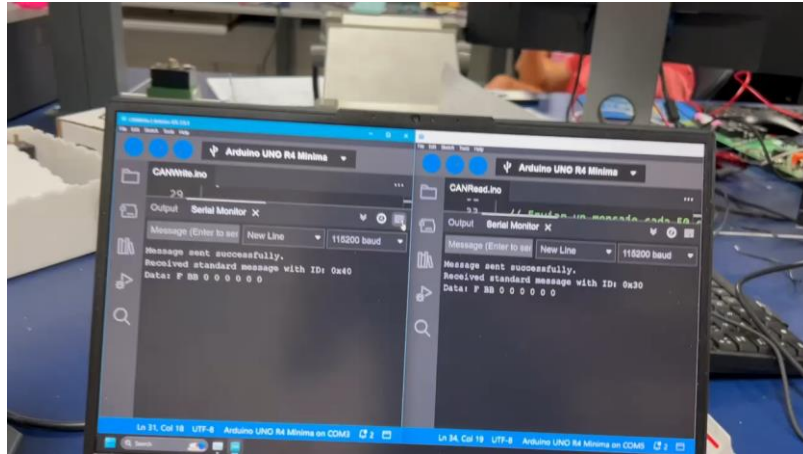
- Implementación con UNO R4 MINIMA
- Transceptores CAN SN65HVD230.



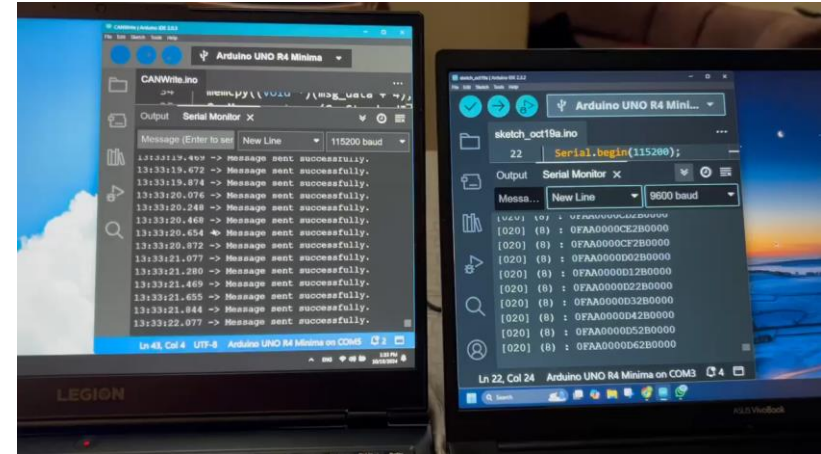
Líneas de transmisión CANH y CANL



Comprobación de comunicación



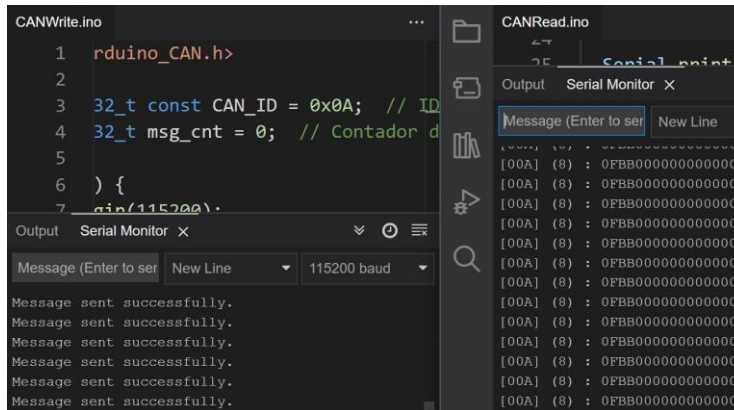
Transmisión Half-Duplex



Transmisión Simplex

Visualización del bus de datos: Resultados

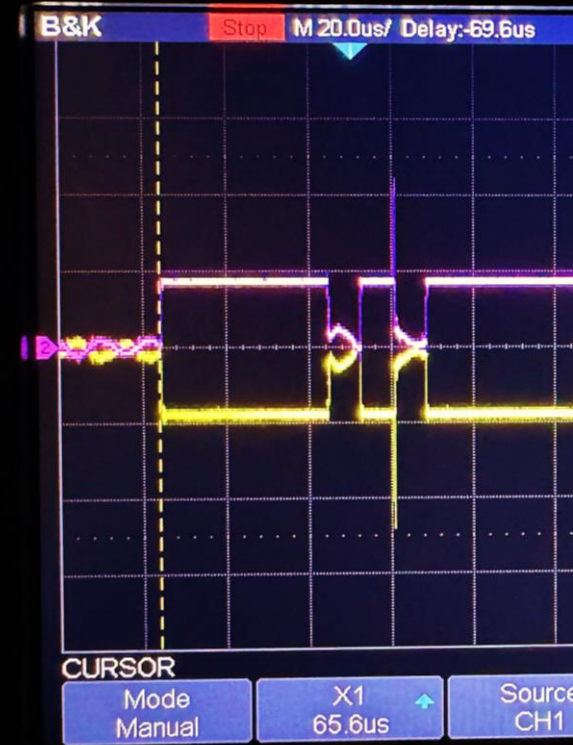
- Scope con CANH y CANL en ambos canales
- Frame Fetch & Decode del mensaje enviado



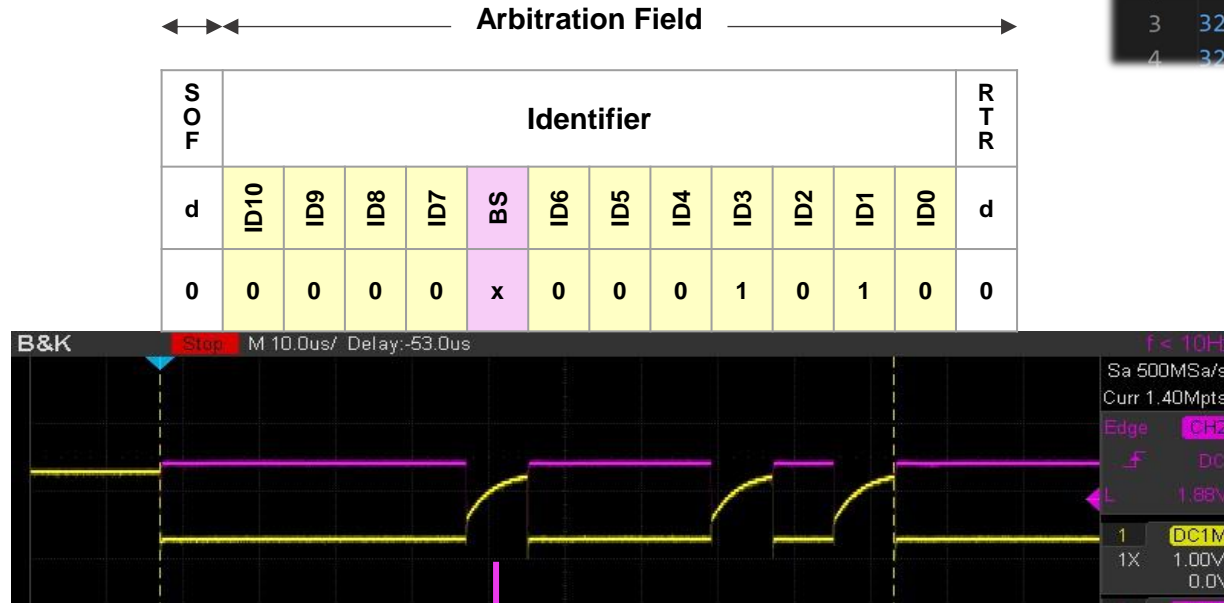
The screenshot shows an IDE with two files: `CANWrite.ino` and `CANRead.ino`. `CANWrite.ino` contains the following code:

```
1  #include <Arduino.h>
2
3  #define CAN_ID 0x00A // ID del mensaje
4  #define MSG_CNT 0 // Contador de mensajes
5
6  void setup() {
7    Serial.begin(115200);
8  }
9
10 void loop() {
11   CANWrite(CAN_ID, "Mensaje enviado exitosamente.");
12   delay(1000);
13 }
14
15 void CANWrite(uint8_t ID, String msg) {
16   Serial.print("Mensaje enviado exitosamente.");
17 }
```

The serial monitor shows the output: "Mensaje enviado exitosamente." repeated five times. The `CANRead.ino` file is partially visible, showing a serial print statement and a list of received messages in hexadecimal format.



Visualización del bus de datos



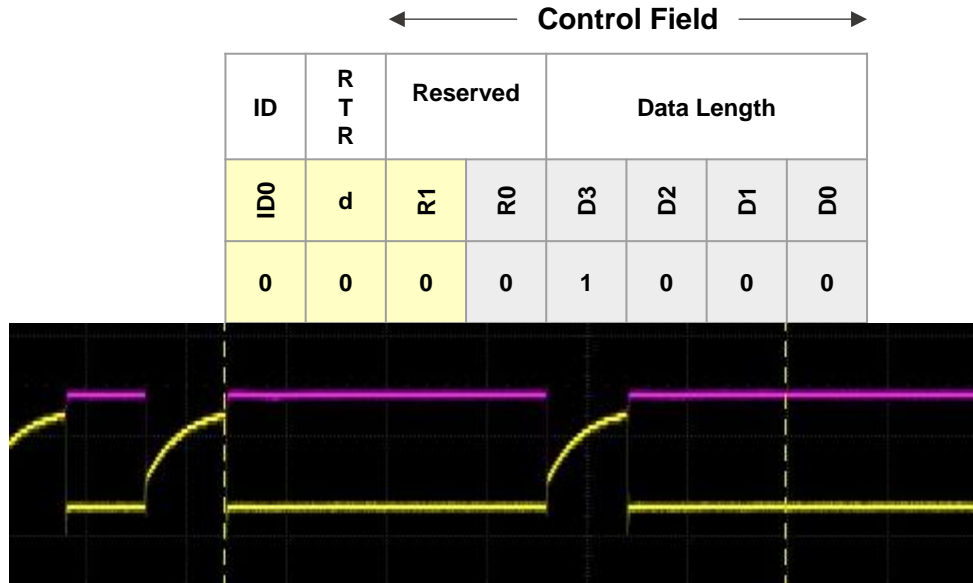
Bit Stuffing

BIT STREAM CODING

The frame segments START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD and CRC SEQUENCE are coded by the method of bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit stream to be transmitted it automatically inserts a complementary bit in the actual transmitted bit stream.

```
2
3 32_t const CAN_ID = 0x0A; // ID
4 32_t msg cnt = 0; // Contador
```

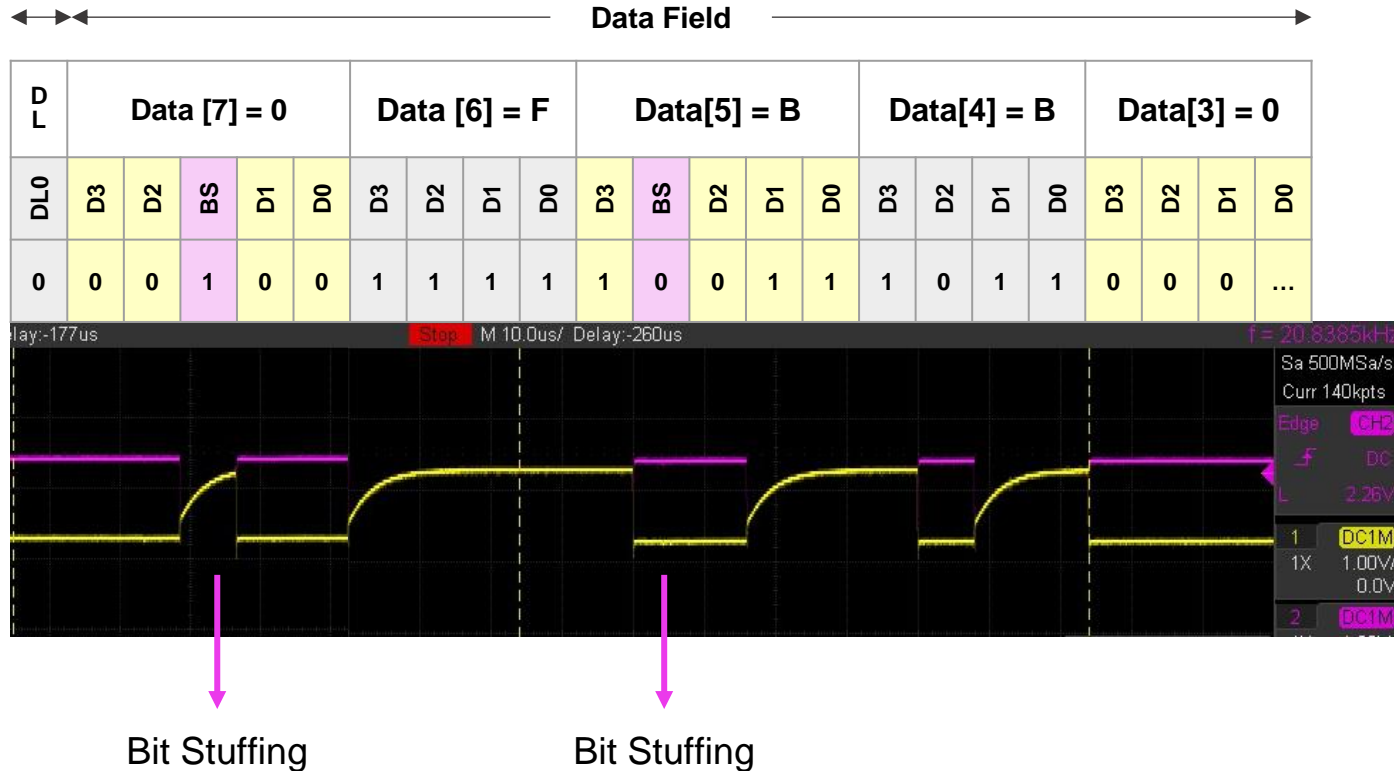
Visualización del bus de datos



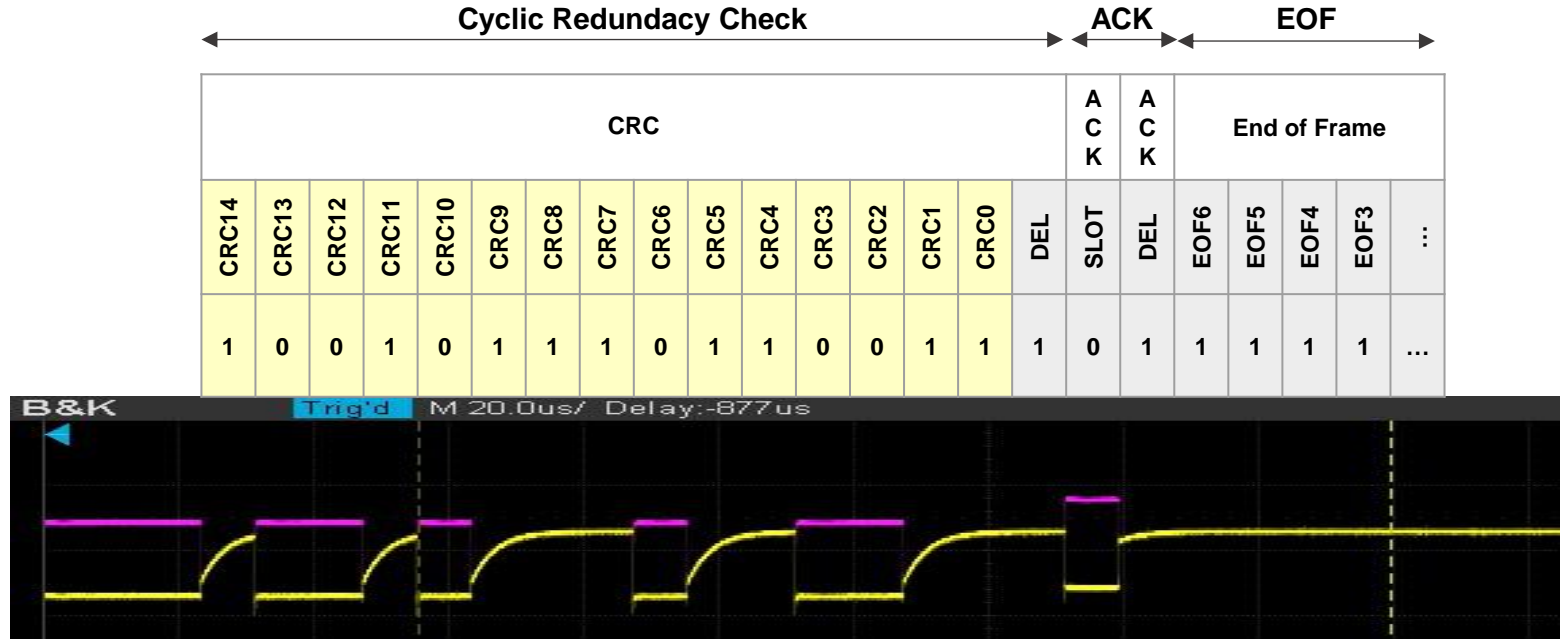
```
[00A] (8) : 0FBB000000000000
[00A] (8) : 0FBB000000000000
[00A] (8) : 0FBB000000000000
```

Number of Data Bytes	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

Visualización del bus de datos

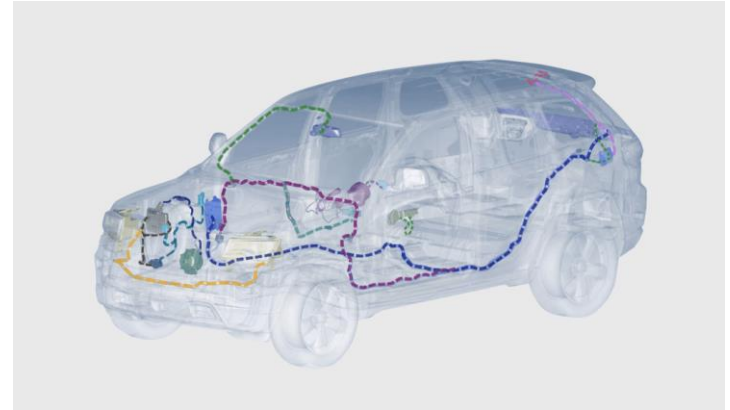


Visualización del bus de datos



Conclusiones

- CAN como solución confiable
- Prioridad de mensajes, multimaster, robustez
- Adaptabilidad mejor la interoperabilidad y simplicidad de dispositivos.
- Relevancia continua del CAN.



¡Gracias!

¿Preguntas?



Referencias, documentación y
demostración