Comunicación via CAN-bus mediante un arduino

En esta guía se describe el funcionamiento del programa para que un arduino sea capaz de enviar mensajes numéricos por un bus CAN.

Conocimientos previos básicos.

·Arduino
·Lenguage C y/o C++

Una idea básica de esto permitirá una comprensión más rápida, aunque se intentará explicar lo máximo posible.

Componentes de Hardware

·Arduino:

> El Arduino es una placa de microcontrolador de código abierto basado en el microchip ATmega328P y desarrollado por Arduino.CC. La placa está equipada con conjuntos de pines de E/S (Entrada/Salida) digitales y analógicas que pueden conectarse a varias placas de expansión y otros circuitos. La placa es programable con el Arduino IDE (Entorno de desarrollo integrado) a través de un cable USB tipo B. Puede ser alimentado por el cable USB o por una batería externa de 9 voltios, aunque acepta voltajes entre 7 y 20 voltios. El diseño de referencia de hardware se distribuye bajo una licencia Creative Commons Attribution Share-Alike 2.5 y está disponible en el sitio web de Arduino. Los archivos de diseño y producción para algunas versiones del hardware también están disponibles.

·MCP2515:

> El módulo CAN MCP2515 permite comunicar tus proyectos basados en Arduino con tu automóvil mediante el protocolo CAN. CAN (siglas del inglés Controller Area Network) es un protocolo de comunicaciones desarrollado por la empresa alemana Robert Bosch GmbH, basado en una topología bus para la transmisión de mensajes en entornos distribuidos. El bus CAN es ideal para aplicaciones de autotrónica, mecatrónica automotriz, automatización industrial, domótica y más.
>
> El módulo permite recibir y enviar frames de datos en formato estándar y extendido. El controlador CAN posee mascaras y filtros de acceso, reduciendo asi la carga del microcontrolador principal. Incluye el chip controlador CAN MCP2515 con interfaz SPI y el chip transceiver CAN TJA1050 con el fin de facilitar la comunicación con microcontroladores y tarjetas de desarrollo. Permite implementar una red de microcontroladores con la topología bus (ahorrando cable) con una longitud de hasta 1200 metros. El cable puede ser tipo par trenzado UTP Cat5/Cat6, cada terminación de nodo debe tener una resistencia de 120 Ohms.

Programa principal.

Este programa utiliza la librería <SPI.h> y también la <mcp2515.h> (MCP a partir de aquí), esta última se puede localizar en el repositorio de GitHub mediante este enlace: https://github.com/autowp/arduino-mcp2515

```
#include <SPI.h>
#include <mcp2515.h>

#define DEBUG(a) Serial.println(a)
#define ID 123

struct can_frame canLectura;
struct can_frame canEscribe;
MCP2515 mcp2515(10);
```

Lo primero que se realiza es incluir las librerías. A continuación se define un DEBUG, que lo que hace es imprimir por el puerto serie lo que necesitemos y se define la ID como 123 (en versiones posteriores esto variará ya que se podrá cambiar la ID sin necesidad de variar el programa). Por último, se crean las estructuras canLectura y can_Escritura con una forma can_frame. Una "estructura" es como una "clase" de python. El MCP2515 mcp2515(10) indica que el pin CS del MCP se conecta al pin 10 del Arduino.

En el setup se inicia el puerto serie a una velocidad de 115200 b/s, ya que es la más cercana a los 125 kb/s del CAN que se utilizan en estos momenetos. Para finalizar, se inicia el CAN con la funciones .reset, .setBitrate y .setNormalModede la librería MCP.

```
void setup()
{
Serial.begin(115200);

mcp2515.reset();
mcp2515.setBitrate(CAN_125KBPS);
mcp2515.setNormalMode();
}
```

```
void loop()
{
  if (mcp2515.readMessage(&canLectura) == MCP2515::ERROR_OK)
  {
    lectura();
  }

  if (Serial.available())
  {
    Serial.println("Leyendo");
    escritura();
  }
}
```

En este programa empleando la librería MCP se comprueba si hay algún mensaje por el bus CAN y llama a la función "lectura" que se verá más adelante.

A continuación, se comprueba si por el puerto serie del Arduino hay algún mensaje y, en caso afirmativo, llama a la función "escritura" que ya se tratará.

Estas dos funciones se repiten en un bucle infinito mientras el arduino está enchufado y conectado a la corriente.

Lectura.

```
void lectura()
{
  Serial.println("------- CAN Read ----------");
  Serial.println("ID \t DLC \t  DATA");

  Serial.print(canLectura.can_id, HEX); // print ID
  Serial.print("\t");

  Serial.print(canLectura.can_dlc, HEX); // print DLC
  Serial.print("\t");

  for (int i = 0; i<canLectura.can_dlc; i++) // print the data
  {
    Serial.print(canLectura.data[i],HEX);
    Serial.print(" ");
  }

  Serial.println();
}
```

Al llamar a esta función con el "if(mcp2515.readMessage(&canLectura)==MCP2515::ERROR_OK)" del 'loop', se lee el mensaje (si existe) y se guarda en la localización de canLectura. Después, utilizando Serial.print's se escriben por el puerto serie la ID (.can_id), el DLC (.can_dlc) y los datos que se leyeron (.data[]).

Escritura.

```
void escritura()
{
  uint16_t entrada = Serial.parseInt();
  //DEBUG(entrada);

  canEscribe.can_id = ID;
  canEscribe.can_dlc = 2;

  canEscribe.data[0] = (entrada/256, HEX);
  //DEBUG((entrada/256,HEX));
  canEscribe.data[1] = (entrada%256, HEX);
  //DEBUG((entrada%256,HEX));


  mcp2515.sendMessage(&canEscribe);
}
```

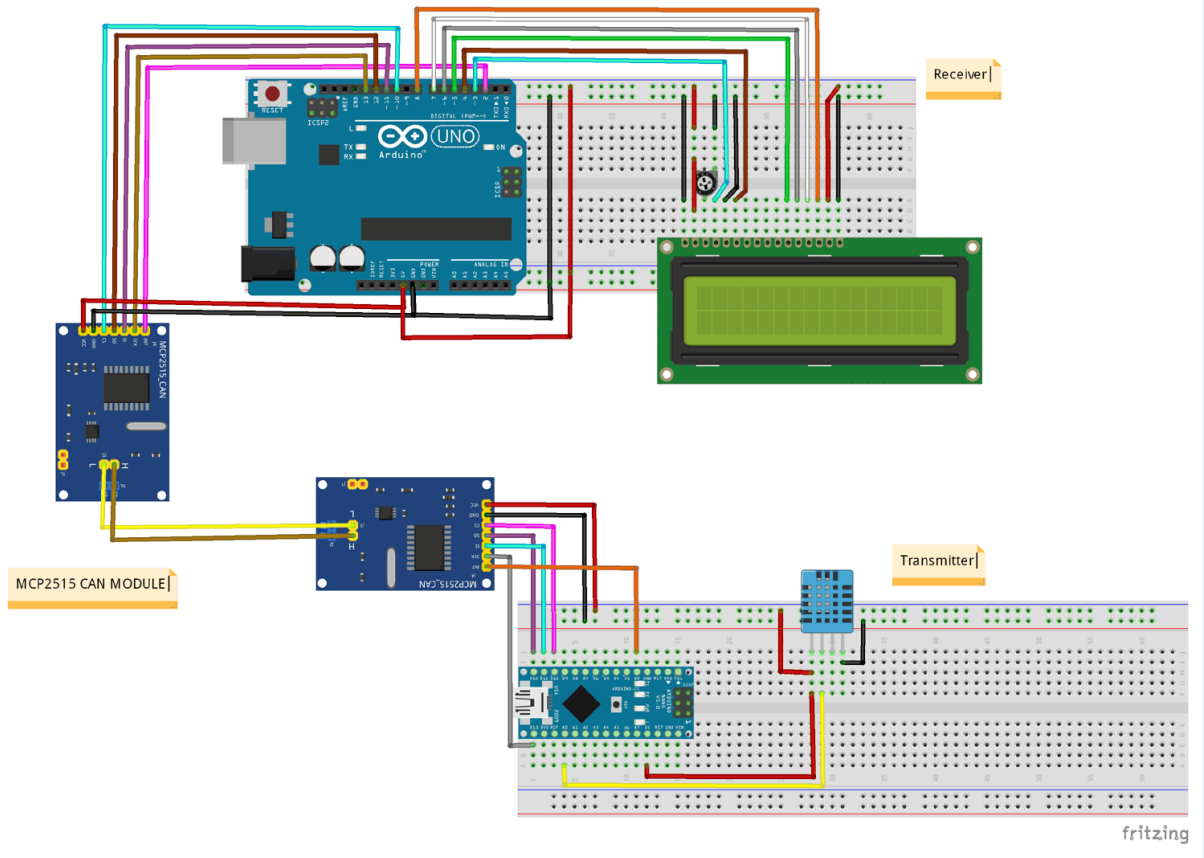
Esta función escribe en el bus CAN los mensajes numéricos (enteros sin signo) que le entren por el puerto serie.

El .parseInt() busca y guarda el primer entero que tenga el puerto serie en el buffer.

Se guardan la ID y el DLC y a continuación se 'encripta' el número que se desea mandar. Primero se guarda el resultado de la división entera, y a continuación el resto. Todo codificado en hexadecimal.

Se termina enviando el mensaje al MCP y del resto ya se encarga el.

Montage del arduino y el MCP2515



Ejemplo de montaje de un Arduino unoy un Nano con un MCP2515 entre otros dispositivos

Como se puede ver en la imagen, VCC y GND del MCP se conectan a los pines del Arduino 5V y GND respectivamente. El Int del MCP se conecta al pin 2 del Arduino. El SI, el SCO y el SCK se conectan al los pines 11, 12 y 13 respectivamente. Por último el CS aparece conectado al pin 10 del Arduino, pero se puede colocar en cualquier otro (De todas formas, aunque se pueda, no es recomendable).