Diseño de sistemas basados en microprocesador

Proyecto 1

Environment/Weather Station

Alex Conejo Martín
César Braojos Corroto

Índice

CONFIGURACIÓN PREVIA

Antes de comenzar a explicar todo lo relacionado con el código de la práctica tenemos que hablar de la configuración seleccionada para el proyecto.

Hemos decidido trabajar con un voltaje de 3,3V por lo que el código y configuración realizada en el STM32CUBEMX está adaptado a este voltaje.

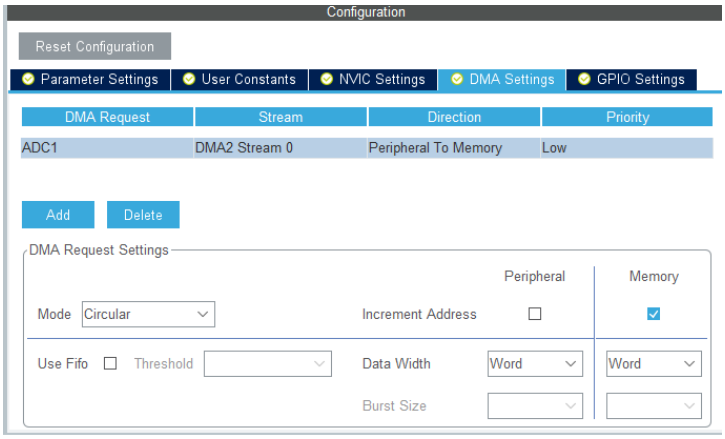
Primero tenemos que hablar de la configuración realizada en STM32CUBEMX donde hemos configurado varios puertos para que, junto a los puertos previamente configurados, podamos realizar el código. A continuación mostraremos una tabla con los diversos puertos y su función dentro de nuestra práctica:

Puertos	Configuración	Dispositivos
PA0 (sim-anemometer)	ADC_IN0	Sensor de viento (potenciómetro)
PA1	ADC_IN1	Sensor de luminosidad
PA2	USART_TX	
PA3	USART_RX	
PA4	ADC_IN4	Sensor de temperatura
PA8	ADC_IN8	Sensor de sonido
PA5 (anemometer)	TIM2	Puente conectado a PA6
PA6	TIM3	Puente conectado a PA5
PA9	USART1_TX	Bluetooth
PA10	USART1_RX	Bluetooth
PB5	EXTI5	Sensor de lluvia (botón táctil)
PB4	EXTI4	Reset (botón)
PA8	GPIO_Output	LED Verde Sonido
PB10	GPIO_Output	LED Verde Temperatura
PB9	GPIO_Output	LED Verde Viento
PC7	GPIO_Output	LED Rojo Sonido
PB6	GPIO_Output	LED Rojo Temperatura
PA7	GPIO_Output	LED Rojo Viento

ADC Mode and Configuration

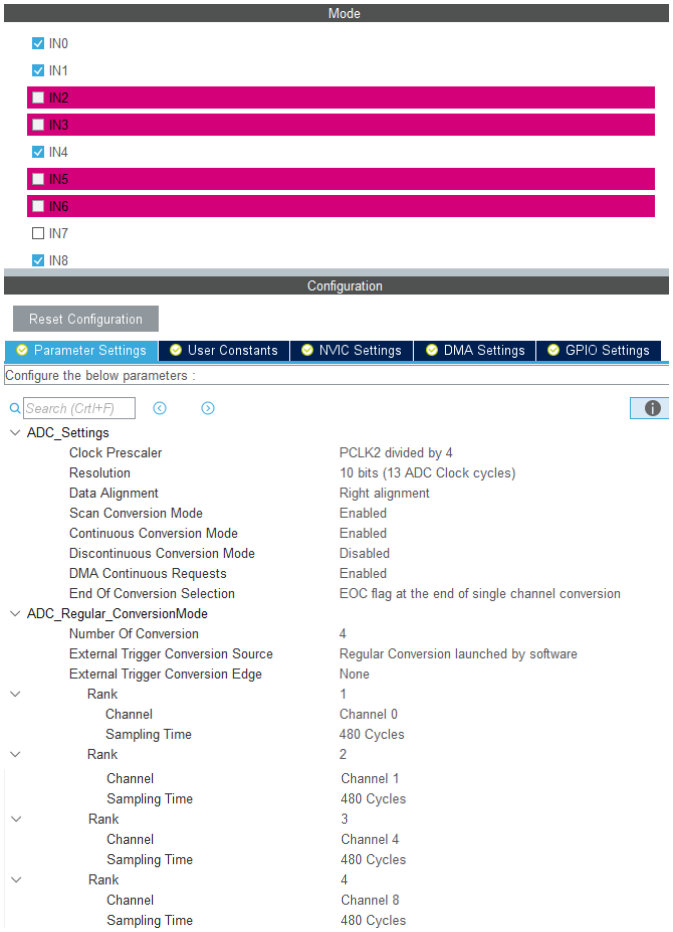
A continuación vamos a explicar la configuración usada para nuestro ADC.

Lo primero que tenemos que dejar claro es que usaremos DMA. Por ello hemos añadido una *"DMA Request"* a nuestra configuración DMA:



Usaremos un modo *"Circular"* y hemos declarado la anchura de datos como *"Word"*. La dirección debe ser *"Peripheral To Memory"* y Prioridad *"Low"*.

En la sección de parámetros hemos seleccionado la siguiente configuración:



Hemos activado 4 modos como podemos observar en la sección *"Mode"*. Cada uno para uno de nuestros sensores:

Modo	Sensor
ADC_IN0	Sensor de Viento (potenciómetro)
ADC_IN1	Sensor de Luminosidad
ADC_IN4	Sensor de Temperatura
ADC_IN8	Sensor de sonido

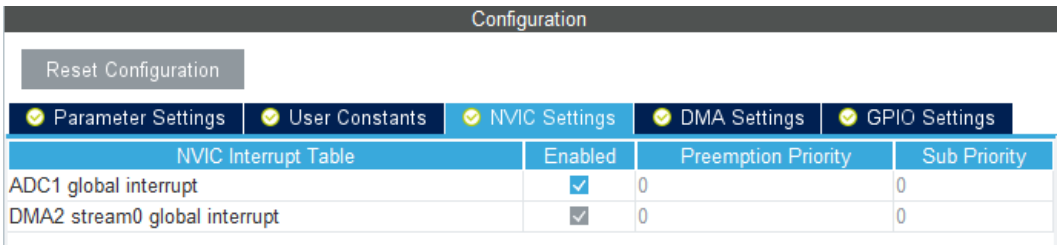
Hemos seleccionado una resolución de 10 bits ya que vamos a trabajar con 3,3V.

Además hemos habilitado tres modos: *"Scan Conversion Mode"*, *"Continous Conversion Mode"* y *" DMA Continuous Requests",* este último nos ha permitido habilitarlo tras añadir nuestro *"DMA Request".*

Para cada modo añadido (cuatro en nuestro caso) tenemos una conversión y un rank. Cada rank está configurado con 480 ciclos (para tener una mayor exactitud de los datos obtenidos) y un canal.

Modo	Rank	Channel
ADC_IN0	1	Channel 0
ADC_IN1	2	Channel 1
ADC_IN4	3	Channel 4
ADC_IN8	4	Channel 8

Para finalizar con nuestra configuración hay que activar en la configuración de NVIC las siguientes interrupciones:



TIM2 and TIM3 Mode and Configuration

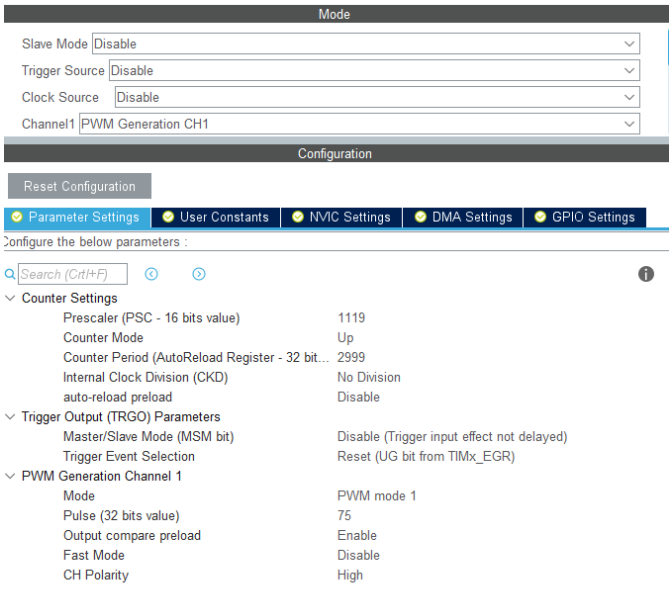
A continuación vamos a hablar de la configuración usada para nuestro TIM2 y TIM3.

Lo primero que tenemos que dejar claro es qué modo va a tener cada uno de nuestro timers y por lo tanto que papel va a tener en nuestro código.

Timer	Modo
TIM2	PWM Generation CH1
TIM3	IC direct mode

Antes de hablar de la función que hará cada timer debemos hablar de su configuración.

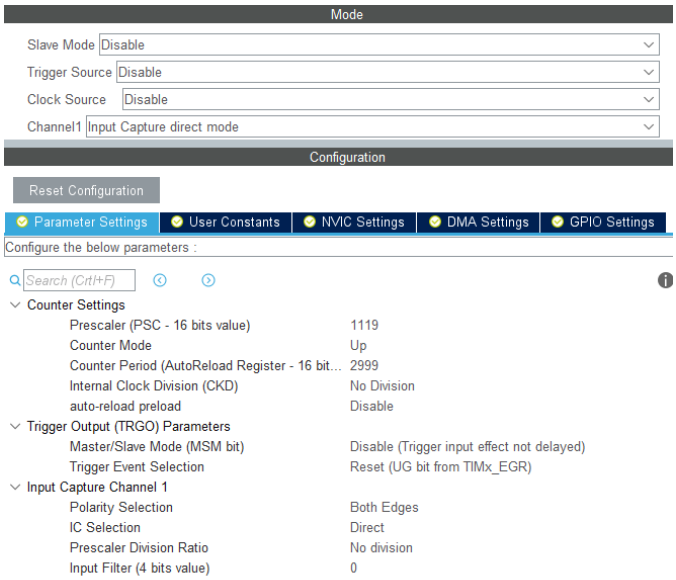
La configuración del timer 2 es la siguiente:



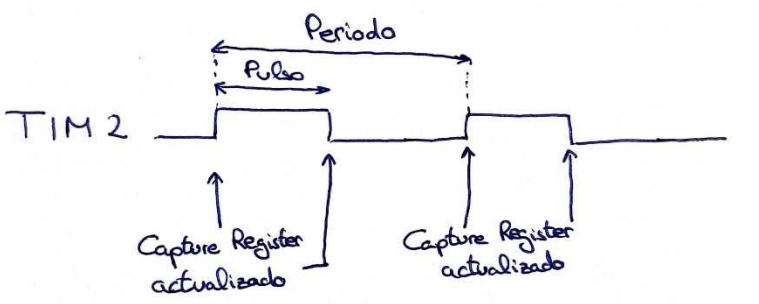
Está configuración ya estaba definida previamente pero con esto podemos determinar varios datos:

Datos	Valor
Frecuencia del timer	84MHz
Frecuencia de reloj	70KHz
Prescaler	1119
Period	2999
Tiempo	23

La configuración del timer 3 es la siguiente:



La configuración es similar a la del TIM2 ya que buscábamos capturar los valores al realizarse un periodo.



Al tener TIM3 el mismo periodo que TIM2 vamos a poder obtener los valores de subida y bajada en un periodo de este timer. Hemos configurado la polaridad como “Both Edges” para capturar tanto valor de subida como valor de bajada por lo que nuestra interrupción va a saltar dos veces por cada periodo cumplido del TIM2.

Con estos datos vamos a poder determinar mediante una fórmula declarada en nuestro código el pulso (periodo del timer 2 – valor capturado en la subida + valor capturado en la bajada). Esa fórmula es debida a que el primer valor que se captura es el de bajada. Esto es lo mismo que restar el valor capturado en la bajada menos el valor capturado en la subida.

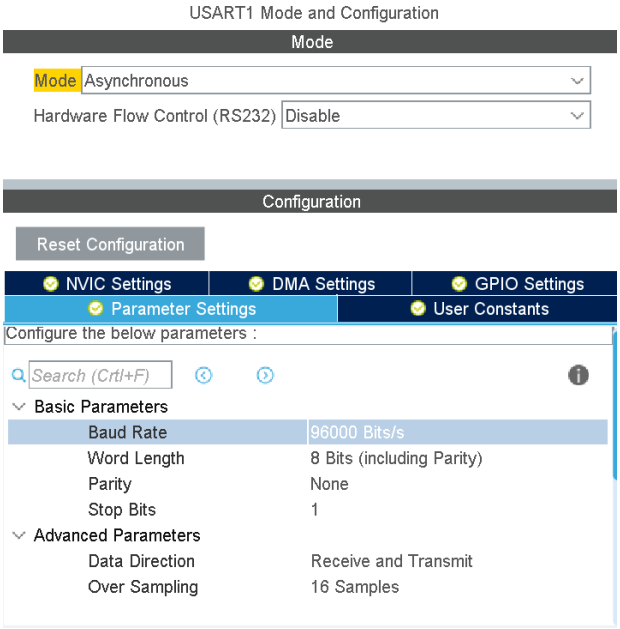
USART (Bluetooth)

A continuación vamos a proceder a explicar la configuración usada para transmitir los mensajes a nuestro monitor que en este caso , se trata de el sensor bluetooth BT5 y la aplicación movil ***Blueetooth Terminal HC-05***.

Para la configuracion de nuestro sensor de bluetooth hemos conifgurado los siguientes puertos de la siguiente manera:

Puertos	Función
PA10	USART1_RX
PA9	USART1_TX

Para ello estableceremos la siguiente configuracion en el USART1. En esta configuracion cambiamos principalmente el *"Baud Rate"* , para que al enviar un mensaje se nos imprima lo mismo que hemos escrito y no simbolos o signos distintos. Además hemos activado tambien el NVIC.



Además este sensor tiene una característica peculiar y es que a la hora de la conexión , la patilla del sensor bluetooth RX va conectada a *"USART1_TX (PA9)"* y TX va conectada a *"USART1_RX (PA10)".*

EXTI Configuration

A continuación vamos a hablar de la configuración usada para nuestro EXTI4 y EXTI5.

Cada uno de nuestros EXTI tiene una función:

Modo	Sensor/Función
GPIO_EXTI4	Reset (botón)
GPIO_EXTI5	Sensor de lluvia (botón táctil)

Hay que activar las interrupciones en la configuración del NVIC:

GPIO_Output Configuration

A continuación vamos a hablar de la configuración usada para nuestros LEDs.

Hemos configurado nuestros LEDs como GPIO_Output en los siguientes puertos:

Puertos	Función
PA8	LED Verde de Sonido
PB10	LED Verde de Temperatura
PB9	LED Verde de Viento
PC7	LED Rojo de Sonido
PB6	LED Rojo de Temperatura
PA7	LED Rojo de Viento

Los LEDs rojos se encenderán cuando los valores de nuestros sensores sobrepasen el Threshold y solo volverán a encenderse los verdes cuando se de al botón Reset aunque esto lo explicaremos con más detalle cuando hablemos del código.

Fórmulas

A continuación vamos a hablar de las fórmulas que hemos usado para calcular el valor de la velocidad, temperatura, luminosidad y sonido. En las siguientes fórmulas el valor será el capturado con ADC para cada uno de los sensores

Dato	Fórmula	Unidad
Velocidad	(Pulso-75)*(200/1499-75)	Km/h
Luminosidad	Exp(valor/54.46053)	lux
Temperatura	1/(ln(resistencia)/4275+1/298.15)-273.15	ºC
Sonido	20*ln(valor/568)+67.5	dB

Empezaremos hablando de la fórmula de velocidad. Lo primero que tenemos que decir es como se calcula el pulso ya que gracias a la fórmula definida previamente en el código inicial el pulso = 75 + (valor*1425/1024). Esto hará que nuestro pulso vaya del 75 al 1499 (pulso máximo con periodo de 2999).

Como nuestra intención era conseguir que el valor 1499 coincidiera con 200 km/h y 75 con 0 km/h determinamos la fórmula mencionada en la tabla.

La fórmula de la luminosidad la hemos determinado gracias a la página mencionada en la bibliografía. En esa página se calculan los lux con: exp(valor/80) pero ese 80 lo hemos ajustado para que al acercar una luz el máximo fuera 100.000 lux que es lo normal al estar expuesto a la luz del sol. Ya que el valor máximo del adc con una luz encima del sensor es de 627 podemos determinar esa fórmula.

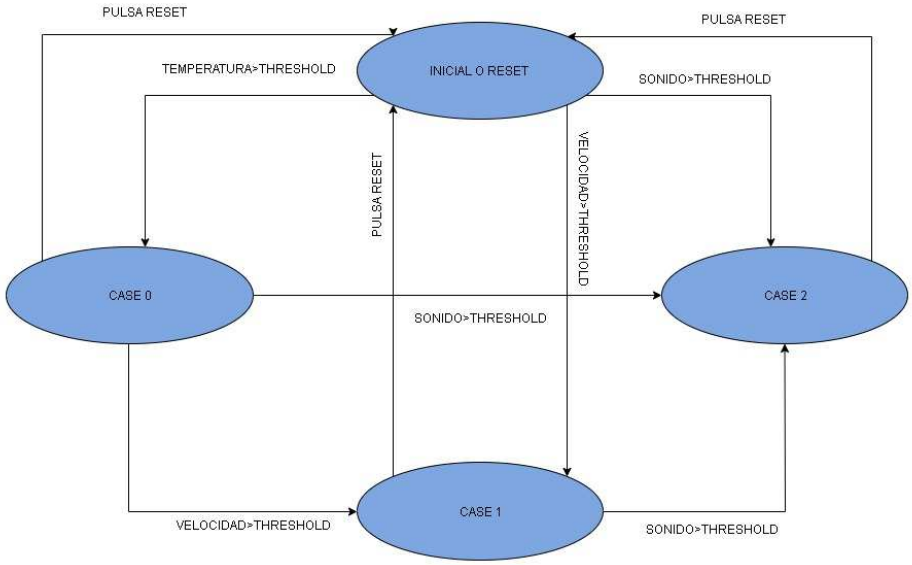
La fórmula de temperatura la hemos determinado gracias a la página mencionada en la bibliografía. En esta fórmula aparece la variable resistencia.

$$Resistencia = 1023/valor - 1$$

El 1023 es debido a que estamos trabajando con un ADC de 10B.

La fórmula del sonido la hemos determinado gracias a la página mencionada en la bibliografía. Esta fórmula está adaptado a nuestro sensor de sonido ya que en una conversación o con la música a un sonido medio/alto llega a los 80dB y sin hablar el valor del sonido es de 50/55 dB. Esta fórmula la hemos comprobado con un sonómetro y varía 5dBs como máximo. La manera en la que lo hemos calculado ha sido con una media ya que con 85dB el valor del adc es de 802 y con 50dB el valor del adc es 334 para que variara lo menos posible. Con ello hemos definido los 67.5dB a un valor del adc de 568.

DIAGRAMA DE ESTADOS



Hemos usado un diagrama de estados Mealy ya que el estado siguiente solo depende de la salida. En el diagrama cuando hablamos de *"THRESHOLD"* nos referimos al Threshold de ese caso. Por ejemplo cuando nos referimos a *"Temperatura>Threshold"* nos referimos al *"THRESHOLD_TEMPERATURE".*

En total tenemos 4 estados:

Caso	Función
Inicial o Reset (case 3)	Es el caso inicial o el caso al pulsar el botón Reset. Su función es la de encender los 3 LEDs a Verde y apagar los rojos
Case 0	Es el caso en el que la temperatura pasa su threshold. Su función es encender el LED Rojo de Temperatura y apagar el LED Verde de Temperatura
Case 1	Es el caso en el que la velocidad del viento pasa su threshold. Su función es encender el LED Rojo de Viento y apagar el LED Verde de Viento
Case 2	Es el caso en el que el sonido pasa su threshold. Su función es encender el LED Rojo de Sonido y apagar el LED Verde de Sonido

Detalles Código

A continuación vamos a explicar algunos métodos que creemos que es interesante explicar, aunque creemos que con los comentarios el código está bastante claro.

El primero es el método “Mensajes”, un método en el que concatenamos los diferentes mensajes para enviarlos mediante el Bluetooth a la aplicación previamente mencionada.

Usamos dos funciones que nos han sido de ayuda para este método: “strcat” y “itoa”. Con “strcat” hemos podido concatenar los diferentes mensajes para enviar un solo mensaje mediante “HAL_UART_Transmit_IT” y para personalizar los mensajes.

Con “itoa” hemos podido convertir las diferentes variables en un array de char y tratarlo como tal.

```
void Mensajes(void)
{
  volatile char MsgTotal[400] = "";
  char MsgVelocidad[105] = " Velocidad Viento \r"; // CREAMOS EL MENAJE DE LA VELOCIDAD , USANDO UNA CONCATENACION
  strcat(MsgVelocidad, itoa(Velocidad, array, 10));
  strcat(MsgVelocidad, " km/h\n");

  char MsgTemperatura[50] = " Temperatura \r"; // CREAMOS EL MENSAJE DE LA TEMPERATURA , USANDO UNA CONCATENACION
  strcat(MsgTemperatura, itoa(temperatura, array, 10));
  strcat(MsgTemperatura, "C\n");

  char MsgLuminosidad[50] = " Luminosidad \r"; // CREAMOS EL MENSAJE DE LA VELOCIDAD , USANDO UNA CONCATENACION
  strcat(MsgLuminosidad, itoa(luz, array, 10));
  strcat(MsgLuminosidad, " Lux \n\n");

  char MsgSonido[50] = " Sonido\r"; // CREAMOS EL MENSAJE DEL SONIDO , USANDO UNA CONCATENACION
  strcat(MsgSonido, itoa(sonido, array, 10));
  strcat(MsgSonido, " Db \n");

  //CONCATENAMOS TODO EN UN UNICO MENSAJE PARA HACER UN TRASMIT
  strcat(MsgVelocidad, MsgTemperatura);
  strcat(MsgVelocidad, MsgSonido);
  strcat(MsgVelocidad, MsgLuminosidad);

  if (v == 1)
  { // COMPROBAMOS SI LLUEVE , Y SI LLUEVE CONCATENAMOS TAMBIEN EL MENSAJE DE LLUVIA
    strcat(MsgVelocidad, Llueve);
    v = 0;
  }
  HAL_UART_Transmit_IT(&huart1, MsgVelocidad, sizeof(MsgVelocidad));
}
```

Otro de los métodos que queremos dejar claro es *“HAL_ADC_ConvCpltCallback”,* el cual salta al llamar la interrupción del ADC. En este caso es más curioso ya que al implementarlo con un DMA de 4 conversaciones los valores del adc se guardan en el array *“buffer”,* es por ello que cada uno de los valores lo guardamos en el array *“adc”* para poder tratarlos individualmente. Además en este método donde cambiamos el pulso del TIM2.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
  if (hadc->Instance == hadc1.Instance)
  {
    for (int i = 0; i < 4; i++)
    {
      adc[i] = buffer[i];
      htim2.Instance->CCR1 = 75 + (adc[0] * 1425 / MAX_ANGLE);
    }
    HAL_ADC_Stop_DMA(&hadc1);
  }
}
```

Uno de los métodos que más hemos pensado ha sido el de la interrupción del TIM3. En este método hemos usado la fórmula para calcular el pulso que explicamos en "TIM2 y TIM3 Mode and Configuration" explicado previamente. Además está pensado de una manera que el periodo del TIM3 se puede cambiar para ofrecer una solución más flexible.

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{

  if (estado_onda == 1)
  {
    ValBajada = __HAL_TIM_GetCounter(htim);
    if (ValBajada < ValSubida)
      valtotal = htim2.Init.Period - ValSubida + ValBajada;
    else
    {
      valtotal = ValBajada - ValSubida;
      HAL_TIM_IC_Stop(&htim3, TIM_CHANNEL_1);
    }
    estado_onda = 0;
  }

  else
  {
    ValSubida = __HAL_TIM_GetCounter(htim);
    estado_onda = 1;
  }
}
```

Finalmente vamos a explicar nuestro bucle "while". Lo primero que tenemos declarado es activar el PWM del TIM2 para que sus subidas y bajadas puedan ser capturadas por nuestro IC más tarde. Activamos las dos interrupciones (ADC e IC) y esperamos 100ms para que de tiempo a que se actualicen los valores calculados en la interrupción anterior. Llamamos al métodos "formulas" que nos calculará las fórmulas explicadas en el apartado "Fórmulas" previamente explicado.

Hemos usado dos funciones que nos permiten limpiar las interrupciones pendientes y activarlas de nuevo, ya que en la interrupción "EXTI" las hemos desactivado tras realizar su función. Estas funciones son *"HAL_NVIC_ClearPending"* y *"HAL_NVIC_EnableIRQ".*

Por último llamamos al método "Control" que trata los estados y al método "Mensajes" previamente explicado, y esperamos 5s para volver a actualizar los datos.

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);

/* Infinite loop */

/* USER CODE BEGIN WHILE */
while (1)
{

  HAL_ADC_Start_DMA(&hadc1, buffer, 4);
  HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);
  HAL_Delay(100);
  Formulas();

Dejamos los print del monitor por si hay que probarlo sin bluetooth o da algun fallo el bluetooth
  /*printf("Luminosidad %d\n\r", luz);
  printf("Temperatura %d\n\r", temperatura);
  printf("Sonido %d\n\r", sonido);
  printf("Velocidad %d \n\r", Velocidad);
  printf("\n\r");*/


/+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++En estas lineas limpiamos las interrupciones y las volvemos a activar +++++++++++++++++++++++++++++++++++++++++++++++++++++
  HAL_NVIC_ClearPendingIRQ(EXTI9_5_IRQn);
  HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
  HAL_NVIC_ClearPendingIRQ(EXTI4_IRQn);
  HAL_NVIC_EnableIRQ(EXTI4_IRQn);

  Control();
  Mensajes();
  HAL_Delay(5000);
}
/* USER CODE END 3 */
```

Escenarios del vídeo

Escenario 1: La estancia permanece en silencio (relativo ya que en la sala había un poco de ruido) hasta que se aplica ruido mediante un altavoz para aumentar los decibelios.

Escenario 2: Se aplica luz intensa hasta que metemos el sensor en un puño para capturar el mínimo valor de luminosidad.

Escenario 3: Simulamos lluvia tocando el botón táctil y dejamos de pulsarlo para comprobar su buen funcionamiento.

Escenario 4: Se muestra la temperatura de la sala hasta que tocamos el sensor aplicando calor sobre el dispositivo.

Escenario 5: Se pone al mínimo el ángulo rotatorio y lo giramos lentamente mostrando como cambia la velocidad. Una vez que está en el máximo volvemos a girarlo al mínimo.

Escenario 6: Se aplican acciones usadas previamente en escenarios como la del uso del altavoz o el giro del potenciómetro demostrando el correcto funcionamiento de los LEDs al sobrepasar el umbral. Además hacemos uso del botón Reset para volver a encender los LEDs verdes y apagar los rojos hasta la siguiente iteración del bucle.

Enlace del video: https://youtu.be/022UhO4I4XY

BIBLIOGRAFÍA

A continuación vamos a mostrar las páginas y manuales usados para realizar la práctica.

Tutorial de uso de DMA:

https://www.youtube.com/watch?v=yDnGiLYBjDc

Fórmula del sensor de temperatura:

https://wiki.seeedstudio.com/Grove-Temperature_Sensor_V1.2/

Fórmula del sensor de luminosidad:

https://forum.arduino.cc/index.php?topic=331679.0

Fórmula del sensor de sonido:

https://forum.arduino.cc/index.php?topic=534279.0

Fórmulas para determinar Prescaler y Period:

https://www.intesc.mx/tutorial-2-timer/

Funciones HAL:

https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf