**Informe Práctica 5 SED: COMUNICACIONES SERIE**

**y**

**CONVERTIDORES A/D**

Autores:

* Francisco Javier Perea Vanguelov (52545).
* Lydia Vega Ochoa (52654).
* Alberto Martínez Trapiello (52713).

Grupo: A-408 (Grupo B).

**Introducción:**

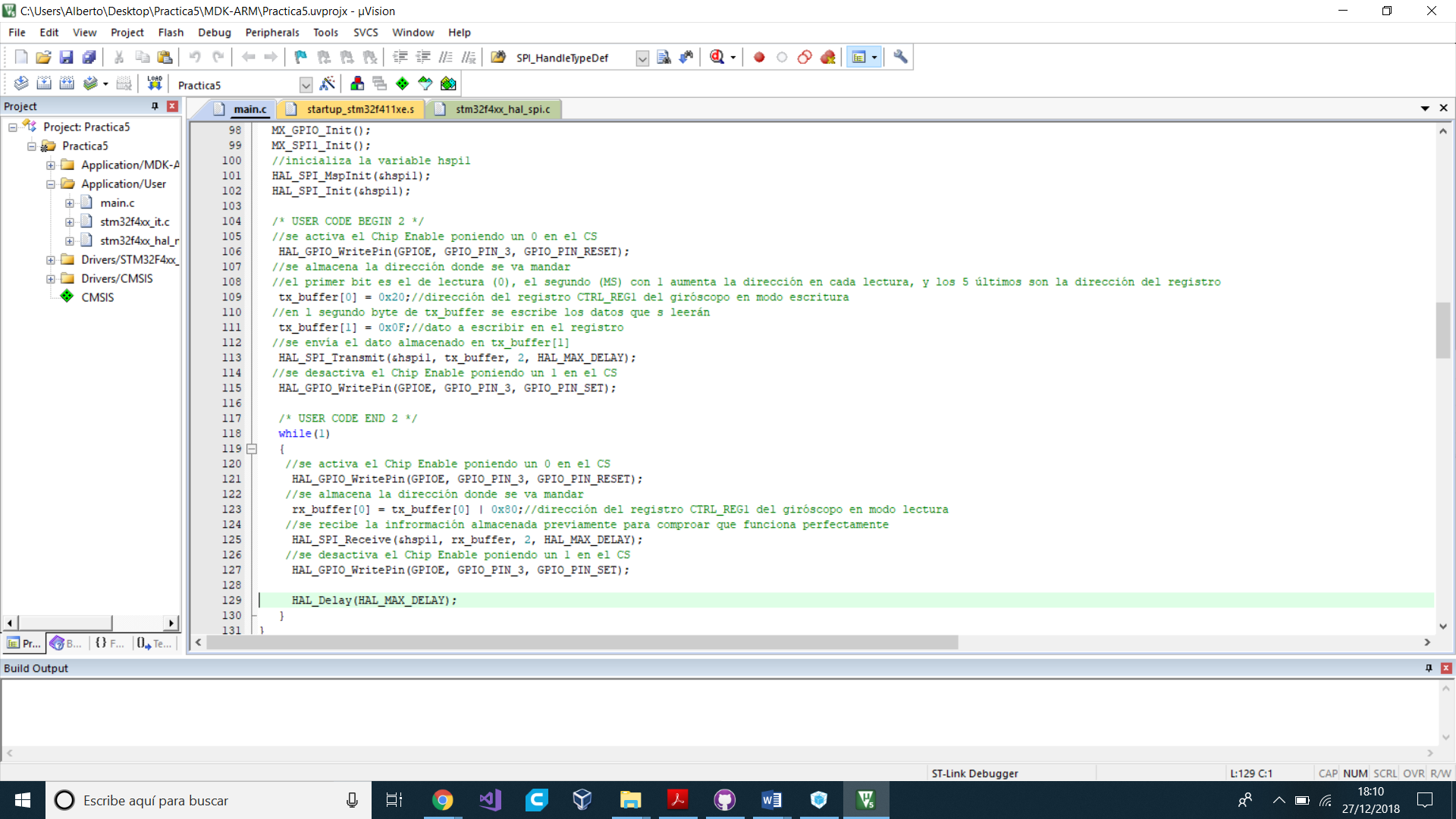
Comenzamos en esta práctica con el segundo acercamiento a la programación de la arquitectura ARM de nuestra micro STM32F411VE. Una vez familiarizados con el entorno pasamos a utilizar elementos más avanzados como el giróscopo y las salidas analógicas.

Para comenzar con el uso del giróscopo nos disponemos a preparar los pines PA3, PA5, PA6, PA7 para comunicarnos con el L3GD20. De ellos, los pines 5, 6, 7 se usan para la comunicación Serie de modo que el micro hace de maestro y el giróscopo de esclavo.

La segunda parte de la práctica consiste en familiarizarse con el convertidor A/D que posee integrado nuestro microprocesador, que nos permitirá captar y leer señales analógicas procedentes del exterior, concretamente, ensayaremos las diferentes posiciones que se pueden adoptar con un potenciómetro, así como la respuesta de un sensor LDR.

**Tarea 1**

**Establezca una primera comunicación con el giróscopo. Para ello, escriba un valor en uno de sus registros y léalo a continuación. Para ello, escriba en el registro CTRL\_REG1 (20h) el valor que nos permita activar los 3 ejes (X, Y y Z). Consulte las especificaciones.**



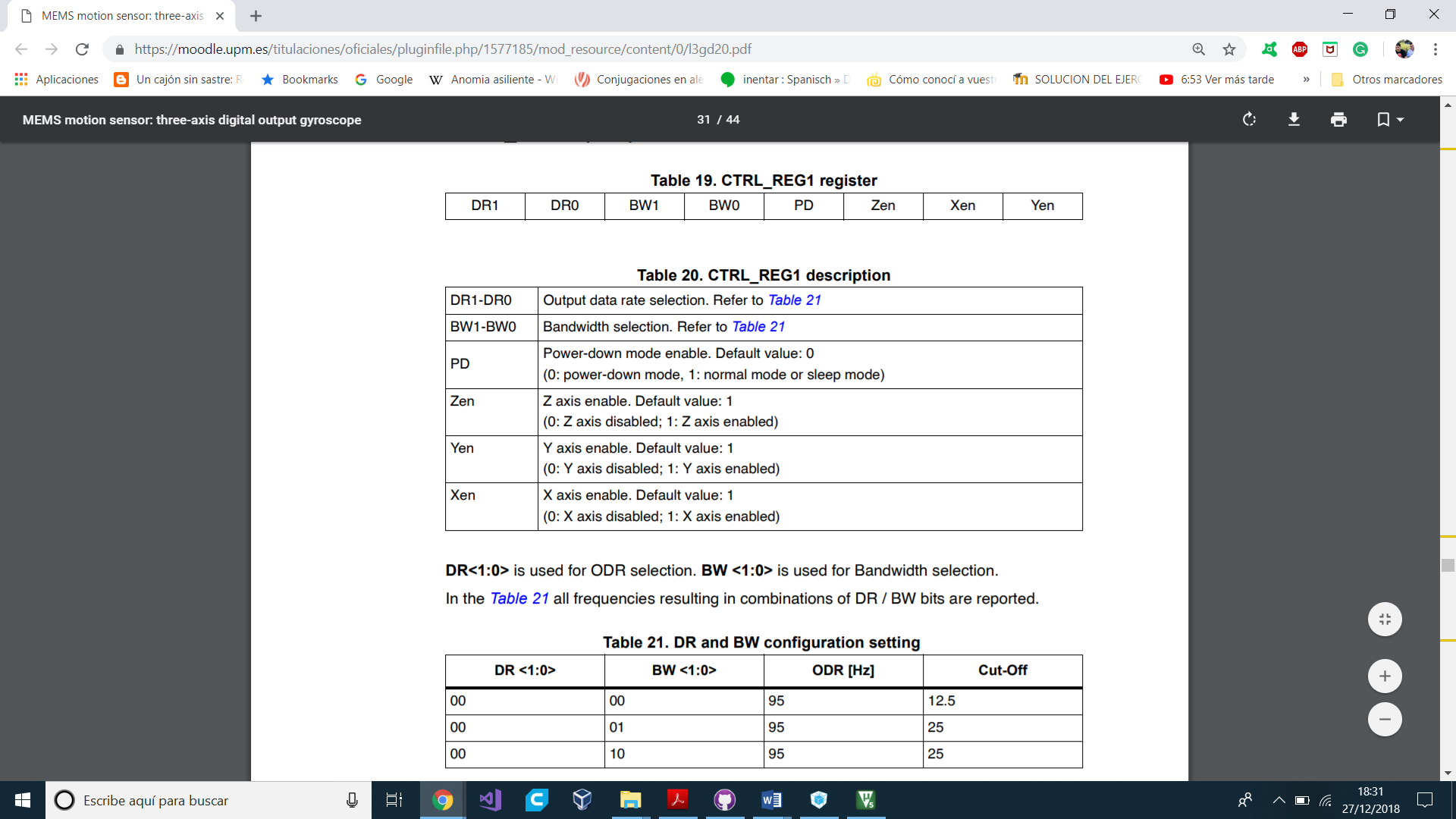
Como primera comunicación vamos a escribir en el registro CTRL\_REG1 un número que posteriormente leeremos para comprobar que la comunicación Serie funciona. Para poder escribir primero se genera un vector de dos enteros tamaño byte, donde se almacenará la dirección y el dato. Para poder pasarle el dato se activa el chip select para que pueda usarse el giróscopo. A continuación, se establece la dirección donde se quiere escribir el mensaje asegurándose de que el primer bit sea un 0 para poder escribir, para escribir en el registro deseado se mira en el data sheet del L3GD20. Posteriormente se guarda en el segundo byte el dato a escribir, para más adelante enviarlo mediante la función Transmit. Poniendo a un el chip select para su posterior uso.

Finalmente una vez escrito el dato par a leerlo basta con repetir el proceso pero con el primer bit a uno y la función Receive.

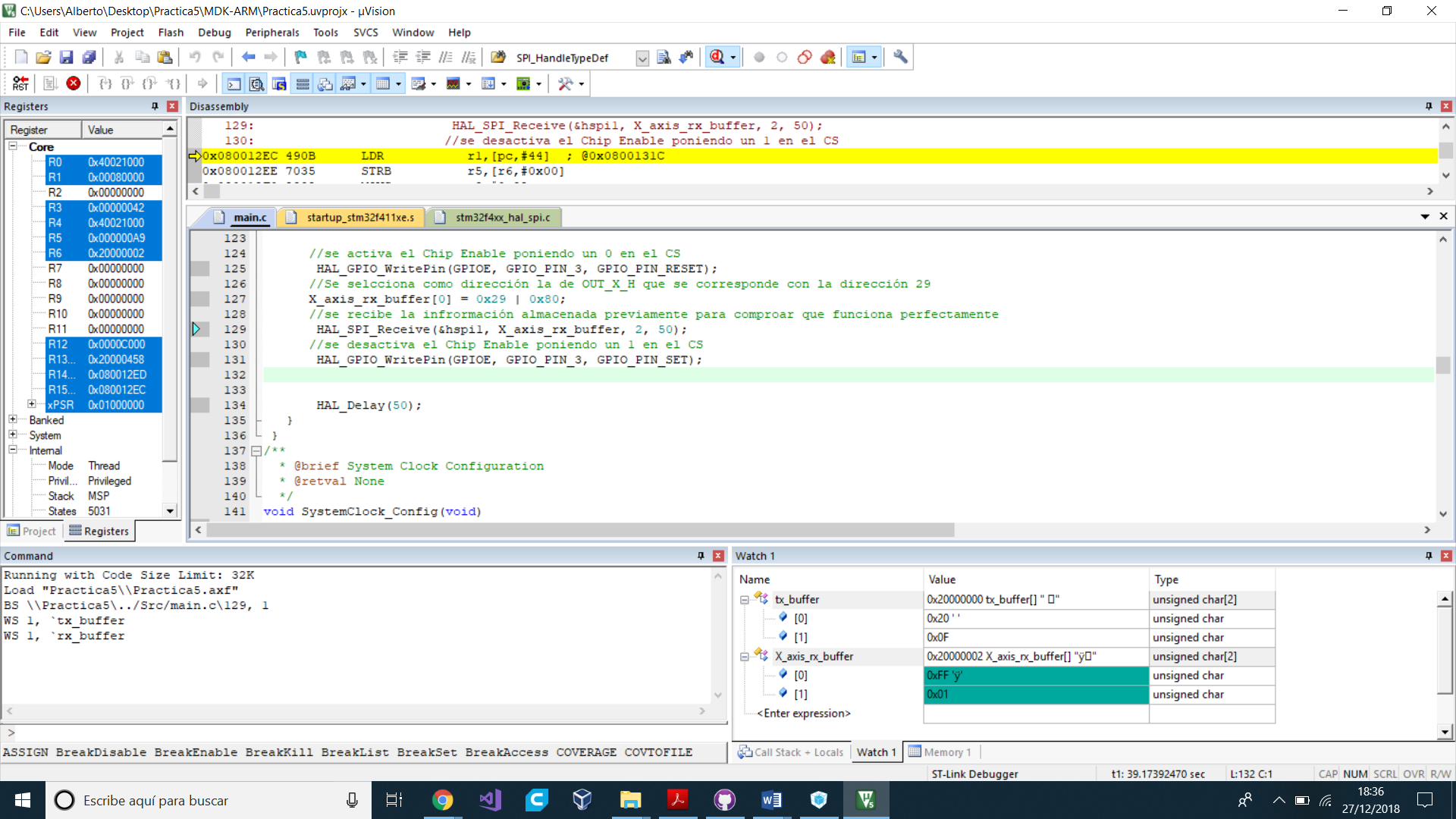
**Tarea 2**

**Ahora lea el valor angular de una de las componentes, por ejemplo, la X. Para ello debe leer el registro OUT\_X\_H. Consulte las especificaciones. Utilice el debugger.**

Una vez aprendido cómo leer y escribir en los registros nos disponemos a leer el ángulo que está inclinado del eje X. Para ello primero se activan los ejes y la energía, tal y como se explica en la hoja de características:



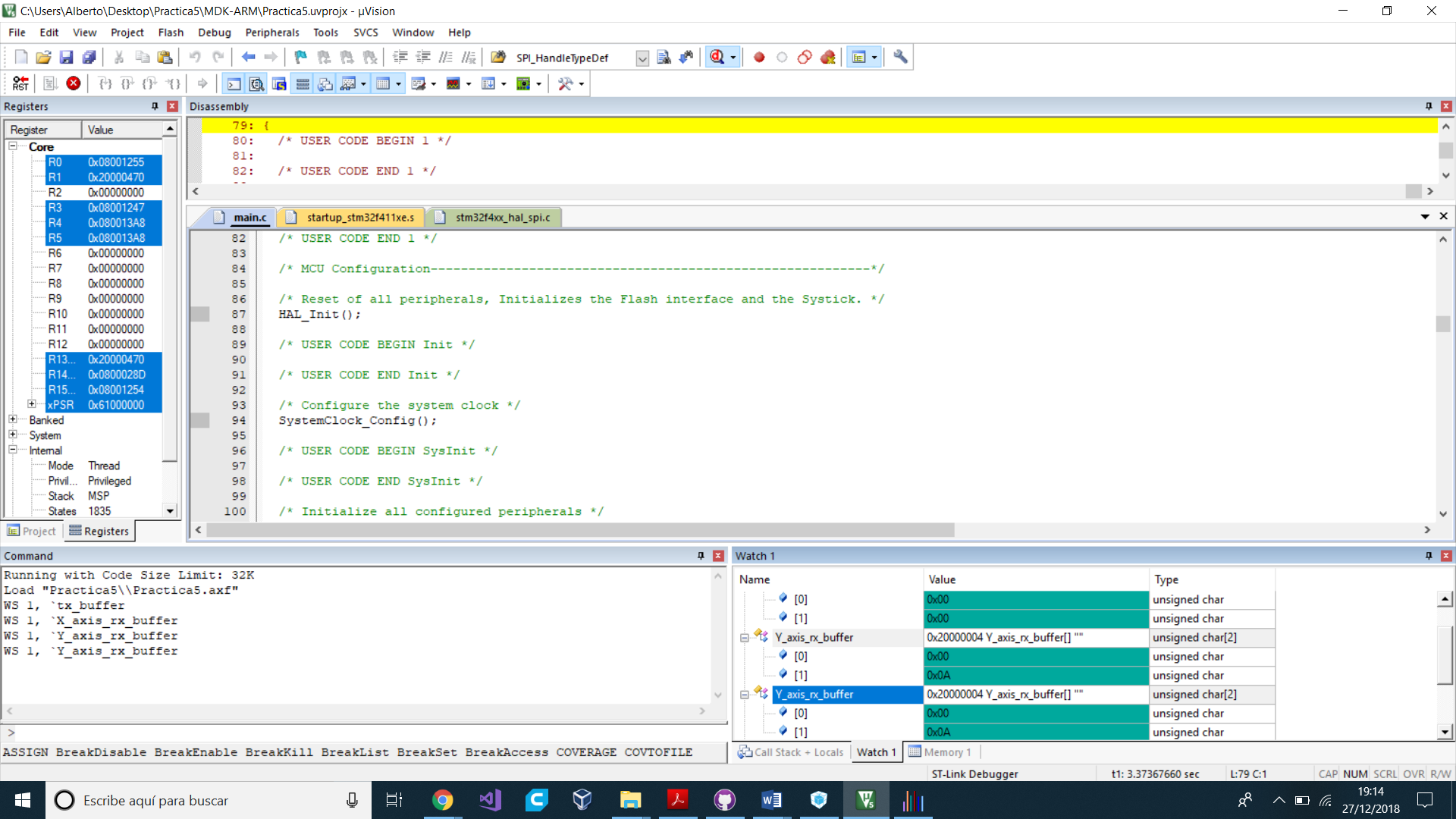
Por lo que basta con escribir unos en las posiciones de los ejes y de PD, por ello bastará con escibir 0x0F. Finalmente para poder leer el valor bastará con leer del registro correspondiente:



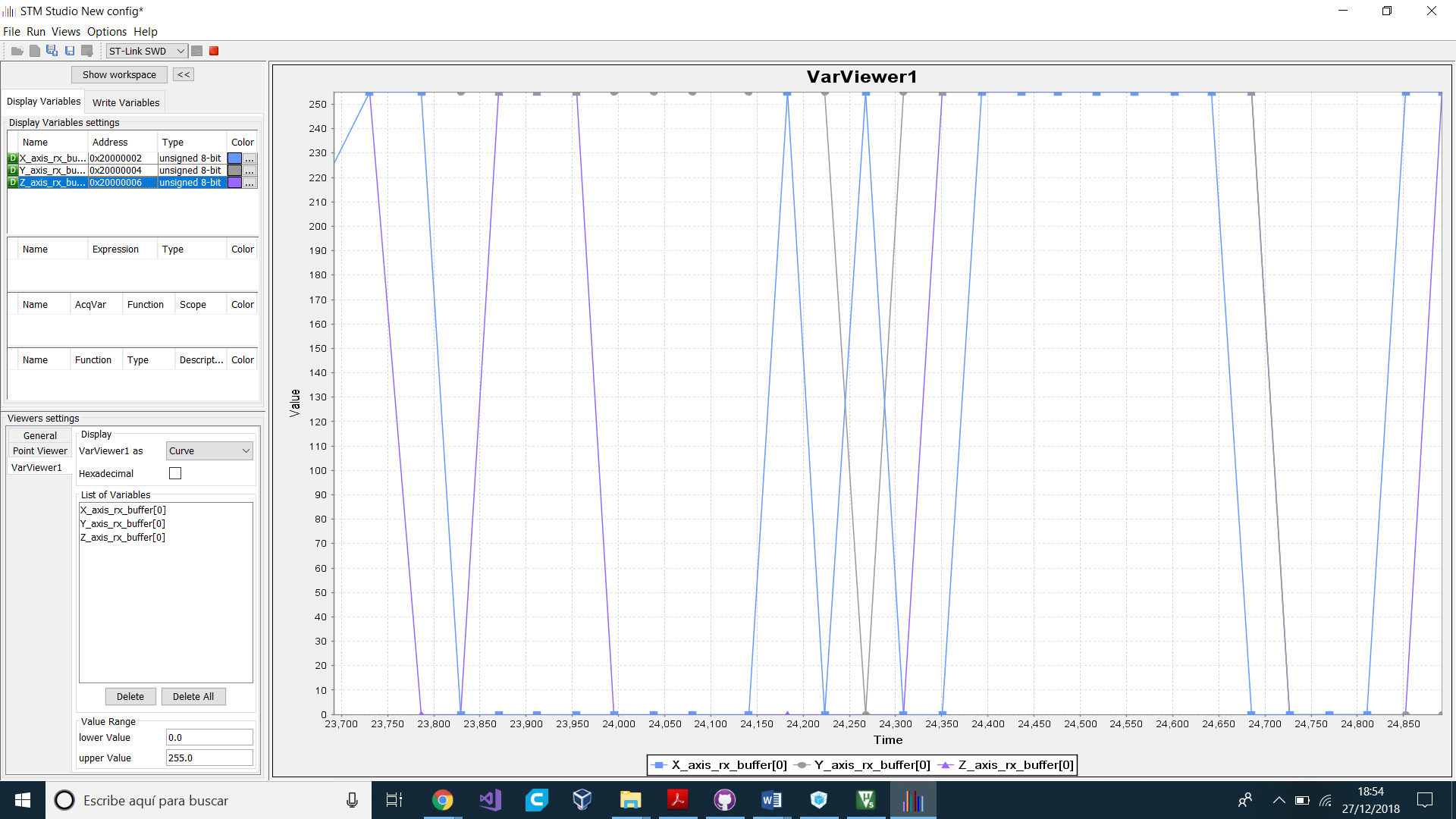
**Tarea 3**

**Visualice de forma periódica el valor de las 3 componentes del giróscopo (X, Y, Z). Para ello puede utilizar el debugger, aunque tiene bastante retardo. Se recomienda el uso del STMStudio. Para ello seguir las instrucciones siguientes (ver Figura 9):**

Con el debugger:



Con STMStudio:



Como se puede apreciar en ambas imágenes ante cambios en la inclinación varían los valores de los distintos registros.

Las siguientes tareas corresponde con la segunda parte de la práctica, la que hace referencia a los convertidores A/D. Para comenzar, como indica en el guion, configuramos un nuevo proyecto con el entorno CubeMx en el que configuraremos el Pin PA0 como ADC y estableceremos una resolución de 8 bits. Recordemos que, cuanto mayor resolución, mayor rango de valores analógicos podremos codificar.

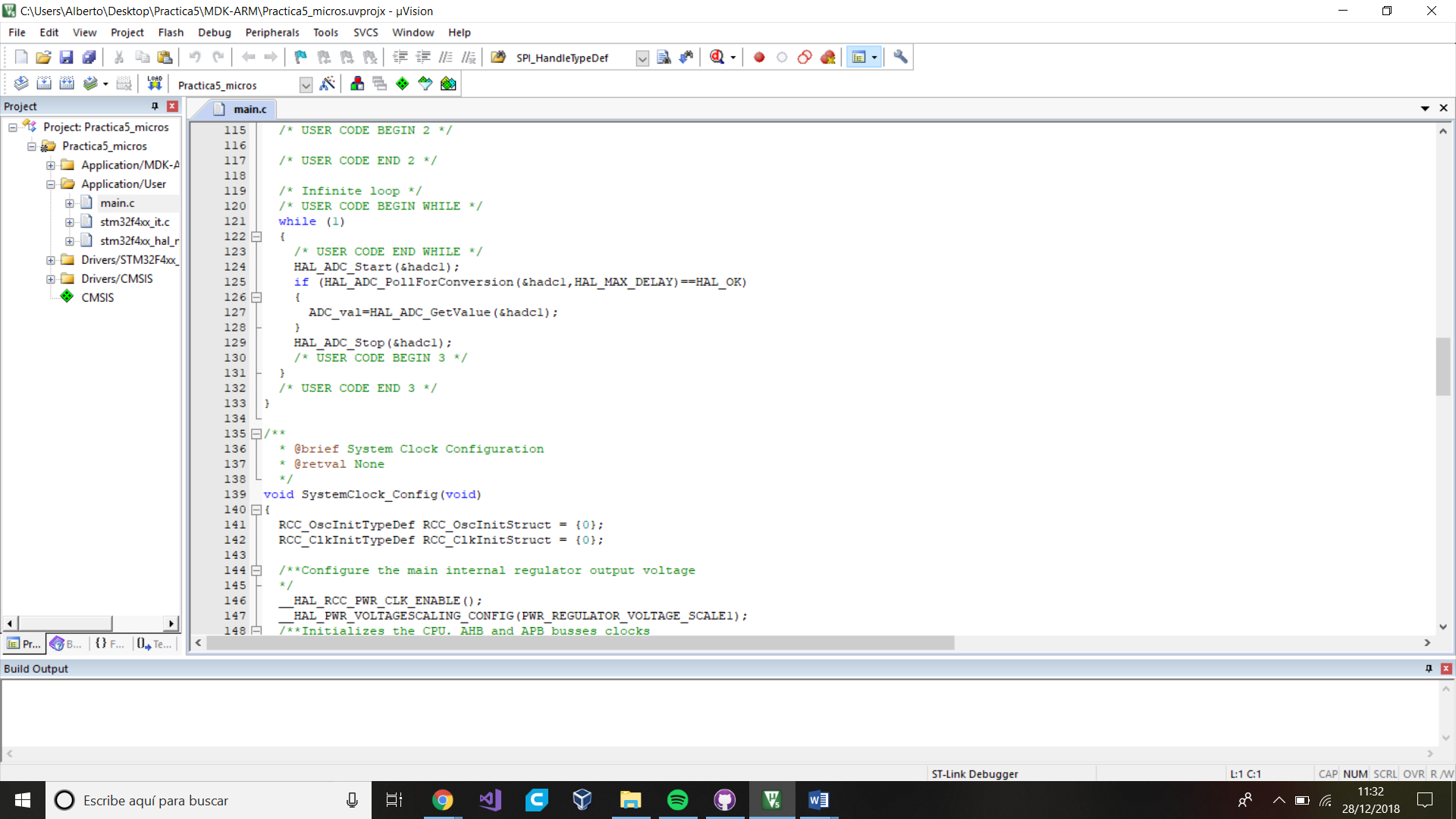
**Tarea 5**

**Lea el valor del ADC de forma recurrente (dentro del bucle while). Para ello debe inicializar el conversor, esperar a que esté la medida lista y leer el valor.**

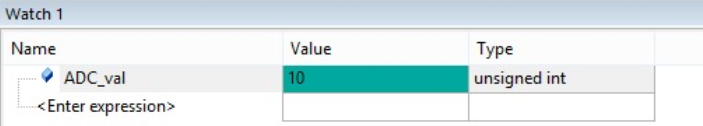


Después de haber configurado el pin, generaremos el proyecto para poder comenzar a programar el microprocesador. Será necesario contar con un potenciómetro, una placa de pruebas y diversos cables para poder realizar el montaje que aparece en la figura que se muestra a la izquierda. También será útil contar con un destornillador adecuado, que permita regular la posición del potenciómetro.

La siguiente imagen muestra el código introducido dentro del bucle *while*. Básicamente, se inician todos los parámetros del convertidor mediante *HAL\_ADC\_Start(&hadcl)*, se lee el valor y se guarda en la variable *ADC\_val*, que previamente habremos definido en la zona del código correspondiente a la declaración de variables privadas (*uint32\_t ADC\_val=0;*).



Al pasar a la sesión de depuración, ejecutamos el *debugger* y, al mover el potenciómetro, se podrá comprobar que en la ventana de visualización de comandos, la variable *ADC\_val* cambia también de valor. En esta caso, como la resolución escogida es de 8 bits, podemos obtener valores entre 0 y 255.

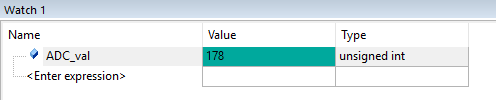
****

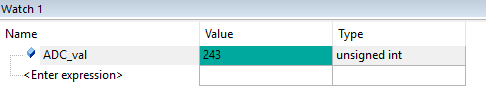
**Tarea 6**

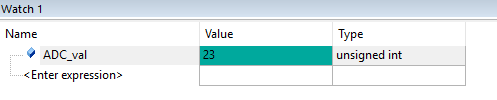
**Haga lo mismo, pero conecte ahora el LDR. Compruebe como varían los valores de luminosidad cuando lo tapa o cuando lo enfoca con una linterna.**

Con el LDR el código es el mismo que el mostrado en la tarea anterior, por lo que lo único que tendremos que hacer es conectar al LDR. La siguiente imagen, extraída del guion de prácticas, se corresponde con la conexión realizada. Se ha empleado una resistencia de 1 kohm para regular la intensidad que circula por el sensor.

A continuación, se muestran tres tablas obtenidas de los resultados arrojados por el debugger. La primera, muestra el valor leído del LDR a luz ambiente, la segunda corresponde a cuando incide sobre él la luz de una literna y, la tercera, a cuando está tapado (prácticamente no recibe luz).





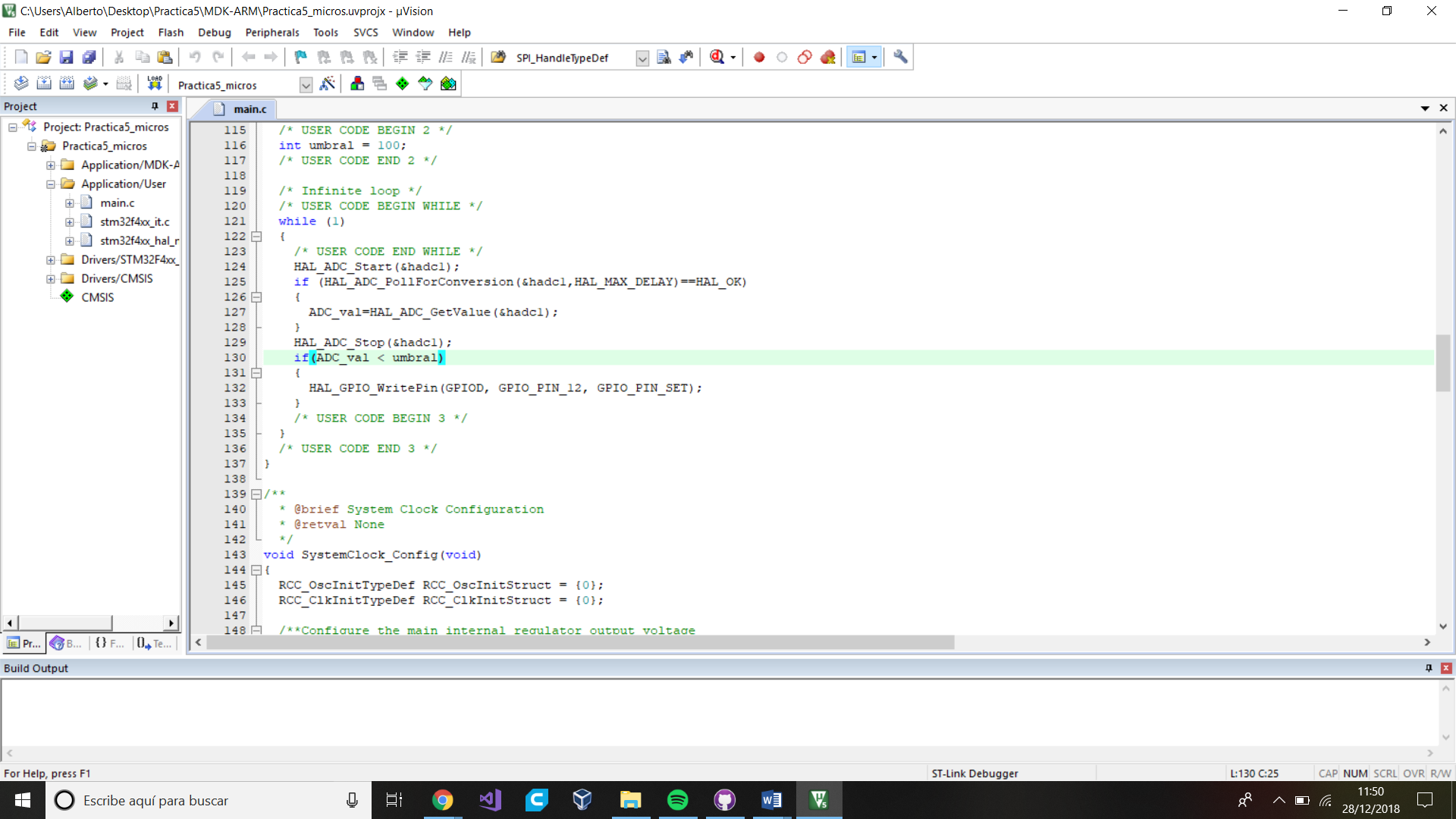


Como es natural en este tipo de sensores. A más luminosidad mayor valdrá la variable ADC\_val. Recordemos también que este sensor es fuertemente no lineal.

**Tarea 7**

**Encienda un LED cuando la luminosidad caiga por debajo de un umbral.**

Para completar esta tarea bastará con establecer con comprobar que el valor analógico leído no supere un valor establecido. Definiremos el umbral como una variable de tipo entero y de valor 100. El código propuesto se muestra en la siguiente imagen.



Sobre la placa se comprueba que, efectivamente, se enciende el LED verde cuando la luminosidad baja de 100 y se queda encendido. Podríamos hacer que se apague si el valor es más alto que el umbral poniendo a nivel bajo el PIN en esta condición.

Al igual que con la anterior práctica, este código al ser desarrollado conjuntamente hemos decidido aprovechar git para coordinarlo, de modo que están en diferentes ramas las distintas tareas (hay un error con el nombre de la rama 4 ya que este debería llamarse Tarea-5-y-6), pero por error no se ha podido cambiar el nombre en el online) quedando en la principal la última. Por lo que junto al código se incluirá el link al repositorio por si fuera de interés evaluarlo.

[**https://github.com/AlbertoTrapiello/Practica5**](https://github.com/AlbertoTrapiello/Practica5)