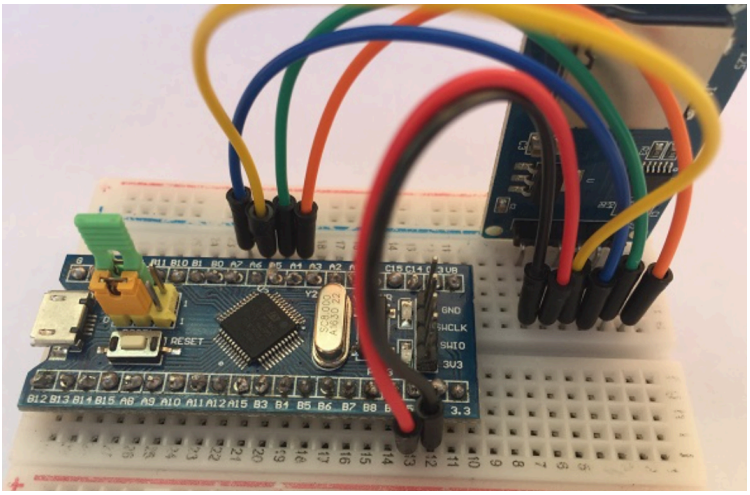
Documentación sd_nextion_lib

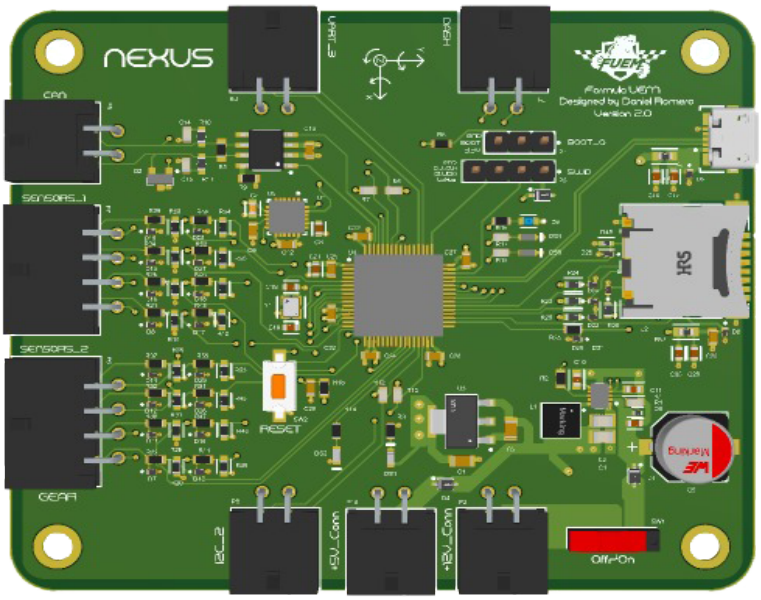
Por Daniel Gutiérrez Torres

Índice

1. Sobre la librería

Esta librería está diseñada para facilitar la escritura de datos en una tarjeta microSD utilizando una placa STM32. Los datos se guardan en formato CSV, lo que permite una fácil manipulación y análisis posterior. La librería también incluye un método para verificar la correcta escritura de los datos y comprobar la accesibilidad del documento, asegurando que los datos se han guardado y pueden ser leídos correctamente.





2. Precondiciones para utilizar la librería

Para utilizar la librería FATFS en tu proyecto, es crucial configurar adecuadamente el sistema de archivos. Primero, debes habilitar y configurar FATFS en el entorno de desarrollo siguiendo las instrucciones de la guía de instalación y configuración de la librería, además de añadir la declaración para el sistema de archivos en tu archivo principal (main.c) de tu proyecto.

```
FATFS fs;
```

La comunicación SPI también debe estar configurada correctamente, ya que se interactúa con los dispositivos de almacenamiento, a través de esta interfaz. Asegúrate de seleccionar y configurar el SPI adecuado en tu proyecto para garantizar una comunicación efectiva con los dispositivos de almacenamiento, además de añadir la declaración para el SPI utilizado en tu archivo principal (main.c) de tu proyecto.

```
/* Private variables ---------------------*/
SPI_HandleTypeDef hspi1;
```

Finalmente, es necesario montar el sistema de archivos utilizando el siguiente código en tu archivo principal (main.c) de tu proyecto. Este código asegura que el sistema de archivos FATFS esté correctamente montado y listo para su uso.

```
// Montar el sistema de archivos
if (f_mount(&fs, "", 0) != FR_OK) {
    // Manejar el error de montaje del sistema de archivos
    Error_Handler();
}
```

3. Funciones:

3.1- Función para guardar un dato en la tarjeta microSd

La función guarda datos en un archivo CSV en una tarjeta microSD. Para ello abre o crea un archivo llamado "data.csv" en modo de escritura y se posiciona al final para añadir nuevos datos. Luego, formatea el identificador `id`, el valor `value` y la marca de tiempo `timestamp` en una cadena y la escribe en el archivo. Si ocurre algún error durante la apertura, la escritura o el cierre, se maneja con `Error_Handler`.

Parámetros:

- id : id asociado al tipo de dato a almacenar
- valor : valor del dato a almacenar
- timestamp : fecha en la que se obtiene u almacena el dato en la microSD

Llamada a la función:

```
save_sd(1, "Value", "timestamp");
```

Ejemplo de uso:

Guardar el valor del sensor de temperatura del motor con id 1

```
uint8_t temp_motor_id = 1;
char* valor_temp_motor= "35°";
char timestamp[20];
get_current_timestamp(timestamp, sizeof(timestamp));

save_sd(temp_motor_id , valor_temp_motor, timestamp);
```

Ejemplo de fichero CSV generado por la función:

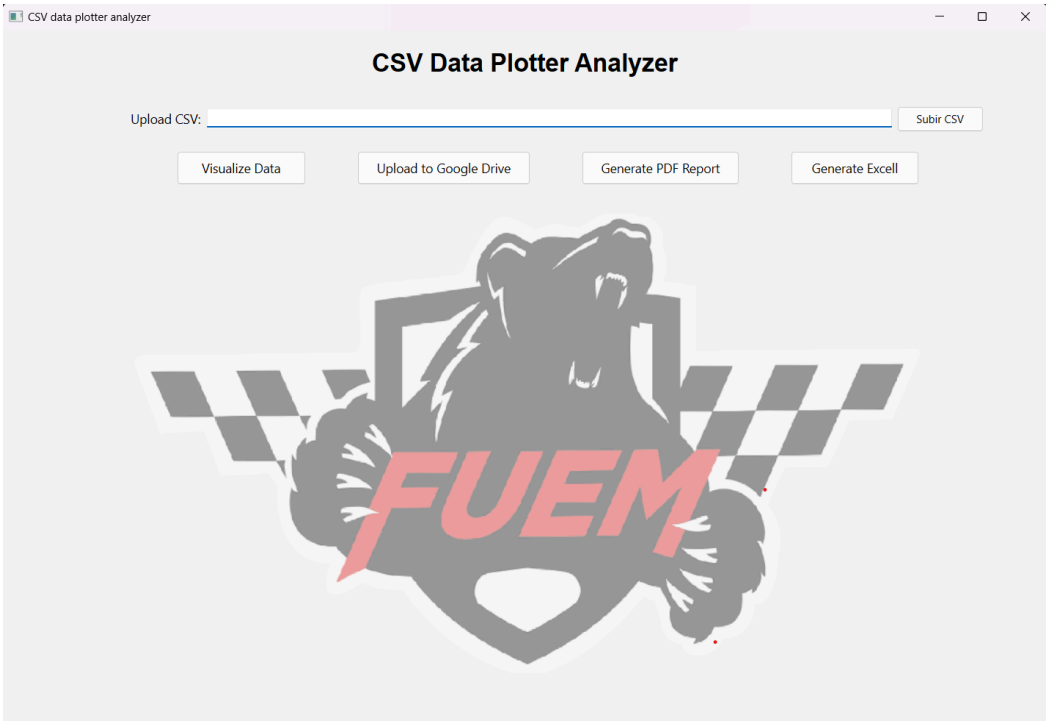
```
id,value,timestamp
4,30,2023-06-12 18:57:41
2,59,2023-06-18 08:28:49
7,9,2023-06-03 23:16:28
1,17,2023-06-18 14:35:37
6,64,2023-06-24 07:32:03
2,63,2023-06-18 19:47:30
7,84,2023-06-08 00:05:17
```

De esta forma, es posible recopilar de manera eficiente los datos provenientes de diversos sensores del coche y almacenarlos en un único archivo. Cada parámetro del sistema está asociado con un identificador único, como se detalla en la tabla de identificadores a continuación:

Parámetro	ID
engine_temp	1
Batery_volt	2
brake1_temp	3
brake2_temp	4
brake3_temp	5
brake4_temp	6
gear	7
speed	8

Utilizando estos identificadores y las marcas de tiempo correspondientes, los datos almacenados en el archivo CSV pueden ser graficados, analizados y procesados de manera efectiva. Esto permite obtener información valiosa sobre el rendimiento y el estado del vehículo, facilitando una evaluación detallada y una comprensión profunda del funcionamiento del coche.

Para facilitar el análisis de los datos, se ha desarrollado una aplicación que permite subir el archivo CSV generado, realizar un estudio estadístico y generar gráficas detalladas. Para utilizar esta herramienta, consulta la aplicación asociada disponible en el repositorio dentro de la carpeta CSV_data_plotter.



3.2- Función para verificar el correcto guardado de los datos en la tarjeta microSd

La función intenta abrir un archivo en modo de escritura para verificar si se puede escribir en él. Recibe el nombre del archivo como parámetro y usa `f_open` para abrir o crear el archivo con permisos de escritura. Si la apertura falla, devuelve 0. Luego, comprueba si el archivo tiene permisos de escritura. Si no los tiene, cierra el archivo y devuelve 0. Si el archivo se puede escribir, lo cierra y devuelve 1, indicando que el archivo puede ser abierto y escrito correctamente.

Todo esto permite manejar el encendido de un LED de control en la placa de desarrollo, encendiéndolo si el archivo puede ser escrito y es accesible o apagándolo en caso contrario.

Parámetros:

- nombre_fichero_csv : nombre del fichero CSV donde la librería almacena los datos dentro de la microSd.

Llamada a la función:

```
verificarSd("nombre_fichero_csv")
```

Ejemplo de uso:

```
if (verificarSd("data.csv") == 1) {
  printf("OK: Se pudo acceder a data.csv\n");
} else {
  // Manejar el error: archivo no disponible
  printf("KO: No se pudo acceder a data.csv\n");
}
```

4. Desarrollo de las funciones:

A continuación se muestran y desarrollan las diferentes funciones que la librería implementa.

4.1- Función para guardar un dato en la tarjeta microSd

La función `save_sd` guarda datos en un archivo CSV en una tarjeta SD. Abre (o crea) el archivo "data.csv" en modo de escritura y se posiciona al final para añadir nuevos datos. Luego, formatea el identificador `id`, el valor `value` y la marca de tiempo `timestamp` en una cadena y la escribe en el archivo. Si ocurre algún error durante la apertura, la escritura o el cierre, se maneja con `Error_Handler`.

```
void save_sd(int id, const char* value, const char*
timestamp) {

   FIL fil;  // Variable para guardar fivhero
   char buffer[256]; // Variable para guardar fila a insertar

   // Abrir el archivo
   if (f_open(&fil, "data.csv", FA_OPEN_ALWAYS |
   FA_WRITE) != FR_OK) {
       // Manejar el error de apertura o creación del archivo
       Error_Handler();
       return;
   }

   // Posicionarse al final del archivo para añadir datos
   if (f_lseek(&fil, f_size(&fil)) != FR_OK) {
       // Manejar error
       f_close(&fil) ;
       Error_Handler();
       return;
   }

   // Escribir el id, value y timestamp
   snprintf(buffer, sizeof(buffer), "%d, %s, %s\n", id,
   value, timestamp);
   if (f_puts(buffer, &fil) == EOF) {
       // Manejar error escritura
       f_close(&fil);
       Error_Handler();
       return;
   }

    // Cerrar el archivo
    f_close(&fil);
}
```

4.2- Función para verificar el correcto guardado de los datos en la tarjeta microSd

Esta función intenta abrir un archivo en modo de escritura para verificar si se puede abrir y escribir correctamente. Recibe el nombre del archivo como parámetro y utiliza la función `f_open` para intentar abrir el archivo con las banderas `FA_OPEN_ALWAYS` y `FA_WRITE`. Si la apertura del archivo falla, devuelve 0. Luego, comprueba si el archivo tiene el permiso de escritura habilitado. Si no es así, cierra el archivo y devuelve 0. Si el archivo se abre correctamente y tiene permisos de escritura, lo cierra y devuelve 1, indicando que el archivo puede ser abierto y escrito sin problemas.

```
int verificarSd(const char* filename) {
    FIL fil;
    FRESULT res;

    // Intentar abrir el archivo en modo de escritura
    res = f_open(&fil, filename, FA_OPEN_ALWAYS | FA_WRITE);
    if (res != FR_OK) {
        return 0;
    }

    // Verificar si el archivo está abierto y se puede escribir
    if ((fil.flag & FA_WRITE) == 0) {
        f_close(&fil);
        return 0;
    }

    // Cerrar el archivo y devolver 1 (todo ok)
    f_close(&fil);
    return 1;
}
```