UNIVERSIDAD POLITÉCNICA DE MADRID

escuela técnica superior de ingeniería y diseño industrial

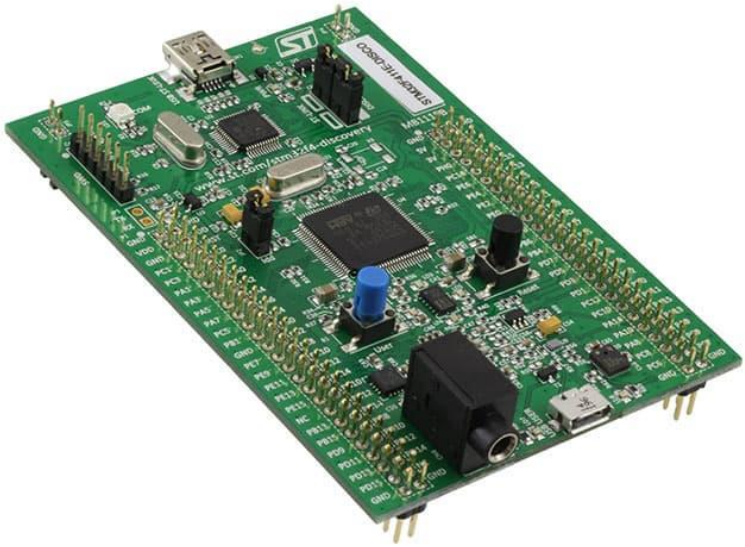
TRABAJO MICROPROCESADORES

CONTROL DE LUCES

Ismael Torrijano Pedroche 55492
Asier Miño Sierra 55364
Sergio Cabo Rodríguez 55161

INTRODUCCIÓN

El presente trabajo consiste en el diseño e implementación de un sistema de control de luces utilizando el microprocesador STM32F411 y la herramienta de desarrollo STM CubeIDE. El objetivo es lograr una solución eficiente y fácil de utilizar para controlar y regular la intensidad luminosa de diferentes luces de una habitación o espacio determinado. Se ha elegido el STM32F411 por su amplia variedad de periféricos y su alto rendimiento, y se ha utilizado STM CubeIDE como entorno de desarrollo integrado para facilitar la programación y depuración del sistema. A lo largo de este trabajo, se describirán los detalles del diseño y la implementación del sistema, incluyendo la selección de hardware y software, el proceso de desarrollo y las pruebas realizadas para validar su funcionamiento.



Microcontrolador STM32F411E-Disco

PLANTEAMIENTO DEL PROBLEMA

El “Control de luces” se utiliza generalmente para sistemas de domótica en la vida cotidiana. En este caso, se ha construido una habitación que detectara la luz proveniente de fuera, y tuviera distintos modos de iluminación internos. El problema que se plantea en este trabajo es la utilización de E/S tales como pulsadores, LED, display u otros. Se deberá diseñar y programar el sistema de control utilizando el STM32F411 y STM CubeIDE, y realizar las pruebas necesarias para garantizar su correcto funcionamiento, así como uso de interrupciones, temporizadores, ADC/DAC, etc.

DISEÑO E IMPLEMENTACIÓN

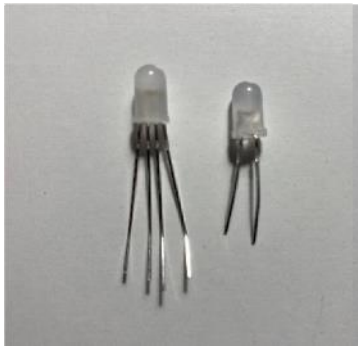
Funcionamiento básico

El funcionamiento del sistema de control de luces es el siguiente: se tiene una luz (LED) controlada por una foto resistencia (LDR), es decir, en los momentos de más luz del día esta luz estará apagada y cuando haya oscuridad (dependiendo del umbral de luz elegido) se enciende. Una vez encendida, esta luz también es controlada por un potenciómetro que regula su intensidad.

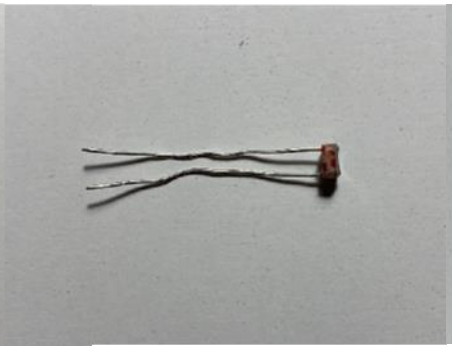
Otra de las opciones de este control es a través de un joystick que hace las veces de botonera, pero más cómodo, ya que dependiendo de la dirección en la que se mueva, se selecciona una opción u otra. Según la dirección a la que se mueva (arriba, abajo, izquierda o derecha) se selecciona entre los colores rojo, verde, azul o blanco (el blanco puede usarse durante las horas de menos luz para complementarse con la otra luz y aumentar la intensidad). Además, este control esta también manejado a placer por un botón que, una vez es pulsado, apaga el RGB y para poder habilitarlo de nuevo se debe volver a pulsar. Esto simularía un botón de OFF/ON.

Esta segunda opción de iluminación del RGB es ajena a la primera opción, es decir, se puede apagar o encender el modo RGB mediante el botón habilitado independientemente de la luz del exterior.

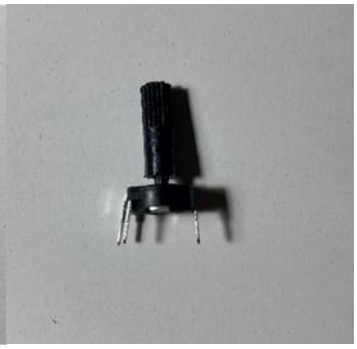
Componentes utilizados



LED normal y RGB



Sensor de luz LDR



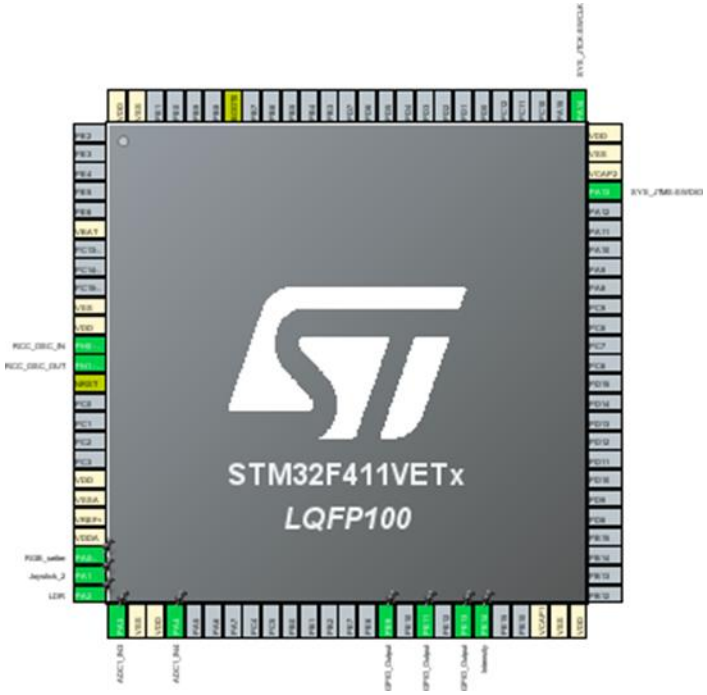
Potenciómetro

Joystick

Resistencias

Respectivamente LED y RGB LED, foto resistencia (LDR), potenciómetro, Joystick y Resistencias (de 220Ω para el LED y de 1kΩ para la LDR).

Configuración de E/S



Configuración de los pines

- **Temporizador (PE14)**
  Asociado como salida que alimenta el LED.
  Counter Settings

Prescaler (PSC - 16 ...	16
Counter Mode	Up
Counter Period (Aut...	100
Internal Clock Divisio...	No Division
Repetition Counter (...	0
auto-reload preload	Disable

  Configurado como un PWM controlado por el potenciómetro que va variando el valor de la duración del pulso.

```
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, Intensidad);
```

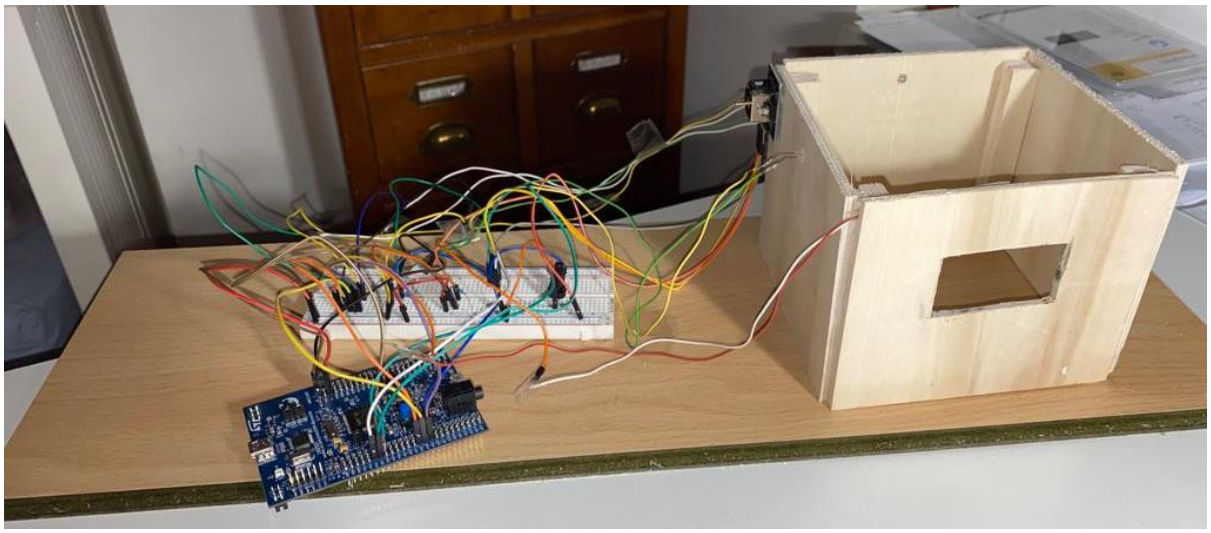
- **Conversor analógico digital (PA1, PA2, PA3, PA4)**
  El conversor además utiliza el acceso directo a memoria (ADM) a través de un buffer de cuatro bits. PA1 y PA3 corresponden a los parámetros x e y del joystick, PA2 corresponde a la LDR y PA4 corresponde al potenciómetro.

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

- **Botón (PA0)**
  Esta entrada PA0 está configurada como una interrupción externa, GPIO_EXTI0.
- **RGB (PE9, PE11, PE13)**
  Estas tres salidas están configuradas como GPIO_Output, aquí se conectan las diferentes patas de nuestro led RGB

MAQUETA REALIZADA

Para la realización de este trabajo, se ha diseñado y construido una maqueta utilizando maderas recicladas donde se instalarán los componentes del trabajo.



DESARROLLO DEL CÓDIGO

Para el funcionamiento del programa, se ha desarrollado el código principal dentro del callback del DMA, *HAL_ADC_ConvCpltCallback().*

Control a través de la LDR

Ya que el programa se ha usado el acceso directo a memoria también se ha hecho uso de la correspondiente interrupción:

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef hadc1)*

En esta interrupción están agrupadas las entradas analógicas al sistema (joystick, LDR y potenciómetro) por lo que al comienzo del programa se declara el valor umbral que definirá el encendido o apagado del LED a través de la función *bool enable_lights* comparándolo con él umbral.

Potenciómetro

El potenciómetro solo entrara el juego en caso de que la LDR lo permita, dicho esto el valor analógico potenciómetro define el valor de la intensidad del LED a través de:

Intensidad = ((Potentiometer) / 100) % 100;

Es decir, se selecciona el valor del potenciómetro y se va cambiando entre 0-100 ya que, como se ha comentado antes, el valor del periodo del temporizador es de 100 (100 intensidad máxima y 0 apagado) y según se varía el parpadeo dará sensación de más o menos luminosidad.

Control RGB

El led RGB se ha planteado como un sistema ajeno al otro led. Esta configuración está pensada de modo que el usuario pueda variar el color de las luces mediante un joystick, dependiendo del lado que elija se iluminara de distintos colores.

Lo primero en este caso es "mapear" los valores del joystick para tener unas referencias de derecha, izquierda, arriba y abajo. Para esto se ha creado una función map, la cual mediante los valores leído del propio joystick determinemos en qué posición se encuentra.

```
void map ()// funcion encargada de traducir el val
{
    R = U = D = L = 0;

    if (ADC_buffer[0] >= 4000)
        L = 1;
    if (ADC_buffer[0] <= 800)
        R = 1;
    if (ADC_buffer[1] >= 4000)
        U = 1;
    if (ADC_buffer[1] <= 800)
        D = 1;
}
```

En el momento en el que se habilite el modo del led RGB mediante la interrupción, se pasaría a la función del control de este LED. Esta función es la encargada, dependiendo de la posición que se haya determinado en la función map, de encender un color u otro. Hay 4 opciones de colores, si está a arriba se encenderá el azul, abajo el verde, derecha el azul, e izquierda se encenderán los tres colores a la vez, lo que simula a la vista un color blanco fuerte. Este último color es útil en el caso de tener el LED del primer modo al máximo con el potenciómetro y quieras más luz blanca.

FUNCIONAMIENTO DEL PROGRAMA

https://drive.google.com/file/d/1z1zEMQwpvm4tuOxGGY6iy8dVnO56KyFi/view?usp=share_link

REPOSITORIO ONLINE

Para el desarrollo de este trabajo se ha utilizado la herramienta online Github para llevar un seguimiento de las tareas realizadas. Se adjunta un enlace al repositorio.

https://github.com/ismael45/Control-Luces

CONCLUSIONES

La realización de este trabajo ha servido de ayuda para afianzar los conocimientos sobre el ámbito de los microcontroladores, en especial el microcontrolador STM32F411, y de la herramienta de programación STMCubeIDE. Las pruebas realizadas con el microcontrolador han sido de gran utilidad, ya que han permitido llevar un control del programa exhaustivo. En cuanto al código que se ha construido en la herramienta STMCubeIDE, se han realizado diversas versiones hasta llegar a la definitiva, teniendo en cuenta el uso de repositorios de ayuda así como de enlaces a webs que proporcionen información de ayuda sobre el proyecto. La realización de las prácticas de laboratorio ha sido de gran ayuda ya que dichas prácticas han sentado una base sobre el uso de interrupciones, temporizadores o conversores A/D. En definitiva, este trabajo abre un amplio abanico de posibilidades en cuanto a la programación en microcontroladores debido a la gran diversidad de estos.