

# Informe de Práctica 2: Chequeo del convertidor A/D y consola serie

Máximo García Aroca

April 12, 2024

## 1 Introducción

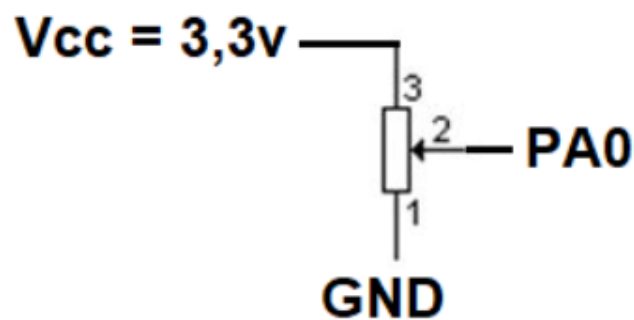
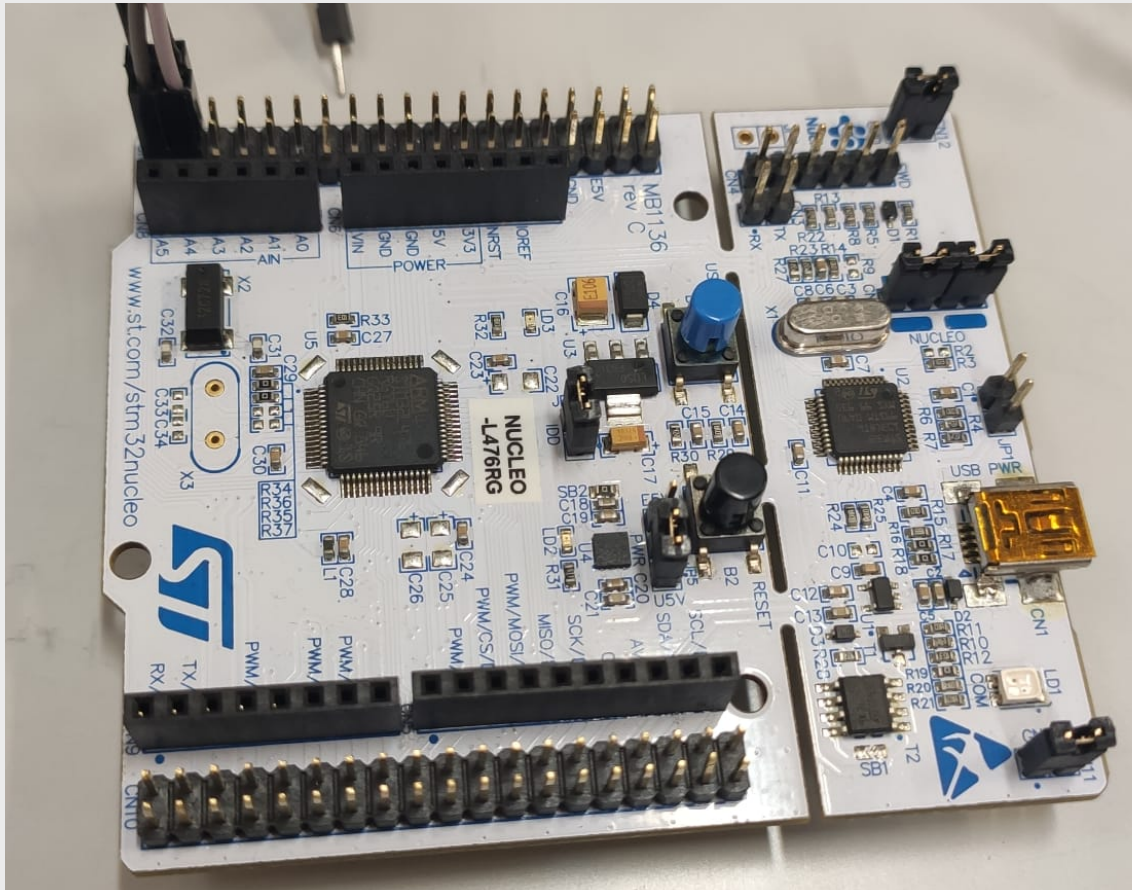
En esta práctica hemos usado un sensor semiconductor de humos. La idea de ésta es que seamos capaces de visualizar los datos que registra el sensor en una consola serie, lo cual podría ser una función de utilidad en un caso práctico de sensor de humos.

El sensor es muy sensible al gas, es de bajo coste y es muy duradero además de tener una circuitería muy simple de entender.

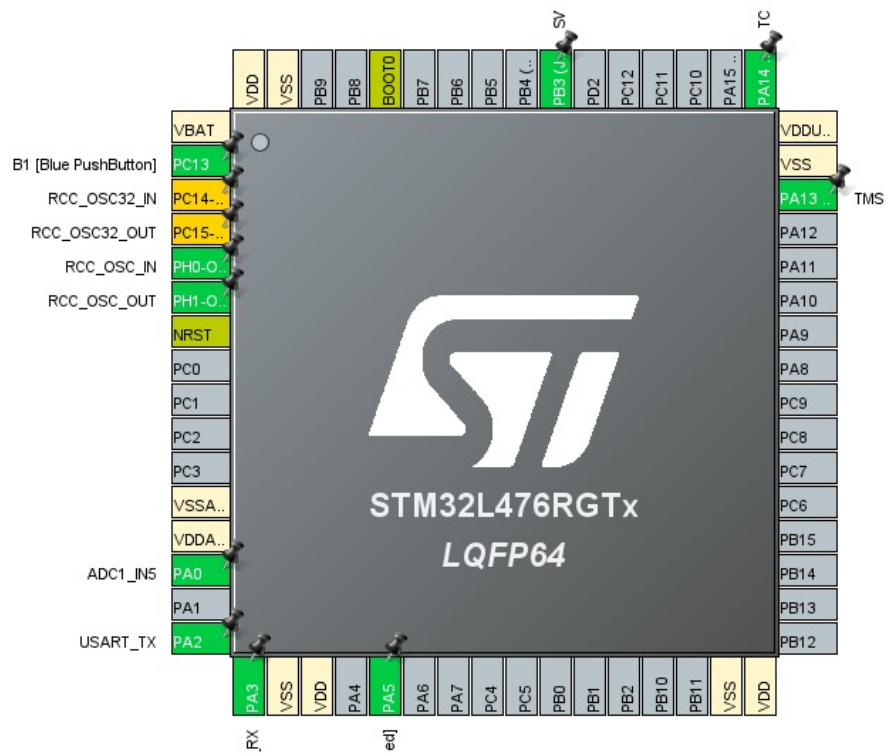
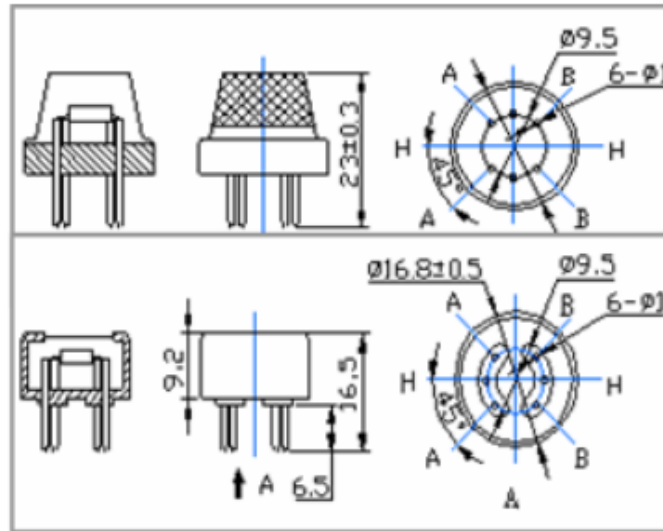
Todo el proceso lo he llevado a cabo junto con mi compañero de prácticas Sergio Maximiliano.

## 2 Diseño físico

Hemos usado como siempre la placa L476RG, de la cual hemos usado tres puertos como se nos indica en el siguiente diagrama: el puerto de 3.3V del arduino, también uno de los GND y el pin PA.0 el cual está configurado como salida. Cada puerto de la placa ha sido conectado a los pines del sensor tal y como se indica:



## Configuration



ADC1\_IN5



### 3 Pasos

Lo primero ha sido crear el proyecto en el software STM32Cube.

Después hemos habilitado el pin PA\_0 con ADC1 entrada, el cual nos permite conectar una tensión externa de entrada. Seleccionamos en ADC1 la entrada IN5 como entrada single-ended. Luego hemos seguido la práctica donde se nos indicaba *"En clock configuration seleccionar el reloj PLLCLK y seleccionar para el convertidor ADC el reloj del sistema marcando SYSCLK y en System Clock Mux la opción PLLCLK y en el reloj HLCK teclear 40 Mhz. En el apartado RCC seleccionar en High Speed Clock (HSE) la opción Crystal/Ceramic Resonator y en LSE la opción Disable."*

The image shows two screenshots from the STM32CubeMX software. The top screenshot is the 'ADC1 Mode and Configuration' window. It has a 'Mode' section with a list of inputs from IN1 to IN10. IN5 is set to 'IN5 Single-ended', while all others are 'Disable'. The bottom screenshot is the 'RCC Mode and Configuration' window. It has a 'Mode' section with 'High Speed Clock (HSE)' set to 'Crystal/Ceramic Resonator' and 'Low Speed Clock (LSE)' set to 'Disable'. There are also checkboxes for 'Master Clock Output', 'LSCO Clock Output', 'SAI1 Extern CLock', and 'SAI2 Extern CLock', all of which are currently unchecked.

ADC1 Mode and Configuration	
Mode	
IN1	Disable
IN2	Disable
IN3	Disable
IN4	Disable
IN5	IN5 Single-ended
IN6	Disable
IN7	Disable
IN8	Disable
IN9	Disable
IN10	Disable

RCC Mode and Configuration	
Mode	
High Speed Clock (HSE)	Crystal/Ceramic Resonator
Low Speed Clock (LSE)	Disable
<input type="checkbox"/> Master Clock Output	
<input type="checkbox"/> LSCO Clock Output	
<input type="checkbox"/> SAI1 Extern CLock	
<input type="checkbox"/> SAI2 Extern CLock	

### SYS Mode and Configuration

Mode

Debug Trace Asynchronous Sw

☒ System Wake-Up 1

☒ System Wake-Up 2

☒ System Wake-Up 4

☐ System Wake-Up 5

Power Voltage Detector In Disable

Timebase Source SysTick

Pinout & Configuration
Clock Configuration
Project Manager
Tools

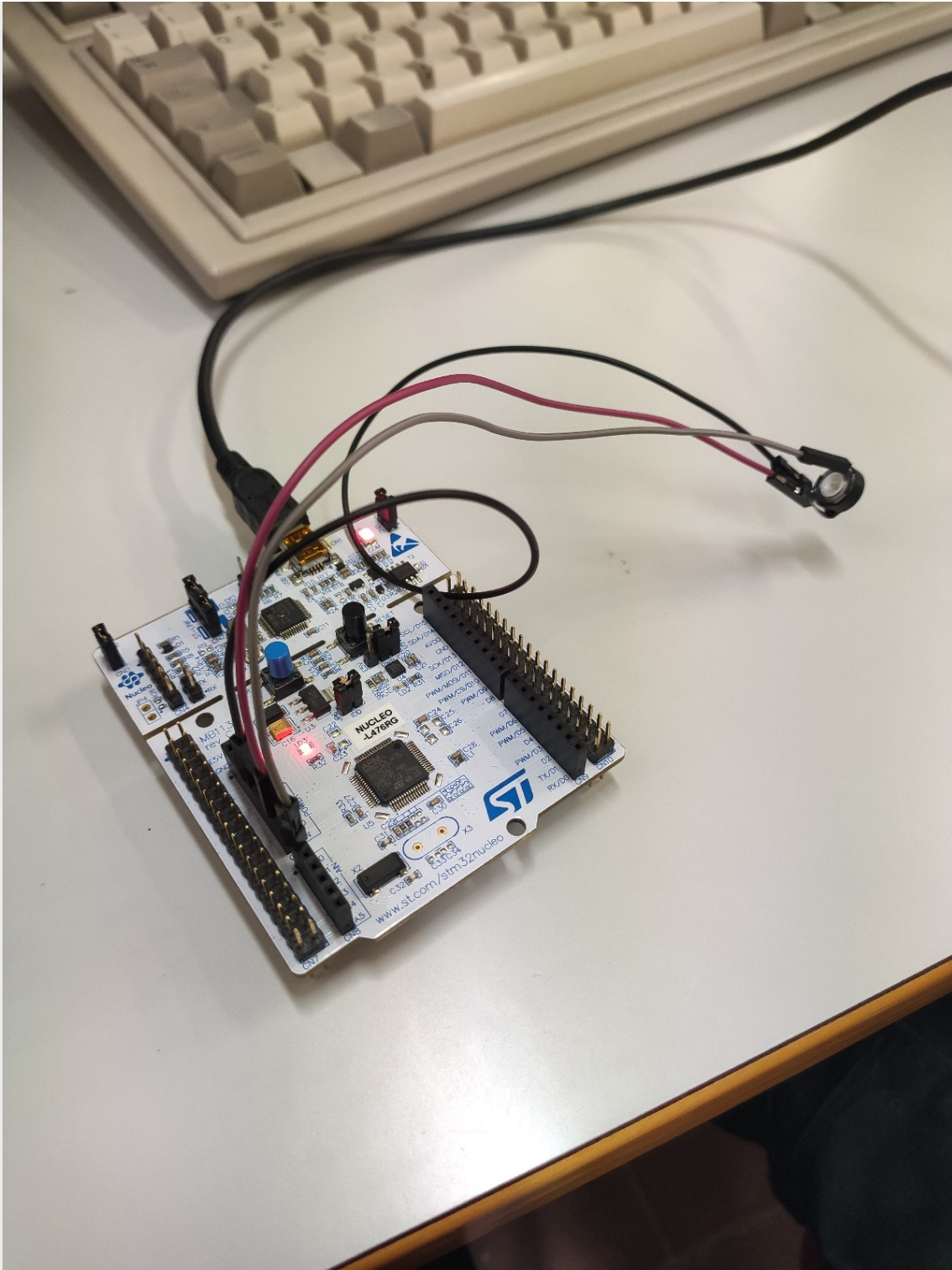
Resolve Clock Issues

The diagram illustrates the clock architecture of the system. It starts with two input frequencies: 32.768 kHz (LSE) and 4-48 MHz (HSE). The LSE is used for RTC and IWDG. The HSE is used for the main PLL (PLLA) and other dividers. The PLLA output is divided to provide clocks for the CPU (PCLK1), APB1 and APB2 (APB1PCLK1, APB2PCLK1), and various peripheral clocks like USB, I2C, ADC, and UART. The diagram shows the division ratios and the resulting clock frequencies for each component.

Después hemos configurado putty para conectarse al COM1 el cual era el puerto de comunicación con la placa.

En cuanto al montaje únicamente hemos tenido que conectar con tres cables a las tres conexiones previamente indicadas.





Para el **montaje** hemos conectado la primera patilla a **GND**, la segunda al puerto **PA\_0** y la tercera a la entrada de tensión de **+3.3V**.

El código ha usado las funciones:

- HAL\_ADC\_Start(hadc1): habilita ADC.
- HAL\_ADC\_PollForConversion(hadc1, ...): indica lectura por del convertidor mediante polling.
- HAL\_ADC\_GetValue(hadc1): obtiene el valor del convertidor (ADC).
- HAL\_UART\_Transmit(huart2, ...): Transmite a la usart la cadena.
- HAL\_Delay(100): realiza un retraso de 100 milisegundos.

Este es el código:

```
/* Includes -----  
#include "main.h"  
#include <string.h>  
#include <stdio.h>  
  
/* Private define -----  
/* USER CODE BEGIN PD */  
#define MAX_LENGTH 30  
/* USER CODE END PD */  
  
/* USER CODE BEGIN Init */  
char str[MAX_LENGTH];  
/* USER CODE END Init */  
  
/* Configure the system clock */  
SystemClock_Config();  
  
/* USER CODE BEGIN SysInit */  
uint32_t dato;  
/* USER CODE END SysInit */
```

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    HAL_ADC_Start(&hadc1); // Inicio del convertidor
    HAL_ADC_PollForConversion(&hadc1, 500); // Lectura mediante polling
    dato = HAL_ADC_GetValue(&hadc1); // Obtención del dato
    sprintf((str), "dato = %lu \n\r", dato); // Copia de cadena string con dato a str
    HAL_UART_Transmit(&huart2, str, strlen(str), 100); // Transmitimos str con el dato en la cadena
    HAL_Delay(100); // Añadimos un delay
    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

```

También hicimos uso como se nos indica en el principio de la práctica de la instrucción *sprintf* para imprimir el valor en el formato adecuado, copiando el valor en la cadena y transmitiéndolo a través del COM.

Al ejecutar el programa y configurar la terminal podemos ver cada 100 milisegundos aproximadamente un valor del sensor de humos. El sensor dispone de una pestaña que al girar aumenta o reduce la sensibilidad del sensor y cambia el valor transmitido lo cual simularía el que el sensor se encontrase en un lugar con más o menos humedad.

```

dato = 3906
dato = 3908
dato = 3907
dato = 3914
dato = 3908
dato = 3909
dato = 3905
dato = 3907
dato = 3908
dato = 3905
dato = 3905

```



