



Manual de instalación y configuración sd_nextion_lib

Por Daniel Gutiérrez Torres



Índice

Paso 1: Descargar el repositorio de la biblioteca	3
Paso 2: Configuración de proyecto en STM32CubeIDE	3
2.1 Configuramos el System core	3
2.2 Establecer comunicación SPI	4
2.3 Configuración de FATFS	6
Paso 3: Creamos una carpeta donde referenciar las bibliotecas de nuestro proyecto	7
Paso 4: Referenciar el código fuente de la librería (.c)	8
Paso 5: Referenciar la cabecera de la librería (.h)	9
Paso 6: Modificación de archivos del proyecto	11
Paso 7: Incluir la biblioteca en el código main de nuestro proyecto	13

A continuación se describen el conjunto de pasos a seguir para la importación de nextion_communication_lib en la herramienta de desarrollo STM32CubeIDE.

Paso 1: Descargar el repositorio de la biblioteca

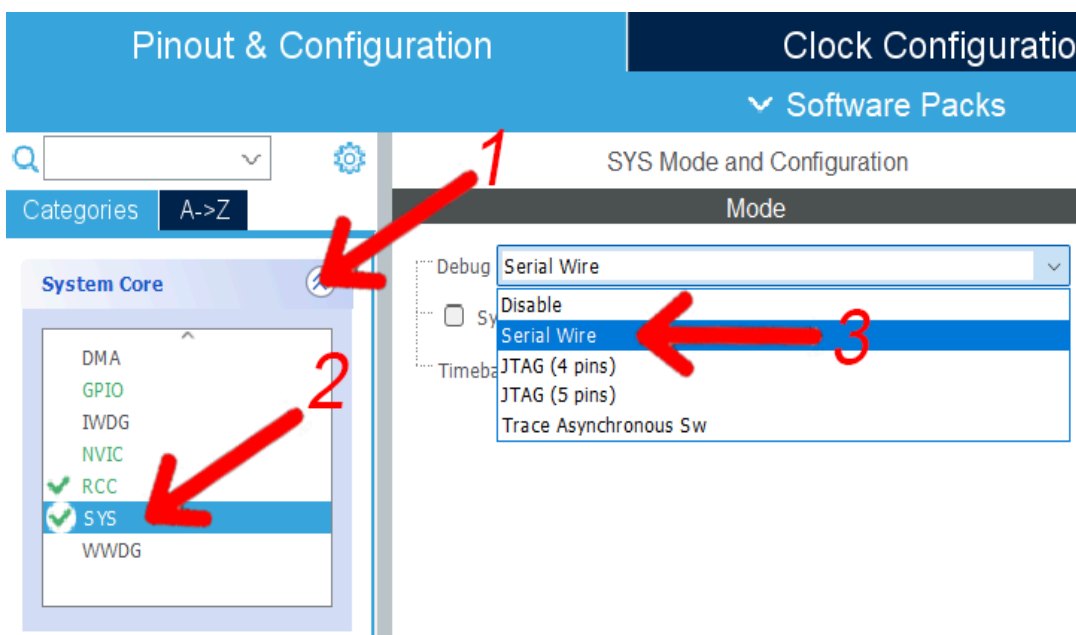
Puedes descargarla haciendo git clone del repositorio:

```
git clone https://github.com/guti10x/Sd_manager.git
```

Paso 2: Configuración de proyecto en STM32CubeIDE

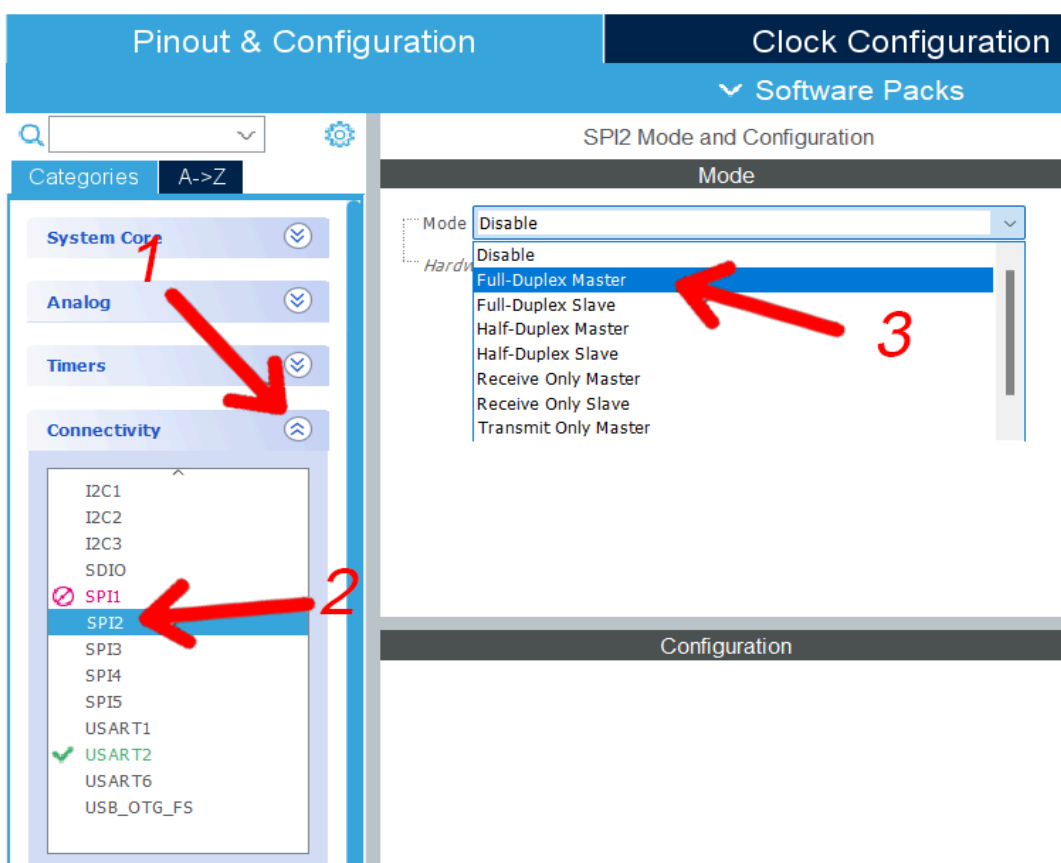
2.1 Configuramos el System core

En primer lugar hacemos click sobre “System Core” (1) para a continuación clicar sobre “SYS” (2), para finalmente seleccionar en el desplegable la opción de “Serial Wire”. Todo esto se realiza con el objetivo de habilitar la interfaz de depuración serial del microcontrolador para programar y depurar el microcontrolador de manera eficiente.

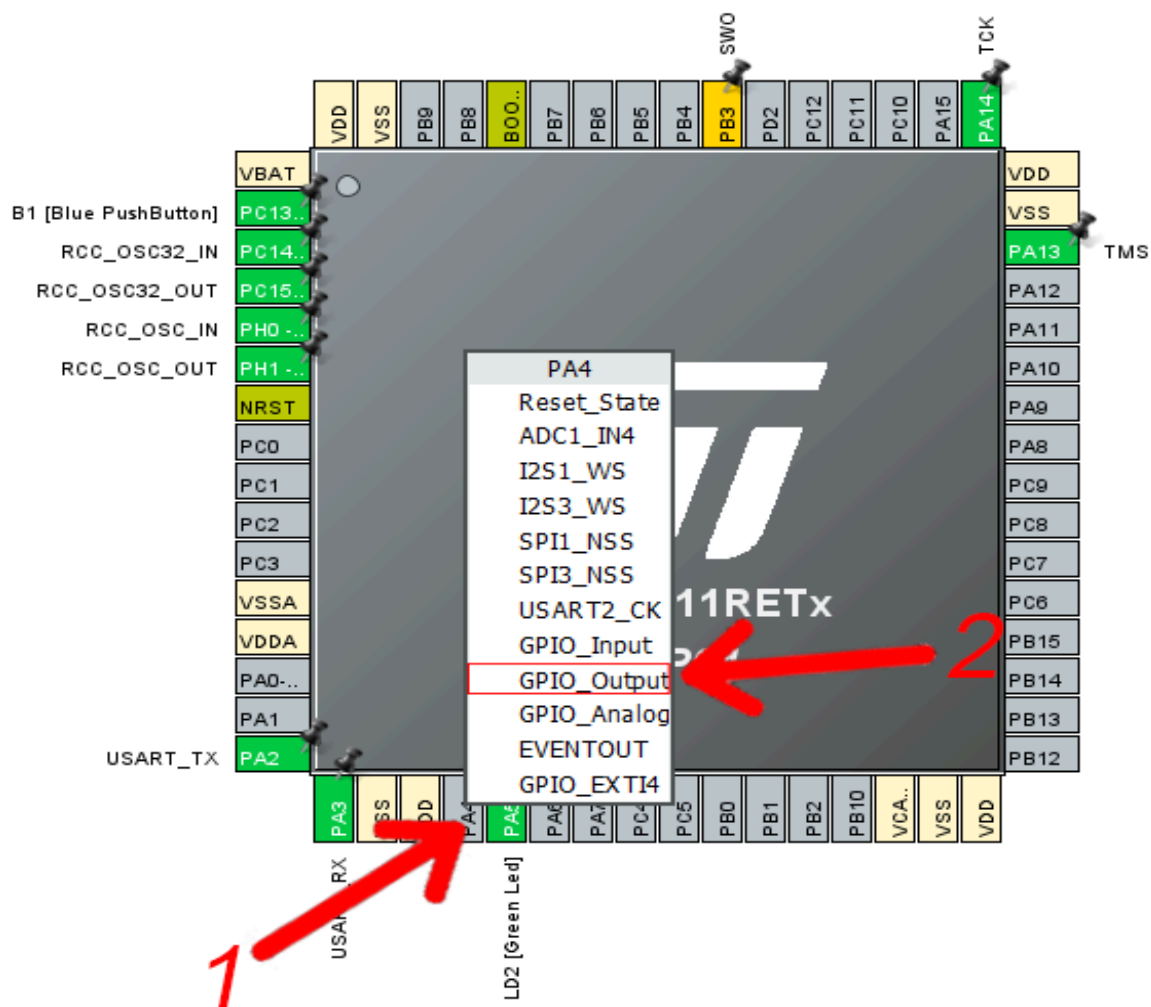


2.2 Establecer comunicación SPI

En primer lugar, hacemos clic sobre "Connectivity" (1) y a continuación seleccionamos el SPI que deseemos utilizar (teniendo en cuenta que, según la placa, SPI1 puede estar ya ocupado, en cuyo caso podemos usar SPI2) (2). Finalmente, establecemos el modo en el desplegable como "Full-Duplex Master". Esto lo hacemos para permitir la transmisión y recepción simultáneas de datos entre el microcontrolador y otros dispositivos, mejorando así la velocidad y eficiencia de la comunicación en aplicaciones que requieren un intercambio rápido y fiable de información, esencial en sistemas embebidos complejos.

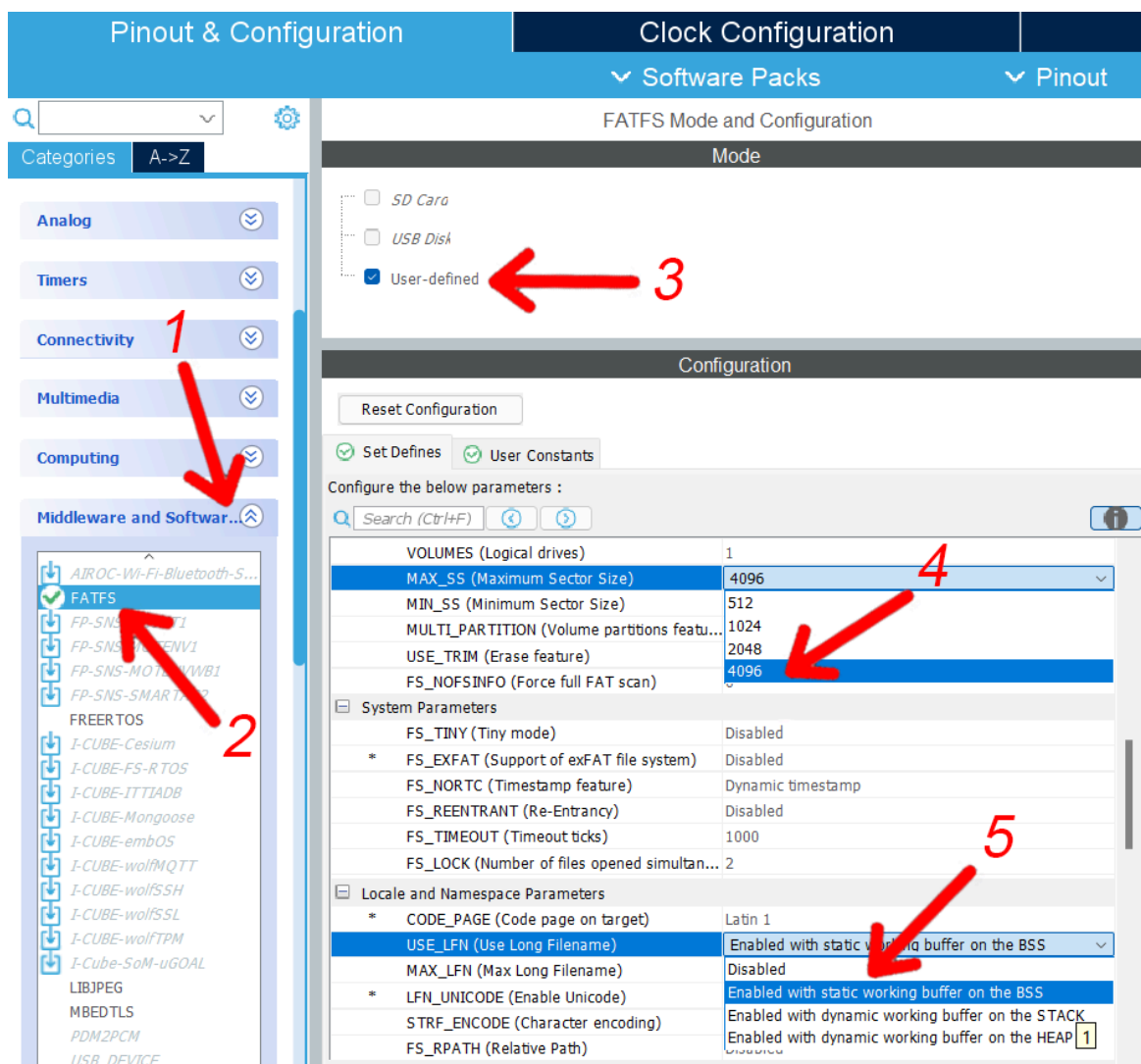


Además, configuramos el pin PA4 como "GPIO_Output" para su uso como señal de control o selección de chip.



2.3 Configuración de FATFS

A continuación, hacemos clic en "Middleware" (1), seleccionamos "FATFS" (2) y marcamos la opción "User-defined" (3). En "Configuration", ajustamos los parámetros en "Set Defines", habilitando "USE LFN" (Use Long Filename) como "Enabled with static working buffer on the BSS" (4) y configuramos "Max SS" (Maximum Sector Size) en 4096 (. Esto lo hacemos para permitir el manejo de archivos con nombres largos y sectores de mayor tamaño, mejorando la capacidad de almacenamiento y la eficiencia en la gestión de archivos en la tarjeta SD, lo cual es crucial para aplicaciones que requieren un almacenamiento flexible y eficiente de datos.



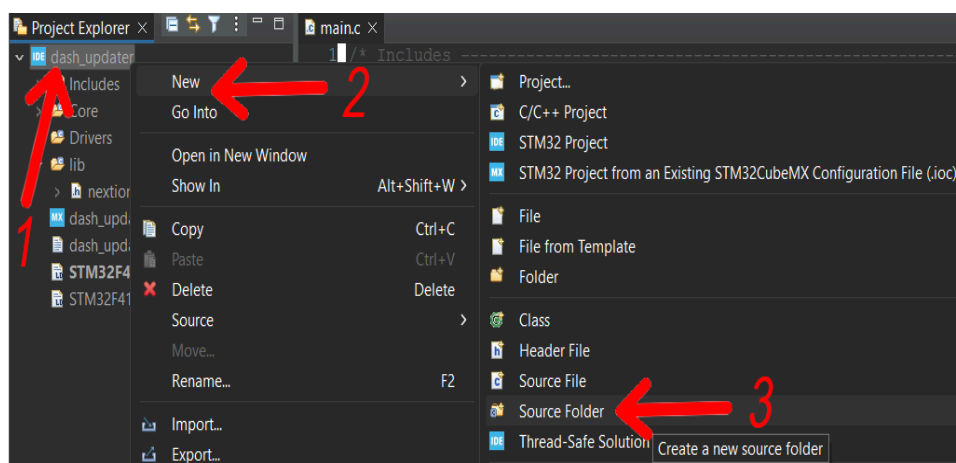
The screenshot shows the 'Pinout & Configuration' software interface. The left sidebar has a 'Middleware and Software' section expanded, showing a list of options. 'FATFS' is selected. The main area shows the 'FATFS Mode and Configuration' settings. Under 'Mode', 'User-defined' is selected. Under 'Configuration', the 'Set Defines' tab is active. The 'Configure the below parameters' table shows the following settings:

Parameter	Value
VOLUMES (Logical drives)	1
MAX_SS (Maximum Sector Size)	4096
MIN_SS (Minimum Sector Size)	512
MULTI_PARTITION (Volume partitions feature)	1024
USE_TRIM (Erase feature)	2048
FS_NOFSINFO (Force full FAT scan)	4096
System Parameters	
FS_TINY (Tiny mode)	Disabled
* FS_EXFAT (Support of exFAT file system)	Disabled
FS_NORTC (Timestamp feature)	Dynamic timestamp
FS_REENTRANT (Re-Entrancy)	Disabled
FS_TIMEOUT (Timeout ticks)	1000
FS_LOCK (Number of files opened simultaneously)	2
Locale and Namespace Parameters	
* CODE_PAGE (Code page on target)	Latin 1
USE_LFN (Use Long Filename)	Enabled with static working buffer on the BSS
MAX_LFN (Max Long Filename)	Disabled
* LFN_UNICODE (Enable Unicode)	Enabled with static working buffer on the BSS
STRF_ENCODE (Character encoding)	Enabled with dynamic working buffer on the STACK
FS_RPATH (Relative Path)	Enabled with dynamic working buffer on the HEAP

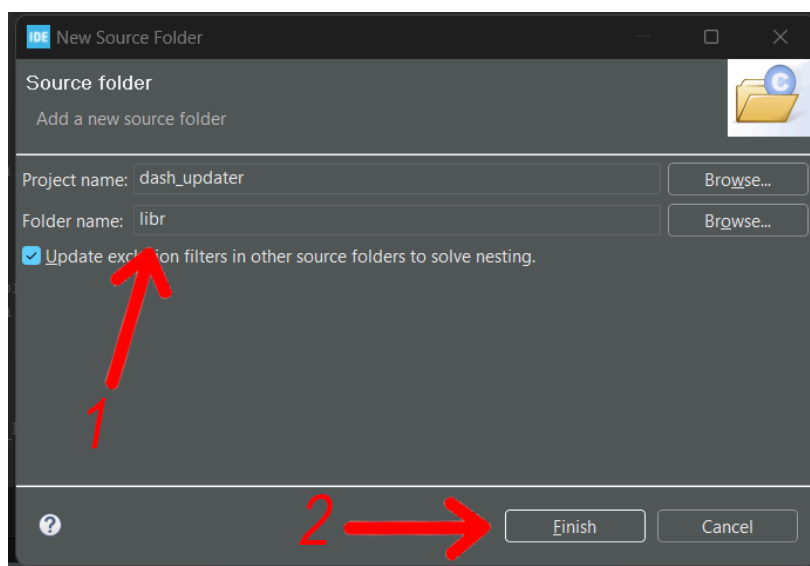
Paso 3: Creamos una carpeta donde referenciar las bibliotecas de nuestro proyecto

Este paso es opcional, ya que puedes reutilizar cualquier carpeta existente para almacenar las bibliotecas ya creadas y configuradas en tu proyecto, o bien incluir los archivos de encabezado y de código fuente de la librería directamente en las carpetas `Core/Src` y `Core/Inc` del proyecto.

3.1- Para ello hacemos click derecho sobre el proyecto (1) y posteriormente seleccionamos “New” (2) para finalmente clicar sobre “Source File” (3).

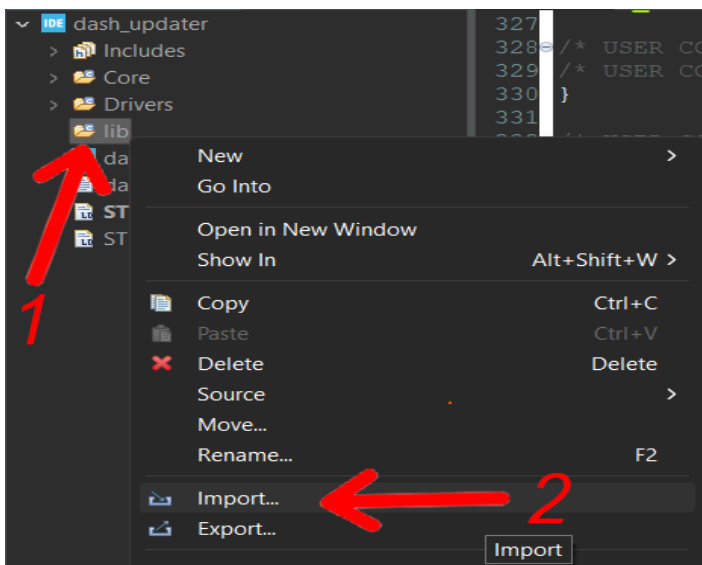


3.2- Definimos un nombre a la carpeta donde referenciamos la biblioteca, como en este caso por ejemplo “libr” (1) y a continuación le damos a “Finish” (2)

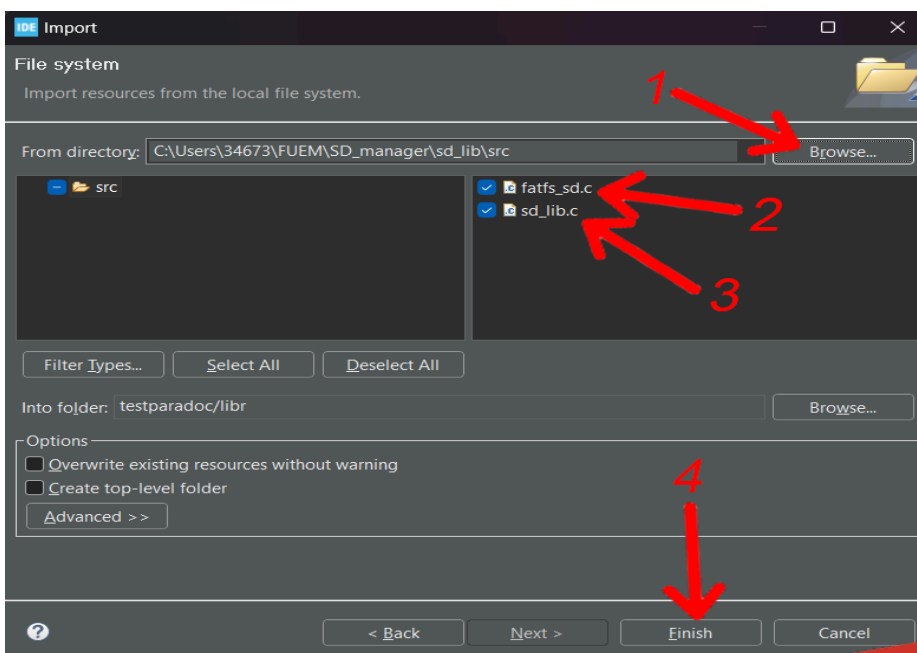


Paso 4: Referenciar el código fuente de la librería (.c)

4.1- Para ello hacemos click derecho sobre la carpeta creada “libr” o la que ya teníamos creada previamente(1) y posteriormente seleccionamos “Import” (2).

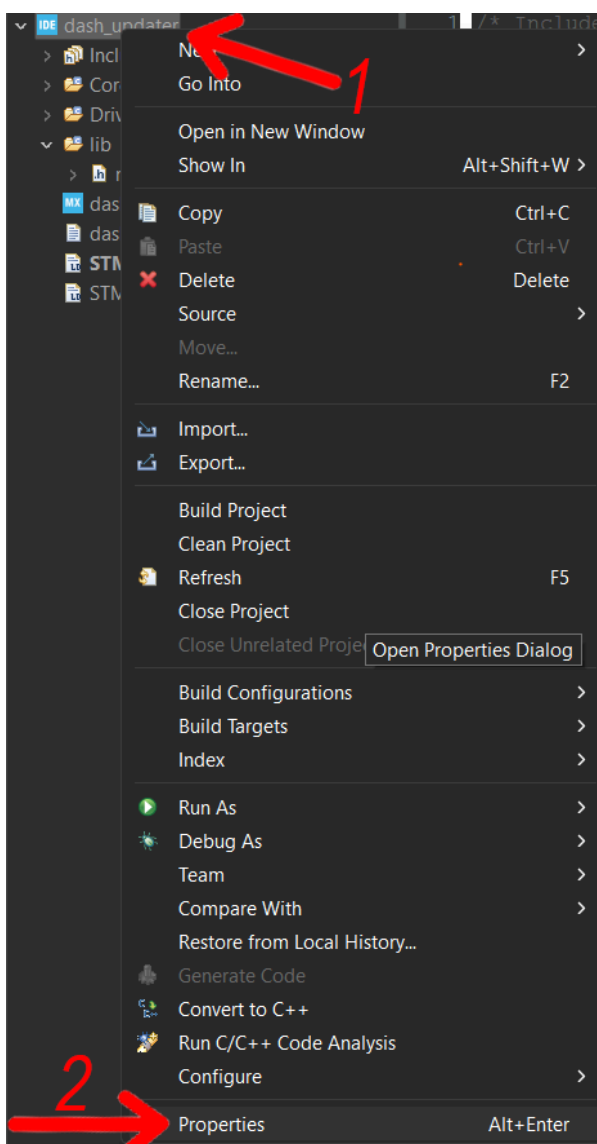


4.2- En segundo lugar hacemos click derecho sobre el botón Browse, buscamos en nuestro directorio local donde hemos guardado el repositorio de la biblioteca y seleccionamos la carpeta “src de la misma”(1). A continuación marcamos los checkboxes de la librería de “fatfs_sd”(2) y de “sd_lib.c” (3) y para terminar pulsamos en “Finish” (4).

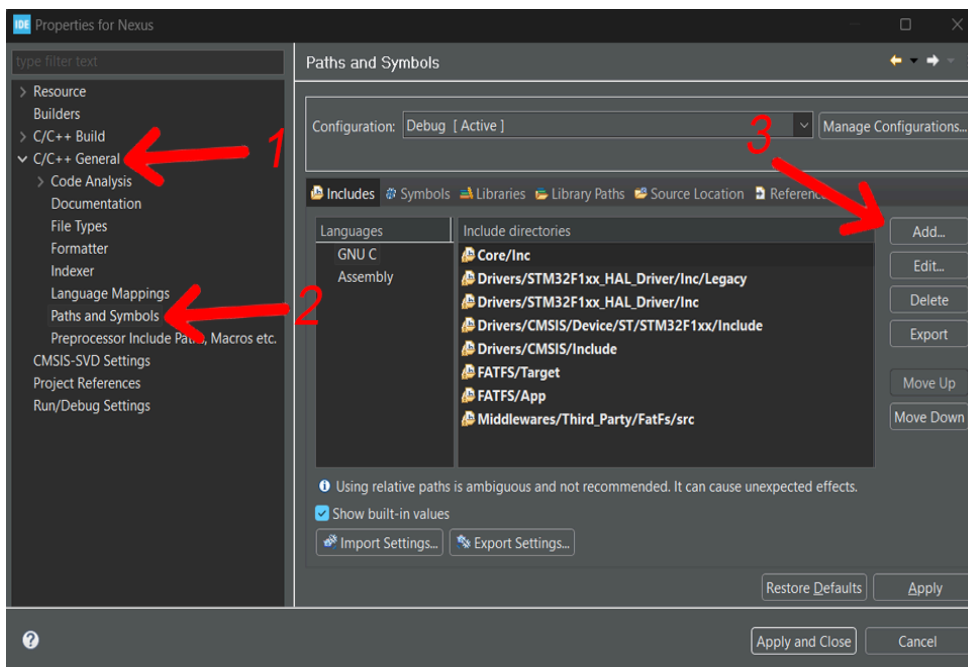


Paso 5: Referenciar la cabecera de la librería (.h)

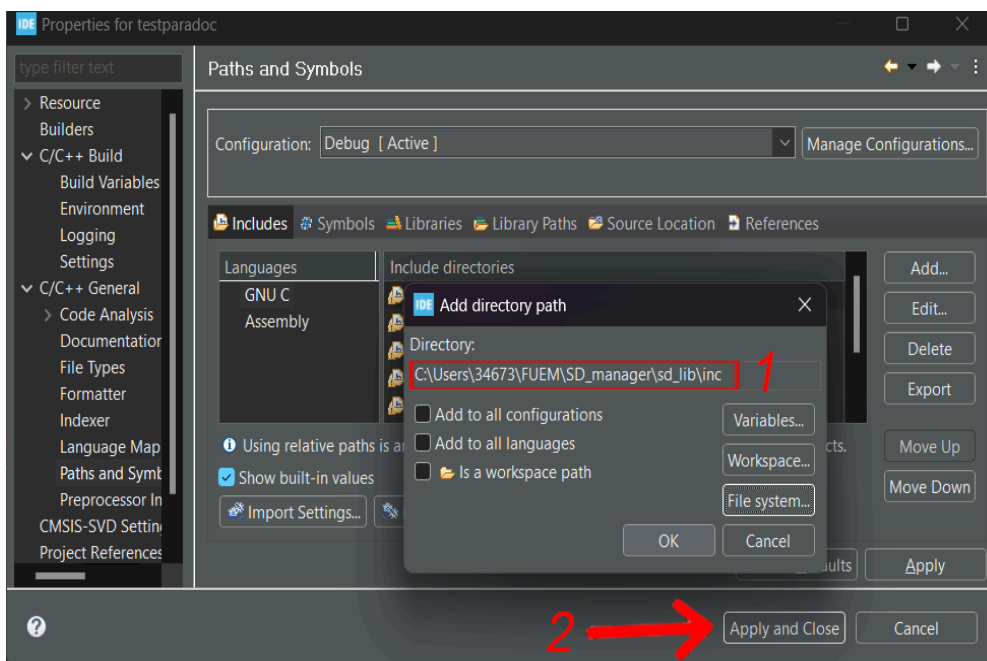
5.1- Para ello hacemos click derecho sobre el proyecto (1) y posteriormente seleccionamos “Properties” (2) .



5.2- A continuación hacemos primeramente click sobre “C/C++ General (1), en segundo lugar sobre “Paths and Symbols” (2) y finalmente sobre “Add” (3).



5.3- Hacemos click en el botón “File system” y buscamos en el explorador de archivos emergente la carpeta “inc” de la biblioteca, seleccionando la carpeta “inc de la misma”(2), para finalmente presionar en “Apply and Close” (2).



Paso 6: Modificación de archivos del proyecto

Para el correcto funcionamiento de la librería es necesario modificar las siguientes líneas de código de los siguiente archivos descritos a continuación

6.3 Modificación del archivo stm32f1xx_it.c

Este archivo se encuentra dentro de tu proyecto en la ruta:
“.\Core\Src\stm32f1xx_it.c”

6.1.1 En primer lugar añadir la declaración externa de las variables Timer1 y Timer2:

```
/* Private user code -----*/  
/* USER CODE BEGIN 0 */  
extern uint16_t Timer1, Timer2;  
/* USER CODE END 0 */
```

6.2.2 En segundo lugar modificar la función SysTick_Handler añadiendo el siguiente código para decrementar los valores de Timer1 y Timer2:

```
/**  
 * @brief This function handles System tick timer.  
 */  
void SysTick_Handler(void)  
{  
    /* USER CODE BEGIN SysTick_IRQn 0 */  
    if Timer1 > 0  
        Timer1--;  
    if Timer2 > 0  
        Timer2--;  
    /* USER CODE END SysTick_IRQn 0 */  
    HAL_IncTick();  
    /* USER CODE BEGIN SysTick_IRQn 1 */  
  
    /* USER CODE END SysTick_IRQn 1 */  
}
```

6.3 Modificación del archivo User_Diskio.c

Este archivo se encuentra dentro de tu proyecto en la ruta:
“.\FATFS\Target\user_diskio.c”

6.3.1 En primer lugar comentamos las siguientes dos líneas y las reemplazamos con la línea de inicialización del disco SD:

```
{  
/* USER CODE BEGIN INIT */  
// Stat = STA_NOINIT;  
// return Stat;  
return SD_disk_initialize pdrv ;  
/* USER CODE END INIT */  
}
```

6.3.2 En segundo lugar comentamos las siguientes dos líneas y las reemplazamos con la línea de verificación del estado del disco:

```
{  
/* USER CODE BEGIN STATUS */  
// Stat = STA_NOINIT;  
// return Stat;  
return SD_disk_status pdrv ;  
/* USER CODE END STATUS */  
}
```

6.3.3 En tercer lugar comentamos la siguientes línea y las reemplazamos con la línea de pa función para realizar la lectura de datos desde la tarjeta SD:

```
{  
/* USER CODE BEGIN READ */  
// return RES_OK;  
return SD_disk_read pdrv buff sector count ;  
/* USER CODE END READ */  
}
```

6.3.4 En cuarto lugar comentamos las siguientes dos líneas y las remplazamos con la línea de pa función para realizar la escritura de datos desde la tarjeta SD

```
{  
  /* USER CODE BEGIN WRITE */  
  /* USER CODE HERE */  
  // return RES_OK;  
  return SD_disk_write(pdrv, buff, sector, count);  
  /* USER CODE END WRITE */  
}
```

6.3.5 En quinto lugar comentamos las siguientes dos líneas y las remplazamos con la línea de la función que maneja los comandos de entrada/salida específicos para el disco SD

```
{  
  /* USER CODE BEGIN IOCTL */  
  // DRESULT res = RES_ERROR;  
  // return res;  
  return SD_disk_ioctl(pdrv, cmd, buff);  
  /* USER CODE END IOCTL */  
}
```

Paso 7: Incluir la biblioteca en el código main de nuestro proyecto

Incluimos dentro de los includes de nuestro proyecto el de la librería con el objetivo de poder referenciarla y poder invocar sus funciones.

```
/* USER CODE BEGIN Includes */  
#include "sd_nextion_lib.h"
```

A partir de este momento, podemos comenzar a utilizar las diferentes funciones que la librería contiene. Para obtener más información sobre las funciones y posibilidades de la librería, no dudes en consultar la documentación asociada.