TRABAJO SED
MICROS

GRUPO Nº. 15

CONTROL DE PERSIANAS

INTEGRANTES

FRANCISCO MARTÍN CUSCURITA 55972

JIMENA LÓPEZ MALDONADO 55951

JOAN BELLIDO INES 55745

ÍNDICE

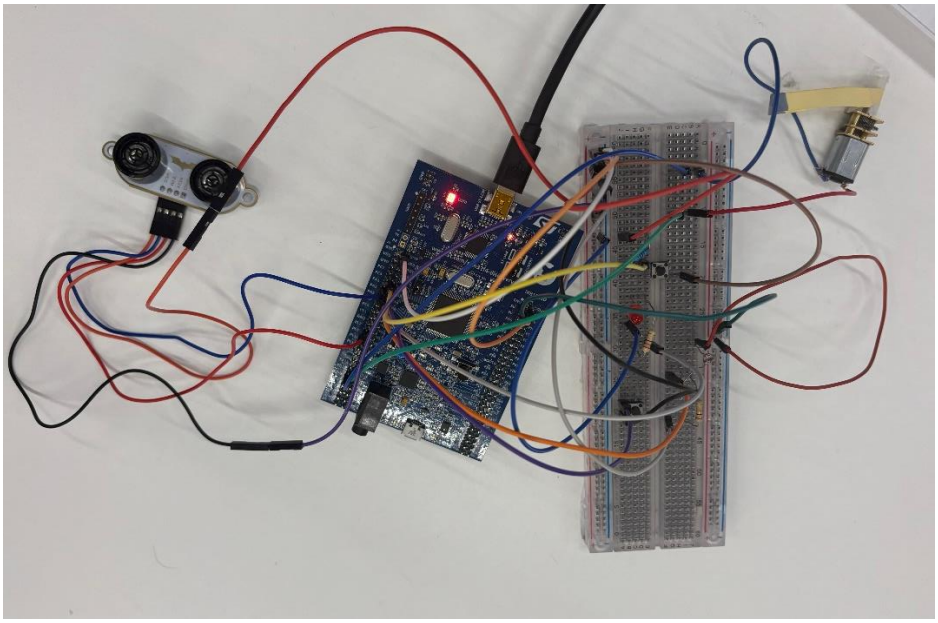
1.- INTRODUCCIÓN Y OBJETIVOS

1.1.- Enunciado

En esta sección del trabajo correspondiente a la asignatura, se llevará a cabo la implementación de un diseño en el microcontrolador **STM32F411** ,**Aplicación Domótica** .

El enunciado del diseño plantea lo siguiente:

"Realización de una aplicación domótica con el microcontrolador, a determinar por el alumno, en el que se deben usar las entradas y las salidas para implementar diversas funciones: control de luces, persianas, riego, climatización, seguridad. (detección de movimiento, detección perimetral, apertura de puertas y ventanas). Se podrán establecer diversos modos de control: manual, automático, en función de condiciones medioambientales, simulación de presencia, etc. Se deberán definir varios sistemas de interacción: pulsadores, pantallas, aplicaciones móviles, páginas web, etc."

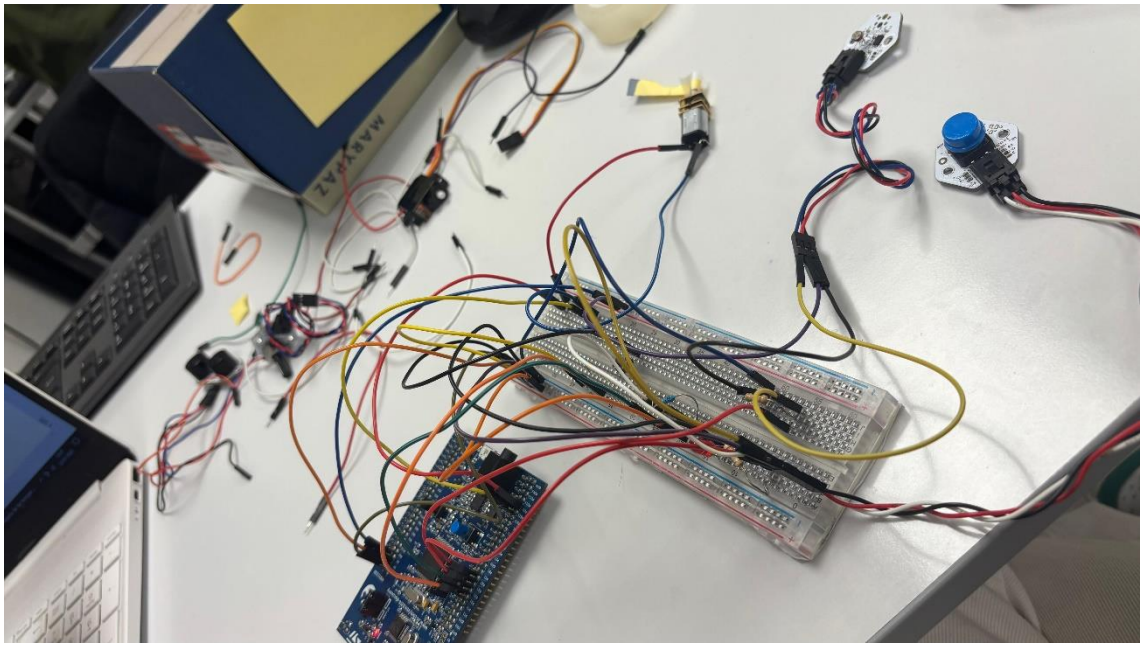


2.- DESCRIPCIÓN DEL PROYECTO

2.1.- Funcionamiento

El objetivo principal de nuestro proyecto es implementar un sistema de gestión de persianas en una habitación, que funcione tanto en modo automático como manual. Además, hemos desarrollado un "modo noche", diseñado específicamente para monitorear posibles intrusos cuando la habitación se encuentra sin iluminación.

El podrá cambiar entre los distintos modos de operación de las persianas usuario mediante un botón conectado al pin PA2. Con la primera pulsación, se activará el modo automático, mientras que una segunda pulsación permitirá cambiar al modo manual.



2.1.1- Modo automático

En el modo automático, se utilizará un fotorresistor para medir la cantidad de luz que ingresa a la habitación. Según esta medición, las persianas se ajustarán subiendo o bajando automáticamente. Cuando cae la noche y la luz ambiental disminuye, las persianas se cerrarán si están levantadas.

Si el sensor no detecta luz durante un periodo superior a 10 segundos, se activará el modo noche, cuyo funcionamiento explicaremos más adelante. Por otro lado, si se vuelve a registrar la presencia de luz, las persianas se subirán de nuevo.

2.1.2- Modo manual

Se trata del modo de funcionamiento más sencillo. Controlaremos la subida y bajada de las persianas con un botón auxiliar conectado al pin PA0.

2.1.3- Modo noche

Como se mencionó en el modo anterior, el modo noche permanecerá activo mientras la habitación esté oscura y las persianas estén cerradas. En esta situación, un sensor de presencia se encargará de monitorear la entrada de posibles intrusos. Si se detecta movimiento a menos de 10 centímetros de distancia, se activará una alerta para el usuario mediante el encendido de un LED.

El modo noche finalizará cuando el fotorresistor registre la presencia de luz durante al menos 10 segundos. En ese momento, las persianas se elevarán automáticamente y el LED se apagará, en caso de estar encendido.

2.2.- Componentes empleados:

● **Microcontrolador STM32F411:** Este microcontrolador basado en arquitectura ARM ha sido la base de nuestro proyecto. En él se han conectado los diferentes dispositivos adicionales utilizando una variedad de puertos que gestionan tanto las entradas como las salidas del sistema.

● **Sensor de luminosidad (LDR):** Su función principal es determinar si es de día o de noche, dependiendo de la intensidad luminosa que percibe. Está compuesto por materiales fotosensibles que modifican su resistencia eléctrica según la cantidad de luz recibida. En ambientes oscuros, su resistencia es elevada, restringiendo el flujo de corriente. En cambio, al exponerse a la luz, la resistencia disminuye, permitiendo un mayor paso de corriente.

● **Sensor ultrasónico HC-SR04:** Este sensor se utiliza para detectar intrusos simulando un sistema de vigilancia. Funciona enviando un pulso de alto voltaje al pin Trigger, lo que genera una serie de ondas ultrasónicas. Estas ondas rebotan al encontrar un obstáculo y regresan al sensor en forma de eco. Midiendo el tiempo entre la emisión y la recepción de la onda, y conociendo la velocidad del sonido, se calcula la distancia al objeto detectado.

● **Motor de corriente continua:** Es un dispositivo electromecánico que convierte energía eléctrica en movimiento mecánico. En este caso, genera la rotación del eje al que se encuentra enrollada la cortina de la persiana.

● **Diodo LED:** Este componente se enciende para alertar al usuario cuando el sensor ultrasónico detecta la presencia de un intruso en el entorno.

● **Dos botones:**

- El primero permite alternar entre los modos de funcionamiento de la persiana: **automático** o **manual** . En el modo automático, el movimiento de la persiana depende de la luz detectada por el sensor LDR.
- El segundo botón se emplea exclusivamente en el modo manual. Al presionarlo, la persiana sube si está bajada, o desciende si está subiendo.

● **Materiales para la maqueta:** Se emplearon diversos materiales para diseñar y construir la estructura de la maqueta que soporta el sistema.

3.- CÓDIGO

3.1.- Estructura

El programa se organiza en tres archivos principales : **controlServo.h** , **controlPresencia.h** y **main.c.** Esta estructura se elige para separar las distintas funcionalidades y mantener el código principal más claro y compacto.

● **controlServo.h:** Este archivo se encarga exclusivamente de gestionar el motor que controla las persianas. Contiene funciones específicas para realizar las operaciones de subir, bajar y detener el movimiento de las persianas.

● **controlPresencia.h:** Su función principal es calcular la distancia detectada por el sensor ultrasónico, utilizando los datos proporcionados por el módulo.

● **main.c:** Este archivo centraliza la lógica del control del sistema. Coordina todos los componentes y utiliza las funciones definidas en los otros archivos, además de gestionar las interrupciones necesarias para el correcto funcionamiento del proyecto.

3.2.- Configuración de los pines

El proyecto utiliza un total de 8 pinos configurados para diferentes funciones:

- **Canales de interrupciones:**
  Los pinos **PA0** y **PA2** se han configurado para generar interrupciones. Estos se activan mediante los botones de control de persianas y de cambio de modo, respectivamente.
- **Temporizadores:**
  - Para la modulación de pulso del *disparador* del sensor ultrasónico, se emplea un temporizador PWM en el pin **A3** con una frecuencia de 40 Hz (según las especificaciones del componente) y un ciclo de trabajo del 10%.
  - La detección del flanco de llegada del pulso se realiza con un temporizador configurado en modo *captura de entrada* en el pin **E9** , al que se conecta el *eco* del sensor. Este opera a una frecuencia aproximada de 45 Hz, utilizando el pulso recibido para calcular la distancia.

- **Convertidor ADC:**
  Para medir la luminosidad con un LDR, se utiliza un convertidor ADC de 8 bits de resolución.
- **Salidas GPIO:**
  - Para indicar la detección de presencia, se emplea la salida **A6** conectada a un LED.
  - El motor DC se controla mediante las salidas **PD8** y **PD9** , cuya polaridad define el sentido de giro (subir o bajar las persianas).

3.3.- Control de persianas

El control de persianas gestiona el movimiento del motor y determina la posición de las cortinas. A continuación, se presenta el código del archivo de cabecera:

```
#ifndef INC_CONTROLPERSIANAS_H_
#define INC_CONTROLPERSIANAS_H_

#include "stm32f4xx_hal.h"
#include <stdbool.h>

// Definiciones de pines y periféricos para el motor
#define MOTOR_PORT GPIOD
#define MOTOR_PIN1 GPIO_PIN_9
#define MOTOR_PIN2 GPIO_PIN_8

extern bool persianasArriba;
extern bool persianasAbajo;

void subirPersianas();
void pararPersianas();
void bajarPersianas();
```

#endif /* INC_CONTROLPERSIANAS_H_ */

La implementación de estas funciones es la siguiente:

do

Copiar código

```
#include <controlPersianas.h>

bool persianasArriba = false;
bool persianasAbajo = true;

void subirPersianas() {
    HAL_GPIO_WritePin(MOTOR_PORT, MOTOR_PIN1, GPIO_PIN_SET);
    HAL_GPIO_WritePin(MOTOR_PORT, MOTOR_PIN2, GPIO_PIN_RESET);
    HAL_Delay(3000); // Tiempo estimado para subir las persianas
    pararPersianas();
    persianasArriba = true;
    persianasAbajo = false;
}

void bajarPersianas() {
    HAL_GPIO_WritePin(MOTOR_PORT, MOTOR_PIN1, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(MOTOR_PIN2, GPIO_PIN_SET);
    HAL_Delay(3000); // Tiempo estimado para bajar las persianas
    pararPersianas();
    persianasArriba = false;
    persianasAbajo = true;
}
```

```
void pararPersianas() {
   HAL_GPIO_WritePin(MOTOR_PORT, MOTOR_PIN1, GPIO_PIN_RESET);
   HAL_GPIO_WritePin(MOTOR_PORT, MOTOR_PIN2, GPIO_PIN_RESET);
}
```

Nota:
El tiempo de retraso (HAL_Delay) depende del tiempo físico necesario para subir o bajar las persianas. La dirección del movimiento (subida o bajada) se determina por la polaridad de la alimentación del motor, que se ajustará al finalizar la maqueta.

3.4.- Control de presencia

El control de presencia detecta objetos o personas cercanas al sensor ultrasónico y los interpreta como intrusos si están a una distancia determinada. El sensor ultrasónico opera enviando una señal, que rebota en un objeto y regresa al sensor a través del pin *ECHO* . El archivo de cabecera declara las variables necesarias:

```
#ifndef INC_CONTROLPRESENCIA_H_
#define INC_CONTROLPRESENCIA_H_

#include "stm32f4xx_hal.h"

#define TIM_CHANNEL TIM_CHANNEL_1

extern uint32_t valor1;
extern uint32_t valor2;
extern uint32_t diferencia;
extern uint8_t primerEvento;
extern uint8_t distancia;
```

```
void controlPresencia(TIM_HandleTypeDef *htim);


#endif /* INC_CONTROLPRESENCIA_H_ */
```

La función controlPresenciacalcula la distancia basándose en el tiempo transcurrido entre el envío y la recepción de la señal ultrasónica:

do

Copiar código

```
void controlPresencia(TIM_HandleTypeDef *htim) {
   if (primerEvento == 0) {
      valor1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL);
      primerEvento = 1;
      __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL,
TIM_INPUTCHANNELPOLARITY_FALLING);
   } else if (primerEvento == 1) {
      valor2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL);
      __HAL_TIM_SET_COUNTER(htim, 0);
      if (valor2 > valor1) {
         diferencia = valor2 - valor1;
      } else {
         diferencia = (0xFFFF - valor1) + valor2;
      }
      distancia = diferencia * 0.034 / 2; // Conversión a centímetros
      primerEvento = 0;
      __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL,
TIM_INPUTCHANNELPOLARITY_RISING);
   }
}
```

3.5.- Main

El archivo principal integra y gestiona las diferentes funciones del proyecto mediante una máquina de estados. La configuración inicial incluye:

```
// Configuración inicial
tiempoInicial = HAL_GetTick();
estadoActual = REPOSO;
modoManual = false;
modoAutomatico = false;

HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_4, 10);
```

El bucle principal sigue una estructura basada en estados:

```
typedef enum {
   REPOSO,
   CONTROL_PERSIANAS,
   CONTROL_PRESENCIA
} Estado;

while (1) {
   switch (estadoActual) {
      case REPOSO:
         estadoActual = CONTROL_PERSIANAS;
         break;
      case CONTROL_PERSIANAS:
         if (modoNoche(&tiempoInicial) == 1) {
            estadoActual = CONTROL_PRESENCIA;
```

```
            if (persianasArriba) bajarPersianas();
        } else if (modoManual && antirrebotes(&botonPersianas, GPIOA,
GPIO_PIN_0)) {
            persianasAbajo ? subirPersianas() : bajarPersianas();
        } else if (modoAutomatico) {
            if (persianasAbajo && luzEntrante() == 1) subirPersianas();
            else if (persianasArriba && luzEntrante() == 0) bajarPersianas();
        }
        break;
    case CONTROL_PRESENCIA:
        if (modoDia(&tiempoInicial) == 1) {
            estadoActual = CONTROL_PERSIANAS;
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        } else if (distancia < 10) {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        }
        break;
  }
}
```

Este sistema automatiza completamente el control de persianas y la detección de intrusos, permitiendo al usuario interactuar solo para elegir el modo deseado.

4.- ENLACE AL REPOSITORIO GIT.

A continuación se adjunta el link del repositorio git en donde se encuentra el código completo.

https://github.com/Franxcus/M-quina_Expendedora.git