

# APU Space - STM32

164	10K COTS	Universidad Nacional de Ingeniería	Lima, Perú	Perú	APU Space
-----	----------	------------------------------------	------------	------	-----------

## CARGA ÚTIL

El vehículo de lanzamiento deberá llevar una carga útil de no menos de 4,4 libras (2 Kg aprox.).

## REGISTRO DE ALTITUD

Todos los cohetes deberán incluir una solución de seguimiento GPS.

Los vehículos de lanzamiento deberán llevar un altímetro de presión barométrica COTS con almacenamiento de datos a bordo, que proporcionará un registro oficial del apogeo para su puntuación.

## MOTORES

Motor COTS: de uso comercial listo para usar

Motor SRAD: realizado por los estudiantes

Se deberá emplear propelentes no tóxicos.

## SISTEMA DE CONTROL ACTIVO

¿Se tiene un sistema de control activo?

## LANZAMIENTO

¿Sistema de lanzamiento?

### Enlaces

#### Sistema principal

[https://unipe-my.sharepoint.com/personal/harold\\_zambrano\\_q\\_uni\\_pe/\\_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fharold%5Fzambrano%5Fq%5Funi%5Fpe%2FDocuments%2FUND%20PRIVADO%2FPERSONAL%2FPROYECTO%2FGRUPO%20DE%20PROYECTO%2FASME%20%2D%20COHETE%2FTrabajos%2DEstado%20de%20arte%2FAPU%20Space%20EyT%20%2D%20Sistema%20Principal%2DDespliegue%2FSistema%20Principal&ga=1](https://unipe-my.sharepoint.com/personal/harold_zambrano_q_uni_pe/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fharold%5Fzambrano%5Fq%5Funi%5Fpe%2FDocuments%2FUND%20PRIVADO%2FPERSONAL%2FPROYECTO%2FGRUPO%20DE%20PROYECTO%2FASME%20%2D%20COHETE%2FTrabajos%2DEstado%20de%20arte%2FAPU%20Space%20EyT%20%2D%20Sistema%20Principal%2DDespliegue%2FSistema%20Principal&ga=1)

#### Informes técnicos

##### IREC

[https://drive.google.com/drive/folders/1fnJhcqYWSvXhEHjA7iv2EFF-rwovlsmb?usp=drive\\_link](https://drive.google.com/drive/folders/1fnJhcqYWSvXhEHjA7iv2EFF-rwovlsmb?usp=drive_link)

##### TEKNOFEST

[https://drive.google.com/drive/folders/16tNU2RgIk2mPlc9RbfVakJILqZtHJ7Wk?usp=drive\\_link](https://drive.google.com/drive/folders/16tNU2RgIk2mPlc9RbfVakJILqZtHJ7Wk?usp=drive_link)

#### Reporte técnico

[https://unipe-my.sharepoint.com/:w/g/personal/harold\\_zambrano\\_q\\_uni\\_pe/EVME9C\\_go\\_NCsLD0vvVdfgUB\\_g460Ow-QDIQWafMmmxvyA?e=NQZaEr](https://unipe-my.sharepoint.com/:w/g/personal/harold_zambrano_q_uni_pe/EVME9C_go_NCsLD0vvVdfgUB_g460Ow-QDIQWafMmmxvyA?e=NQZaEr)

# REVISIÓN

Antes usaban PIC18F4550

Uso de STM32

Lectura de los avances hasta pág. 42.

**SRAD:** Sistema redundante

**COTS:** Computadora ya hecha

Sistema direccional de antena

Sistema de control activo

Ver los sensores

## **Pág. 0**

¿El propelente es comprado, el sistema es COTS?

¿De donde sacaron la simulación del motor?

Al parecer sacan sus componentes de la tienda Aerotech

Materiales usados:

ABS: Ojiva

Duraluminio: ?

Fibra de vidrio: Fuselaje

¿Qué programas se utilizan para la simulación de la nariz, qué parámetros se le dan al software?

El sistema de freno aéreo utiliza inteligencia artificial.

¿Cómo opera el freno aéreo, que hace?

## **Pág. 12**

Tabla comparativa de protocolos de comunicación

Protocolo	Tipo	Líneas	Velocidad	Dispositivos	Ventaja Principal	Desventaja Principal
SPI	Síncrono	4+	Alta (MHz)	1 Maestro - N Esclavos	Muy rápido, full-duplex	Requiere más pines
UART	Asíncrono	2	Moderada (bps - Mbps)	1 a 1	Simple, sin reloj	No soporta múltiples dispositivos
I2C	Síncrono	2	Baja a moderada (hasta 3.4 Mbps)	1 Maestro - N Esclavos	Solo 2 cables para múltiples dispositivos	Más lento que SPI

## 25/01/25

- Ver el tipo de comunicación y programación del STM32.
- Pruebas de protocolos de comunicación.
- Si se puede hacer las pruebas con MATLAB.
- Investigar de SCRUM:  
<https://www.scrumstudy.com/certification/scrum-fundamentals-certified>

## 31/01/25

Pruebas de sensores

Videos de I2C STM32 HAL

<https://www.youtube.com/watch?v=iGZA9ThTYBQ>

### BMP280

Referencia:

[\(BUENO\)](https://naylampmechatronics.com/sensores-posicion-inerciales-gps/358-sensor-de-presion-bmp280.html)

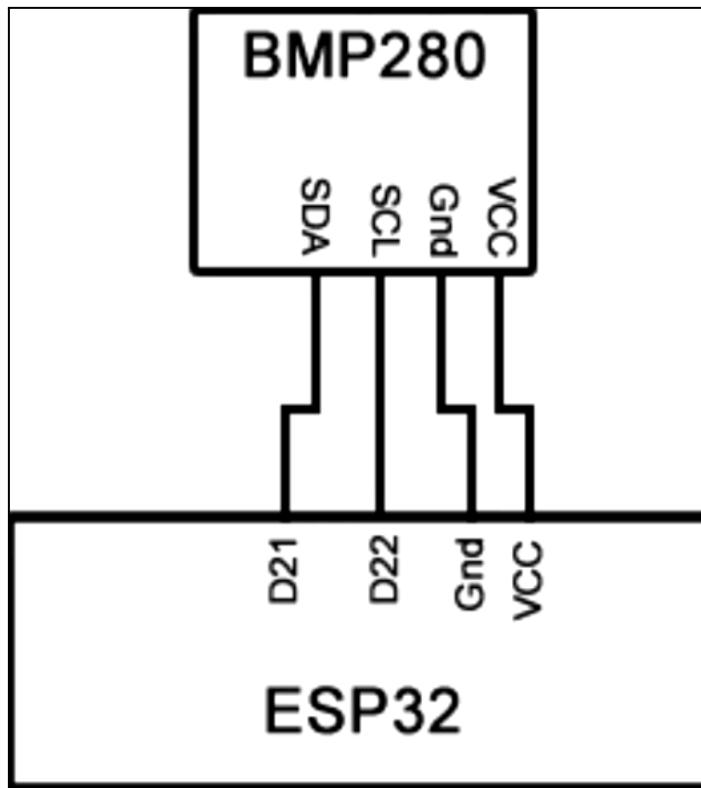
Para el programa en el ESP32 con el sensor BMP280, elegir como tarjeta ESP32 Wrover Module.

### Dirección I2C del BMP:

0X76: para el funcionamiento.

### Conexión ESP32

Alimentación 3.3V



### **Code to get the I2C address**

<https://www.circuitschools.com/interfacing-16x2-lcd-module-with-esp32-with-and-without-i2c/>

### **MPU6050**

Referencia:

<https://naylampmechatronics.com/sensores-posicion-iniciales-gps/33-modulo-mpu6050-acelerometro-giroscopio-i2c.html>

### **Dirección I2C del MPU:**

0x68: para el funcionamiento

**01/02/25**

Revisión del protocolo I2C

### **Registro en un sensor**

En el datasheet de un sensor, **un registro es una dirección de memoria dentro del sensor donde se almacenan datos de configuración, medición o estado.** Estos

registros pueden ser leídos y/o escritos a través de una interfaz de comunicación, como I<sup>2</sup>C, SPI o UART.

Tipos de registros en un sensor:

1. Registros de configuración: Permiten ajustar parámetros del sensor, como sensibilidad, rango de operación, frecuencia de muestreo, entre otros.
2. **Registros de datos:** Contienen los valores medidos por el sensor, como temperatura, aceleración, presión, etc.
3. Registros de estado: Indican información sobre el funcionamiento del sensor, como si hay un error, si una medición está lista, o si hay interrupciones activas.
4. Registros de identificación: Guardan información del fabricante, ID del sensor o versión del firmware.

Cada registro tiene una dirección específica y un tamaño definido (por ejemplo, 8 o 16 bits). Para leer o escribir en ellos, se usa un protocolo de comunicación según el sensor.

### **Periféricos en un microcontrolador**

En un microcontrolador STM32, un periférico es un módulo de hardware integrado en el chip que permite interactuar con el mundo externo o realizar tareas específicas sin cargar demasiado la CPU.

Tipos de periféricos en STM32. Los periféricos pueden clasificarse en tres categorías principales:

#### **Periféricos de Comunicación (Interacción con otros dispositivos)**

- I<sup>2</sup>C (Inter-Integrated Circuit) → Comunicación con sensores y memorias EEPROM.
- SPI (Serial Peripheral Interface) → Comunicación rápida con pantallas, memorias y ADCs externos.
- USART/UART (Universal Synchronous/Asynchronous Receiver-Transmitter) → Comunicación serie con otros microcontroladores, módulos Bluetooth, GPS, etc.
- CAN (Controller Area Network) → Comunicación en sistemas automotrices e industriales.
- USB (Universal Serial Bus) → Permite al STM32 actuar como host o dispositivo USB.

#### **Periféricos de Control y Tiempo (Medición y generación de señales)**

TIM (Timers) → Se utilizan para contar tiempo, generar PWM, capturar eventos o interrupciones periódicas.

RTC (Real-Time Clock) → Mantiene la hora y fecha incluso en bajo consumo.

WDG (Watchdog Timer) → Reinicia el microcontrolador si se detecta un fallo en el software.

#### **Periféricos de Entrada/Salida y Procesamiento**

GPIO (General Purpose Input/Output) → Controla pines de entrada y salida.

ADC (Analog to Digital Converter) → Convierte señales analógicas en digitales (ej. sensores de temperatura).

DAC (Digital to Analog Converter) → Convierte datos digitales en señales analógicas (ej. generación de audio).

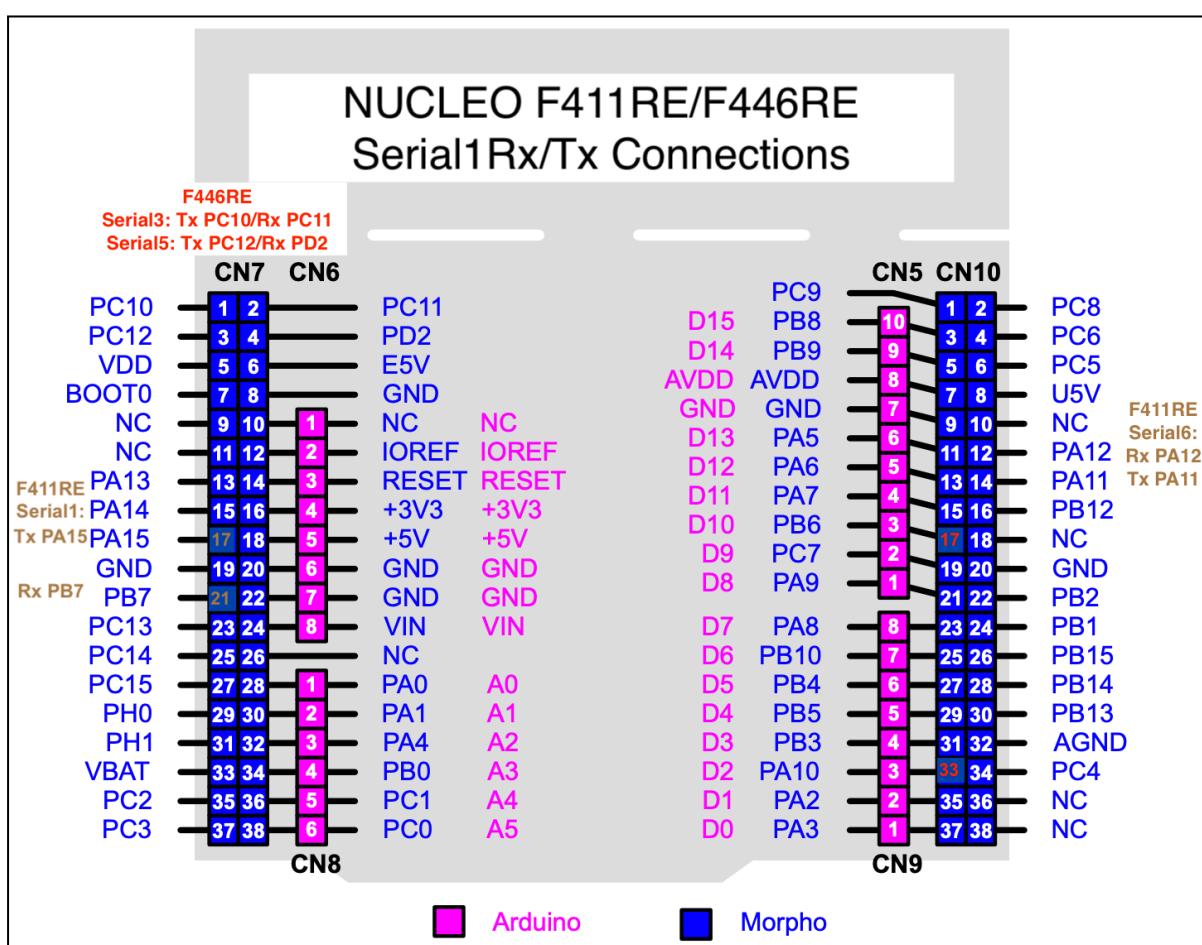
DMA (Direct Memory Access) → Permite transferencias de datos sin intervención de la CPU, mejorando el rendimiento.

FSMC/QSPI (Flexible Static Memory Controller / Quad SPI) → Manejo de memorias externas como SDRAM o Flash.

## Pinout STM32

Referencia:

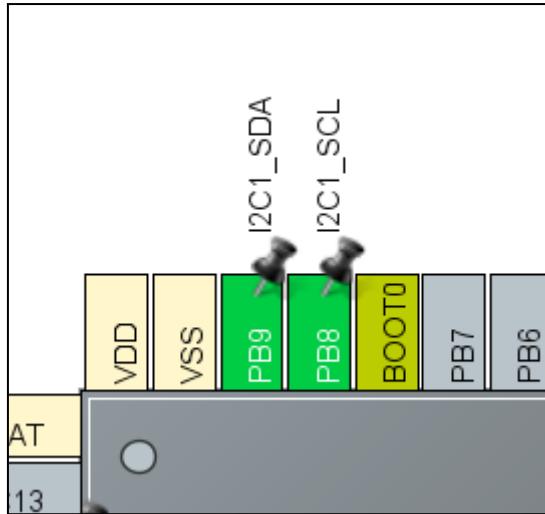
<https://dcc-ex.com/reference/hardware/microcontrollers/stm32-nucleo.html#gsc.tab=0>



Nos guiamos para la configuración I2C del siguiente video:

<https://www.youtube.com/watch?v=iGZA9ThTYBQ>

Cambio de pines para la comunicación I2C dependiendo de la tarjeta NÚCLEO (revisar en la placa donde colocan SDA y SCL), en este caso usamos la NUCLEO F446RE.



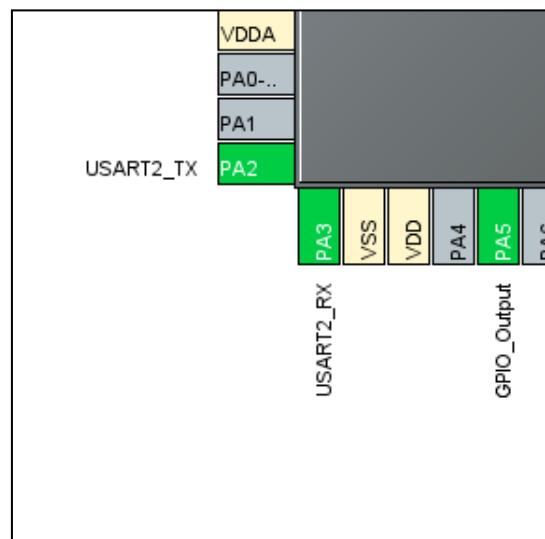
## Uso de USART (“printf”)

Referencia:

[https://www.youtube.com/watch?v=ccKZ\\_rKFDX4](https://www.youtube.com/watch?v=ccKZ_rKFDX4) (BUENO)

Ojo con la configuración del printf y scanf.

Pines para el NUCLEO F466RE. Depende de la placa.



## Configuración del MPU en STM32

Referencias:

<https://www.youtube.com/watch?v=iJn70hPxT7E>

<https://www.youtube.com/watch?v=P7a6qxacnO4>

Dentro de la carpeta **Drivers** del proyecto:

**stm32f4xx\_hal\_i2c.h**, existirán varias funciones usadas del I2C.  
Para el trabajo con sensores usamos.

Lectura de conexión del sensor.

```
HAL_StatusTypeDef HAL_I2C_IsDeviceReady(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout);
```

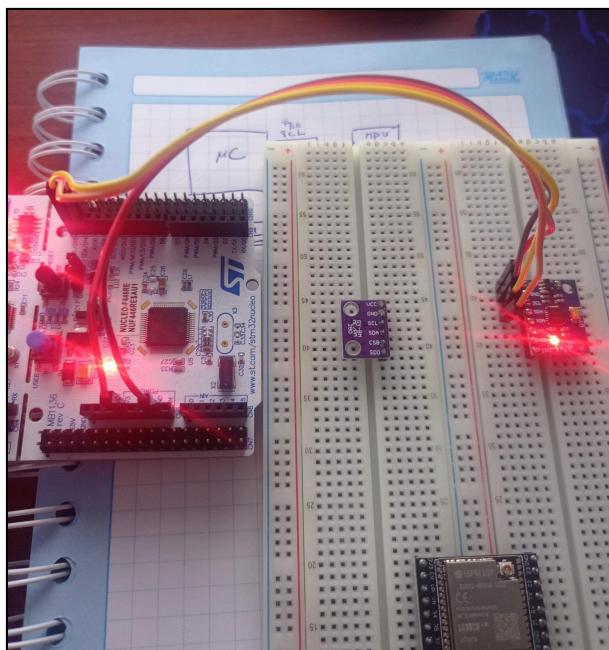
Trabajo con el sensor

Revisión del datasheet.

```
HAL_StatusTypeDef HAL_I2C_Mem_Write(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout);  
HAL_StatusTypeDef HAL_I2C_Mem_Read(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

## Pruebas alcanzadas

Detección del sensor MPU6050



```
HAL_StatusTypeDef ret = HAL_I2C_IsDeviceReady(&hi2c1, 0b1101000 << 1+0, 1, 100);  
if (ret == HAL_OK)  
{  
    printf("The device is ready \r\n");  
}  
else  
{  
    printf("The device is not ready, Check cables \r\n");  
}
```

The screenshot shows a terminal window titled "Console X Memory STM32 (CONNECTED)". The log output is as follows:

```

298
299
300
301
302
303
The device is ready
1
The device is ready
The device is ready

```

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

Alternativa de problema.

## 08/02/25

Código inicial:

Configuración del giroscopio

```

void mpu6050_Init()
{
    HAL_StatusTypeDef ret = HAL_I2C_IsDeviceReady(&hi2c1, DEVICE_ADDRESS
<< 1+0, 1, 100);
    if (ret == HAL_OK)
    {
        printf("The device is ready \r\n");
    }
    else
    {
        printf("The device is not ready, Check cables \r\n");
    }
    uint8_t temp_data = FS_GYRO_500;
    ret = HAL_I2C_Mem_Write(&hi2c1, DEVICE_ADDRESS << 1+0,
REG_CONFIG_GYRO, 1, &temp_data, 1, 100);
    if (ret == HAL_OK)
    {
        printf("Writing to register 27 \r\n");
    }
    else
    {
        printf("Failed writing to the register \r\n");
    }
    // Range accelerometer configuration
}

```

Para la medición del sensor: **stm32f4xx\_hal\_i2c.h**

Registros para la medición del acelerómetro.

Para calcular bien las mediciones debemos encapsular dos registros.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59								ACCEL_XOUT[15:8]
3C	60								ACCEL_XOUT[7:0]
3D	61								ACCEL_YOUT[15:8]
3E	62								ACCEL_YOUT[7:0]
3F	63								ACCEL_ZOUT[15:8]
40	64								ACCEL_ZOUT[7:0]

## MPU6050.c

```

/*
 * MPU6050.c
 *
 * Created on: Feb 1, 2025
 * Author: David Evangelista
 */
#include <MPU6050.h>
#include <main.h>
#include <stdio.h>
extern I2C_HandleTypeDef hi2c1;
void mpu6050_Init()
{
    HAL_StatusTypeDef ret = HAL_I2C_IsDeviceReady(&hi2c1, DEVICE_ADDRESS
<< 1+0, 1, 100);
    if (ret == HAL_OK)
    {
        printf("The device is ready \r\n");
    }
    else
    {
        printf("The device is not ready, Check cables \r\n");
    }
    uint8_t temp_data;
    ret = HAL_I2C_Mem_Write(&hi2c1, DEVICE_ADDRESS << 1+0,
REG_CONFIG_GYRO, 1, &temp_data, 1, 100);
    if (ret == HAL_OK)
    {
        printf("Configuring gyroscope \r\n");
    }
    else
    {
        printf("Failed to configure gyroscope \r\n");
    }
    // Range accelerometer configuration
    temp_data = FS_ACC_4G;
    ret = HAL_I2C_Mem_Write(&hi2c1, DEVICE_ADDRESS << 1+0, REG_CONFIG_ACC,
1, &temp_data, 1, 100);
    if (ret == HAL_OK)
    {
        printf("Configuring accelerometer \r\n");
    }
    else

```

```

    {
        printf("Failed to configure the accelerometer \r\n");
    }
    temp_data = 0;
    ret = HAL_I2C_Mem_Write(&hi2c1, DEVICE_ADDRESS << 1+0, REG_USR_CTRL,
1, &temp_data, 1, 100);
    if (ret == HAL_OK)
    {
        printf("Exiting from sleep mode \r\n");
    }
    else
    {
        printf("Failed to exit from sleep mode \r\n");
    }
}
void mpu6050_Read()
{
    // Buffer
    uint8_t data[2];
    // Every axis have 2 bytes
    int16_t x_acc;
    HAL_I2C_Mem_Read(&hi2c1, DEVICE_ADDRESS << 1+0, REG_DATA, 1, data, 2,
100);
    x_acc = ((int16_t) data[0] << 8) + data[1];
    printf("X axis acceleration: %d \r\n", x_acc);
}

```

## MPU6050.h

```

/*
* MPU6050.h
*
* Created on: Feb 1, 2025
* Author: David Evangelista
*/
#ifndef INC_MPU6050_H_
#define INC_MPU6050_H_
// Define macros
// We use them in C program
#define DEVICE_ADDRESS 0x68
// Values range gyroscope
#define FS_GYRO_250          0
#define FS_GYRO_500           8
#define FS_GYRO_1000          9
#define FS_GYRO_2000          10
#define FS_ACC_2G              0
#define FS_ACC_4G              8
#define FS_ACC_8G              9
#define FS_ACC_16G             10
//Register to configure gyroscope
#define REG_CONFIG_GYRO       27
#define REG_CONFIG_ACC         28

```

```

#define REG_USR_CTRL          107
#define REG_DATA              59
void mpu6050_Init();
void mpu6050_Read();
#endif /* INC_MPU6050_H_ */

```

### Resultado:

Medición del acelerómetro en el eje X



```

Console Memory Console X
STM32 (CONNECTED)
X axis acceleration: -44
X axis acceleration: -48
X axis acceleration: -64
X axis acceleration: -22
X axis acceleration: -42
X axis acceleration: 10
X axis acceleration: 42
X axis acceleration: -22
X axis acceleration: -56
X axis acceleration: -14

```

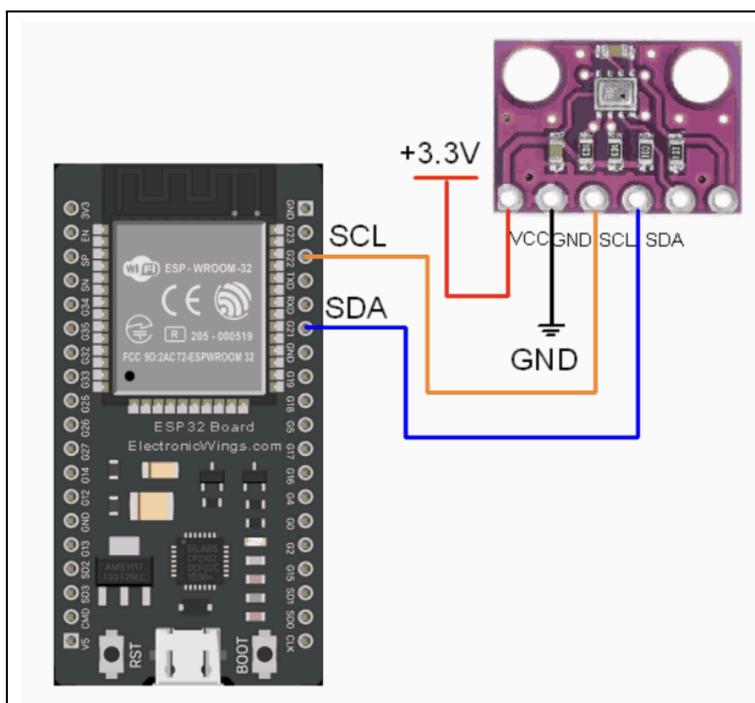
Activar Windows  
Ve a Configuración para activar Windows.

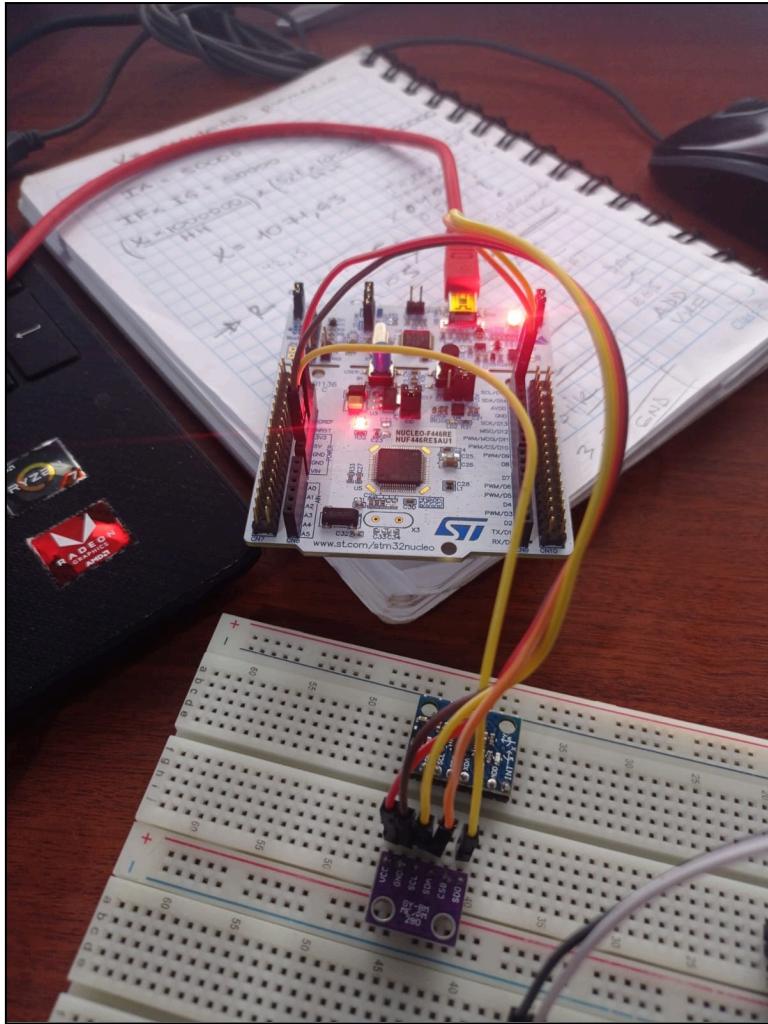
## Reunión 08/02/25

Continuar con la programación de los sensores y completar la guía.  
Revisar el tema del banco de baterías, BMS.

## 11/02/25

Revisión del sensor de presión BMP 280. Recordando también pin SDO a GND.





## BMP280

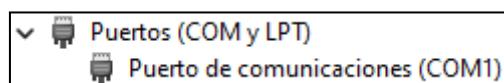
Referencia:

<https://naylampmechatronics.com/sensores-posicion-iniciales-gps/358-sensor-de-presion-bmp280.html>

### Dirección I2C del MPU:

0x76: para el funcionamiento

Para la comunicación con el ESP32 usamos el puerto COM con la siguiente denominación.



### Referencia Github (*Librería*):

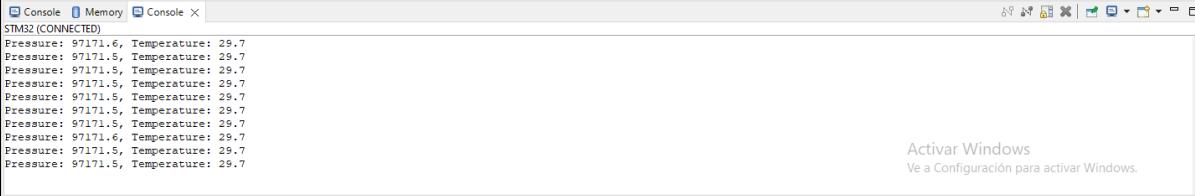
[https://github.com/ProjectOfficial/STM32/tree/main/STM32\\_I2C](https://github.com/ProjectOfficial/STM32/tree/main/STM32_I2C)

### Referencia:

<https://www.youtube.com/watch?v=Yjz0yBZ0LGA>

## Resultado:

Medición de presión y temperatura:



```
STM32 (CONNECTED)
Pressure: 97171.6, Temperature: 29.7
Pressure: 97171.5, Temperature: 29.7
Pressure: 97171.6, Temperature: 29.7
Pressure: 97171.5, Temperature: 29.7
Pressure: 97171.5, Temperature: 29.7
```

Activar Windows  
Ve a Configuración para activar Windows.

**15/02/25**

**Tarjeta micro SD**

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

**Referencia:**

<https://randomnerdtutorials.com/esp32-microsd-card-arduino/>

<https://www.youtube.com/watch?v=spVIZO-jbxE>

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

**Github:**

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

**16/02/25**

Configuración del SPI

**Pinout & Configuration**

**Clock Configuration**

**Software Packs**

SPI2 Mode and Configuration

Mode: Full-Duplex Master

Hardware NSS Signal: Disable

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings

Parameter Settings | User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

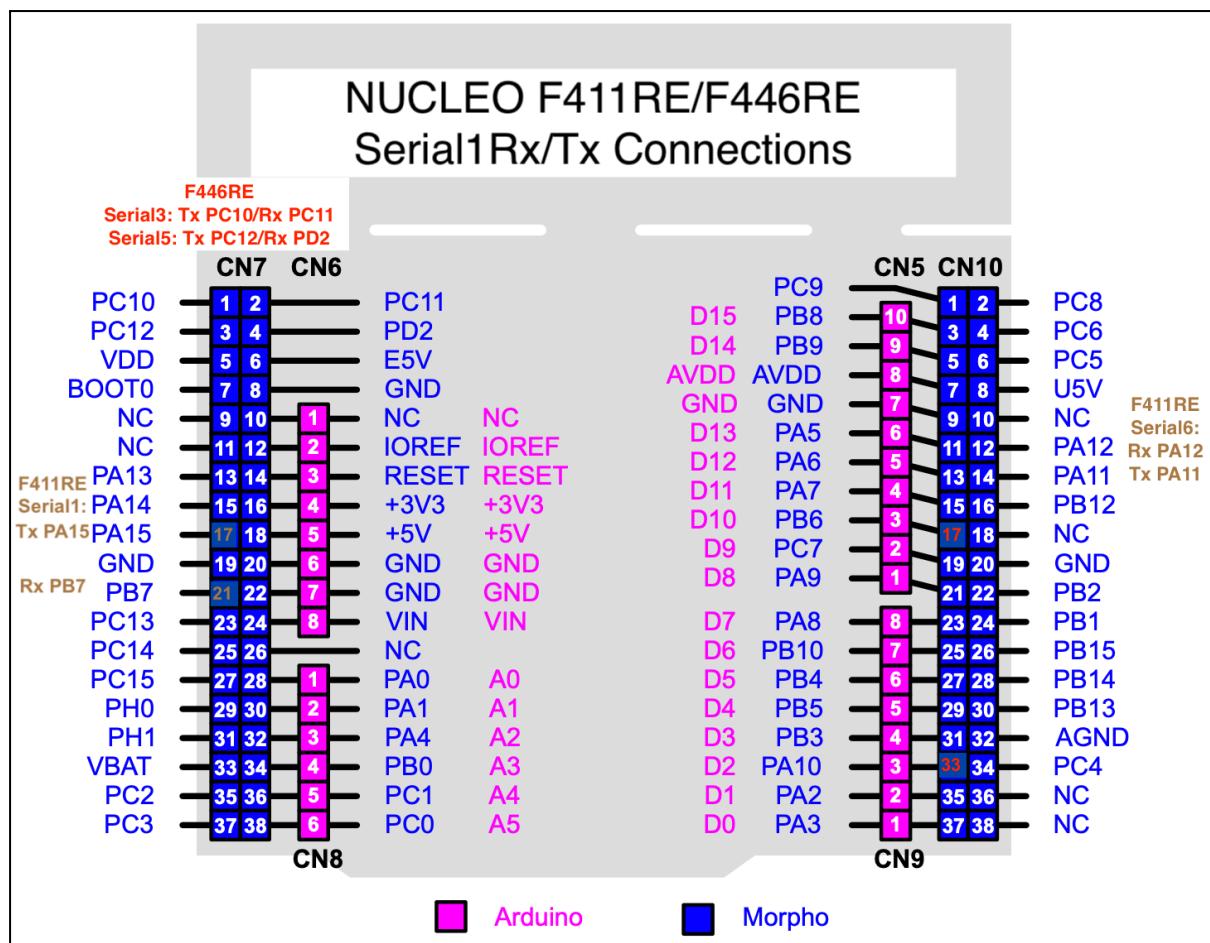
Clock Parameters

Prescaler (for Baud Rate)	8
Baud Rate	5.625 MBits/s

El bitrate debe ser menor a 10 MBits/s

**Data Size: 8 bits**

**Configuración de acuerdo con la placa**



### Configuración de GPIO Output (CS)

GPIO Mode and Configuration

Configuration

Group By Peripherals

<input checked="" type="checkbox"/> SPI	<input checked="" type="checkbox"/> SYS	<input checked="" type="checkbox"/> USART
<input checked="" type="checkbox"/> GPIO	<input checked="" type="checkbox"/> Single Mapped Signals	<input checked="" type="checkbox"/> I2C
<input checked="" type="checkbox"/> RCC		

Search Signals

Show only Modified Pins

Pin ...	Signal o...	GPIO ou...	GPIO m...	GPIO P...	Maximu...	User Label	Modified
PA5	n/a	Low	Output ...	No pull-...	Low		<input type="checkbox"/>
PC0	n/a	High	Output ...	No pull-...	Medium		<input checked="" type="checkbox"/>

PC0 Configuration :

GPIO output level	High
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	Medium
User Label	

### Configuración adicional (USE\_LFN, MAX\_SS)

**Pinout & Configuration**

**Clock Configuration**

**Software Packs**

FATFS Mode and Configuration

Mode

SD Card

USB Disk

User-defined

Configuration

Reset Configuration

Set Defines    User Constants

Configure the below parameters :

Search (Ctrl+F)

USE\_LFN (Use Long Fil... Enabled with static working buffer o...)

MAX\_LFN (Max Long Fil... 255)

LFN\_UNICODE (Enable ... ANSI/OEM)

STRF\_ENCODE (Charac... UTF-8)

FS\_RPATH (Relative Path) Disabled

Physical Drive Parameters

VOLUMES (Logical drives) 1

MAX\_SS (Maximum Sec... 4096)

MIN\_SS (Minimum Sect... 512)

MULTI\_PARTITION (Volu... Disabled)

USE\_TRIM (Erase feature) Disabled

FS\_NOFSINFO (Force fu... 0)

System Parameters

**MAX\_SS (Maximum Sector Size)**

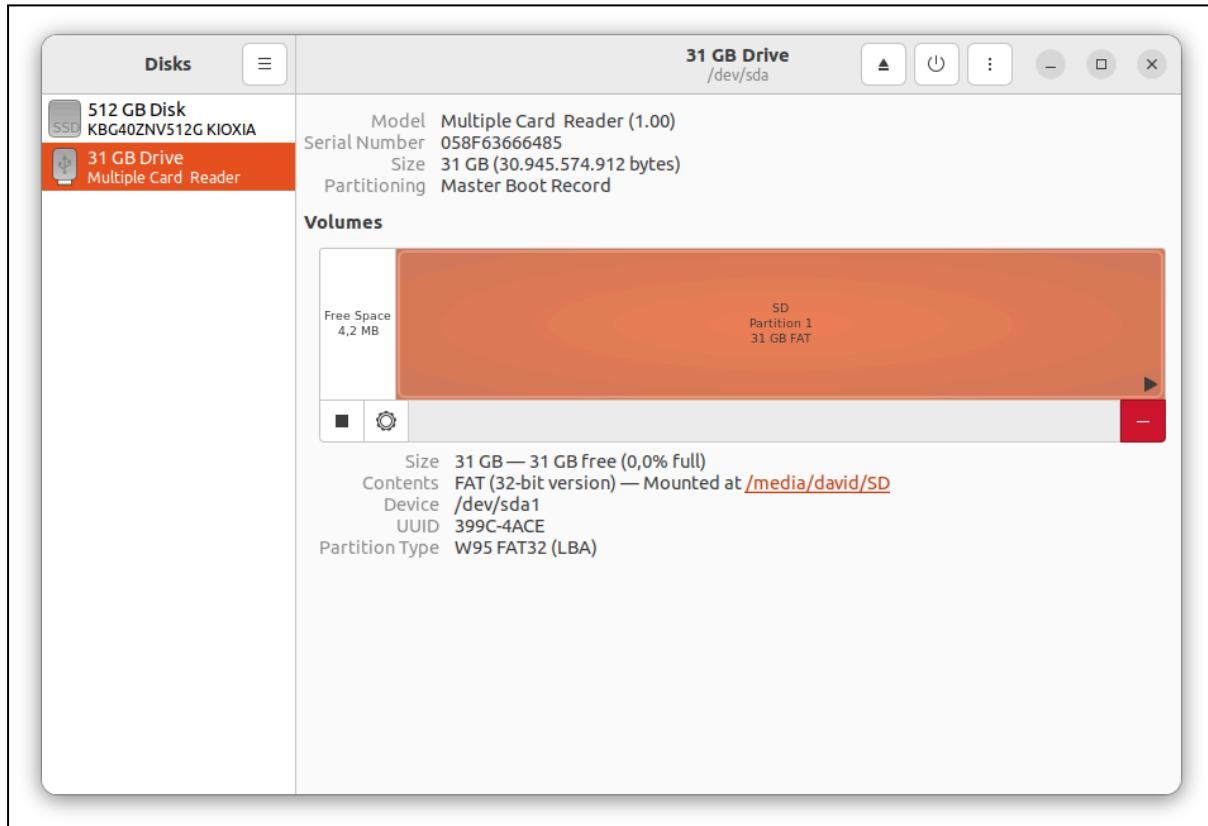
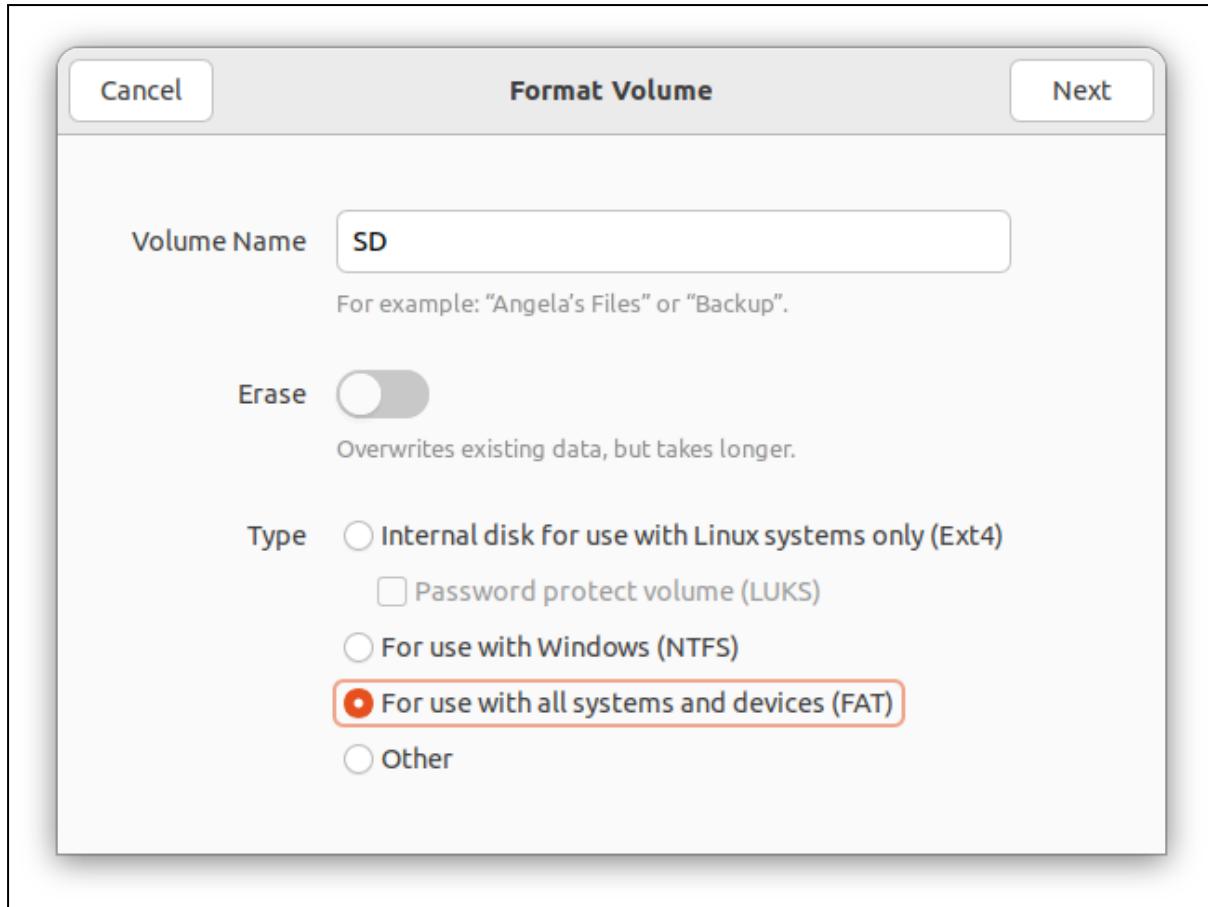
MAX\_SS (Maximum Sector Size)

Parameter Description:

Falta asignación de pines.

**17/02/25**

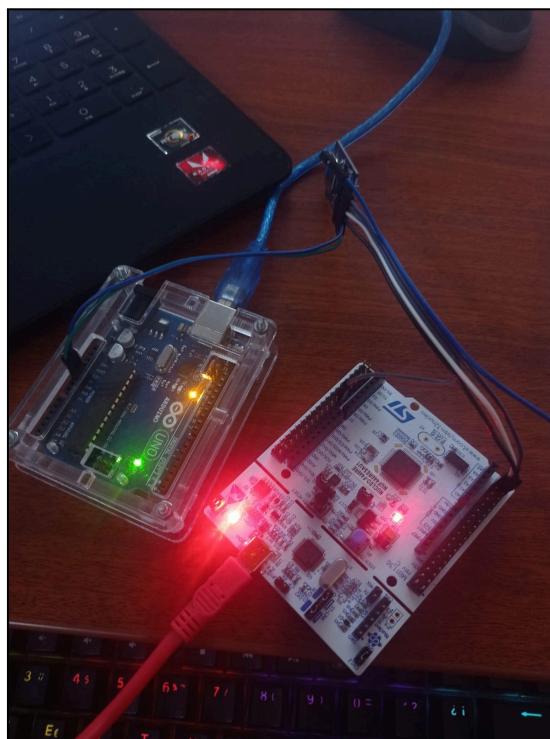
Revisar el voltaje en el módulo debe ser externo para poder alimentar correctamente.  
Formatear en FAT32 (celular).



**Referencia:** <https://controllerstech.com/sd-card-using-spi-in-stm32/>

## Resultado:

## Operaciones con la SD Card.



27/02/25

## Lectura de todos los ejes del MPU6050.

## **Creación de los registros de datos X, Y**

```
//Register to configure gyroscope
#define REG_CONFIG_GYRO      27
#define REG_CONFIG_ACC       28
#define REG_USR_CTRL        107
#define REG_DATA_X           59
#define REG_DATA_Y           61
#define REG_DATA_Z           63
```

## Resultado:



```
STM32 (CONNECTED)
X: 36, Y: -14
X: 48, Y: -4
X: 6, Y: 8
X: 48, Y: -34
X: 42, Y: -30
X: 58, Y: -4
X: 40, Y: 24
X: 56, Y: -12
X: 10, Y: -16
X: 116, Y: -32
X: 12, Y: -36
X: 30, Y: -6
X: 28, Y: -6
X: 48, Y: 42
```

Activar Windows  
Ve a Configuración para activar Windows.

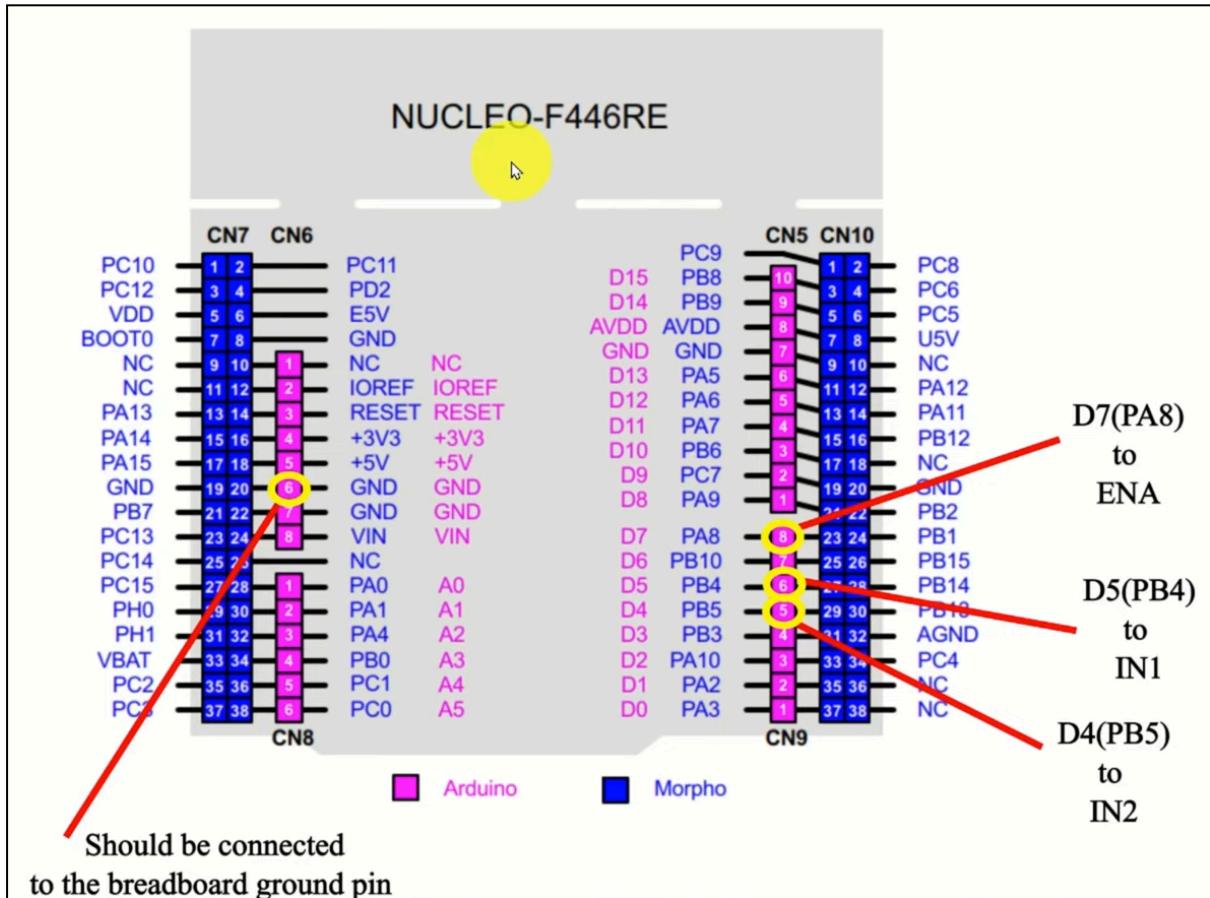
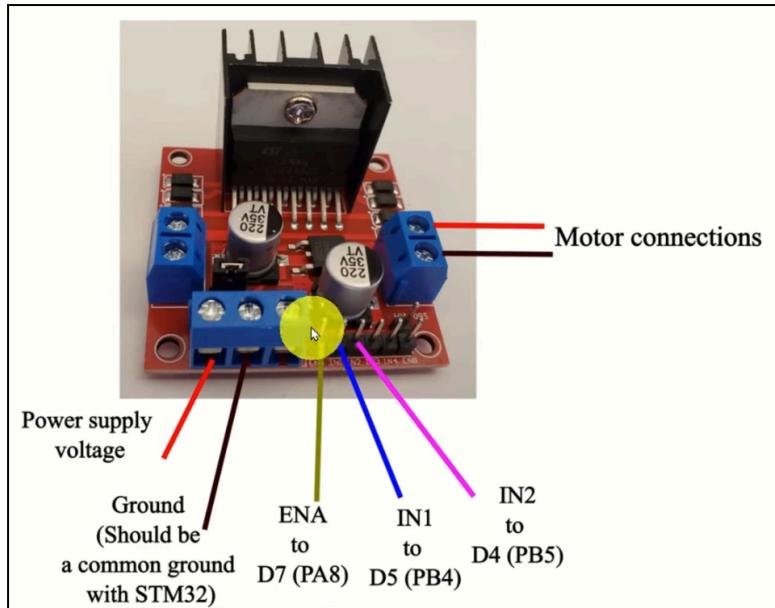
## Código **MPU6050.c**

```
void mpu6050_Read()
{
    // Buffer
    uint8_t data[2];
    // Every axis have 2 bytes
    int16_t x_acc;
    int16_t y_acc;
    HAL_I2C_Mem_Read(&hi2c1, (DEVICE_ADDRESS << 1) + 0, REG_DATA_X, 1,
data, 2, 100);
    x_acc = ((int16_t)data[0] << 8) + data[1];
    HAL_I2C_Mem_Read(&hi2c1, (DEVICE_ADDRESS << 1) + 0, REG_DATA_Y, 1,
data, 2, 100);
    y_acc = ((int16_t)data[0] << 8) + data[1];
    printf("X: %d, Y: %d\r\n", x_acc, y_acc);
}
```

## Control de motor DC

Referencia:

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



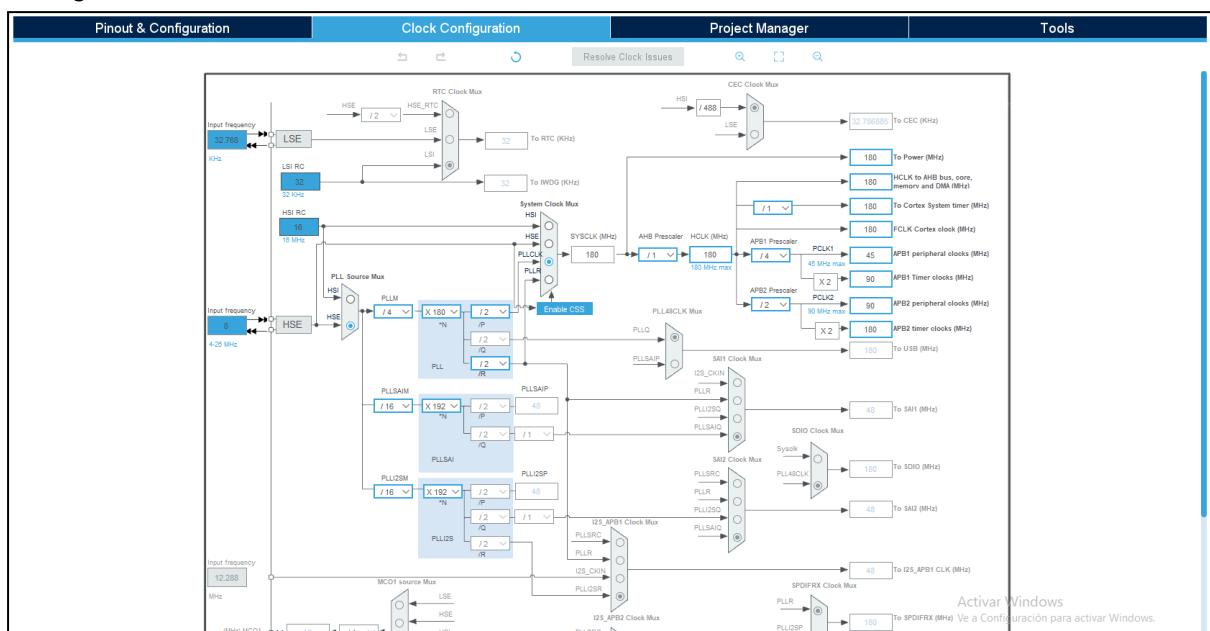
In the STM32 development environment, we select the frequency of the PWM signal by using the three parameters  $f_{TC}$ ,  $k_{PS}$ , and  $k_P$ , and the following equation

$$f_{PWM} = \frac{f_{TC}}{(k_{PS} + 1)(k_P + 1)} \quad (3)$$

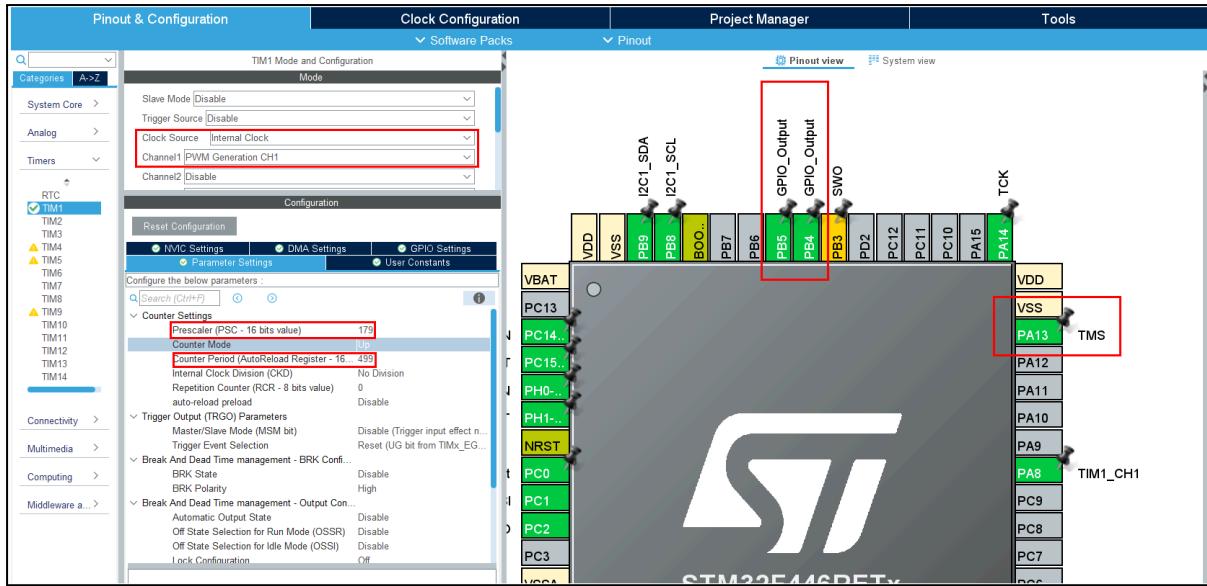
where

- $f_{TC}$  is the frequency of a timer clock of an STM32 microcontroller.
- $k_{PS}$  is the prescaler value.
- $k_P$  is the period value.

## Configuración del CLK



## Configuración del TIMER para la generación de la señal PWM



## Definición del Duty Cycle

In the STM32CubeIDE environment, the duty cycle is determined by

$$D = \frac{CCR}{ARR + 1} \quad (4)$$

where **CCR is the Capture/Compare Register (CCR)**. We select the value of *CCR* in the interval  $0, 1, \dots, ARR + 1$ .

Cálculos:

$f_{pwm} = 2000 \text{ Hz}$   
 $f_{TC} = 180 \times 10^6$   
 $k_{\text{prescaler}} = 179$   
 $(\text{prescaler})$   
 $k_p = ?$   
 $(\text{period})$   
 $2000 = \frac{180 \times 10^6}{(179 \times f_{TC}) (k_p + 1)}$   
 $k_p + 1 = \frac{10^6}{2000}$   
 $k_p = 499$   
 ARR  
 $CCR = 0, 1, \dots, ARR + 1$   
 $D = \frac{CCR}{ARR}$

Código:

```

HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 0);
TIM1->CCR1 = 250;
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
HAL_Delay(3000);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
TIM1->CCR1 = 250;
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
HAL_Delay(3000);
    
```

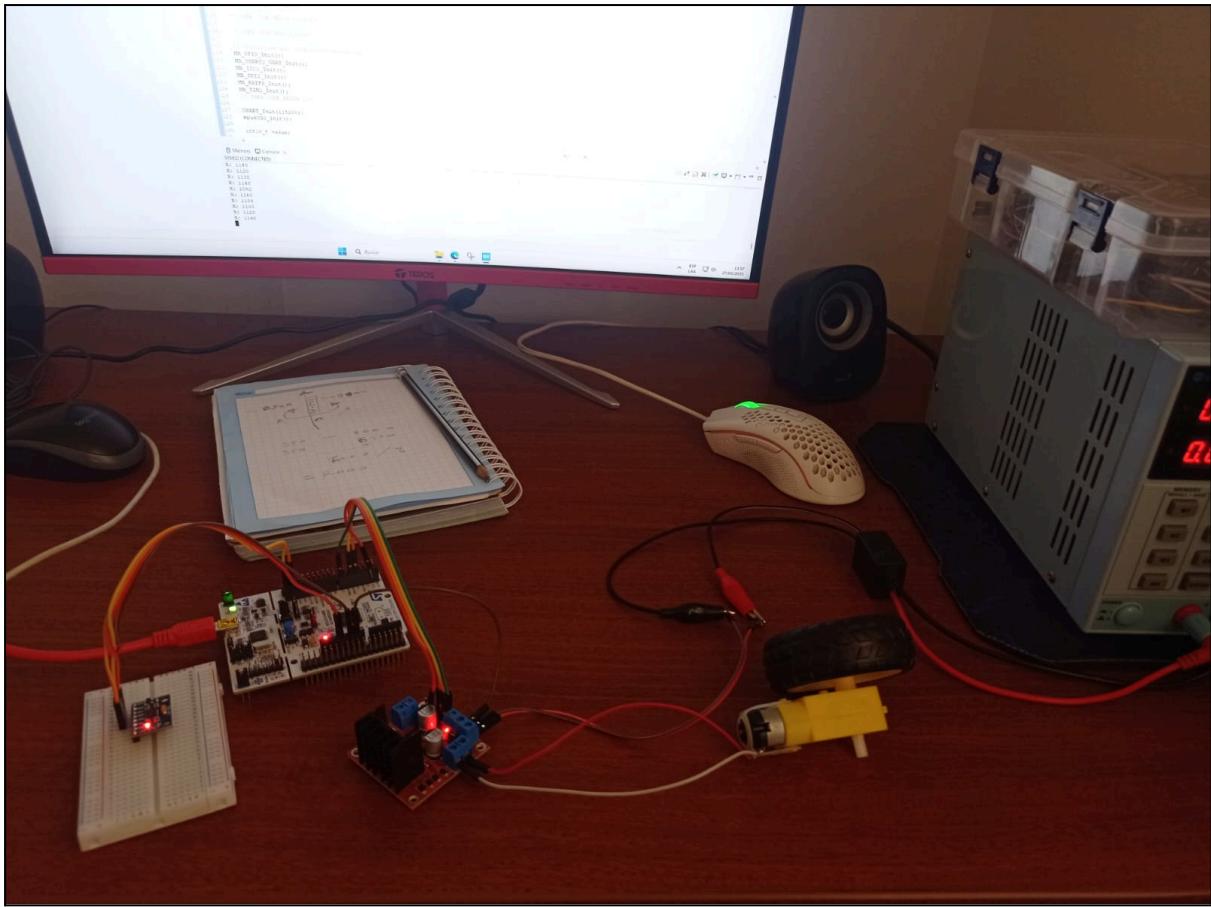
Observación del motor usado

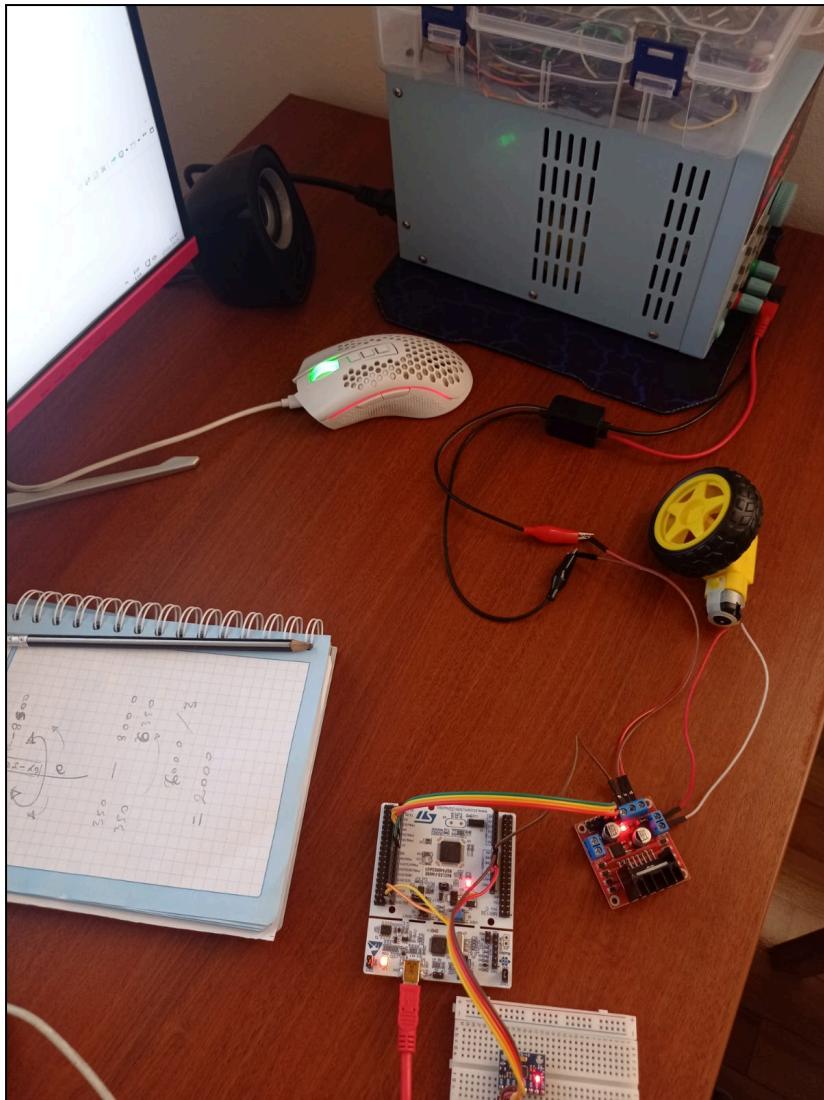
**Voltaje máximo:** +6V

Referencia:

<https://naylampmechatronics.com/motores-dc/606-motor-dc-6v200rpm.html>

Resultados finales





## Programa final

```
value = mpu6050_Read();
if (value>350)
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,1);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5,0);
    if (value<2000)
    {
        TIM1->CCR1 = 125;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }else if (value<4000)
    {
        TIM1->CCR1 = 250;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }else if (value<6000)
    {
        TIM1->CCR1 = 375;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }else
```

```

{
    TIM1->CCR1 = 500;
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
}
}else if (value<-350)
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
    if (value>-2000)
    {
        TIM1->CCR1 = 125;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }else if (value>-4000)
    {
        TIM1->CCR1 = 250;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }else if (value>-6000)
    {
        TIM1->CCR1 = 375;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }else
    {
        TIM1->CCR1 = 500;
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    }
}
HAL_Delay(500);

```

## Observaciones

- Tener en cuenta el tipo de dato a emplear se tuvo problemas con el signo del dato por usar uint16\_t en lugar de int16\_t.
- Para usar int16\_t debemos importar la librería #include<stdint.h>.

**06/03/25**

**Motor usado:**

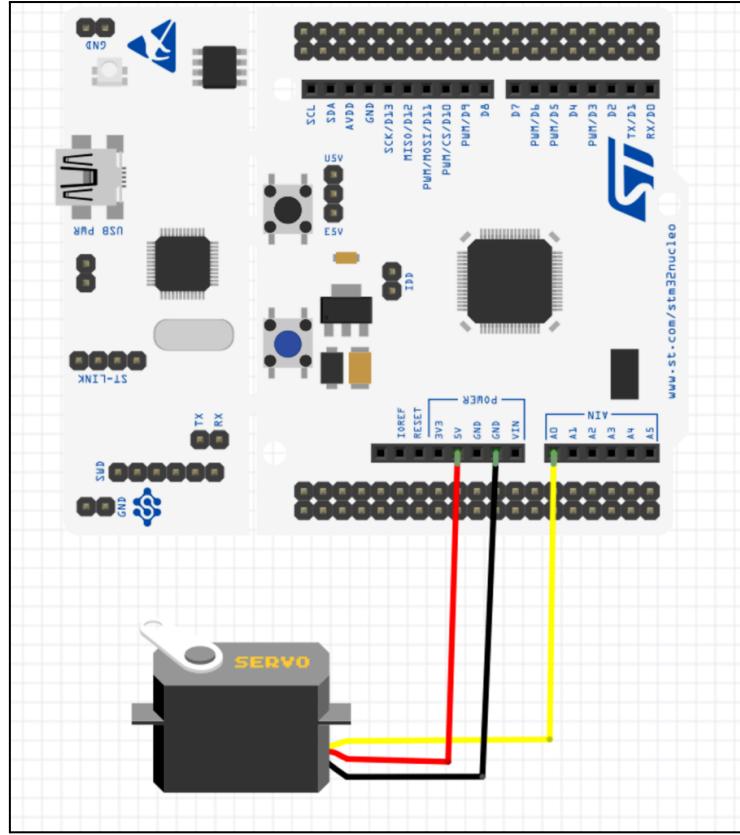
<https://naylampmechatronics.com/servomotores/780-servo-rds3235-35kg.html>



El Servo RDS3235 pertenece a la gama de servos RDS especialmente diseñados para aplicaciones en robótica, posee reducción metálica e incluye brackets metálicos que facilitan la conexión entre articulaciones. Ideal para proyectos de brazos robóticos o manipuladores seriales, bípedos o humanoides/artrópodos en general. Puede rotar aproximadamente 270 grados (135° en cada dirección). Tiene la facilidad de poder trabajar con diversidad de plataformas de desarrollo como Arduino, PICs, Raspberry Pi, o en general a cualquier microcontrolador.

Para su uso con Arduino, recomendamos conectar el cable naranja al pin 9 o 10 y usar la Librería "Servo" incluida en el IDE de Arduino. Para la posición -45° el pulso es de 0.5ms, para 90° es de 1.5ms y para 225° 2.5ms. Los cables en el conector están distribuidos de la siguiente forma: Marrón = Tierra (GND), Rojo = VCC (6-8.4V), Naranja = Señal de control (PWM).

Debido a la potencia del Servo RDS3235 es necesario alimentarlo con una fuente separada de +8.4V y con capacidad de entregar por lo menos 5A por cada servo conectado. Se deben unir las "tierras" de la fuente de los servos y del microcontrolador para tener una "tierra común". El pin de señal de control va conectado directamente al microcontrolador o Arduino pues solo es de datos y requiere muy poca corriente. Para evitar problemas de ruido eléctrico es recomendable agregar un capacitor de por lo menos 470uF entre la alimentación del Servo y GND.



**15/03/25**

Reunión APU Space

## **23/03/25**

Puntos por hacer:

- Búsqueda de librerías relacionados con los sensores del sistema principal.
- Programación de los sensores y STM32 del sistema principal (Condición: previa entrega de los componentes electrónicos).

### **Sensor BNO085**

Módulo de sensor 9 ejes de alta precisión acelerómetro Gyro Magnetómetro para realidad virtual 3D.

**Github:** (contiene otras librerías BNO08x)

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

### **Sensor MPL3115A2**

Sensor Presión barométrica/Altitud/Temperatura.

**Github:** (contienen .c y .h)

<https://github.com/psas/stm32/tree/master/common/devices>

<https://github.com/talhasevinc/STM32>

<https://github.com/KwintenSchram/Ambient>

### **Sensor SHT31**

Sensor de Temperatura y Humedad Relativa.

**Github:** (consideramos porque indican el modo de uso)

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

<https://github.com/henriheimann/stm32-hal-sht3x>

**Github:** (contiene .c y .h con ejemplos, buscar en sht3x)

<https://github.com/Sensirion/embedded-sht>

### **Micro SD Storage Board TF Card**

3.3 V Reader Memory Shield Module

(Se realizó las pruebas con el microcontrolador)

### **TRANSCEIVER LORA 433MHZ E220-400T30D - LLCC68**

UART 1W SMA

**Github:** (librería usada para el modelo E220, aunque está en lenguaje C)

[https://github.com/xreef/EByte\\_LoRa\\_E220\\_Series\\_Library/blob/master/library.json](https://github.com/xreef/EByte_LoRa_E220_Series_Library/blob/master/library.json)

Sensor de corriente y voltaje

<https://naylampmechatronics.com/sensores-corriente-voltaje/559-monitor-de-corriente-voltaje-high-side.html>

**29/04/25**

## Selección de pines para el STM32 con los sensores a utilizar.

Núcleo usado:

**STM32 NUCLEO H755ZI-Q**

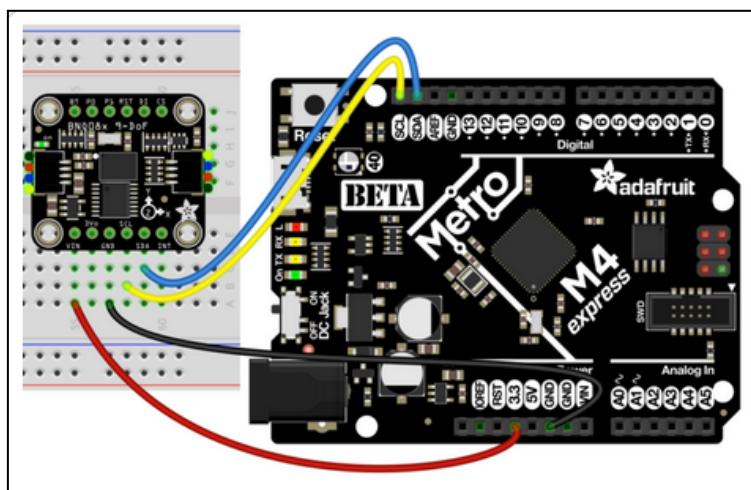
Ref: <https://os.mbed.com/platforms/ST-Nucleo-H743ZI/>

Sensores a utilizar:

**BNO085**

Ref:

<https://learn.adafruit.com/adafruit-9-dof-orientation-imu-fusion-breakout-bno085?view=all>



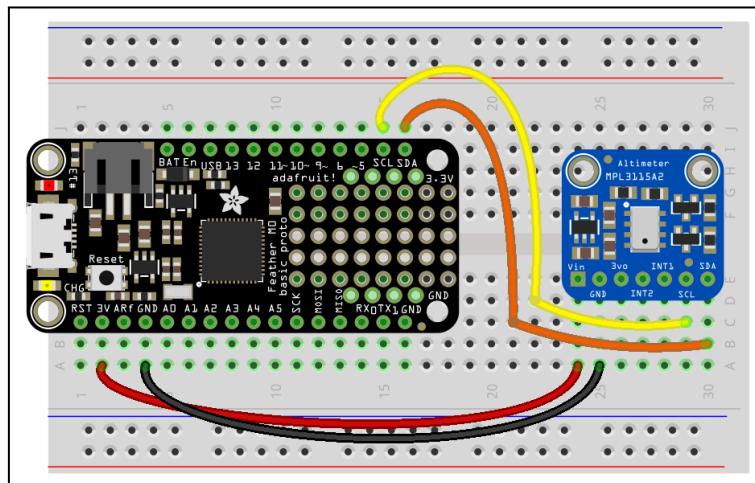
Pines necesarios para utilizarlo con el protocolo I2C.

Utilizando el núcleo **M4 (I2C1)**

<b>SDA</b>	<b>PB7</b>
<b>SCL</b>	<b>PB6</b>
<b>VIN</b>	3.3V
<b>GND</b>	GND

## MPL3115A2

Ref: <https://learn.adafruit.com/using-mpl3115a2-with-circuitpython/hardware>

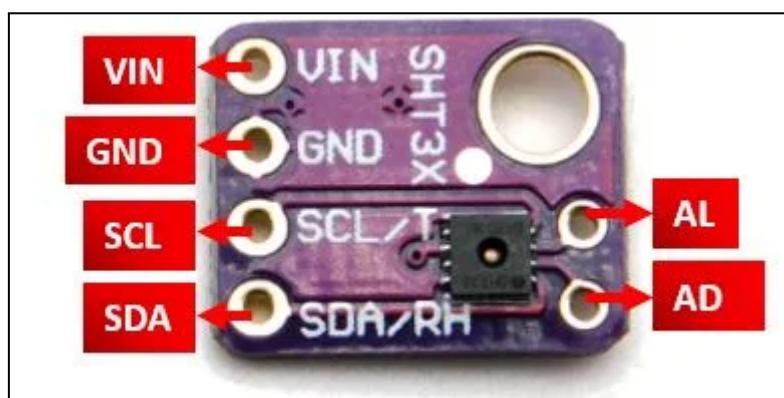


Pines necesarios para utilizarlo con el protocolo I2C.

Utilizando el núcleo **M4 (I2C2)**

SDA	<b>PB11</b>
SCL	<b>PB10</b>
VIN	3.3V
GND	GND

## SHT31



Pines necesarios para utilizarlo con el protocolo I2C.

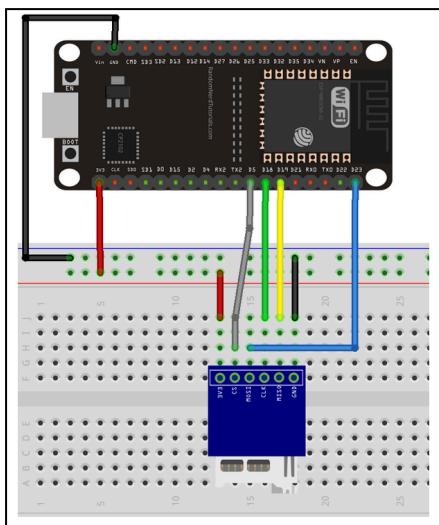
Utilizando el núcleo **M4 (I2C3)**

SDA	<b>PB7</b>
-----	------------

<b>SCL</b>	<b>PB6</b>
<b>VIN</b>	3.3V
<b>GND</b>	GND

### Micro SD Card

Ref: <https://randomnerdtutorials.com/esp32-microsd-card-arduino/>



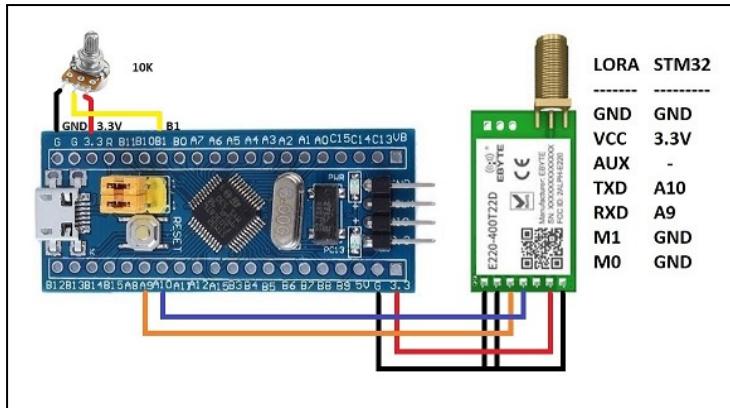
Pines necesarios para utilizarlo con el protocolo SPI.

Utilizando el núcleo **M4 (SPI1)**

<b>MISO</b>	<b>PA6</b>
<b>MOSI</b>	<b>PD7</b>
<b>CLK</b>	<b>PA5</b>
<b>CS</b>	<b>PA14</b>
<b>VIN</b>	3.3V
<b>GND</b>	GND

### Módulo LORA

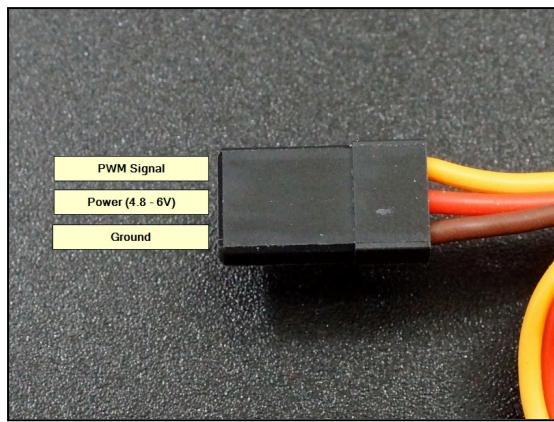
Ref: <https://www.micropeta.com/video2>



Pines necesarios para utilizarlo con el protocolo UART.  
Utilizando el núcleo **M4 (USART2)**

<b>VCC</b>	3.3V
<b>GND</b>	GND
<b>AUX</b>	-
<b>TXD</b>	<b>PD5</b>
<b>RXD</b>	<b>PA3</b>
<b>M1</b>	GND
<b>M0</b>	GND

## Servomotor



Pines necesarios para utilizar el TIMER y generar la señal PWM.  
Utilizando el núcleo **M4 (TIM1)**

PWM	<b>PE9</b>
VCC	-
GND	GND

## **09/05/25**

### **Encender LED en la placa**

Trabajamos con el núcleo M4.

Trabajamos con el pin PB1.

**Establecer las configuración del pin sin importar si siguen por default.**

Como es la primera vez de uso de la placa se realizó una actualización del firmware donde se dejó todo por defecto.

Una vez hecho esto ya se puede usar bien el *Debug Configuration*, porque antes no creaba bien la configuración.

Ahora sí, se logra depurar correctamente.

Crear nuevo proyecto, trabajar primero con M7 y trabajar primero el LED ver si cumple lo que estamos haciendo con el pin.

## **10/05/25**

**Ref:** <https://www.youtube.com/watch?v=jl1k6p-fduE&t=144s>

**Importante configuración Debug.**

PE1 AMARILLO PARA EL M4

PB14 ROJO PARA EL M7