**– Lab 4: Pedestrian Detection Feature (Timers/WDT) –**

**Objetivo:**

El objetivo de la práctica es el siguiente:

Debido a una pandemia y atendiendo a las recomendaciones sanitarias, nuestro cliente ha decidido añadir un sistema sin contacto, que permita realizar la misma operación que el botón. Para ello, se hará uso de un sensor de ultrasonidos y de un timer de la placa de desarrollo STM32-F411RE. Se considerará la presencia de un peatón si se detecta un obstáculo a menos de 200 mm.

Además, nuestro cliente ha decidido que no quiere interrupciones anidadas ni tampoco que una vez finalizado el ciclo del semáforo se inicie inmediatamente un nuevo ciclo, se deberá esperar al menos 5 segundos entre una petición y otra.

**Esquema:**

El esquema de nuestro circuito realizado en Fritzing es el siguiente:

Diagrama, Esquemático

Descripción generada automáticamenteº

**Componentes:**

* Placa STM32
* Placa Grove
* Botón para cable de cuatro canales
* Ultrasonidos HC-SR04
* 2 LED rojos, 2 LED verdes, LED Amarillo
* Respectivas resistencias (resistencia explicada en prácticas anteriores)

**Esquema STM32 CubeMX**

Interfaz de usuario gráfica

Descripción generada automáticamenteTabla, Excel

Descripción generada automáticamenteEl archivo STM32 Cube MX está adjuntado junto a esta memoria, por lo que no entraremos en más detalles.

**Explicación de la solución:**

Texto

Descripción generada automáticamenteEn la línea 42 definimos las variables que usaremos para medir la distancia y el tiempo.

Texto

Descripción generada automáticamenteEntre las líneas 63 y 66 declaramos las funciones que utilizamos a lo largo de nuestro programa, para satisfacer la lógica deseada. Son las siguientes:

Texto

Descripción generada automáticamenteNuestro método main únicamente se compone de una llamada a la función secuencia, tal y como ideamos en la práctica anterior:

Texto

Descripción generada automáticamenteEn dicha función es donde reside toda la lógica del programa. Entraremos solo en detalle en aquellas partes que varíen y hayamos implementado respecto a las prácticas anteriores.

En el caso del estado inicial, lo que hacemos en esta práctica es calcular la distancia llamando a la función *calcularDistancia()*, que se encargará de generar un pulso de 10 us y esperar recibir el echo, como observamos en la Texto

Descripción generada automáticamentesiguiente imagen:

El funcionamiento es simple: el trigger envía el pulso y el echo la recibe. El tiempo que tarda en recibirla es la distancia.

Se inicializa la variable tiempoTIM, que nos servirá para calcular el pulso. Posteriormente, con el pin en modo trigger en estado bajo, se pone en alto durante 10 us (pulso) y se vuelve a bajar.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTenemos las siguientes funciones:

Como se puede apreciar, son funciones antagónicas. Respectivamente se encargan de cambiar el modo del pin a Echo o Trigger, cambiando el modo de output a input y viceversa correspondientemente. De tal forma, somos capaces de calcular la distancia del obstáculo mediante el tiempo de respuesta calculado en la función *calcularDistancia()*.

Texto, Carta

Descripción generada automáticamenteContinuando con la función *secuencia()*, calculamos la distancia mediante la siguiente función (facilitada en el enunciado):

Debemos comprobar que sea menor que 200mm para iniciar la secuencia y simular el funcionamiento del botón.

Por último, mediante la siguiente función lo que hacemos es incrementar la variable que hemos usado para controlar el tiempo, para posteriormente calcular el incremento respecto al valor inicial. Creemos que no es el método más ortodoxo de realizarlo, pero resulta efectivo:

Texto

Descripción generada automáticamente

**Ejemplo de ejecución (Tera Term)**

**Texto

Descripción generada automáticamente**