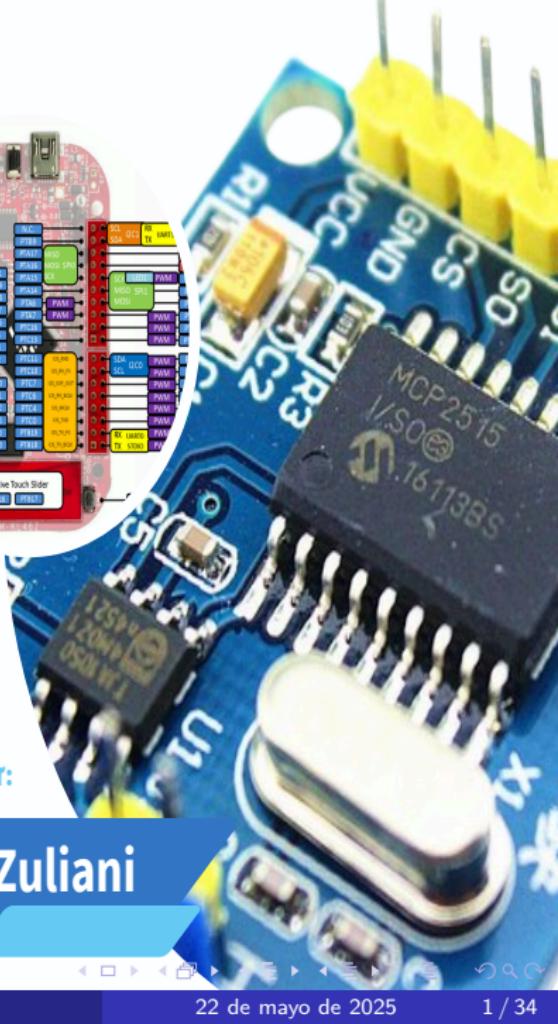
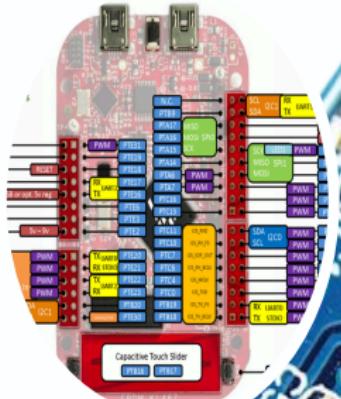


SD2

# CAN MCP2515

Presentado por:

Agustín Zuliani



# Contenido

1 Introducción

2 Motivación

3 MCP2515

- Conceptos
- Características del controlador
- Funcionamiento
- Modos de operación
- Transmisión
- Recepción
- Errores
- Interrupciones

4 Ejemplo

5 Referencias

# Introducción

Este trabajo fue realizado como parte de la adscripción 2024-2025, con los profesores Daniel y Guillermo. A lo largo de esta presentación abordaremos el controlador CAN MCP2515 con sus características, usos y librería disponible para trabajar en la cátedra de Sistemas Digitales 2 (SD2).



# Motivación

- *¿Porqué la comunicación CAN?*

La comunicación CAN es utilizada en un amplio rango de la electrónica tales son el caso de:

- Industria automotriz.
- Automatización industrial.
- Sistemas donde se requiere una gran confiabilidad en la comunicación, como inversores de AC a DC (utilizados en paneles solares).



- *¿Porqué el MCP2515?* Este controlador CAN provisto por Microchip, provee diversas ventajas, nombramos algunas a continuación:
  - **Costo-Calidad:** Es una solución económica que permite garantizar una comunicación de tipo CAN.
  - **Compatibilidad:** Puede ser utilizado con cualquier microcontrolador que posea comunicación SPI para manipularlo.
  - **Flexibilidad:** Ofrece filtros configurables y múltiples buffers, permitiendo un control eficiente del tráfico CAN. Mas adelante profundizaremos sobre estos conceptos.
  - **Confiabilidad:** posee bajas tasas de error, además de diversos mecanismos para reestablecer o cortar la comunicación si fuese necesario.

- ¿Porqué realizar una librería desde cero? ¿Cómo fue realizada? La intención de realizar una librería fue por el hecho de manipular, en este caso mediante SPI, al controlador CAN. Fue realizada en base a una librería provista por el entorno de desarrollo Arduino y adaptada a NXP para la KL46Z en particular. Dicha librería puede ser utilizada para cualquier microcontrolador (con algunos mínimos cambios) siempre y cuando se posea una librería para SPI.

# Conceptos relacionados

Para entender correctamente el funcionamiento de este controlador haremos uso de conceptos adquiridos a lo largo de la asignatura, tales como:

- Comunicación SPI.
- Comunicación CAN.



# Conceptos básicos

- *¿Qué es el MCP2515?*

El MCP2515 es un controlador de protocolo CAN, desarrollado por Microchip. Este permite ser configurado mediante un microcontrolador que incorpore el protocolo SPI. Mediante una librería nos permitirá tanto inicializar como enviar y recibir información. A continuación mostramos una imagen del módulo completo.

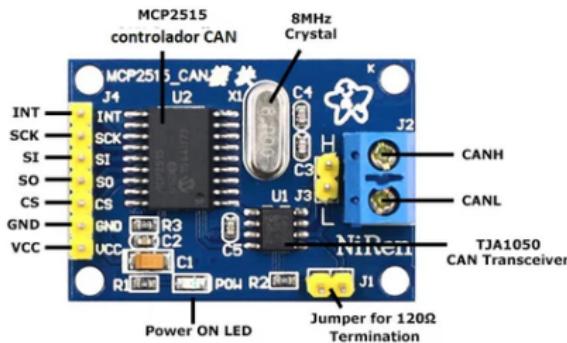


Figura 3.1: Módulo de comunicación BUS CAN

- *¿Qué observamos?*

Dentro del módulo podemos distinguir entre el controlador CAN MCP2515, pero además existe otro integrado mas que se denomina Transceiver. Este integrado se encarga de adaptar la señales desde TTL (0V a 5V o 3.3V) a señales diferenciales (**CANH** y **CANL**) propias del bus físico. En pocas palabras funciona como un puente entre el controlador CAN y el bus físico. En este caso el Transceiver es el **TJA1050**, también puede ser el MCP2551.

Otro punto a destacar es la resistencia de terminación de  $120\Omega$  que aparece cuando se puentea el jumper. Será requerido para los módulos de los extremos (en caso de tener 3 o más), y permitirá evitar la reflexión de las señales.

# Características

- Implementa CAN V2.0B a 1Mb/s y puede funcionar en modo standard o extendido.
- **Buffers de recepción, máscaras y filtros:** posee dos buffers de recepción RXB0 y RXB1, 6 filtros de 29 bits y 2 máscaras de 29 bits.  
*¿Cuál es la ventaja de poseer estos filtros y máscaras?*
- **Buffers de transmisión:** posee tres buffers de transmisión TXB0,TXB1,TXB2 con prioridades configurables.
- **Interfaz SPI:** soporta velocidades de hasta 10 MHz.
- **Interrupciones:** posee un pin físico destinado a interrupciones por buffers llenos, errores en transmisión o wake-up.
- **Bajo consumo de energía:** opera típicamente de entre 2.7 a 5.5 V con una corriente de 5 mA.

# Pinout del MCP2515

Se muestra a continuación el pinout del integrado:

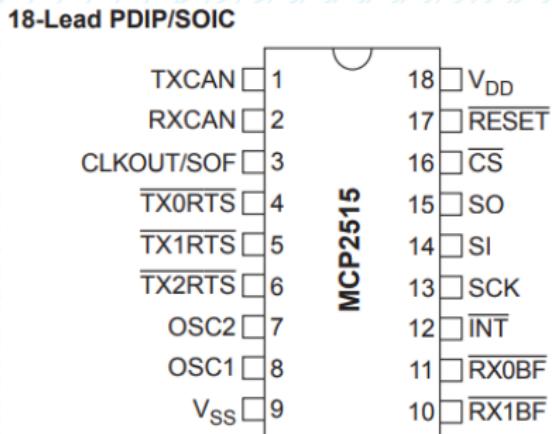


Figura 3.2: Pinout del integrado de 18 pines.

# Diagrama en bloques del controlador

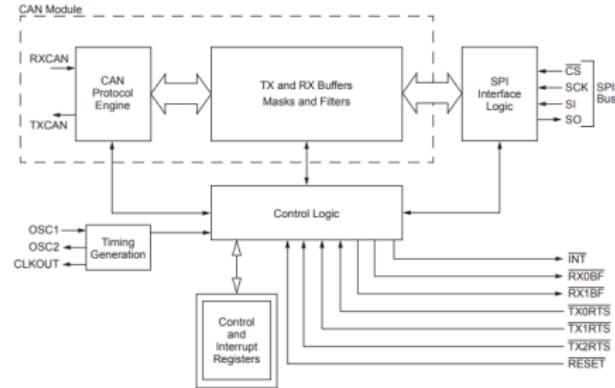


Figura 3.3: Diagrama en bloques del funcionamiento.

El dispositivo consta de tres bloques principales:

- El bloque del módulo CAN.
- El bloque del controlador lógico y registros que son utilizados para configurar el dispositivo.
- El bloque de protocolo SPI.

# Bloque CAN

## *¿Cuál es la función del bloque CAN?*

Se encarga de manejar todas las funciones de transmisión y recepción de mensajes CAN. Los mensajes de transmisión son cargados al primer buffer TXBn vacío. Todos los mensajes CAN son chequeados y emparejados a los filtros para determinar si deben ser cargados al buffer de recepción RXBn.



# Bloque lógico

## *¿Cuál es la función?*

Controla la configuración y operación del MCP2515 al utilizarse como interfaces de bloques para pasar información. Consta de un pin de interrupción multipropósito.

Para mas información puede verse la Tabla 1.1 de la hoja de datos del integrado [1].



# Buffers y Protocol Engine

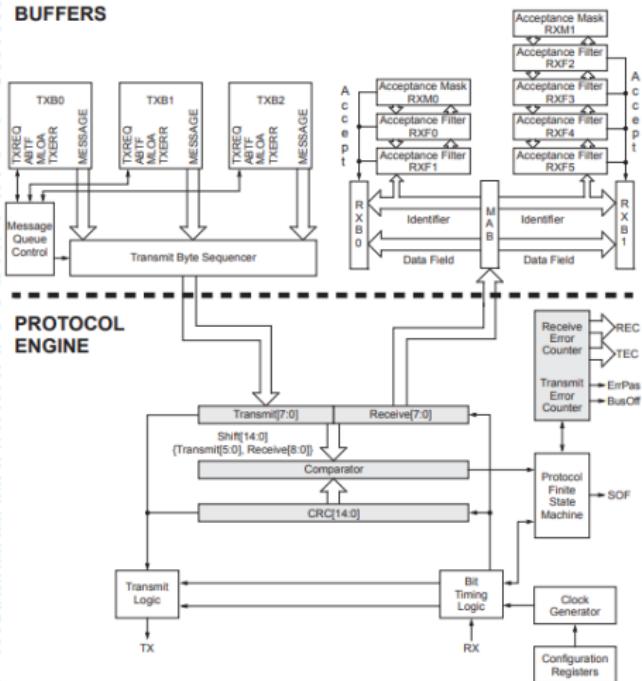


Figura 3.4: Diagrama en bloques de los buffers y el protocol engine.

- *¿Qué destacamos en el diagrama en bloques de los buffers?*
  - ① Para la transmisión vemos todos los mensajes pasan por una cola de datos (o buffer circular) que luego ingresa a un shift register.
  - ② Por el lado de la recepción tenemos los filtros RXF[0:5] y las máscaras RXM[0:1].
- *y ... ¿respecto al protocol engine?*
  - ① El protocolo de maquina de estado finita (FSM), ver sección 1.5.1 de [1].
  - ② Registro de CRC.
  - ③ Buffers de Tx y Rx.

# Modos de operación

El MCP2515 puede trabajar en cuatro modos de operación distintos, dicho modos fueron implementados en la librería. Se presentan a continuación los modos de operación:

- ① **Modo Normal:** modo estándar de comunicación CAN.
- ② **Modo Sleep:** modo de ahorro de energía. Despierta cuando ocurre alguna actividad en el bus CAN o cuando se setea WAKIF (ver REGISTER 7-2 de [1]) mediante SPI.
- ③ **Modo de Solo Escucha:** solamente puede recibir mensajes (incluidos mensajes con error) e independientes de los filtros y máscaras. Puede ser utilizado para monitorizar el bus o detectar el baud rate. No pueden transmitirse mensajes en este modo.



- ④ **Modo Loopback:** se utiliza para simular una comunicación CAN en el mismo integrado. Puede ser utilizado para realizar un test de funcionamiento. Los mensajes no se transmiten al bus físico.



# Preguntas?



# Transmisión

- ① **Prioridad de Transmisión:** Dentro de los tres buffers que posee el MCP2515 se transmitirá aquél que tenga mayor prioridad. Si dos buffers tienen la misma prioridad se pondrá el más alto. Por ejemplo si el buffer 0 y el 1 tienen la misma prioridad, el buffer 1 será el primero en transmitir.

*¿Cómo modifico la prioridad?*

② **Flujo de Transmisión:**

- ① Se carga la información en el buffer correspondiente.
- ② Se setea el bit TXREQ.
- ③ Se transmite el mensaje CAN al bus (cuando se tenga el arbitraje del mismo).
- ④ Si se encuentra habilitado, se activará una interrupción de mensaje enviado correctamente.

Se puede observar en detalle el diagrama de flujo de la figura 3-1 en [1].

**REGISTER 3-1: TXBnCTRL: TRANSMIT BUFFER n CONTROL REGISTER  
(ADDRESS: 30h, 40h, 50h)**

U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
—	ABTF	MLOA	TXERR	TXREQ	—	TXP1	TXPO
bit 7							bit 0

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>Unimplemented:</b> Read as '0'
bit 6	<b>ABTF:</b> Message Aborted Flag bit 1 = Message was aborted 0 = Message completed transmission successfully
bit 5	<b>MLOA:</b> Message Lost Arbitration bit 1 = Message lost arbitration while being sent 0 = Message did not lose arbitration while being sent
bit 4	<b>TXERR:</b> Transmission Error Detected bit 1 = A bus error occurred while the message was being transmitted 0 = No bus error occurred while the message was being transmitted
bit 3	<b>TXREQ:</b> Message Transmit Request bit 1 = Buffer is currently pending transmission (MCU sets this bit to request message be transmitted – bit is automatically cleared when the message is sent) 0 = Buffer is not currently pending transmission (MCU can clear this bit to request a message abort)
bit 2	<b>Unimplemented:</b> Read as '0'
bit 1-0	<b>TXP[1:0]:</b> Transmit Buffer Priority bits 11 = Highest message priority 10 = High intermediate message priority 01 = Low intermediate message priority 00 = Lowest message priority

**Figura 3.5: Registro TXBnCTRL.**

- **Características:**

- Posee dos buffer de recepción **RXB0** y **RXB1** y un buffer adicional **MAB** *¿que función cumple este último?*.
- Cada buffer contiene los siguientes registros: **SIDL-SIDH reg.**, **EID8-EID0 reg.**, **DLC reg.**, **DATA BYTE reg.**, **Control reg..**  
Pueden verse en [1] pagina 27 a 32.

- **Filtros y máscaras:** El MCP2515 permite configurar filtros y máscaras por hardware. El buffer **RXB0** posee dos filtros y el **RXB1** posee cuatro filtros. Además de una máscara cada uno. *¿Porqué son importantes?*

- Permite reducir la carga al microcontrolador.
- Si el mensaje no supera la máscara o filtro queda directamente descartado por lo que no se carga en ningún buffer.

## Ejemplo de máscara y filtro

Supongamos que quisiera recibir solo mensajes que comiencen con 0x5, por lo que aceptaría mensajes de 0x500 a 0x5FF. Asumiendo el formato estándar de 11 bits, configuramos la máscara para que el filtro solo “preste” atención a los primeros 3 bits: 11100000000 (0x700). Por último configuramos el filtro como sigue: 10100000000 (0x500).



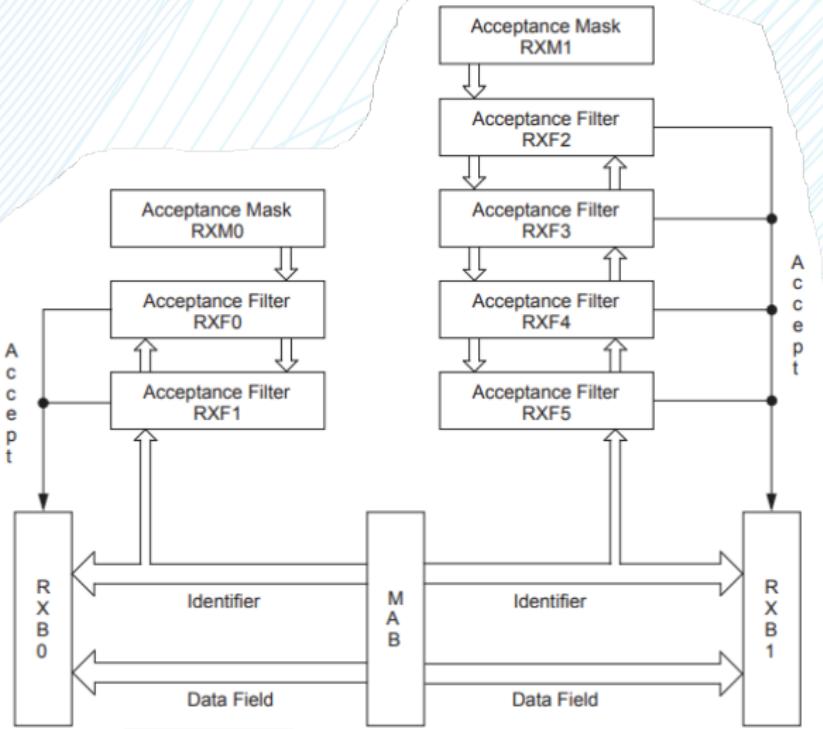


Figura 3.6: Diagrama en bloques de los buffers de recepción

# Flujo de Recepción

¿Cómo es la recepción el mensaje?

- ① **Inicio de mensaje:** detecta el comienzo de un mensaje a través del bit de SOF (Start Of Frame). *¿Cómo funciona?*

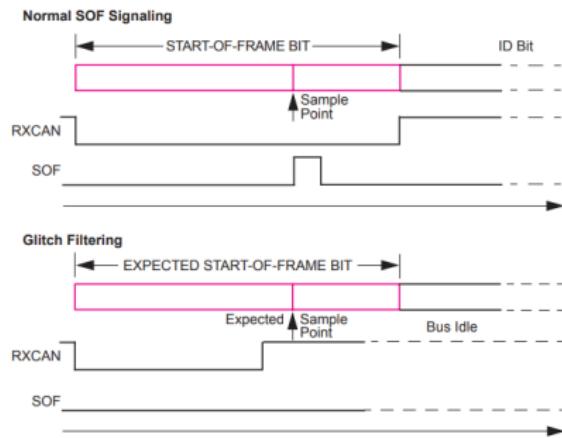


Figura 3.7: Señal de Comienzo de trama

- ③ **Carga mensaje en MAB:** se carga el mensaje en el tercer buffer de recepción que determina donde debe cargar el mensaje en base a las máscaras y filtros.
- ④ **Almacenamiento:** carga el mensaje en **RXB0** o **RXB1**, *¿qué sucede si alguno o ambos están llenos?*. Puede verse en profundidad el diagrama de flujo en la figura 4-3 de [1].
- ⑤ **Interrupción:** Si el mensaje pudo ser cargado en algún buffer lanza una interrupción (**RX0IF**,**RX1IF**), al igual que si no pudo ser cargado en ninguno debido al espacio (**RX0OVR**,**RX1OVR**).

# Tipos de errores

El MCP2515 provee tres estados de errores distintos, que se nombran a continuación:

- **Error active:** modo normal de trabajo donde el integrado puede enviar mensajes (incluidos mensajes de error).
- **Error pasive:** se transmitirán mensajes y tramas de error pasivos.
- **Bus-off:** durante este estado queda totalmente desconectado del bus.

*¿Qué tipo de errores detecta?*

- CRC error
- Bit error
- ACK error
- Stuff bit

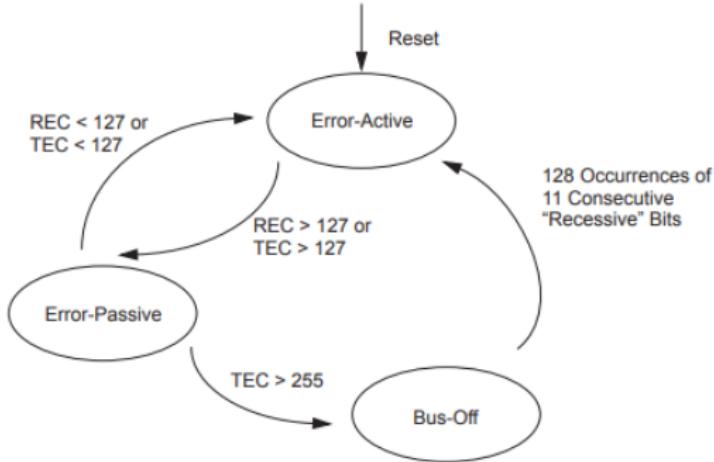


Figura 3.8: Diagrama de estado de errores.

# Preguntas?



## Interrupciones

Posee varios tipos de interrupciones que se relacionan con lo mencionado hasta el momento, partiendo desde la transmisión, pasando por la recepción y errores. Posee en principio dos registros: **CANINTE**, **CANINTF**. La última letra simboliza Enable y Flag, respectivamente.

Las interrupciones que posee son las siguientes:

- **Transmit Interrupt:** Se lanza cada vez que envía el mensaje y queda libre el buffer de dicho registro de transmisión. Se posee una interrupción para cada buffer de transmisión.
  - **Receive Interrupt:** Es generado una vez que el mensaje es correctamente cargado en el buffer de recepción correspondiente. Se configura para cada buffer de recepción por separado.
  - **Message Error Interrupt:** Ocurre cuando se detecta un error tanto en transmisión como recepción.

- **Bus Activity Wake-up Interrupt:** cuando el dispositivo se encuentra en modo sleep y se detecta alguna actividad en el bus se lanza dicha interrupción. Útil para cuando se quiere ahorrar energía.
- **Error Interrupt:** Se dará cuando ocurra un overflow en recepción, o algunos de los errores visto en la sección anterior.

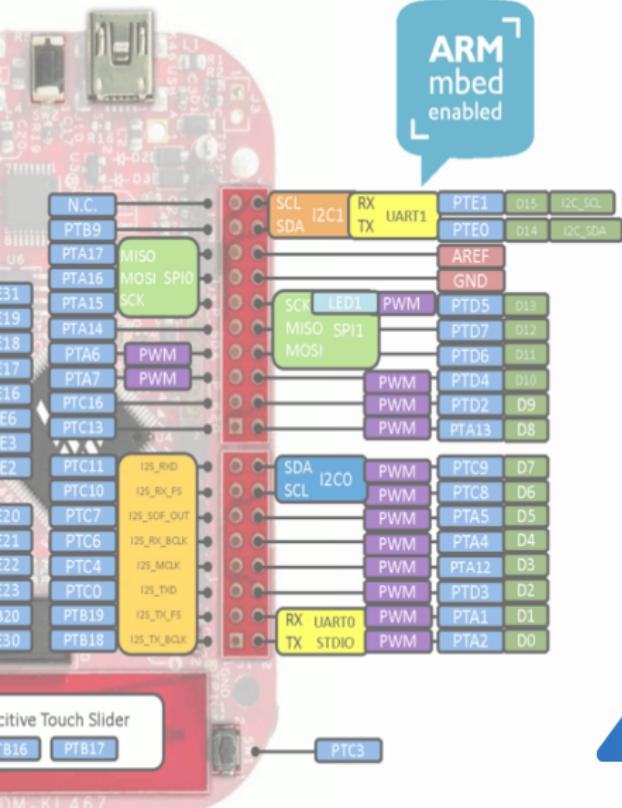


# Ejemplo de 3 Nodos

A modo de ejemplo presentaremos una comunicación tipo CAN bus, tenemos presentes tres nodos que cumplen las siguientes funciones:

- ① **Nodo 1:** Actúa como productor, leyendo el estado del sensor de luz conectado a la placa de desarrollo y se envía al bus con un  $ID = 10$ .
- ② **Nodo 2:** Actúa como productor, enviando el estado de algunos de sus GPIOs con un  $ID = 20$  y como consumidor leyendo la información del Nodo 1.
- ③ **Nodo 3:** Actúa como consumidor, leyendo la información del Nodo 2 y mostrándola por pantalla.





SD2

# GRACIAS

amzuliani02@gmail.com

# Referencias



Microchip Technology Inc.

*MCP2515: Stand-Alone CAN Controller with SPI Interface, 2005.*  
Datasheet.