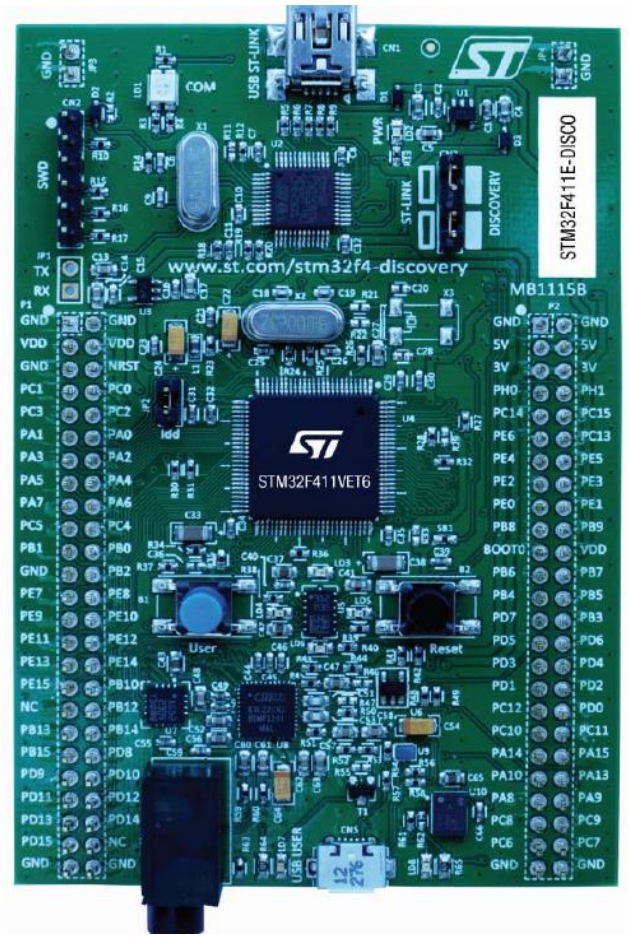
Trabajo Micros:
Control de Luces



Asignatura: SISTEMAS ELECTRONICOS DIGITALES (S.E.D).

Participantes:

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

Grupo: A-408.

ÍNDICE

1. Introducción:

Para la realización de este trabajo se contará con la placa STM32F411VE, sobre la cual se programará (con la ayuda de los programas “STMCubeMx” y “MDK-ARM” para configurar y programar respectivamente la placa). Para comprobar el funcionamiento se prepara en un protoboard los circuitos requeridos para cada una de las facetas del poryecto.

El proyecto elegido ha sido “Control de luces”, que al tratarse de una aplicación de domótica lo primero que se pensó fue en las posibles aplicaciones en un hogar. Con esta idea inicial se planteó como primera aproximación el control del encendido y apagado de leds en función de la luz exterior y mediante un potenciómetro controlar la intensidad de los mismos. Una vez desarrollada esta parte nos dedicamos a complementarla con la introducción de un LED RGB y un Joystick, de modo que con el Joystick se pudieran controlar los colores con los que ilumina el LED, y mediante una interrupción poder mantener ese color aunque se mueva el Joystick. Como detalle final se incluye un Display de 7 segmentos para poder ver al intensidad con la que alumbran los LEDs.

2. Funcionamiento:

Para comenzar se pueden diferenciar dos funcionalidades que al ser independientes se pueden separar: LEDs y LED RGB.

2.1 LEDS

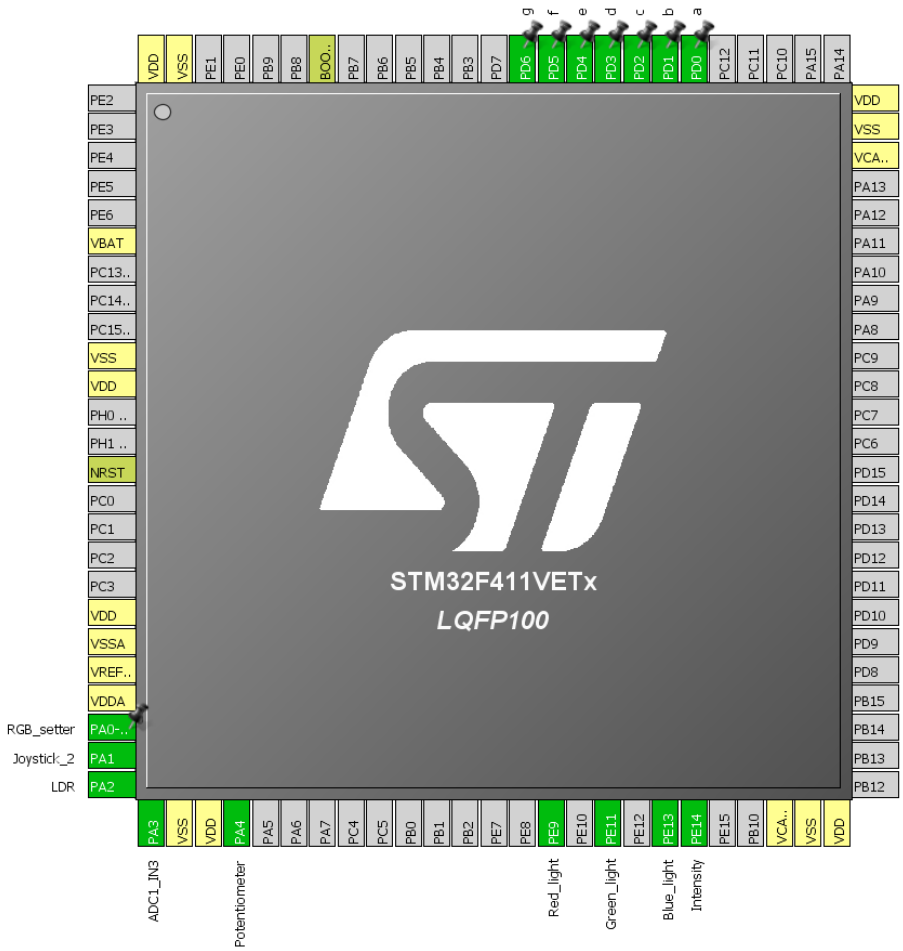
La parte de los LEDs responden ante la intensidad lumínica, de modo que si, mediante un LDR, detecta que hay luz no se encienden los LEDS. Y en el momento que la luz decae por debajo de un umbral se encienden los LEDS. Una vez encendidos se puede controlar la intensidad con la que iluminan mediante un potenciómetro, el cual determina el duty cycle de la salida PWM conectada a los LEDS.
Por último se planteó añadir un display para representar unos números, pero al dar problemas en su uso se ha decidido dejar de lado (aunque se mantiene el código comentado por si se detectan errores).

2.2 LED RGB

El LED RGB se ha planteado como un sistema ajeno al resto de LEDS, pensado para que el usuario pueda modificar mediante un Joystick el color de la luz, como una combinación de los

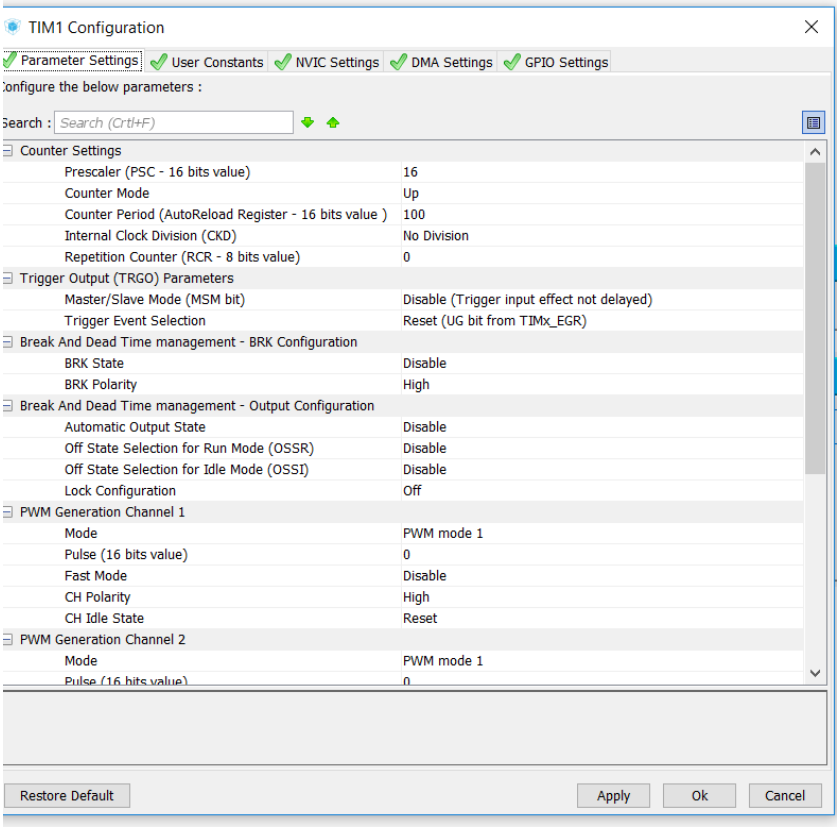
colores Rojo, Verde y Azul, y al pulsar el botón pueda tener la opción de dejar la luz seleccionada. Esta luz se mantiene, aunque varíe el potenciómetro, hasta que se pulse otra vez el botón y entonces volvería a estar disponible el Joystick para poder seleccionar el color.

En cuanto a la configuración del micro se han utilizado 4 canales del ADC para la lectura de los valores analógicos (los dos del Joystick, el LDR y el potenciómetro), así como los cuatro canales del Temporizador TIM1 en modo PWM (para gestionar los tres colores del RGB y la intensidad de los LEDs).



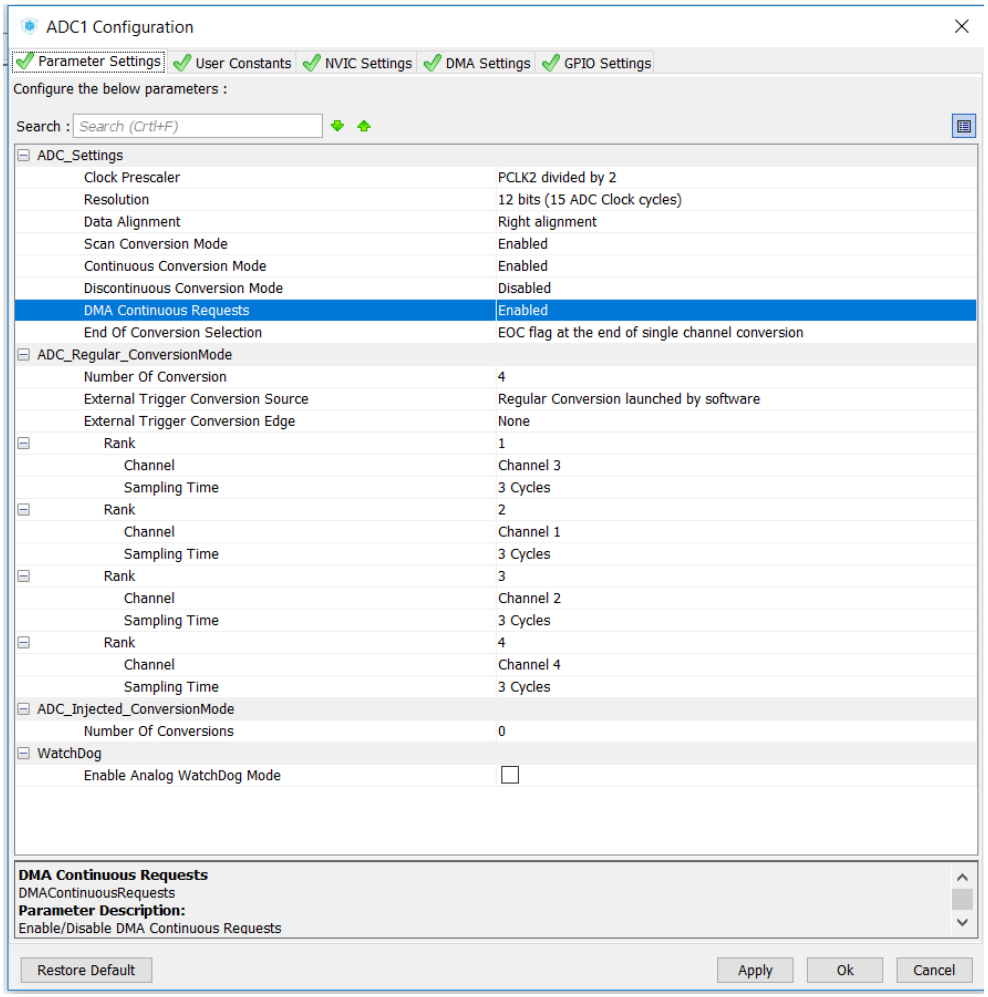
TEMPORIZADOR

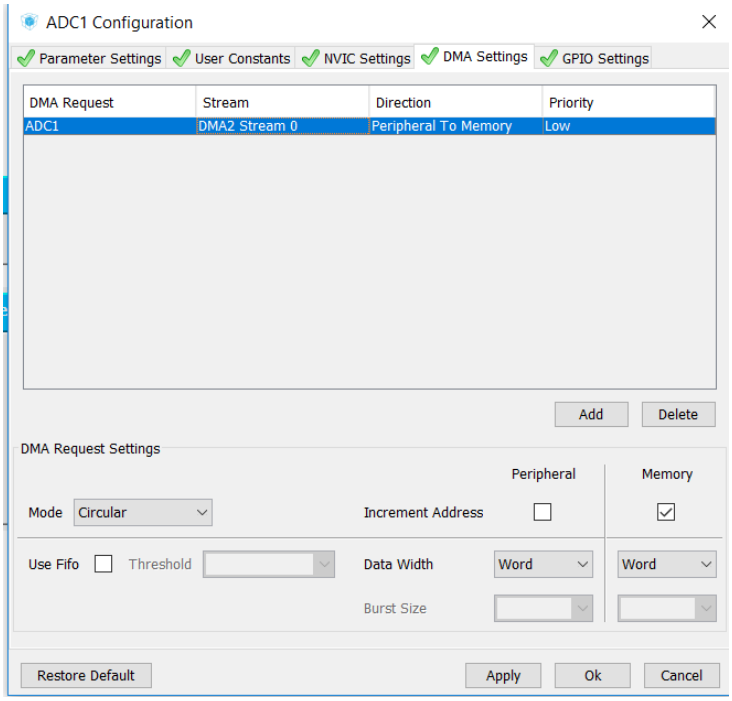
Para la configuración del temporizador se ha comprobado, tras múltiples pruebas, que con una frecuencia del orden de kHz serviría para que al gestionar el Duty Cycle no se perciba el parpadeo. De este modo al combinarse los colores RGB queda un color más o menos uniforme y en los LEDs se percibe como si se redujera la intensidad de manera continua.



ANALOG-DIGITAL-CONVERSOR

Para poder tomar las diferentes medidas primero se planteó el uso de las funciones vistas en el laboratorio, pero al intentar tomar las 4 se descubrieron múltiples errores ya que sólo tomaba dos y las duplicaba. Por lo que se optó por usar el acceso directo a memoria (DMA), para lo cual se sigue la siguiente configuración:


ADC1 Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
Configure the below parameters :
Search : Search (Crtl+F)
ADC_Settings
Clock Prescaler PCLK2 divided by 2
Resolution 12 bits (15 ADC Clock cycles)
Data Alignment Right alignment
Scan Conversion Mode Enabled
Continuous Conversion Mode Enabled
Discontinuous Conversion Mode Disabled
DMA Continuous Requests Enabled
End Of Conversion Selection EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode
Number Of Conversion 4
External Trigger Conversion Source Regular Conversion launched by software
External Trigger Conversion Edge None
Rank 1
Channel Channel 3
Sampling Time 3 Cycles
Rank 2
Channel Channel 1
Sampling Time 3 Cycles
Rank 3
Channel Channel 2
Sampling Time 3 Cycles
Rank 4
Channel Channel 4
Sampling Time 3 Cycles
ADC_Injected_ConversionMode
Number Of Conversions 0
WatchDog
Enable Analog WatchDog Mode
DMA Continuous Requests
DMAContinuousRequests
Parameter Description:
Enable/Disable DMA Continuous Requests
Restore Default
Apply
Ok
Cancel


ADC1 Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
DMA Request
Stream
Direction
Priority
ADC1
DMA2 Stream 0
Peripheral To Memory
Low
Add
Delete
DMA Request Settings
Peripheral
Memory
Mode
Circular
Increment Address
Use Fifo
Threshold
Data Width
Word
Word
Burst Size
Restore Default
Apply
Ok
Cancel

Al implementarlo se puede ver cómo los valores se modifican al implementar la función: *"HAL_ADC_Start_DMA (&hadc1, ADC_buffer, 4);"*, pero se comprueba que no se ejecuta el código del while(1).

Ante este problema se plantea el uso de threads para poder realizar el control del DMA. Para ello se prueba a usar FREERTOS con la función *"xTaskCreate(DMA_Control, "DMA_Control", configMINIMAL_STACK_SIZE, 0, 2, 0);"*, la cual permite que definas la función *"void DMA_control(void *)"* que ejecutaría *DMA_Control* de manera concurrente.

Esta solución tampoco sirvió, por lo que se decidió definir explícitamente la función *"void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef* hadc1)"* y dentro de ella programar el código de funcionamiento, ya que se comprueba que el código de dentro de Callback se ejecuta de manera continua.

INTERRUPCIONES

Para las interrupciones se aprovecha el botón PA0 asociarle la EXTI0 y de este modo gestionar la activación y desactivación de un flag que permite bloquear la función encargada de variar el color de las luces.

GPIO OUTPUT

Se han programado a su vez las salidas necesarias para poder controlar el display. Al tratarse de un Display de 7 segmentos de ánodo común, se diseñó el código para que se pusieran a 0 las puertas que se asociaran al LED encendido.

Tras múltiples pruebas (incluido cambiarlo por uno de cátodo común) no se coniguió un resultado satisfactorio. Como posibles problemas se han supuesto que podría ser mal contacto con los cables, o incluso que al estar conectado todo a las salidas de 5V y GND de la placa se vieran afectados los LEDs del Display al poner la entrada a 0 ya que igual no lea correctamente el voltaje respecto a la referencia (incluso se ha tratado de alimentar el Display de manera ajena con una pila).

Por último se comprueba mediante el debugger que a la hora de asignar las salidas de cada LED mediante *"HAL_GPIO_WritePin(GPIOE, GPIO_PIN_6, GPIO_PIN_RESET);"* llegado a un punto salta al final del *switch case* sin motivo aparente.

Finalmente se ha solucionado al cambiar las puertas por GPIOD, funciona perfectamente y sin ningún problema.

3. Código

Para el desarrollo del código se ha desarrollado dentro del callback del DMA. Tal como se ha explicado antes, y consiste en una serie de funciones sencilla que están explicadas en los comentarios del código.

4. Github

Para el desarrollo del código hemos utilizado GitHub como plataforma de control de versiones. Hemos ido desarrollando el código mediante commits en local para finalmente hacer una push y subirlo al repositorio online. Al ser un código tan sencillo se han desarrollado una sola rama al final para las últimas prubas quedando el final en la rama Alberto.

Para manejar todas estas herramientas nos hemos servido de Github Desktop, programa que facilita la gestión de los cambios y Atom para solventar errores al fusionar.

https://github.com/AlbertoTrapiello/Trabajo-Microcontroladores

5. Conclusión

En el desarrollo de este proyecto nos han sido de utilidad lo aprendido en las prácticas para hacernos una idea de cómo configurar y programar a nivel básico la placa. Dado que nos propusimos añadir funcionalidades un poco más complejas se ha requerido de mucha investigación en distintas webs (stackoverflow, controllerstech, etc) y en los manuales (tanto de HAL como de la STM32F411VE).

Aunque hay detalles que tal como se ha programado, tienen margen de mejora en caso de que en algún futuro nos dispusiéramos a desarrollar un proyecto similar fuera de la protoboard.