SISTEMAS ELECTRÓNICOS DIGITALES

CURSO 2022/23

TRABAJO DE MICROS

Control de un estor

GRUPO 12

Gallego Torres, Luis Francisco (55249)

Álvarez López, Alberto (55117)

Berrendo Carretero, Sergio (55147)

Contenido

Introducción

Este proyecto consiste en una adaptación de la segunda opción presentada en la oferta de trabajos de microprocesadores. En nuestro caso, hemos decidido realizar el montaje de un estor en miniatura, una interesante aplicación de domótica de la que se pueden derivar distintos usos y productos.

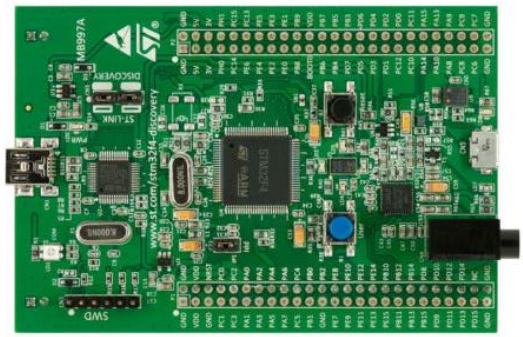
Para demostrar su funcionamiento, se ha construido una maqueta con materiales reciclados y de fácil acceso, para minimizar costes e impacto ambiental. Hemos optado por combinar control manual a través de un potenciómetro con control automático en función de la luminosidad detectada por un sensor LDR.

En este informe se recogen todas las características del proyecto, desde su descripción y funcionamiento, hasta su desarrollo y las conclusiones a las que hemos llegado.

Maqueta y componentes

El diseño de nuestra maqueta del estor se ha tratado de hacer de la forma más ecológica posible, reutilizando materiales que teníamos en casa a los que no dábamos uso, o que podíamos aprovechar para implementar lo que teníamos en mente; de forma que tuviéramos que comprar lo mínimo. Éstos se mencionan en la siguiente lista de materiales:

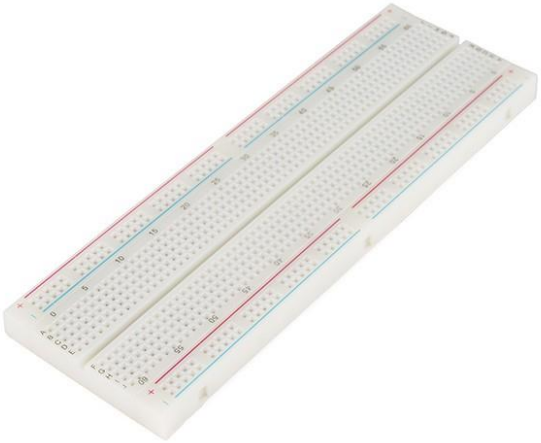
Placa STM32F4DISCOVERY: componente en el que se describe todo el funcionamiento del programa con el fin de que todo funcione de forma correcta.



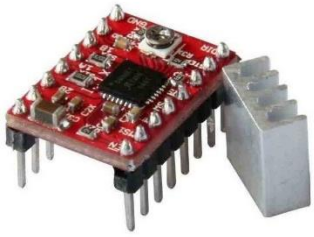
Motor paso a paso de 12 V Nema 17: actuador situado en el eje del estor que le permitirá subir y bajar según está programado.

Batería de 12 V: para suministrar la energía necesaria al motor paso a paso.

Protoboard: placa en la que se realizan todas las conexiones entre cables y componentes de nuestra maqueta.



Driver A4988: empleado para poder realizar todas las comunicaciones pertinentes entre la placa y el motor.

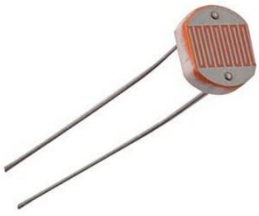


Cables: para realizar las conexiones entre componentes.

Potenciómetro de 10kΩ: componente con el que la altura del estor puede ser regulada a voluntad del usuario.

Botón: para alternar entre el modo automático y manual del estor (incluido en la placa).

LDR pequeño: detecta el nivel de luminosidad con el fin de determinar si el estor debe estar extendido o recogido en el modo automático.



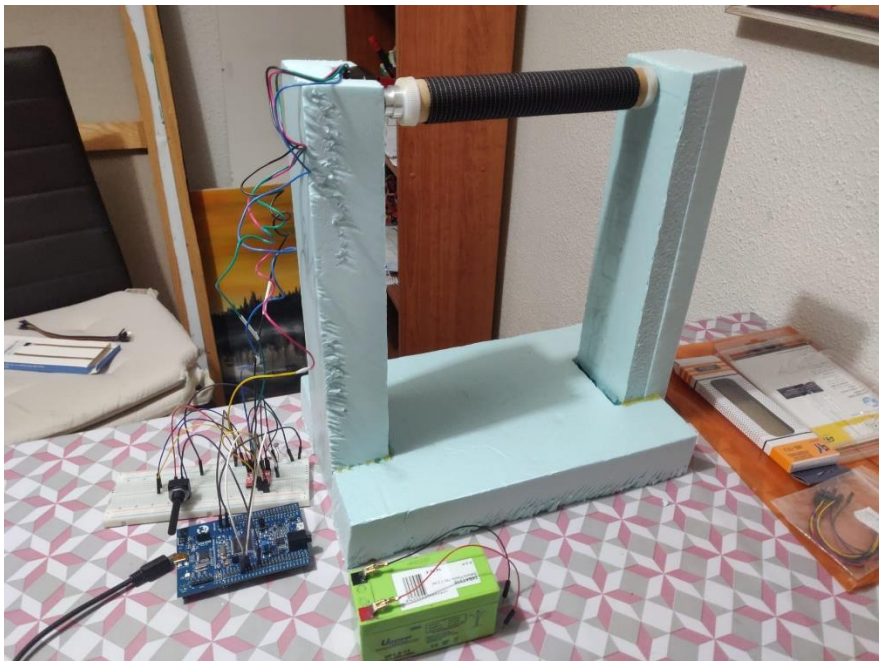
Poliestireno extruido: reciclado tras haber sido empleado en labores domésticas, es el que da forma a la estructura de la maqueta, tanto la base como los pilares de apoyo del estor y el motor.

Rollo de papel: actúa como eje en torno al que se recoge la tela del estor.

Trozo de tela: material del que está hecho el estor.

También se han empleado tapones de botellas y otros útiles para asegurar la fijación de la barra del estor al eje de rotación del motor.

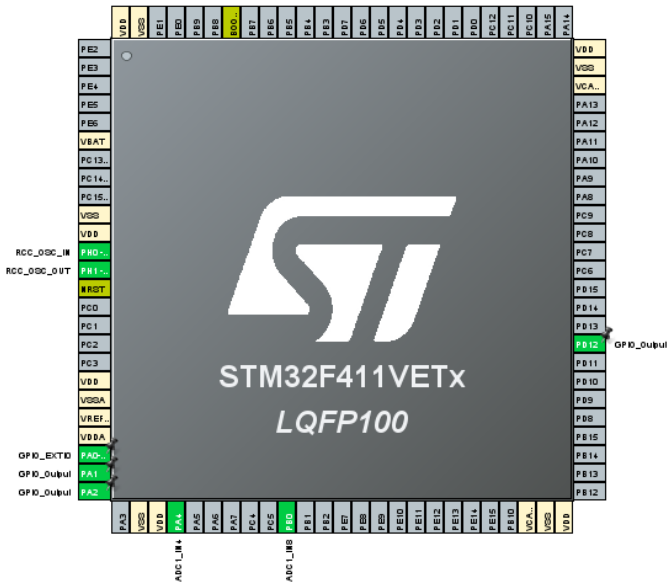
La maqueta consiste en una base de poliestireno sobre la que se alzan dos pilares del mismo material. En lo alto de uno de ellos se halla encajado el motor paso a paso, en cuyo eje de rotación va anclado el eje del estor, que también se apoya en la otra columna. En dicho eje, el rodillo, está sujetada la tela del estor, que cuelga hasta la base de la maqueta. El motor está conectado mediante cables a una batería que lo alimenta, y a la protoboard, donde se encuentran el controlador, el potenciómetro para el modo manual y el LDR con el que se detecta la luminosidad. Así mismo, a la protoboard también va conectada la placa STM32, que dicta el funcionamiento del programa.



Para su mejor visualización se ha grabado un vídeo explicando su funcionamiento, el cual se encuentra en el repositorio del trabajo.

Descripción del programa

Para realizar la programación del estor hemos dividido distintas funcionalidades de los componentes en funciones con el fin de organizar el código y que todo marchara como es debido. A lo largo de este apartado se irán mostrando y detallando las partes del código implementado.

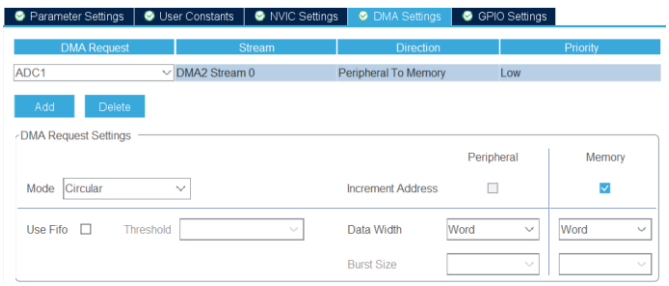


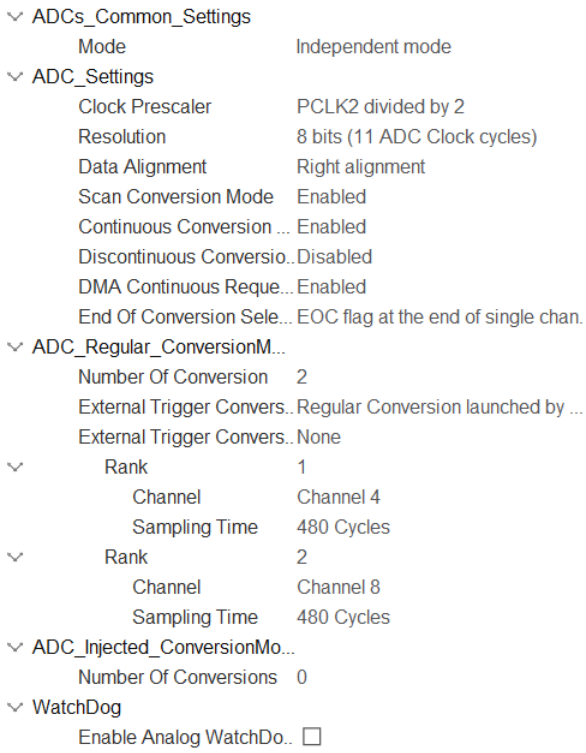
A modo de descripción general y aprovechando la interfaz *.ioc* del programa, podemos mostrarlas distintas configuraciones elegidas. Los pines de propósito general utilizados fueron los siguientes:

Pin...	Signal ...	GPIO ...	GPIO ...	GPIO ...	Maxim...	User L...	Modifi...
PA0-...	n/a	n/a	Extern...	No pull...	n/a		☐
PA1	n/a	Low	Output...	No pull...	Low		☐
PA2	n/a	Low	Output...	No pull...	Low		☐
PD12	n/a	Low	Output...	No pull...	Low		☐

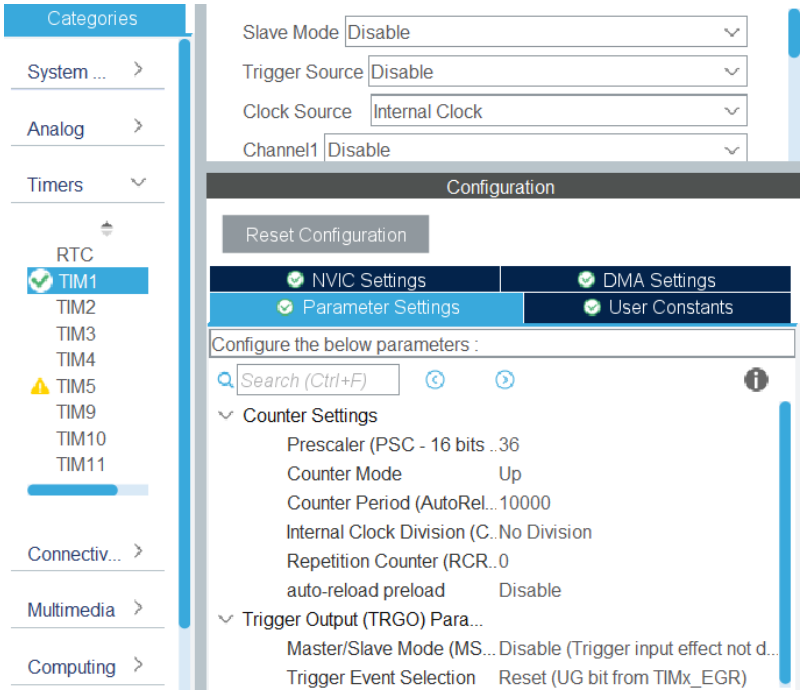
Los pines PA1 y PA2 corresponden al control del motor paso a paso, para el paso y la dirección respectivamente. El pin PA0 está configurado en el modo *GPIO_EXTIO*, siendo este el conectado al botón de la placa. Finalmente, el PD12 se seleccionó como la salida con la que se controla el LED correspondiente de la placa.

Para la conversión analógica/digital se emplearon dos canales del ADC1 de la placa, siendo estos el IN4 e IN8, con sus correspondientes pines PA4 y PB0. Al emplear múltiples canales de conversión y querer que fuera simultánea, se optó por el uso de DMA para la conversión. La configuración realizada para ello fue la siguiente:





Para el proyecto se hizo uso de un temporizador, siendo esta su configuración:



Una vez habiendo presentado la configuración del proyecto, vamos a presentar el código utilizado.

En el *main* del programa se encuentra simplemente la inicialización de las distintas partes del proyecto y dentro del *while* se sitúa el control del motor. Este codifica el movimiento del motor en una dirección u otra conforme a las medidas obtenidas de la conversión analógica/digital. Como se puede ver, el movimiento con la luz es completo, es decir, sube y baja en toda su longitud cuando se superan por encima o debajo los umbrales estipulados. Cabe mencionar que estos están seleccionados arbitrariamente para favorecer el funcionamiento de la maqueta y no representan valores realistas de iluminación solar.

```
while (1)
{
    if (button_int % 2 == 0){
        if (ADC_luz > 70){
                        while (posicion < y_extendido){
                            step(1, 0, 1); // 1 paso horario
                            //HAL_Delay(10);
                            posicion ++;
                        }
                    }
        else if (ADC_luz < 50) {
                        while (posicion > y_recogido){
                            step(1, 1, 1); // 1 paso antihorario
                            //HAL_Delay(10);
                            posicion --;
                        }
        }
    }
    else if (button_int % 2 == 1){
            if (y > posicion) {
                while (posicion < y) {
                    step(1, 0, 1); // 1 paso horario
                    //HAL_Delay(10);
                    posicion++;
                }
            }
            else if (y < posicion) {
                while (posicion > y) {
                    step(1, 1, 1); // 1 paso antihorario
                    //HAL_Delay(10);
                    posicion--;
                }
            }
    }
```

La variable *button_int* es la encargada de definir si se hace uso del control automático o manual.

Para el resto del funcionamiento se desarrollaron funciones externas o se hizo uso de las diferentes herramientas de programación, como *callbacks* o interrupciones.

La primera función que cabe destacar es step. Esta función simplemente se encarga de trasmitir a los pines de salida del motor las órdenes de movimiento determinadas por el programa. La función cuenta con funcionalidades más allá de las necesarias para el funcionamiento de nuestro estor, como puede ser la selección de un *delay* o la posibilidad de realizar varios pasos con una misma orden. Éstas fueron implementadas desde un primer momento con la idea de ser usadas más adelante, pero por unos motivos u otros (que se detallan en el apartado de problemas y hoja de ruta) no se acabaron usando. Creemos que es importante dejarlas en el programa, ya que forman parte de su desarrollo y además pueden ser usadas en un futuro si decidiéramos perfeccionarlo.

```
void step (int step, uint8_t direccion, uint16_t delay)
{
  int x;
  if (direccion == 0)
    HAL_GPIO_WritePin(DIR_PORT, DIR_PIN, GPIO_PIN_SET);
  else
    HAL_GPIO_WritePin(DIR_PORT, DIR_PIN, GPIO_PIN_RESET);
  for(x=0; x<step; x=x+1)
  {
    HAL_GPIO_WritePin(STEP_PORT, STEP_PIN, GPIO_PIN_SET);
    //microDelay(delay);
    HAL_Delay(delay);
    HAL_GPIO_WritePin(STEP_PORT, STEP_PIN, GPIO_PIN_RESET);
    //microDelay(delay);
    HAL_Delay(delay);
  }
}
```

La siguiente parte que queremos mencionar es el uso del *callback* de conversión completa de las señales analógicas. Gracias al uso del *DMA* se pueden recoger los valores de la conversión CAD en tiempo real directamente a memoria sin quitar uso del procesador. En el *callback* se recogen los valores del LDR y el potenciómetro, siendo este último además convertido a otro con el que se puede hallar la altura del estor.

```
void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef * hadc){

    if (hadc->Instance == ADC1){
        adcvalue[0]=adcbuffer[0];
        adcvalue[1]=adcbuffer[1];
        ADC_luz = adcvalue[0];
        ADC_potenciometro = adcvalue[1];
        y = getPotenciometro(ADC_potenciometro);
    }
}
```

En esta función se llama también a *getPotenciometro*, una función designada para transformar el valor del potenciómetro a la escala del estor, además de dividirlo en tramos para paliar en cierta medida el ruido del potenciómetro, detalle que se comentará más adelante.

```
int getPotenciometro(uint16_t pot){
    int altura, tramo;
    altura = (pot * y_extendido) / max_potenciometro ;
    tramo = altura/51; //tramos de 17
    altura = tramo * 51;
    return altura;
}
```

Finalmente, para el uso del botón con el que cambiamos de tipo de control del motor, se hizo uso de dos recursos de HAL, como son el *HAL_GPIO_EXTI_Callback* y el *HAL_TIM_PeriodElapsedCallback*. El empleo de estas funciones se aprovechó para indicar el cambio de control por el usuario y además programar un antirrebotes del actuador. Gracias a la interrupción, cuando el botón de la placa es pulsado, se accede al *EXTI_Callback*. En ese momento se dispara un temporizador básico con el que se llamará al *TIM_PeriodElapsedCallback* cuando se alcance el valor establecido del temporizador. Entonces se activa el LED de indicación de estado, se detiene el temporizador y se indica al programa que se ha pulsado efectivamente el botón a través de *button_int*.

Cabe mencionar el uso del *flag button_state*, el cual nos sirve para determinar que se dispare el temporizador solo la primera vez que se pulse, evitando los rebotes.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
        if(GPIO_Pin == BUTTON_PIN && button_state == 1){
            HAL_TIM_Base_Start_IT(&htim1);
            button_state = 0;
        }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance==TIM1){
        if(HAL_GPIO_ReadPin(BUTTON_PORT, BUTTON_PIN) == GPIO_PIN_RESET){
            HAL_GPIO_TogglePin(LED_PORT, LED_PIN);
            button_state = 1;
            button_int++;
            HAL_TIM_Base_Stop_IT(&htim1);
        }
    }
    /*if(htim->Instance==TIM3){
        HAL_TIM_Base_Stop_IT(&htim3);
        elapsed = 1;
    }*/
}
```

Además de lo mencionado, se han utilizado varias variables, situadas al comienzo del código, con las que se han podido llevar a cabo todas estas operaciones.

Problemas y ruta de desarrollo

A lo largo del desarrollo del proyecto, hemos pasado por diferentes etapas y situaciones que nos han obligado a tomar unas decisiones u otras, llevándonos hasta el resultado final que presentamos. El concepto original del trabajo no difería en gran medida de lo que ha acabado resultando, pero describiremos nuestro camino recorrido en nuestro progreso.

En nuestro primer acercamiento decidimos desarrollar un estor a partir de materiales que teníamos a mano los tres integrantes del grupo, reciclando aquellos que nos pareciesen mejor y rescatando componentes electrónicos de proyectos pasados de los que no haríamos más uso. El mejor ejemplo es la plancha de poliestireno extruido, el cual cortamos a mano en trozos del tamaño considerado para dar forma a nuestra maqueta.

Originalmente se decidió tratar de controlar el motor con la iluminación captada por un LDR y emplear un detector infrarrojo a modo de sensor de presencia humano. Para ello se desarrolló una primera versión del control de un motor paso a paso (escogido por su mayor control de posición además de ofrecer un mayor par en caso de necesitarlo), fruto de la investigación por internet y el ensayo de prueba y error. De este proceso surgió la versión inicial de una función llamada *microDelay*, la cual se encuentra comentada en el código. Esta función permitía un mejor control del motor paso a paso estableciendo *delays* entre los pasos inferiores al milisegundo, consiguiendo así una mayor fluidez en el movimiento. Para ello se hacía uso de un temporizador básico con el que se detenía el programa durante unos microsegundos cada vez que se llamaba a la función.

Pero con la necesidad de realizar dos lecturas analógicas simultáneas y la inclusión del DMA, la detención completa del programa en esa función dificultaba el funcionamiento general. Es por eso por lo que en la versión final el *delay* es de 1 milisegundo, repercutiendo en la fluidez aun no siendo algo problemático.

También conforme avanzaba el proyecto se desechó la idea de usar un detector infrarrojo a modo de sensor de presencia debido a la mala experiencia de uso que teníamos y los problemas que nos estaba dando la unidad con la que contábamos. Para completar el trabajo se tomó la decisión de tratar de controlar el estor con un potenciómetro, para que el usuario pudiera decidir a gusto la altura del estor.

En sí el desarrollo de la funcionalidad fue efectivo, el estor sique las indicaciones del potenciómetro con velocidad, pero presenta un problema: es muy complicado que el motor se quede parado en un punto, ya que debido al ruido presente en la señal oscila arriba y abajo produciendo una vibración algo desagradable. Para ello se probaron varios métodos correctivos, como son el uso de filtros analógicos, el cálculo de una media de la medida o el uso de tramos de movimiento, siendo este último el elegido por su mejor, aunque no perfecto funcionamiento.

El uso de tramos permite que el motor se sitúe en menos puntos, elegidos en una distribución equitativa de la longitud del recorrido. De esta forma, la gran mayoría de los puntos del potenciómetro son estables para el motor, aunque en aquellos donde se produce un cambio de tramo sigue apareciendo ruido.

Finalmente, a la vez que se implementó el potenciómetro, se incluyó el uso del botón para cambiar de una funcionalidad a otra; de una forma por la que si se quisiera añadir otra funcionalidad sería muy sencillo.

Instrucciones y funcionamiento

Nuestro diseño consiste en un pequeño estor motorizado que simula el comportamiento de uno de mayor tamaño. El funcionamiento principal es su recogida y extensión de forma automatizada en función del nivel de luz detectado. Por defecto, nuestro estor comienza completamente retraído, si detecta que la luminosidad ha aumentado traspasando cierto umbral, éste se extiende simulando lo que haría uno a tamaño real, tratando de atenuar la iluminación de una habitación. Si posteriormente detecta que la luminosidad baja de un umbral inferior, el estor volverá a la posición de retracción completa.

Por otro lado, si se pulsa el botón de la placa se accede a un modo de ajuste manual: a través de un potenciómetro es posible regular la posición del estor a gusto del usuario, coincidiendo los límites de extensión y retracción con los de giro del potenciómetro. Si de nuevo se pulsase el botón de usuario de la paca, se regresaría al control automático mediante iluminación. Con el fin de facilitar el uso, si el programa se encuentra en modo manual, se enciende un led de la placa.

Conclusiones y consideraciones

Con la realización de este trabajo hemos obtenido una base sólida de conocimientos en la programación de microcontroladores, dada por la cantidad de horas dedicadas a redactar código y buscar información para sacar el proyecto adelante. Además, no solo hemos sido capaces de aprender más acerca de la programación de la placa STM32, sino que también hemos podido crear una maqueta ecológica y funcional de la que nos sentimos orgullosos. Somos conscientes de que nuestro proyecto podría no resultar excesivamente complejo ni muy atractivo, pero creemos que nuestros objetivos personales han sido cumplidos en su gran mayoría; habiendo satisfecho nuestros planteamientos y recibido una cantidad de experiencia tanto en lo que a desarrollar un proyecto de estas características supone, como el la coordinación y cooperación entre miembros de un grupo.