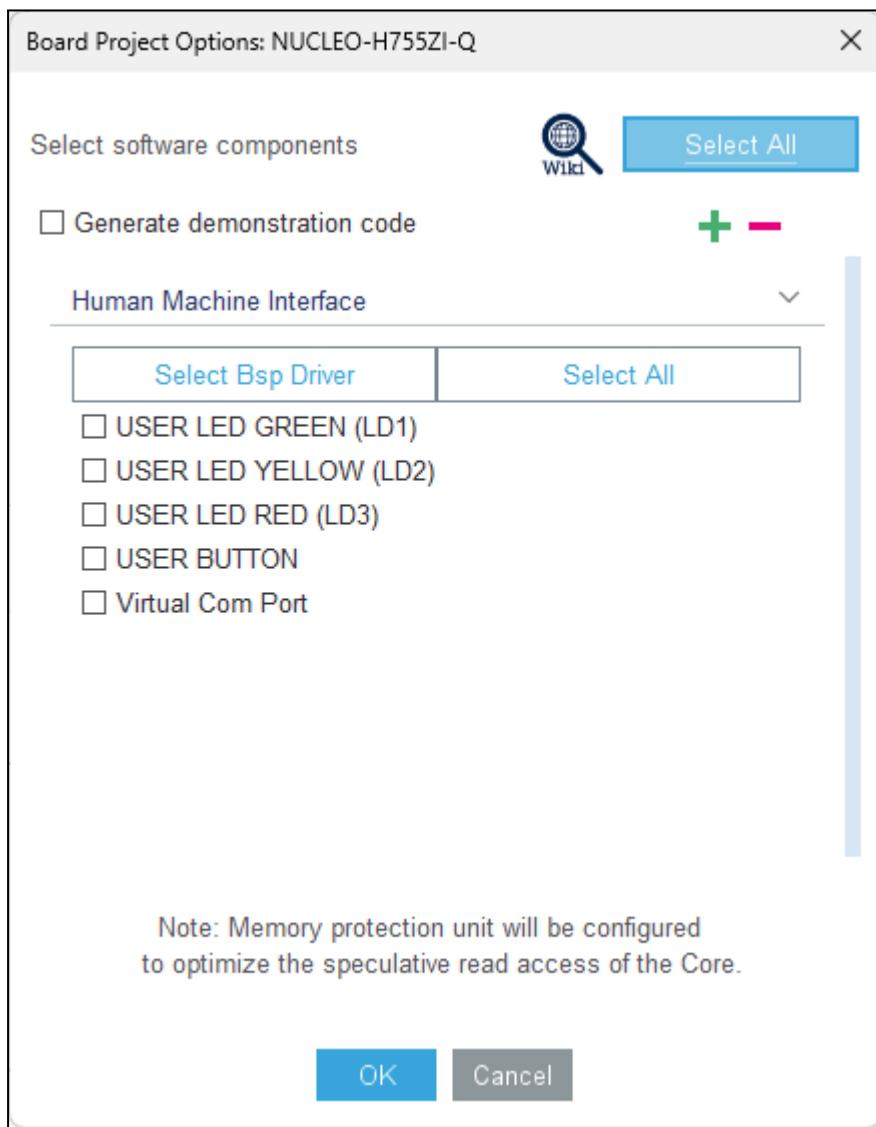


**11/05/25**

## **Guia para la creación de nuevo proyecto**

Se crea un proyecto limpio. Deseleccionar todo.



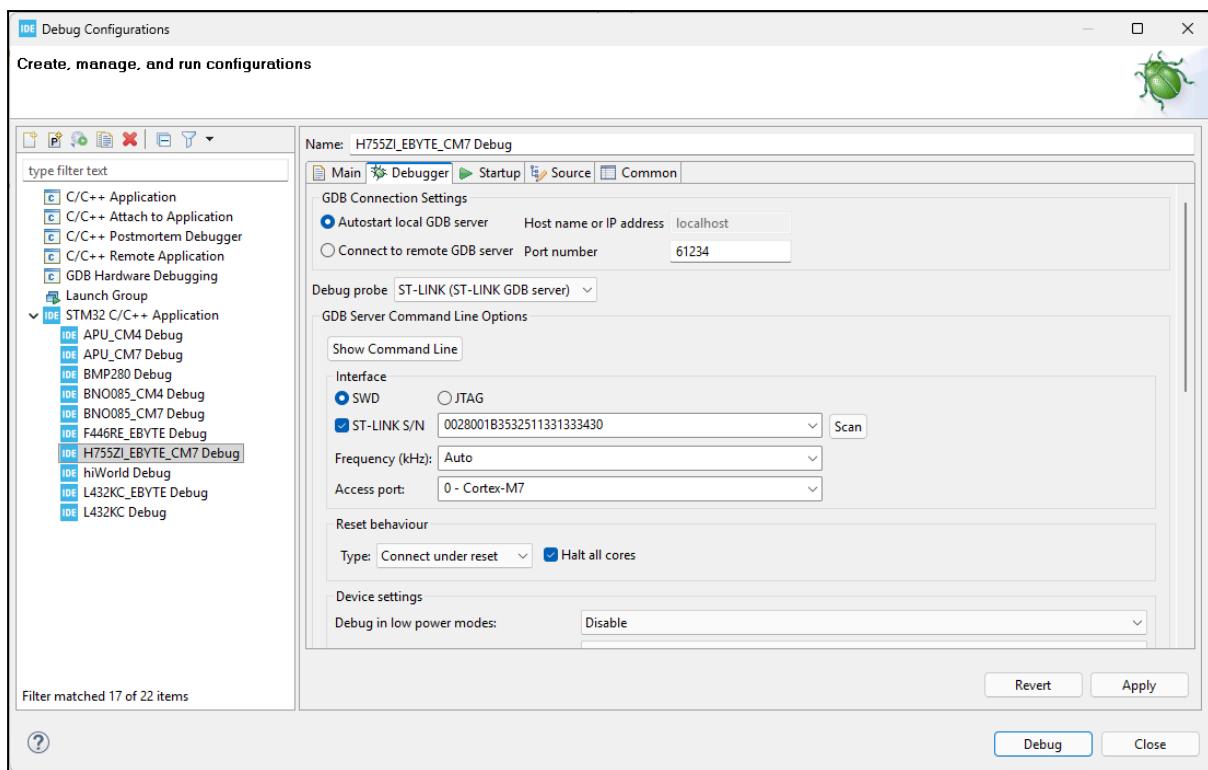
Si salen errores acerca de la indexación en C/C++, reiniciar el IDE.

En *Pinout* limpiamos todos los pines restantes para empezar a trabajar.

## **Guía para el debug**

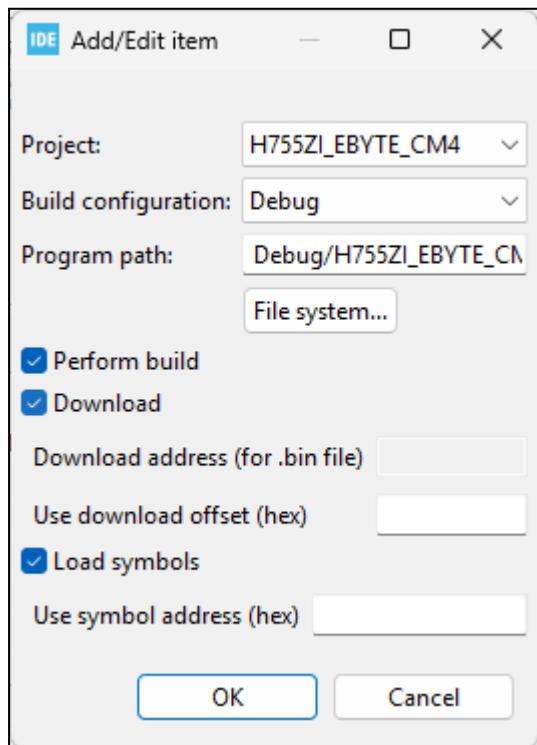
Realizado la programación y conexión del NUCLEO H755ZI el procedimiento para el **Debug** es el siguiente.

Primero creamos el **Debug File** para el CM7.

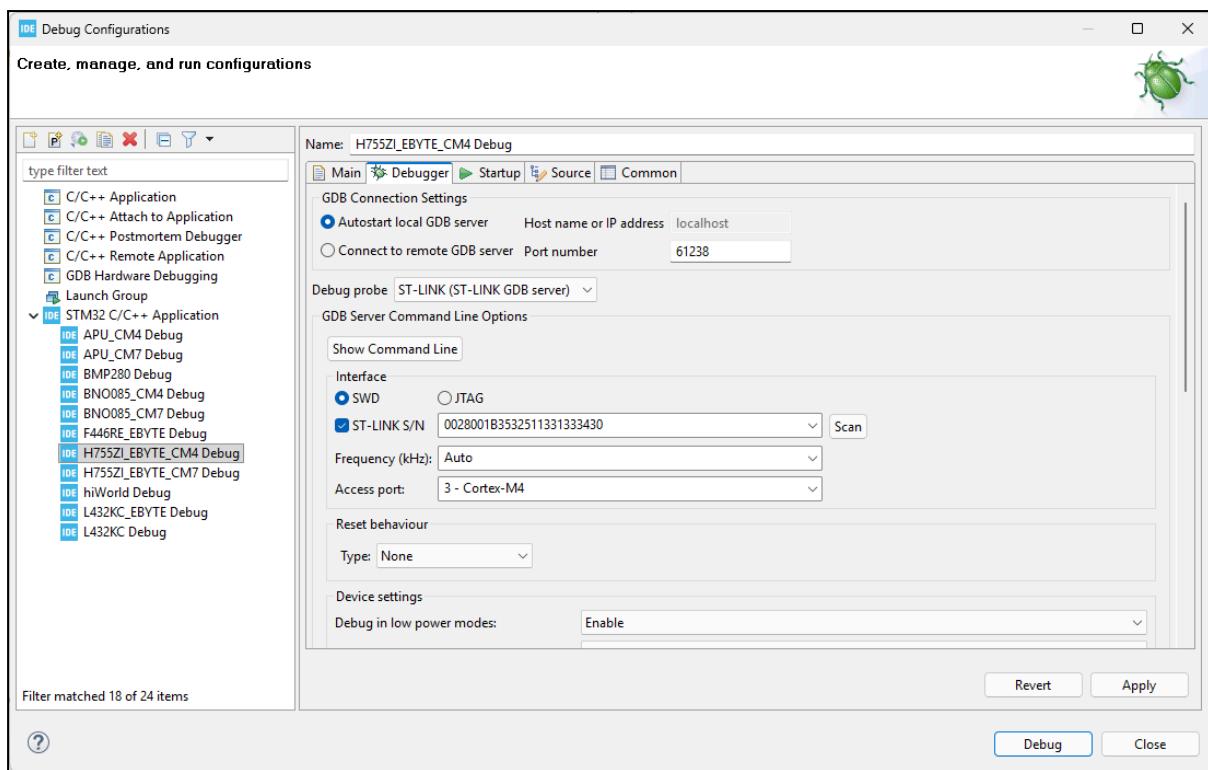


En la parte de *Debugger*, dejamos el puerto por defecto y seleccionamos la tarjeta conectada. Además se activa la opción ***Halt all cores***.

En la parte de *Startup* añadimos lo siguiente.



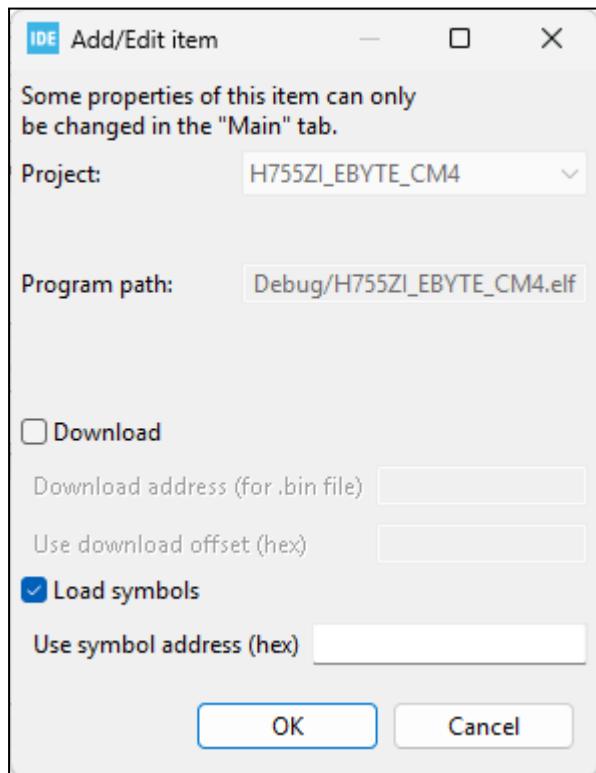
Aplicamos y cerramos ahora creamos el ***Debug File*** para el CM4.



Cambiamos de puerto a **61238**.

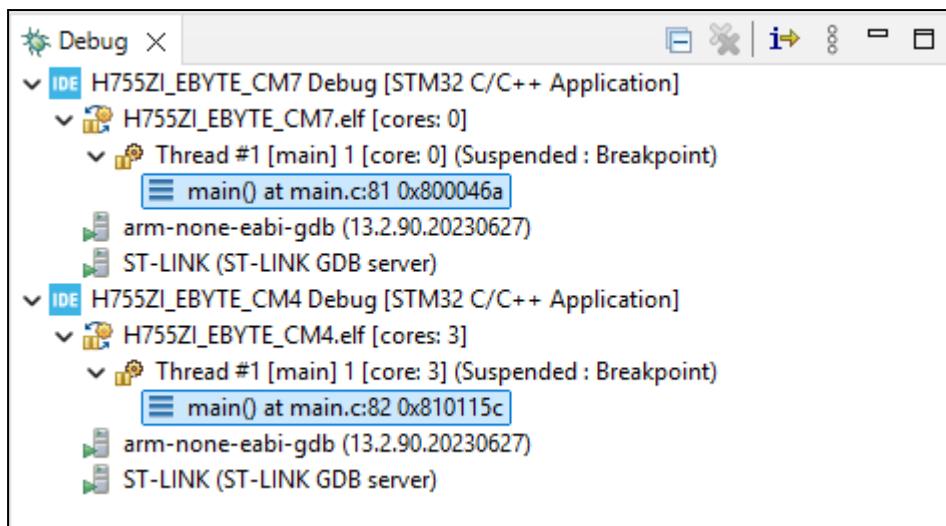
Escaneamos la tarjeta y en la opción **Type** seleccionamos **None**.

En la sección de *Startup* editamos y quitamos la selección **Download**.



Ahora procedemos a hacer el Debug de las configuraciones. **Primero al núcleo CM7, luego al CM4.**

Finalmente debemos inicializar ambos al mismo tiempo seleccionando ambos arranques de los dos núcleos.



## OBSERVACIONES

En la consola saldrá el mensaje de *Waiting for debugger connection* aunque de todas formas se podrá ejecutar ambos núcleos.

```
STMicroelectronics ST-LINK GDB server. Version 7.9.0
Copyright (c) 2024, STMicroelectronics. All rights reserved.

Starting server with the following options:
  Persistent Mode          : Disabled
  Logging Level            : 1
  Listen Port Number       : 61238
  Status Refresh Delay     : 15s
  Verbose Mode             : Disabled
  SWD Debug                : Enabled

Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
```

La desactivación en el CubeIDE también se realiza para ambos núcleos al mismo tiempo.

## Creación de la carpeta Peripherals

Incluir carpeta para que lo reconozca

Ref (min 7:48):

<https://www.youtube.com/watch?v=ubACkaN6QVA&list=PLO92aMMVufr8zID9TIniz7aALdDxMUIqU&index=1>

## Ver el USART y la muestra de texto

Verificación de qué interfaz de USART utiliza cada placa. Para el Dual Core utilizado es **USART3**. Primero que todo en la creación del proyecto debemos deseleccionar las recomendaciones para que los pines estén limpios.

<https://www.youtube.com/watch?v=WnCpPf7u4Xo&t=422s>

Otro video de USART con Dual Core (puede servir)

<https://www.youtube.com/watch?v=g8pnH7RhRUo>

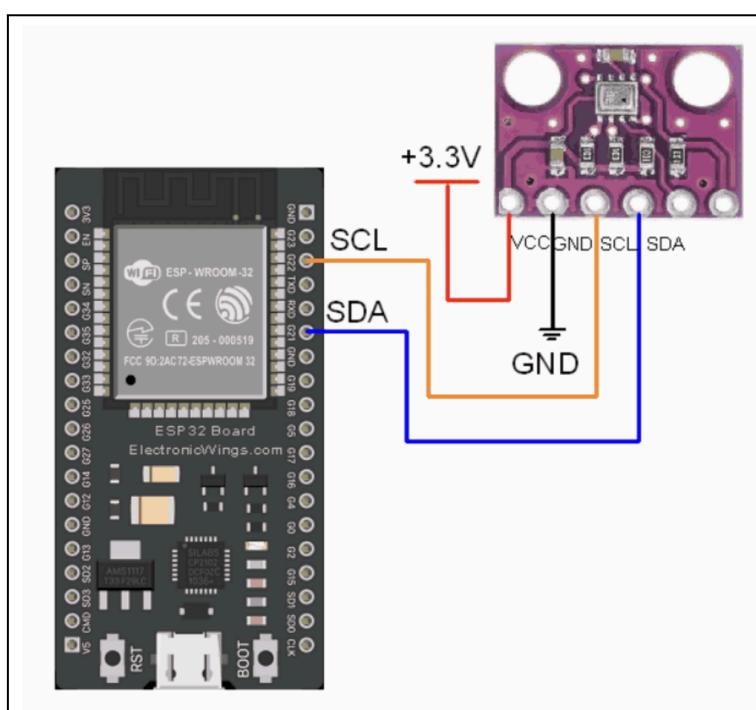
## Reconocimiento de I2C por ESP32

Selección de pines.

Imagen referencial para ver los pines SDA y SCL.

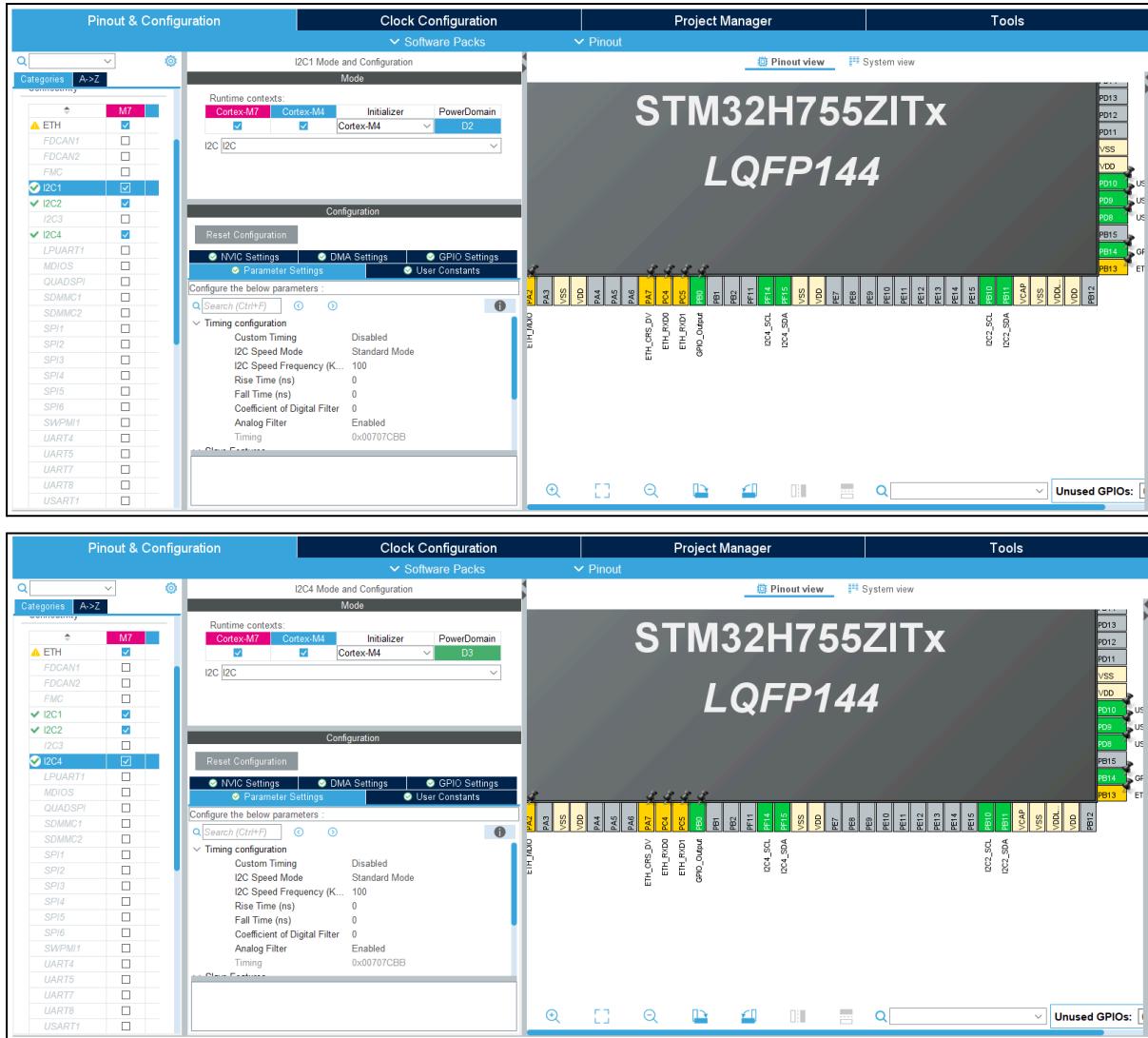
G22 -> SCL

G21 -> SDA



## Configuración para la creación de los I2C

No movemos nada de las configuraciones.



Configuramos los 3 I2C de la misma forma, usamos el I2C1, I2C2, I2C4.

El I2C1 e I2C2 tiene Power Domain en D2 (funciona con energía del M4), el I2C4 tiene Power Domain D3 (modo de bajo consumo, puede funcionar sin brindar energía por parte del M4 y M7).

Power Domain tiene que ver con la alimentación. D1 significa que tiene que estar activado la energía del M7 para que funcione el periférico. De la misma forma D2 tiene que estar activado la energía de M4. Sin embargo, D3 funciona en modo bajo de energía sin estar conectado o activado los núcleos M4 o M7.

### Pines que se asignó a partir de la configuración

I2C1 SDA: PB7

I2C1 SCL: PB6

I2C2 SDA: PB11

I2C2 SCL: PB10

I2C4 SDA: PF15

I2C4 SCL: PF14

# Pruebas SHT31

Alimentación del sensor: 3.3V (conexión a VIN)

Dirección I2C: 0x44

Las primeras pruebas fueron en ESP32 para ver si funciona bien el sensor. **CORRECTO.**

## Conexión

I2C1 SDA: PB7

I2C1 SCL: PB6

## Búsqueda de librería

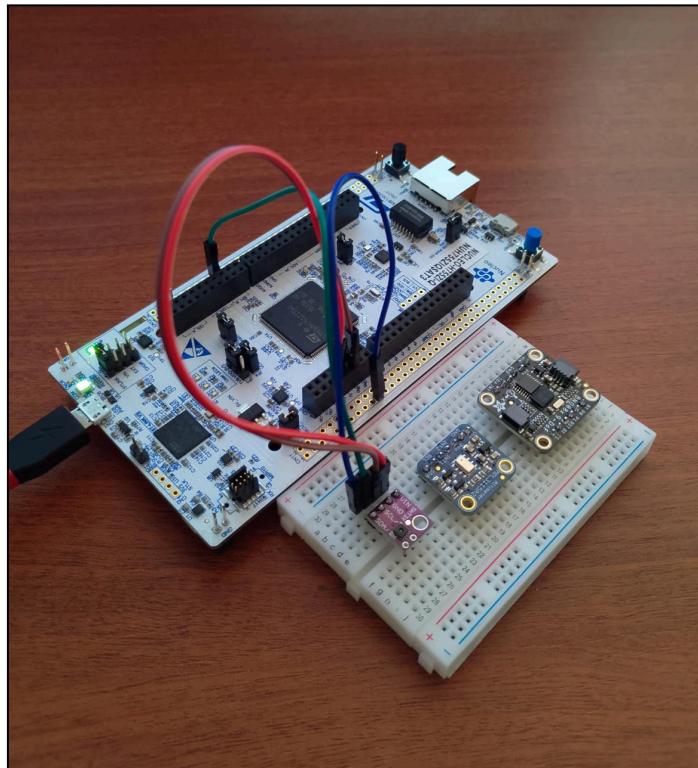
Ref: <https://github.com/trung-pham-dinh/SHT31-STM32> **CORRECTO.**

Agregamos la carpeta Peripheral para que todo esté ordenado.

Se debe revisar las líneas de código adicionales que se agregaron a los archivos *SHT31.h* y *SHT31.c*.

La alimentación del sensor fue a través del NÚCLEO.

## Resultado



```
Console X Progress Memory
COM5 (CONNECTED)

Temperature: 22.815672
Humidity: 64.286263

Temperature: 22.799648
Humidity: 64.272530

Temperature: 22.829023
Humidity: 64.319832

Activar Windows
Ve a Configuración para activar Windows.
```

## Pruebas MPL3115A2

Alimentación del sensor: 3.3V (conexión a VIN)

Dirección I2C: 0x60

Las primeras pruebas fueron en ESP32 para ver si funciona bien el sensor. **CORRECTO.**

### Conexión

I2C1 SDA: PB11

I2C1 SCL: PB10

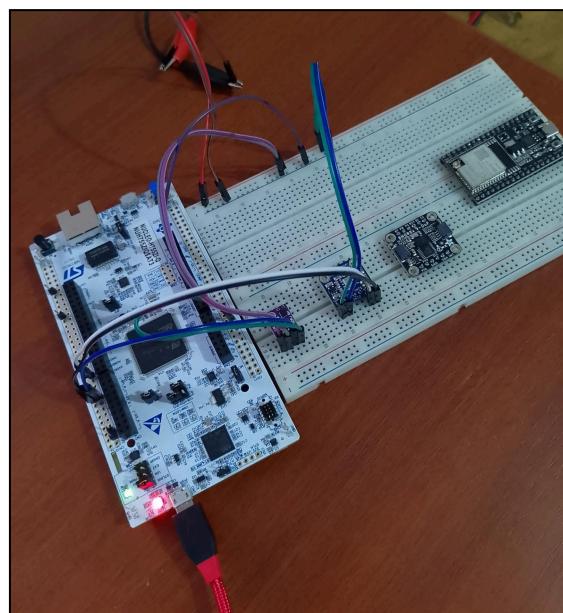
### Búsqueda de librería

Ref: <https://github.com/KwintenSchram/Ambient/tree/master/stm32>

Debemos seleccionar si queremos operar en modo altímetro o modo barómetro (presión).

Ahora la alimentación de los sensores fue con una fuente externa, porque probamos los dos sensores al mismo tiempo.

### Resultado



```
Console X Progress Memory
COMS (CONNECTED)
#####
Temperature: 22.970551 C, Humidity: 61.113907
Pressure: 96612.000000 Pa, Temperature: 21.937500 C
#####
Temperature: 22.983902 C, Humidity: 61.118484
Pressure: 96617.000000 Pa, Temperature: 21.937500 C
#####
Activar Windows
Ve a Configuración para activar Windows.
```

Lo de rojo es el resultado del MPL3115A2. El otro resultado es del SHT31.

## Pruebas BNO085

Alimentación del sensor: 3.3V (conexión a VIN)

Dirección I2C: 0x4A

Las primeras pruebas fueron en ESP32 para ver si funciona bien el sensor. **CORRECTO.**

### Conexión

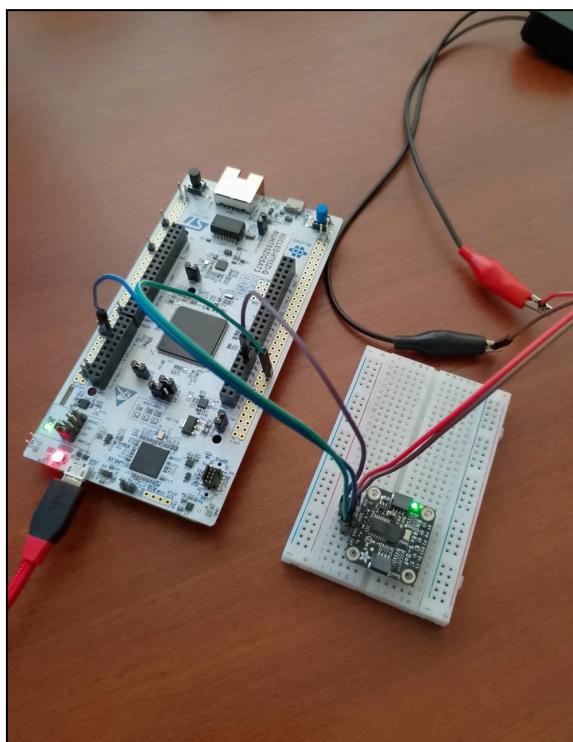
I2C1 SDA: PF15

I2C1 SCL: PF14

### Búsqueda de librería

Ref: [https://github.com/ufnalski/ahrs\\_bno085\\_g474re/tree/main](https://github.com/ufnalski/ahrs_bno085_g474re/tree/main)

Primero se replicó el código del github revisado con la finalidad de obtener resultados y probar el sensor. Los resultados salen bien con una observación, se necesita alimentación externa para poder garantizar el correcto funcionamiento del sensor ( impresión de datos en pantalla), caso contrario los datos se quedan colgados quizás porque requiere una ligera demanda de energía por parte del sensor para el envío de datos.





El voltaje entregado es de 3.3V estable y la demanda de corriente **solo del sensor** es de 9 mA aprox.

### Resultado (prueba para solo el sensor BNO085: conectado a I2C1)

```
Console X Progress Memory
COM5 (CONNECTED)
Roll : -178.406138 deg
Pitch : 2.692147 deg
###
Heading: -171.881541 deg
Roll : -178.406138 deg
Pitch : 2.692147 deg
###
Heading: -171.881541 deg
Roll : -178.406138 deg
Pitch : 2.692147 deg
###
Activar Windows
Ve a Configuración para activar Windows.
```

Se tuvo que crear una librería simple para lectura de datos a partir del programa *main.c* del github anterior (revisar el proyecto APU en el CubeIDE).

### Resultado con la librería creada usando I2C1

```
Console X Progress Memory
COM5 (CONNECTED)
Y: -178.743256, P: 3.130996, R: -179.076983
Activar Windows
Ve a Configuración para activar Windows.
```

Ahora que ya tenemos la librería pasaremos a usar otra interfaz I2C, usaremos el I2C4 porque así se definió cuando usemos los demás sensores.

## Resultado con la interfaz I2C4



The screenshot shows a terminal window titled 'Console' with the status 'COM5 CONNECTED'. The window displays a series of sensor readings in a loop:

```
Yaw: -178.952959, Pitch: 2.728176, Roll: -161.494404
Yaw: -178.945078, Pitch: 2.734042, Roll: -161.501068
Yaw: -178.938323, Pitch: 2.733007, Roll: -161.507822
```

In the bottom right corner of the terminal window, there is a watermark that says 'Activar Windows' and 'Ve a Configuración para activar Windows.'

## Links para su revisión (creación de librerías)

<https://www.youtube.com/watch?v=NUAeER-1e4c>

[https://www.youtube.com/watch?v=\\_JQAVe05o\\_0](https://www.youtube.com/watch?v=_JQAVe05o_0)

## Pruebas SD Card

Alimentación del sensor: 3.3V-**5V** (conexión a VIN)

Alimentación	3.3V – 5V
Interfaz	SPI
Formato SD/microSD	FAT, FAT32
Tamaño SD/microSD	1 GB hasta 32 GB

Protocolo de comunicación SPI.

### Conexión, pines SPI1

SPI1 MOSI: PA7

SPI1 MISO: PA6

SPI1 CLK: PA5

SPI1 CS: PB8 (lo definimos nosotros)

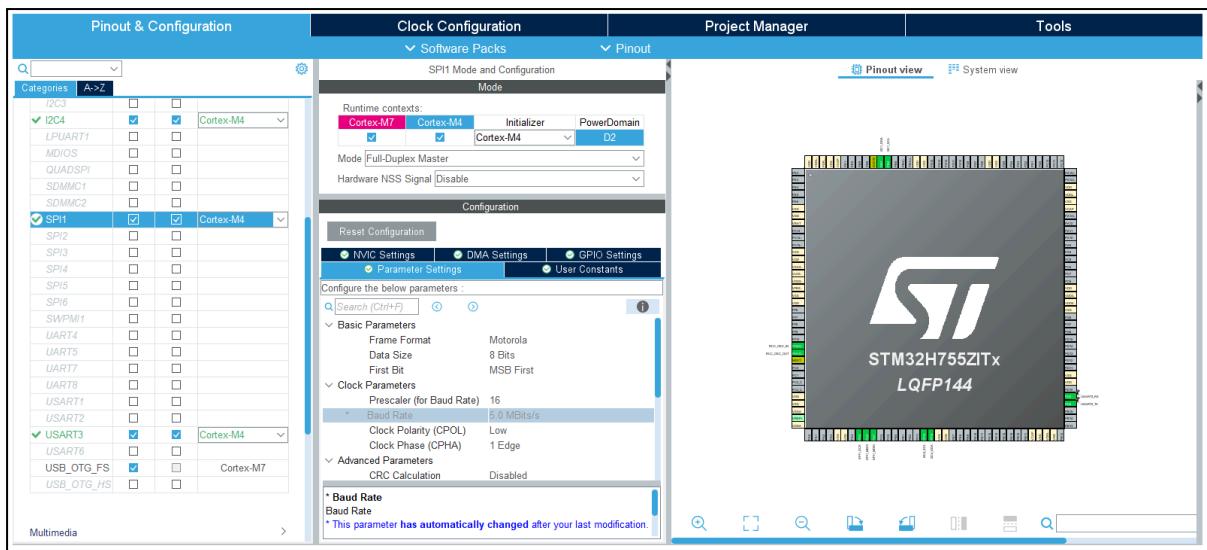
### Búsqueda de librería

Video: <https://www.youtube.com/watch?v=oTaUw2CeEjw>

Ref: [https://github.com/eziya/STM32\\_SPI\\_SDCARD/tree/master](https://github.com/eziya/STM32_SPI_SDCARD/tree/master) CORRECTO.

Se modificó el archivo, revisar el proyecto en STM32.

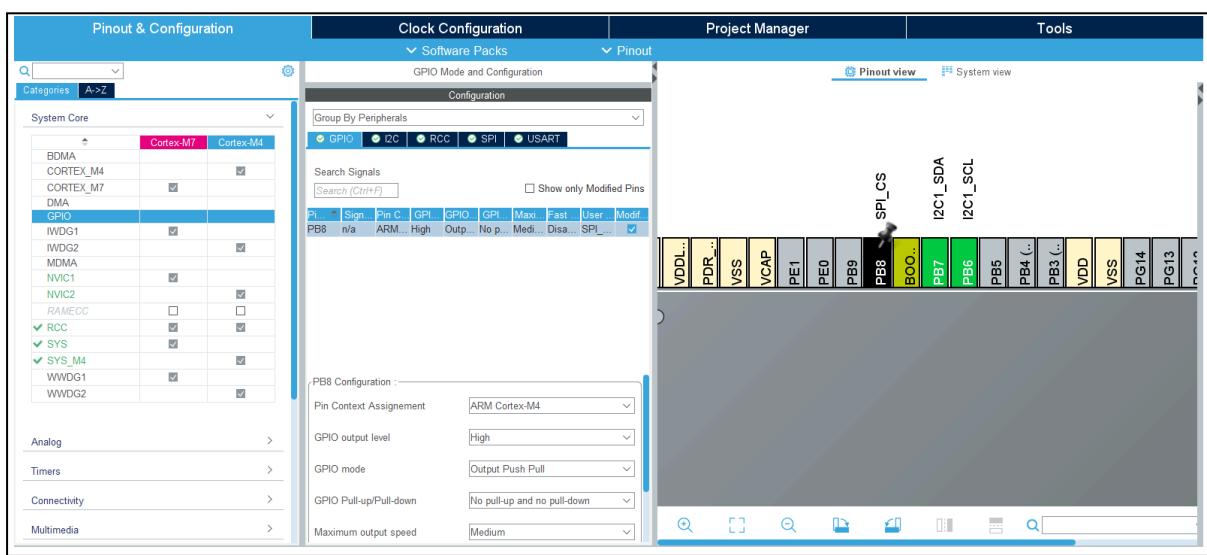
Al proyecto trabajado de BNO085 vamos a trabajar también las pruebas del SD Card. Para eso configuramos una interfaz de SPI.



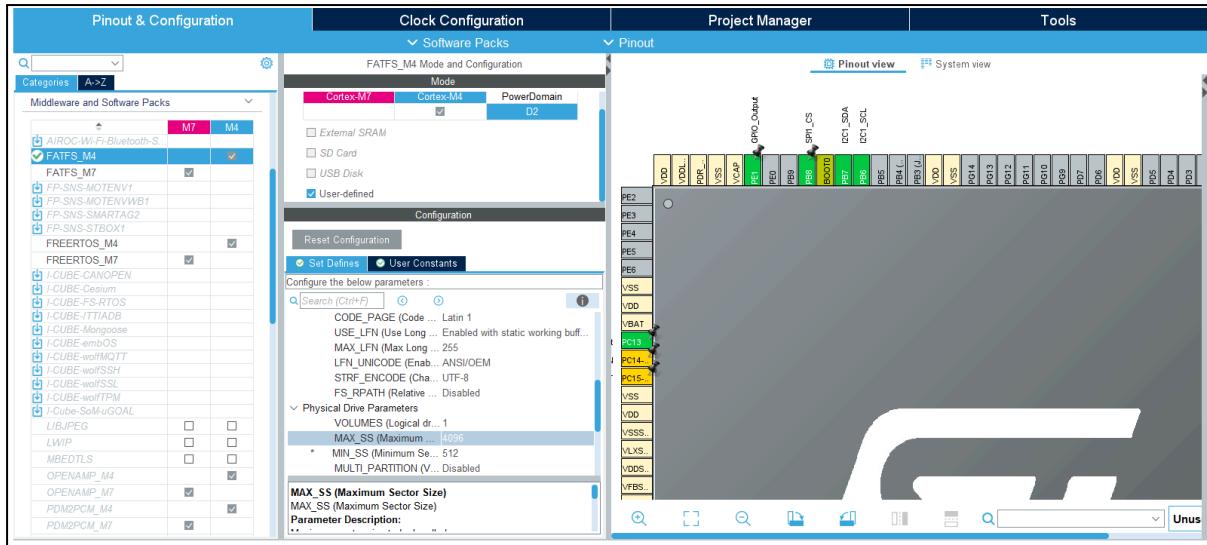
El bitrate debe ser menor a 10 MBits/s

Data size: 8 bits.

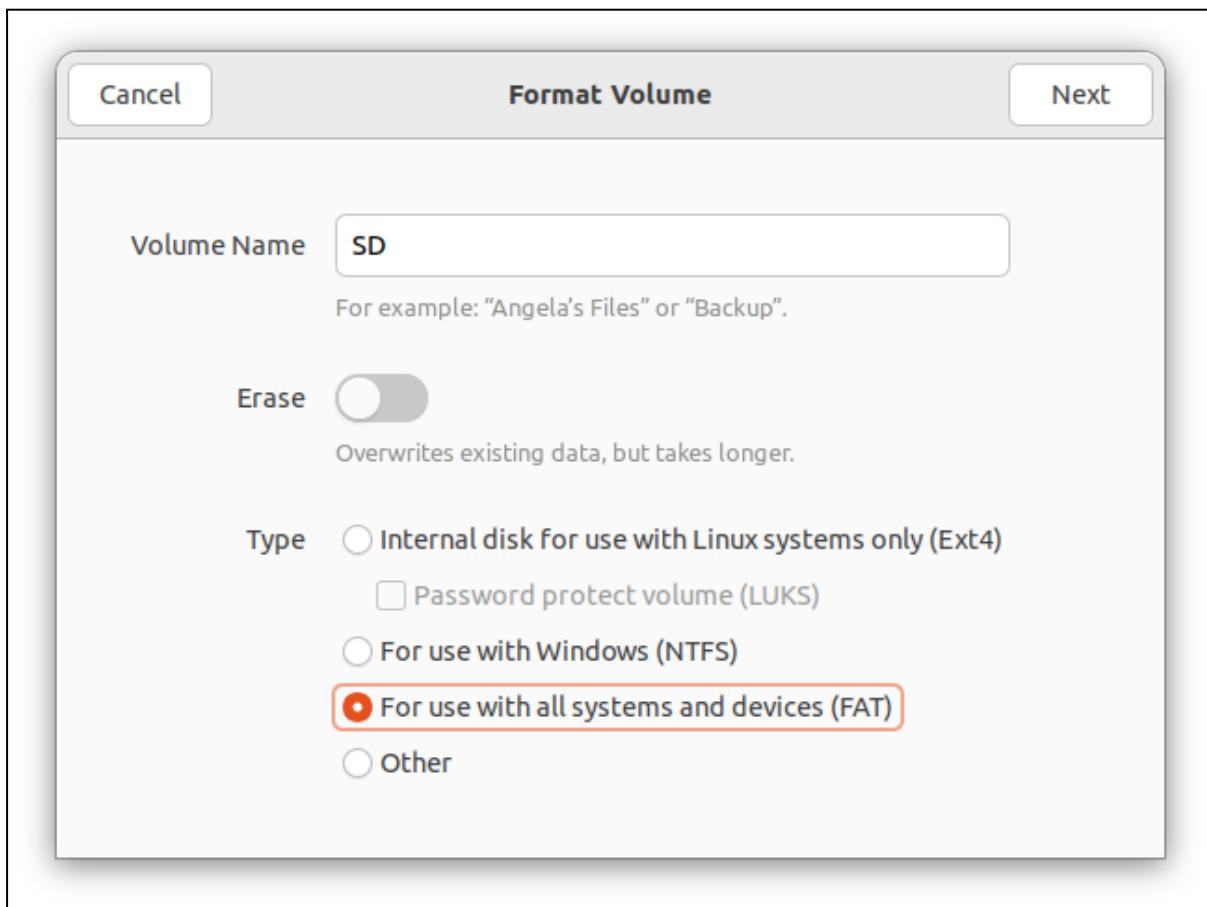
Agregamos el pin de salida CS (PB8 o a elección) con las siguientes configuraciones.

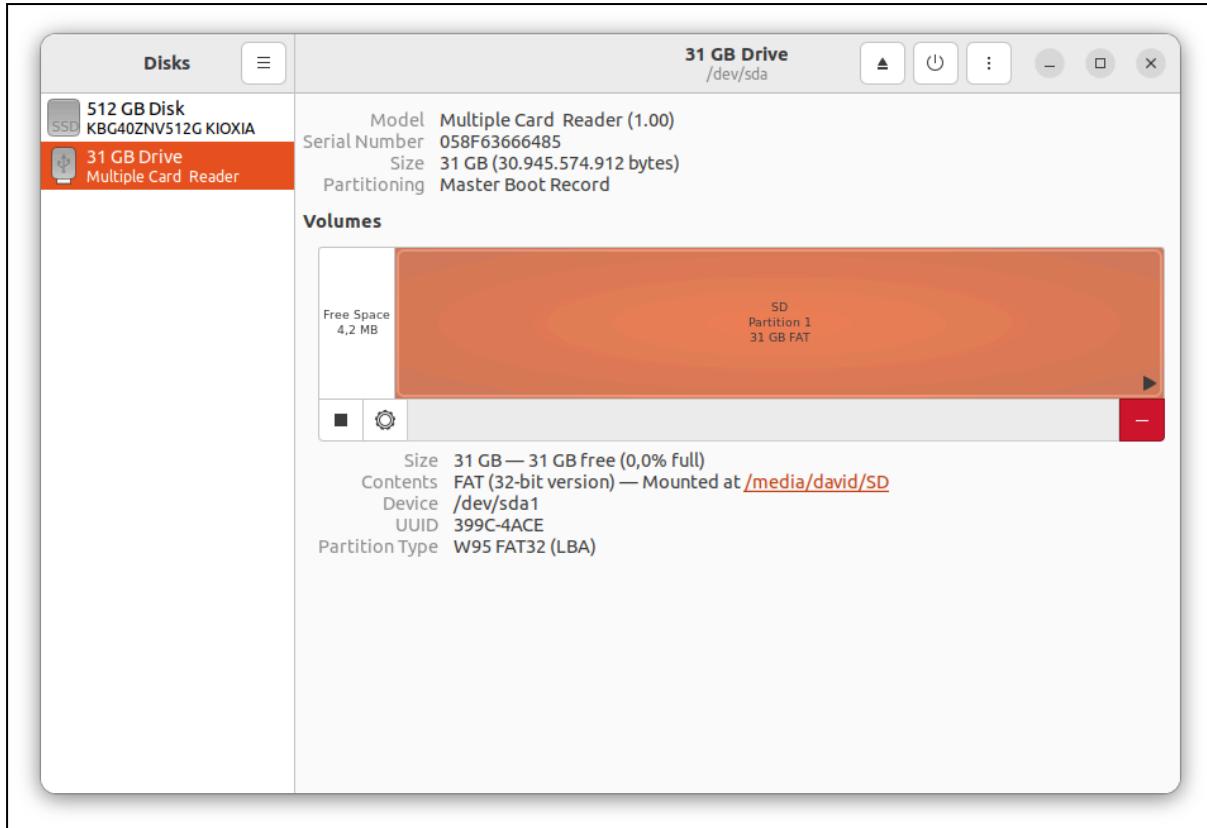


Configuraciones adicionales para el manejo de la SD Card (USE\_LFN, MAX\_SS)

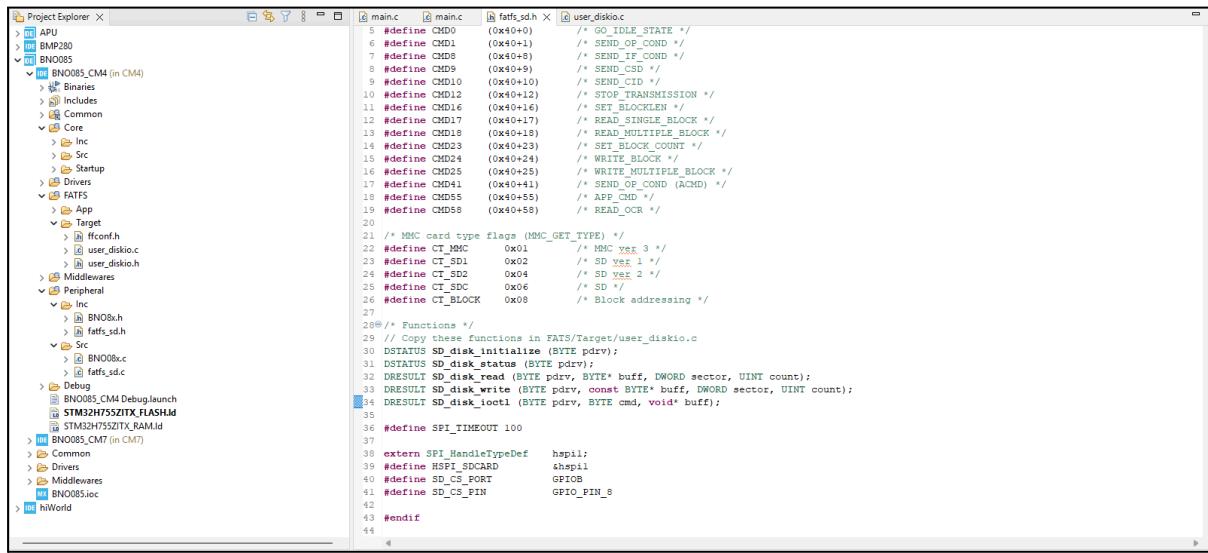


Revisar el voltaje en el módulo debe ser externo para poder alimentar correctamente.  
Formatear en FAT32 (se hizo por Linux).





Tenemos que copiar las funciones que se encuentran en el archivo *.h* de la librería para el SD Card en el archivo *FATFS/Target/user\_diskio.c*. Copiarlo para cada función implementada en el *.c*.



En el archivo *fatfs\_sd.c* de la librería agregar `#include "stm32h7xx_hal.h"`

```

1 #define TRUE 1
2 #define FALSE 0
3 #define bool BYTE
4
5 // Added for David Evangelista
6 #include "stm32h7xx_hal.h"
7

```

En el archivo *Core/Src/stm32h7xx\_it.c* agregar el siguiente código.

```

20 /* Includes -----
21 #include "main.h"
22 #include "stm32h7xx_it.h"
23@/* Private includes -----
24 /* USER CODE BEGIN Includes */
25
26 // Verified the SPI interface we use
27 extern DMA_HandleTypeDef hdma_spl_tx;
28 extern SPI_HandleTypeDef hspl;
29
30 extern uint8_t Timer1, Timer2;
31
32 /* USER CODE END Includes */

```

```

188     * @brief This function handles System tick timer.
189     */
190@void SysTick_Handler(void)
191 {
192     /* USER CODE BEGIN SysTick_IRQn_0 */
193
194     if (Timer1 > 0)
195     {
196         Timer1--;
197     }
198     if (Timer2 > 0)
199     {
200         Timer2--;
201     }
202
203     /* USER CODE END SysTick_IRQn_0 */
204     HAL_IncTick();
205     /* USER CODE BEGIN SysTick_IRQn_1 */
206     HAL_SYSTICK_IRQHandler();
207     /* USER CODE END SysTick_IRQn_1 */
208 }

```

En el *main.c* están las funciones pero necesitamos colocarlas como funciones en otra librería para mayor manejo.

### OBSERVACIÓN

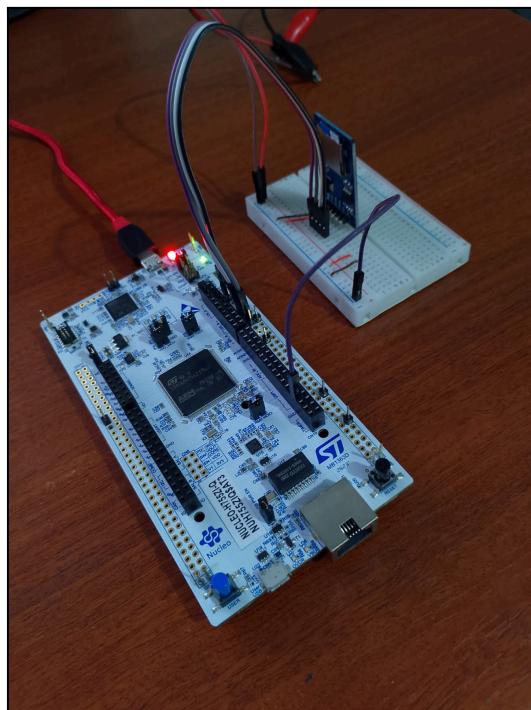
Se hicieron pruebas con solo el SD Card:

La primera vez no escribió (montando la SD Card y el BNO085 y alimentando con 3,3V ambos componentes), solo se montó bien la SD Card pero el programa no siguió. Quitando el BNO085 y **alimentándose con 5V** si corrió normal el programa.

Ahora probaremos con 3.3V pero solo el módulo SD Card. **NO FUNCIONA.** Se queda colgado como la vez anterior, solo muestra que se montó bien el módulo pero no avanza el código.

**La SD Card debe ser alimentada con 5V externos para su correcto funcionamiento.**

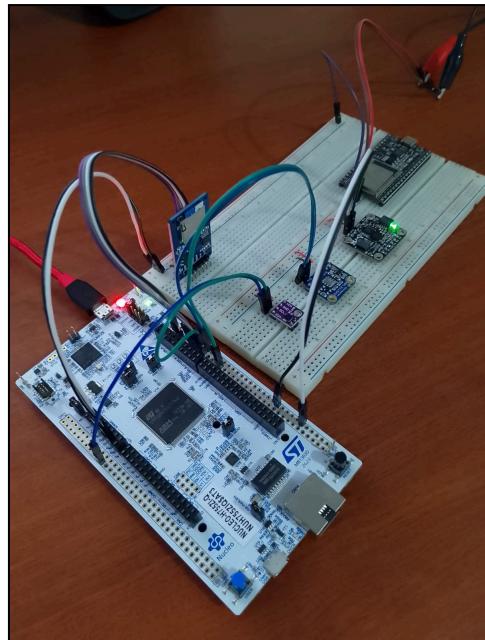
## Resultados



```
COMS (CONNECTED)
File opened for reading and checking the free space.
The free space is 30201296.
The file is closed.
File opened for reading.
N1: 3, N2: 3.14, N3: 1.14
The file is closed.
The Micro SD Card is unmounted.
```

## Prueba con los sensores avanzados y guardado de los datos en el SD Card

Se unieron todas las librerías avanzadas en un solo proyecto (**APU**). Se realizó la conexión de todos los sensores vistos y el módulo SD Card.



También se midió la corriente consumida por los tres sensores (SHT31, MPL3115A2 y BNO085). El módulo fue alimentado por el mismo NUCLEO a 5V.



Corriente consumida aprox = 10 mA

Recordar primero alimentar luego arrancar los programas.

## Resultados

Data mostrada para el guardado en la SD Card

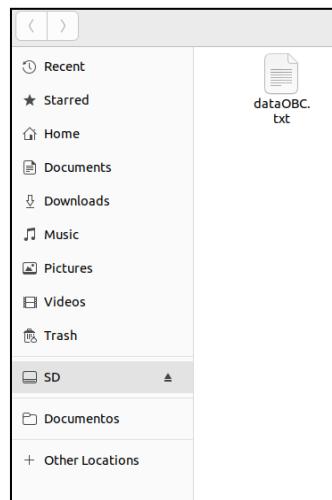
```

Console X Progress Memory
COM5 CONNECTED
SHT31: T 24.418, H 62.625; MPL3115A2: P 97117.000, T 23.625; BNO085: Y -179.301, P 5.616, R 97.760
Data
SHT31: T 24.431, H 62.484; MPL3115A2: P 97118.000, T 23.625; BNO085: Y -179.301, P 5.616, R 97.760
Data
SHT31: T 24.447, H 62.608; MPL3115A2: P 97122.000, T 23.625; BNO085: Y -179.301, P 5.616, R 97.760
Data
SHT31: T 24.418, H 62.580; MPL3115A2: P 97122.000, T 23.625; BNO085: Y -179.301, P 5.616, R 97.760
Data
SHT31: T 24.418, H 62.580; MPL3115A2: P 97117.000, T 23.625; BNO085: Y -179.301, P 5.616, R 97.760
Data
SHT31: T 24.418, H 62.535; MPL3115A2: P 97117.000, T 23.625; BNO085: Y -179.301, P 5.616, R 97.760
Data

```

## Revisión de la SD Card

Creación de archivo y vista de contenido.



```

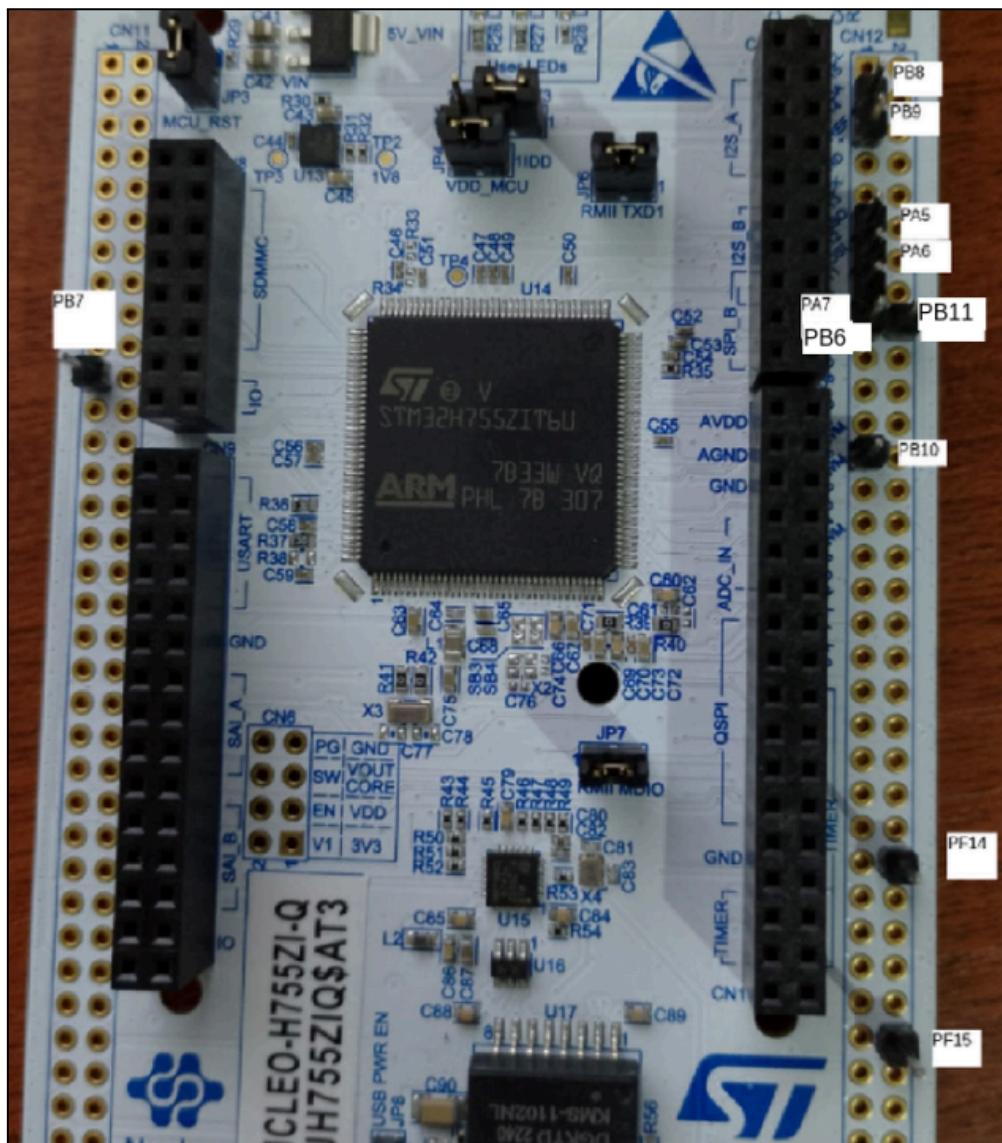
dataOBC.txt
SD /media/david/SD

1 SHT31: T 24.477, H 63.055; MPL3115A2: P 97119.000, T 23.500; BNO085: Y -178.876, P 5.895, R 93.064
2
3 SHT31: T 24.431, H 63.046; MPL3115A2: P 97120.000, T 23.562; BNO085: Y -178.876, P 5.895, R 93.064
4
5 SHT31: T 24.463, H 62.974; MPL3115A2: P 97117.000, T 23.500; BNO085: Y -178.876, P 5.895, R 93.064
6
7 SHT31: T 24.447, H 62.918; MPL3115A2: P 97117.000, T 23.500; BNO085: Y -178.876, P 5.895, R 93.064
8
9 SHT31: T 24.447, H 62.795; MPL3115A2: P 97119.000, T 23.562; BNO085: Y -178.876, P 5.895, R 93.064
10
11 SHT31: T 24.447, H 62.795; MPL3115A2: P 97118.000, T 23.500; BNO085: Y -178.876, P 5.895, R 93.064
12
13 SHT31: T 24.418, H 62.821; MPL3115A2: P 97120.000, T 23.562; BNO085: Y -178.483, P 5.682, R 98.055
14
15 SHT31: T 24.405, H 62.721; MPL3115A2: P 97119.000, T 23.562; BNO085: Y -178.432, P 5.654, R 101.655
16
17 SHT31: T 24.375, H 62.727; MPL3115A2: P 97118.000, T 23.562; BNO085: Y -178.424, P 5.630, R 103.927
18
19 SHT31: T 24.447, H 62.762; MPL3115A2: P 97116.000, T 23.562; BNO085: Y -179.164, P 5.792, R 100.302
20
21 SHT31: T 24.418, H 62.780; MPL3115A2: P 97116.000, T 23.562; BNO085: Y -179.445, P 5.891, R 94.239
22
23 SHT31: T 24.418, H 62.780; MPL3115A2: P 97119.000, T 23.562; BNO085: Y -179.453, P 5.919, R 91.950
24
25 SHT31: T 24.447, H 62.808; MPL3115A2: P 97120.000, T 23.562; BNO085: Y -178.831, P 5.902, R 93.835
26
27 SHT31: T 24.477, H 62.933; MPL3115A2: P 97118.000, T 23.562; BNO085: Y -178.580, P 5.801, R 97.739
28
29 SHT31: T 24.447, H 62.940; MPL3115A2: P 97119.000, T 23.562; BNO085: Y -178.924, P 5.922, R 95.649
30
31 SHT31: T 24.506, H 63.038; MPL3115A2: P 97118.000, T 23.562; BNO085: Y -178.923, P 5.961, R 95.640
32
33 SHT31: T 24.418, H 62.954; MPL3115A2: P 97119.000, T 23.562; BNO085: Y -178.918, P 5.976, R 95.625
34
35 SHT31: T 24.463, H 62.986; MPL3115A2: P 97119.000, T 23.562; BNO085: Y -178.918, P 5.976, R 95.625
36
37 SHT31: T 24.463, H 62.986; MPL3115A2: P 97120.000, T 23.562; BNO085: Y -178.918, P 5.976, R 95.625
38
39 SHT31: T 24.490, H 63.014; MPL3115A2: P 97121.000, T 23.562; BNO085: Y -178.918, P 5.976, R 95.625
40
41 SHT31: T 24.391, H 62.608; MPL3115A2: P 97111.000, T 23.562; BNO085: Y -179.080, P 5.895, R 98.598
42
43 SHT31: T 24.362, H 62.557; MPL3115A2: P 97110.000, T 23.562; BNO085: Y -179.080, P 5.895, R 98.598
44
45 SHT31: T 24.391, H 62.631; MPL3115A2: P 97110.000, T 23.562; BNO085: Y -179.080, P 5.895, R 98.598
46
47 SHT31: T 24.431, H 62.672; MPL3115A2: P 97111.000, T 23.562; BNO085: Y -179.080, P 5.895, R 98.598
48

```

Plain Text ▾ Tab Width: 8 ▾ Ln 13, Col 43 ▾ INS

## Pinout usado



## **Referencias**

Para el módulo LORA

Ref: <https://www.youtube.com/watch?v=JG189GIko7k>

Creación de drivers STM32

[https://www.youtube.com/watch?v=\\_JQAvE05o\\_0](https://www.youtube.com/watch?v=_JQAvE05o_0)

Librería:

<https://github.com/jgromes/RadioLib/tree/master/examples/SX127x>