**Informe Práctica 4 SED: INTERRUPCIONES**

Autores:

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

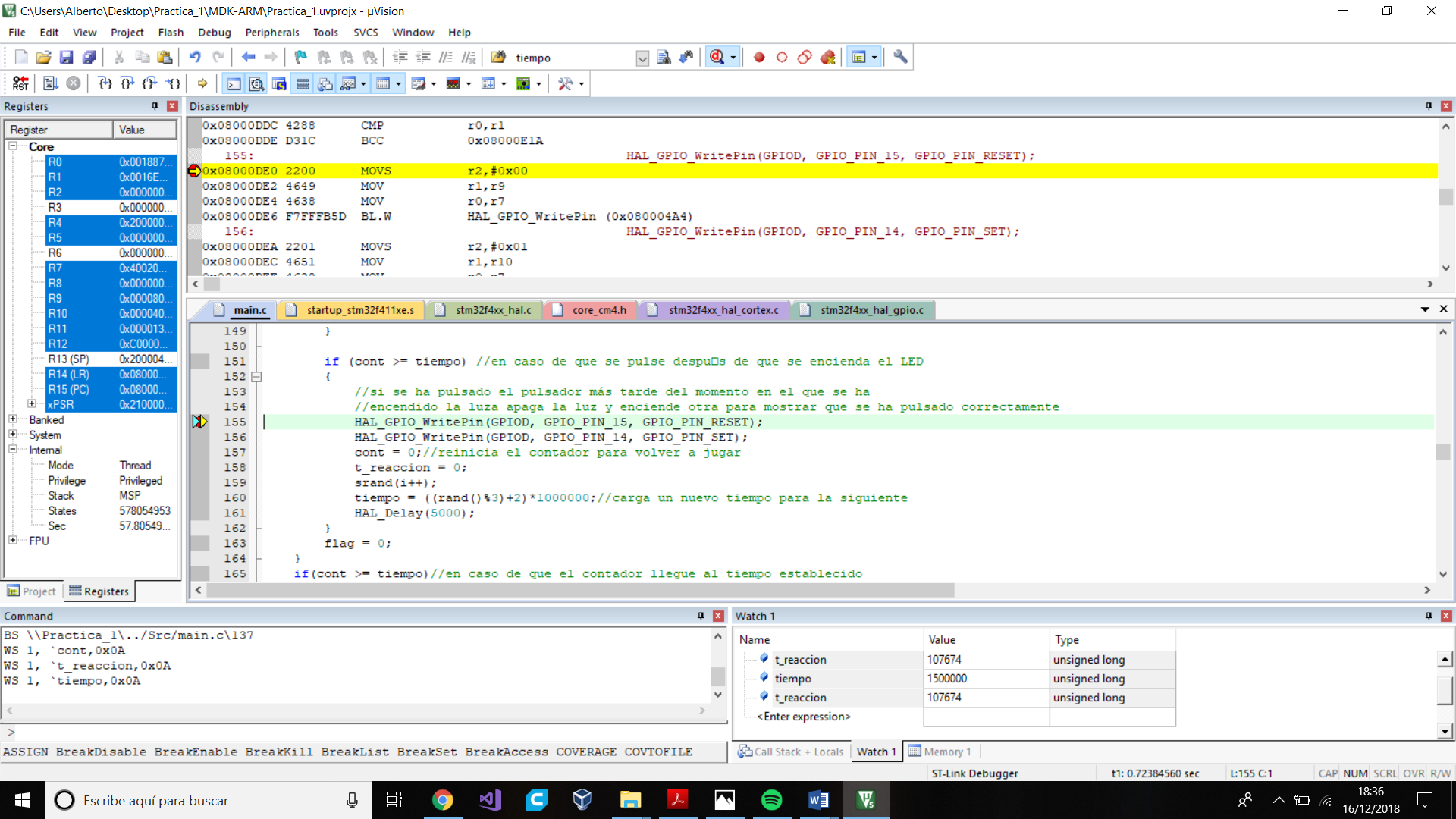
Grupo: A-408 (Grupo B).

**Introducción:**

Comenzamos en esta práctica con el primer acercamiento a la programación de la arquitectura ARM de nuestra micro STM32F411VE. Previamente realizamos la práctica 0 para familiarizarnos con el entorno de programación (STMCubeMX y Keil uVision).

Por lo que comenzamos con la programación de un sencillo código que nos permita aprender a usar las interrupciones y la interacción con botones y LEDs. Como objetivo se nos plantea conseguir programar un juego que consiga medir el tiempo de reacción del usuario. Este juego consiste en que se encienda una luz tras un tiempo aleatorio, de modo que el usuario ha de esperar a que se ilumine para en ese momento pulsar el botón. Si el botón se pulsa antes de que se haya iluminado dará error y parpadeará una luz. En caso de que se pulse tras el encendido de la luz, entonces se parará la ejecución del programa para mostrar en el debugger el tiempo de reacción.

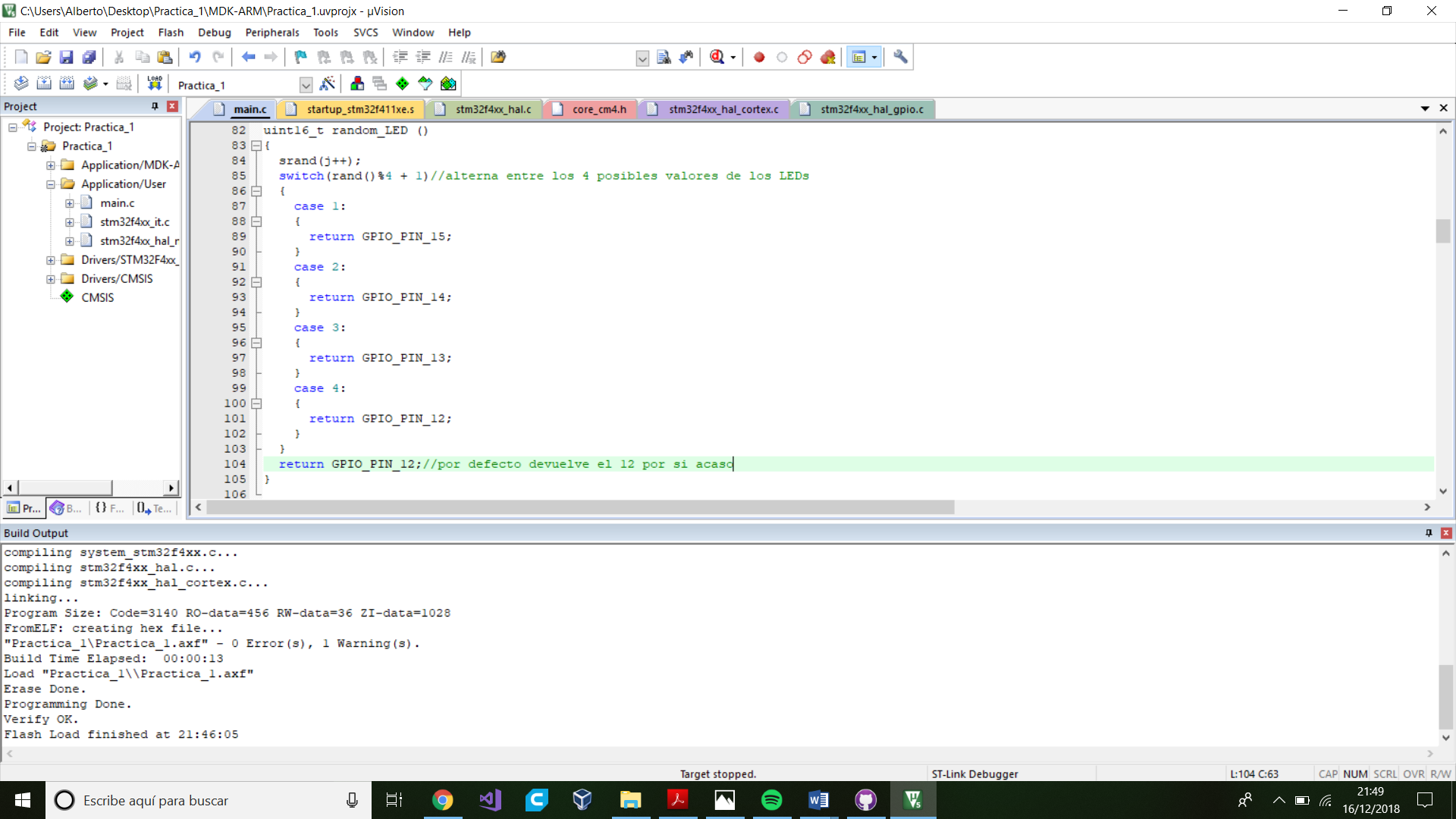
**Tarea 1a:** **Para ver el número de iteraciones contadas, establezca un punto de interrupción (breakpoint) en su función principal después de que se haya detectado la pulsación del interruptor y examine la variable contador utilizando la ventana de vigilancia (watch window).**



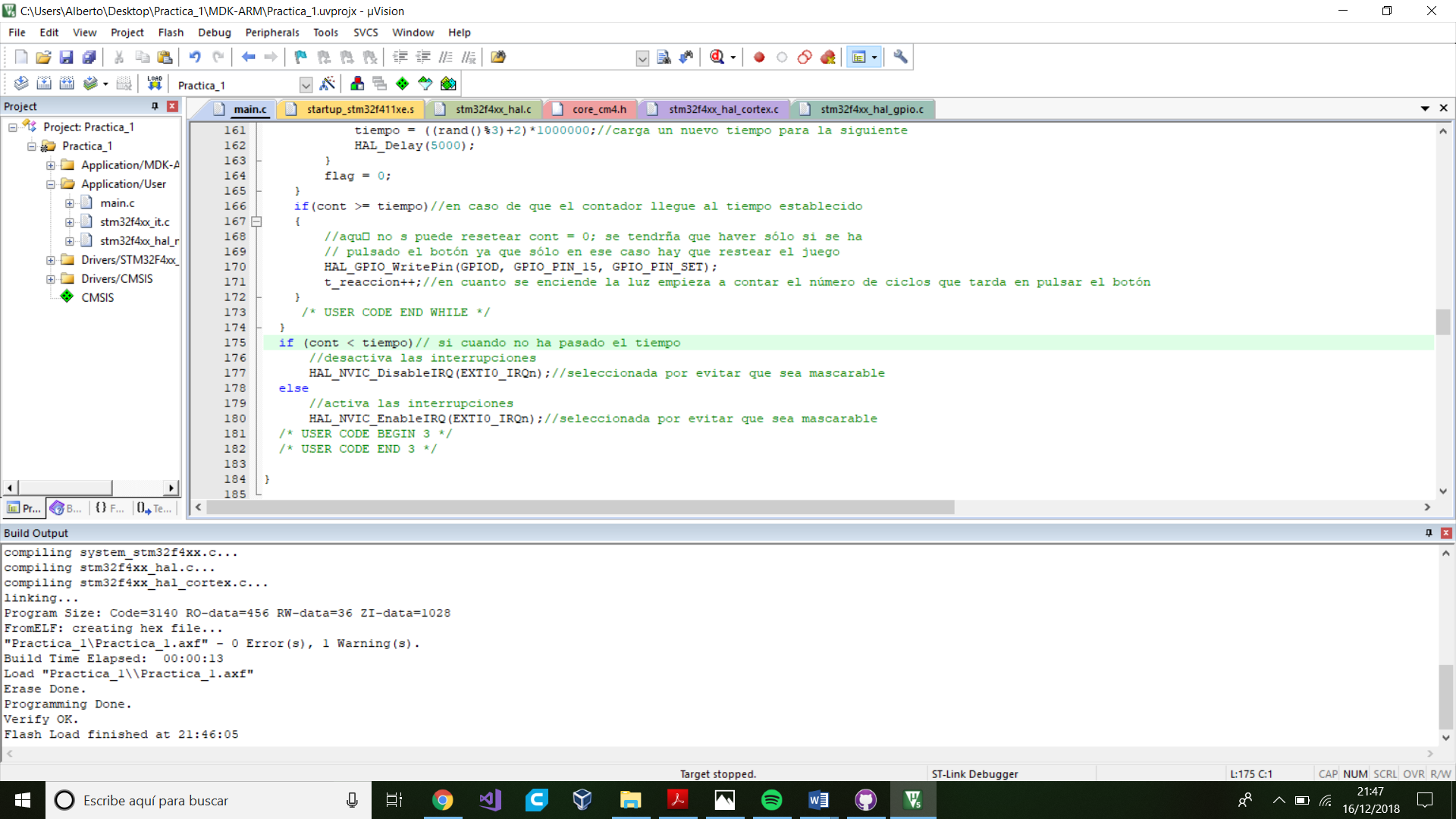
Como se puede comprobar en este test, la diferencia entre el tiempo (el tiempo aleatorio en el cual se ha de encender el LED) y el cont (variable que cuenta el tiempo hasta que se pulsa) es t\_reaccion que está ciclos del micro (que funciona a 1MHz).

**Tarea 1b: Modifique el programa para encender un LED de cada color de forma aleatoria.**

Se introduce un switch case con la función rand() para poder encender LEDs de manera aleatoria. Para hacerlo de una manera más organizada se desarrolla una función *random\_LED* encargada de devolver un GPIO distinto en cada instante.

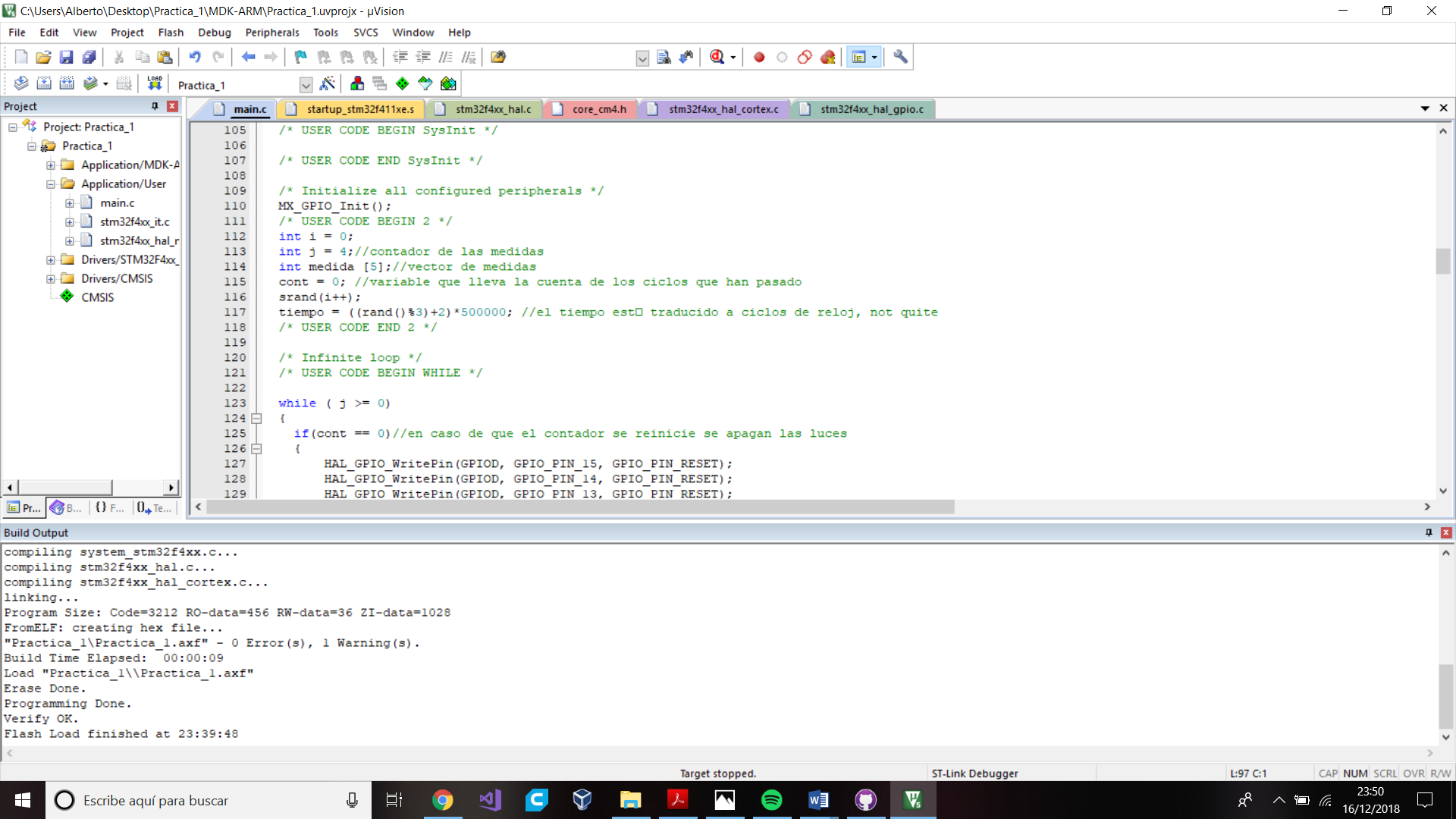


**Tarea 2: Para evitar que el usuario pulse el botón antes de que se haya encendido el LED, deshabilite las interrupciones hasta que se encienda el LED. Utilice las instrucciones HAL\_NVIC\_DisableIRQ() y HAL\_NVIC\_EnableIRQ() con los parámetros correspondientes.**Para ello se hace la distinción entre antes y después de que se encienda el LED y se habilita o deshabilita las interrupciones, respectivamente.

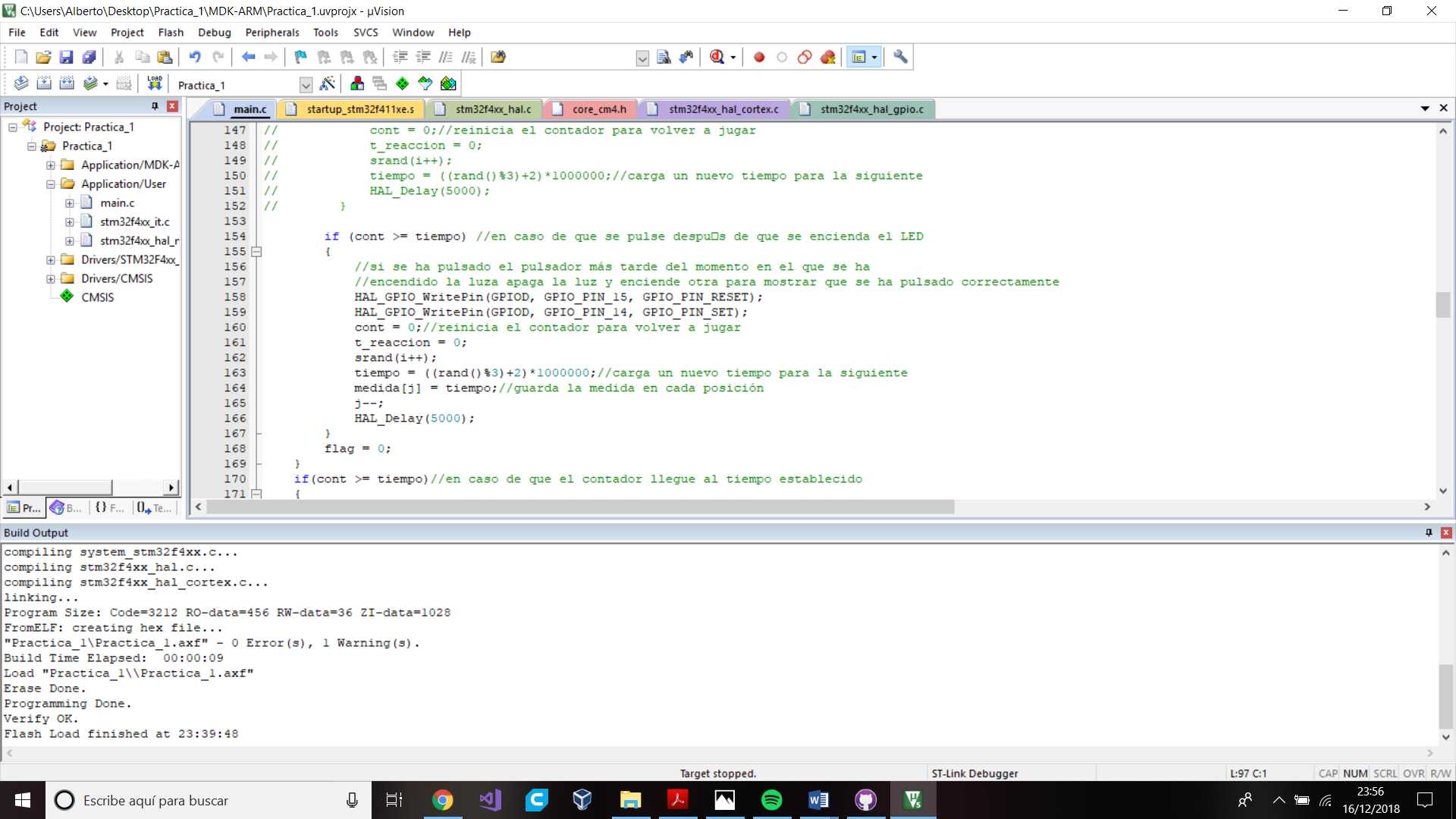


**Tarea 3: Modifique el programa para obtener la media de 5 medidas. Es decir, que se ejecute 5 veces seguidas el mismo procedimiento.**

Para realizar este apartado basta con introducir una variable que lleve la cuenta de las iteraciones y un vector para almacenar las medidas y posteriormente hacer la media. Y se exige como condición para el bucle que no se hayan contado las 5 medidas.



Y en cuanto se mide el tiempo se almacena el valor en distinta posición:



**Tarea 4: Haga lo mismo, pero sin interrupciones. Para poder detectar la pulsación del botón mientras se está esperando, deberá implementar un mecanismo que cuente tiempo pero que no bloquee el microprocesador mientras se espera.**

Desde un primer momento se implementa el diseño sin el uso de delay (excepto para la espera de 5 segundos tras cada pulsación para que dé tiempo a ver la medida). De modo que es un requisito que se ha cumplimentado durante todo el código para de esta manera evitar que se bloquee el micro.

Por lo que junto a la memoria se adjuntará el código para que pueda ser evaluado. Este código al ser desarrollado conjuntamente hemos decidido aprovechar git para coordinarlo, de modo que están en diferentes ramas las distintas tareas, 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/Practica\_1**