TRABAJO
SISTEMAS ELECTRÓNICOS DIGITALES

DISEÑO DE UN LAVADERO DE COCHES

CURSO 24/25

---------- Estudiantes (Matrícula)

Manuel Vallina Vides (56636)
Hugo Vigil Velasco (56646)
Daniel Vázquez Chuvieco (56638)

ÍNDICE

1. INTRODUCCIÓN Y OBJETIVOS

En este trabajo se presenta la idea de hacer un *lavadero de coches* basado en el **microprocesador** (STM32F411).

El objetivo principal es diseñar un sistema que sea capaz de analizar un vehículo en una zona concreta mediante sensores de ultrasonidos y gestionar el acceso de otros vehículos en función de los valores medidos.

- *Funcionamiento*

El funcionamiento es el siguiente, en primer lugar, al iniciar el programa, debemos pulsar un botón para que el programa se inicie. Una vez iniciado nos pide que seleccionemos la temperatura a la que deseamos que se realice el lavado del coche, tras haber seleccionado la temperatura, el usuario debe elegir el modo de lavado que desea, distinguiendo entre modo 1 y modo 2, ambos comparten el código base, es decir, se pulsa el botón que permite iniciar el modo 1 o 2, y si el sensor de ultrasonido1 (el que está dentro de la máquina), no detecta presencia, levanta la barrera (servomotor1) y permite el paso del coche. A continuación, el código espera a que el coche llegue al ultrasonido1 para comenzar el lavado. Una vez detecta presencia empieza el lavado, cuando comienza el lavado un led de color blanco parpadea. Cuando el lavado termina, el led blanco deja de emitir luz y en el display aparece un mensaje que indica que el proceso de lavado ha acabado. Por último, si el ultrasonido2, situado al final de la maqueta detecta al vehículo, la barrera de salida (servomotor2) sube permitiendo que el vehículo que se encuentra en movimiento salga, cuando todo esto ha acabado, el código regresa al punto donde te pide la temperatura, esperando así a un nuevo usuario.

Desde cualquier botón del código se puede presionar el botón de emergencia, pasando a un estado de emergencia donde parpadean los leds blanco y rojo. De este estado solo se puede salir pulsando el botón de rearme, al pulsarlo, el sistema vuelve al estado del principio, donde nos pide inicializar el programa (pulsando el botón que pulsamos al inicializar el programa), una vez lo pulsamos el estado pasa al siguiente estado desde el que hemos pulsado la emergencia, pues esta ya se habrá arreglado.

En todas las partes del código, el i2c nos muestra mensajes de lo que está sucediendo o que requisitos necesita para continuar.

2. MATERIAL EMPLEADO

En este punto se describirán en detalle todos los componentes electrónicos que se emplearon, detallando también la función de cada uno en el proyecto.

Pulsador PUL665PLN

Este pulsador tiene, una de sus dos patas, conectada a tierra y la otra a la placa.

De este componente se emplean 3 ejemplares:

- *Botón de Inicio:* Conexión PC15

  Este da la orden, siempre que el **sensor de presencia_1** no detecte ningún otro vehículo dentro del lavadero, de abrir la barrera que permite la entrada del vehículo al mismo.

- *Botón de Emergencia*: Conexión PC13

  Con esté botón se implementa la interrupción del proceso desde cualquier punto del mismo.

- *Botón de Rearme*: Conexión PC14

  Pone de vuelta en funcionamiento al proceso tras haber activado el estado de Emergencia

- *Botón de Inicio*: ConexiónPE8

  Inicia el programa entero.

- *Botón de modo1*: Conexión PB2

  Para elegir el modo de funcionamiento 1.

- *Botón de modo2*: Conexión PE7

  Para elegir el modo de funcionamiento 2.

Sensor de Presencia HC-SR04

Este sensor ultrasónico se emplea para detectar la presencia vehículos, los cuales serán detectados si se sitúan a menos de 10 cm de este.

En el centro de lavado se encuentran dos ejemplares:

- *Sensor de Presencia_1:* conexión PD13, PD10

  Que se sitúa dentro de las instalaciones para detectar si hay un vehículo siendo lavado.

  Si su respuesta es afirmativa, no permitirá la apertura de la barrera inicial, aunque se presione el botón de inicio.

- *Sensor de Presencia_2*: conexión PA1, PA2

  Este se sitúa al final del centro de lavado, con el objetivo de detectar la salida de un vehículo del mismo, ordenando la apertura de la barrera final a su paso.

Barrera Servomotor MG90S

Este componente es un servomotor analógico, que, mediante la anexión de un elemento alargado a su engranaje, funciona como una barrera.

Se ha escogido este ejemplar en concreto por su característica fiabilidad de actuación en espacios reducidos y por su discreto tamaño, que no interfiere en el diseño de la maqueta.

Se usan dos ejemplares, pero tienen exactamente la misma función; elevar la barrera para permitir el paso de un vehículo al recibir la orden pertinente:

- *Barrera de Entrada*: conexión PB3

  Es accionado con el pulsador de inicio

- *Barrera de Salida*: conexión PA7

  Es accionado con la activación del **sensor de presencia_2**

LEDs

La función de estos LED es muy sencilla.

- *LED Verde:* conexión PD12.

Encendido por defecto, indica que el **sensor de presencia_1** no está detectando ningún vehículo dentro del lavadero y que por lo tanto, en el momento en el que se pulse el botón de inicio, la barrera de entrada se elevará y el coche podrá acceder a este.

- *LED Rojo:* conexión PD11

Cuando encendido indica justo lo contrario, que la barrera no se puede elevar, ya que el mismo **sensor de presencia_1** tiene en su rango de detección a un vehículo.

Potenciómetro

Este componente se ha empleado simplemente para permitirle al usuario elegir de una manera cómoda e intuitiva la temperatura a la que quiere que su vehículo sea lavado.

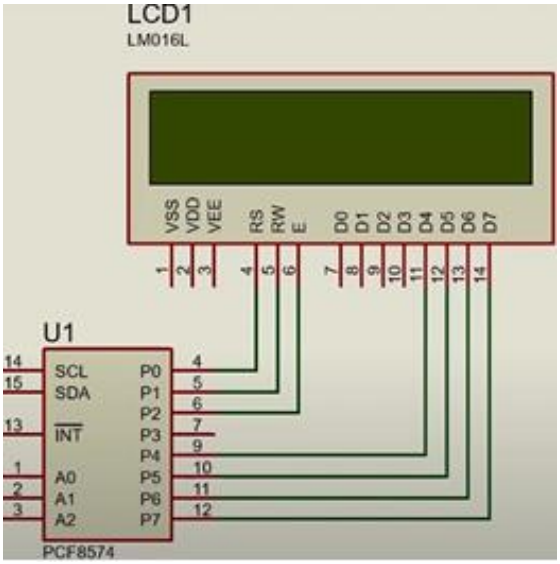
Pantalla LCD con módulo

El display LCD (Liquid Crystal Display) permite mostrar información en su pantalla. El modelo utilizado en este proyecto es de formato 16x2, lo que significa que puede presentar 2 filas con hasta 16 caracteres cada una.

Como se ilustra en la Imagen 2, el modelo empleado incluye un módulo de comunicación serial I2C que simplifica su control. El display operará en modo "4 bits", ya que solo 4 pines de datos del LCD están conectados al chip integrado del módulo I2C. Esta configuración se detalla en la Figura 2.

El display cumple con varias funciones, en primer lugar, estéticamente permite al usuario saber en qué etapa del proceso se encuentra junto con las acciones que están sucediendo en referencia al vehículo mientras avanza por el centro de lavado.

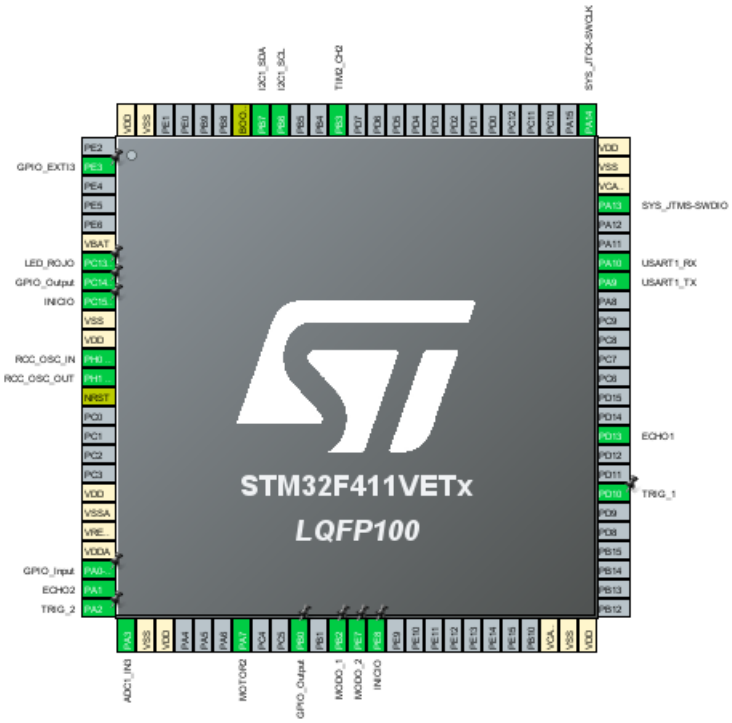
Por otro lado, en cuanto a la programación del proyecto, el display ha sido de gran utilidad para hacer comprobaciones a nivel de código.



3. RESUMEN DE CONEXIONES

A continuación, y por comodidad del interesado, se facilita una tabla en la que se recoge toda la información pertinente a las conexiones entre el microcontrolador y los componentes detallados en el punto 2 (material empleado) de esta memoria.

La información que se presenta, en orden, es la siguiente: pines de unión componente/placa con su modo de funcionamiento, tensión a la que los alimenta el microcontrolador y una sección de comentarios en caso de ser necesaria.



Componente	Pin / Pines
Pulsador *PUL665PLN*	PC13 - 15
Sensor *HC-SR04*	PD1, PD2 PA10, PA13
Servomotor *MG90S*	PB3 PA7
LED	PD11, PD12
Display *I2C*	

4. DESCRIPCIÓN DE LAS FUNCIONES

Función HAL_GPIO_EXTI_Callback

- La función HAL_GPIO_EXTI_Callback gestiona interrupciones externas en pines GPIO. Si el botón de emergencia activa su pin (EMERGENCY_BUTTON_PIN), se establece la bandera emergencyFlag = 1. Si los pines de selección de modos (MODE_BUTTON_PIN o MODE2_BUTTON_PIN) o inicio (INICIO_PIN) generan interrupciones, se asignan los valores correspondientes a las variables modo o inicio, indicando el estado o modo del sistema. Esto permite que el programa principal gestione los eventos según estas señales.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == EMERGENCY_BUTTON_PIN) {
        emergencyFlag = 1; // Activar la bandera de emergencia
    }
    else{
        if (GPIO_Pin == MODE_BUTTON_PIN){
            modo =1;
        }
        else if (GPIO_Pin == MODE2_BUTTON_PIN){
            modo =2;
        }
        else if (GPIO_Pin==INICIO_PIN){
            inicio=1;
        }
        }
    }
```

Funciones para Conversión y Lectura de Temperatura

- La función convertirTemperatura convierte el valor leído del ADC a una temperatura dentro de un rango definido por Tmin y Tmax. Por su parte, readTemperature realiza la lectura del ADC utilizando el potenciómetro y convierte ese valor en una temperatura entre 30°C y 60°C mediante convertirTemperatura. Estas funciones permiten traducir los valores del potenciómetro a temperaturas utilizables en el sistema.

```c
float convertirTemperatura(uint32_t lecturaADC, float Tmin, float Tmax) {
    return Tmin + ((float)lecturaADC / 4095.0f) * (Tmax - Tmin);
}
float readTemperature(void) {
    uint32_t adcValue = 0;

    // Leer el ADC (potenciómetro)
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK) {
        adcValue = HAL_ADC_GetValue(&hadc1);
    }
    HAL_ADC_Stop(&hadc1);

    // Convertir a temperatura (30°C a 60°C)
    return convertirTemperatura(adcValue, 30.0f, 60.0f);
}
```

Función para Medir Distancia con Ultrasonido

- La función readUltrasonicDistance1 mide la distancia utilizando un sensor ultrasónico. Primero, genera un pulso de 10 µs en el pin TRIG para activar el sensor. Luego, mide el tiempo que tarda el pulso ECHO en volver usando un temporizador (htim4). A partir de la duración del pulso, calcula la distancia en centímetros, considerando que la velocidad del sonido es de 34300 cm/s. El resultado es la distancia medida al objeto más cercano.

```c
float readUltrasonicDistance2(void) {
    // Emitir pulso TRIG
        uint32_t startTime = 0, endTime = 0;
        HAL_TIM_Base_Start(&htim5);
        // Genera el pulso TRIG (10µs)
        HAL_GPIO_WritePin(ULTRASONIC_PORT2, TRIG2_PIN, GPIO_PIN_SET);
        delayMicroseconds(10);
        HAL_GPIO_WritePin(ULTRASONIC_PORT2, TRIG2_PIN, GPIO_PIN_RESET);

        // Espera el inicio del pulso ECHO
        while (HAL_GPIO_ReadPin(ULTRASONIC_PORT2, ECHO2_PIN) == GPIO_PIN_RESET);

        startTime = __HAL_TIM_GET_COUNTER(&htim5); // Marca el tiempo inicial

        // Espera el final del pulso ECHO
        while (HAL_GPIO_ReadPin(ULTRASONIC_PORT2, ECHO2_PIN) == GPIO_PIN_SET);

        endTime = __HAL_TIM_GET_COUNTER(&htim5); // Marca el tiempo final

        // Calcula la duración del pulso (en microsegundos)
        uint32_t duration = endTime - startTime;

        // Calcula la distancia en cm (velocidad del sonido: 34300 cm/s)
        return (duration * 0.0343) / 2;
}
```

Función para Controlar el Movimiento del Motor

- La función moveMotor1 controla un motor mediante señales PWM en el canal 2 del temporizador htim2. Primero, establece un valor de 500 para mover el motor (por ejemplo, subir una barrera) y espera 5 segundos con un retardo (HAL_Delay(5000)). Luego, cambia el valor a 2000, ajustando la posición o deteniendo el motor según la configuración del hardware.

```
void moveMotor1(void) {
        __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, 500);
          HAL_Delay(5000);   // Tiempo para subir la barrera
        __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, 2000);
}
```

Control de Indicadores LED

- La función toggleLEDs gestiona el estado de los LEDs según la detección de presencia, encendiendo el LED verde y apagando el LED rojo cuando se detecta presencia, y encendiendo el LED rojo mientras apaga el LED verde en caso contrario. Esto permite proporcionar una indicación visual clara del estado del sistema al usuario.

```
void toggleLEDs(int presenceDetected) {
    if (presenceDetected) {
        HAL_GPIO_WritePin(LED_PORT, LED_RED_PIN, GPIO_PIN_SET); // Apagar LED rojo
        HAL_GPIO_WritePin(LED_PORT, LED_GREEN_PIN, GPIO_PIN_RESET); // Encender LED verde
    } else {
        HAL_GPIO_WritePin(LED_PORT, LED_RED_PIN, GPIO_PIN_RESET);   // Encender LED rojo
        HAL_GPIO_WritePin(LED_PORT, LED_GREEN_PIN, GPIO_PIN_SET); // Apagar LED verde
    }
}
```

Manejo de Emergencias

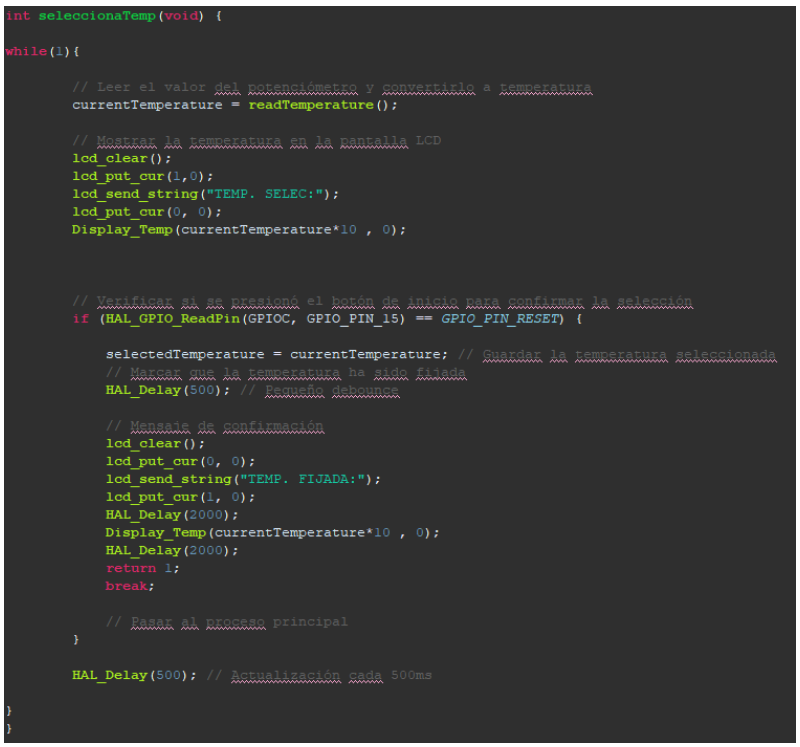
- La función handleEmergency gestiona el estado de emergencia mostrando un mensaje en la pantalla LCD e iniciando el parpadeo de los LEDs rojo y verde para alertar al usuario. Durante este estado, el sistema permanece en un bucle infinito hasta que se presiona el botón de rearme, momento en el cual se restablece la bandera de emergencia, se reinicia el sistema y se vuelve al estado de espera.

```
void handleEmergency(void) {
    // Parpadeo de ambos LEDs (rojo y verde) en el estado de emergencia
     lcd_clear();
     lcd_put_cur(0,0);
     lcd_send_string("PAUSA DE");
     lcd_send_string("EMERGENCIA ");
     while (1) {
            // Parpadeo de los LED
            HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_14); // LED verde
            HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); // LED rojo
            HAL_Delay(500);

            // Verificar si el botón de rearme (PA0) fue presionado
          if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_RESET) {
              inicio=0;
              currentState2 = ESPERANDO;
              emergencyFlag = 0; // Restablecer la bandera de emergencia
             break; // Salir de la función de emergencia
          }
       }
}
```

Selección de Temperatura del Lavado

- La función seleccionaTemp permite al usuario elegir la temperatura del lavado mediante un potenciómetro, mostrando el valor en una pantalla LCD en tiempo real. La selección se confirma al presionar un botón, guardando el valor elegido como selectedTemperature.
  Una vez fijada, muestra un mensaje de confirmación y finaliza retornando un valor de éxito.



```
int seleccionaTemp(void) {

while(1){

        // Leer el valor del potenciómetro y convertirlo a temperatura
        currentTemperature = readTemperature();

        // Mostrar la temperatura en la pantalla LCD
        lcd_clear();
        lcd_put_cur(1,0);
        lcd_send_string("TEMP. SELEC:");
        lcd_put_cur(0, 0);
        Display_Temp(currentTemperature*10 , 0);



        // Verificar si se presionó el botón de inicio para confirmar la selección
        if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_15) == GPIO_PIN_RESET) {

            selectedTemperature = currentTemperature; // Guardar la temperatura seleccionada
            // Marcar que la temperatura ha sido fijada
            HAL_Delay(500); // Pequeña demora

            // Mensaje de confirmación
            lcd_clear();
            lcd_put_cur(0, 0);
            lcd_send_string("TEMP. FIJADA:");
            lcd_put_cur(1, 0);
            HAL_Delay(2000);
            Display_Temp(currentTemperature*10 , 0);
            HAL_Delay(2000);
            return 1;
            break;

            // Pasar al proceso principal
        }

        HAL_Delay(500); // Actualización cada 500ms

}
}
```

Gestión de Flujo de Lavado por Estados

- La función process controla el ciclo completo del lavadero automático de coches utilizando una máquina de estados.
  Administra la interacción con sensores ultrasónicos para verificar la posición del vehículo, activa motores para subir y bajar barreras, y simula el proceso de lavado. Además, verifica una bandera de emergencia para interrumpir las operaciones si es necesario, asegurando un funcionamiento seguro y eficiente.

Descripción de la función “Main”

La función main de este sistema embebido inicializa los periféricos del microcontrolador y gestiona el funcionamiento de un sistema de lavado de coches a través de una interfaz LCD. Se basa en una máquina de estados con dos estados principales: ESPERANDO (espera) y EJECUTANDO (ejecución).

- **Fase de inicialización:**
  Se inicializan los periféricos, como GPIO, temporizadores, I2C, UART y ADC. La pantalla LCD muestra un mensaje de inicio.

- **Bucle principal:**
    - Si se activa una bandera de emergencia, el sistema gestiona la condición de emergencia.
    - En el estado ESPERANDO, el sistema espera la interacción del usuario para iniciar el proceso.
    - Una vez activado, cambia al estado EJECUTANDO, donde solicita al usuario seleccionar la temperatura y el modo de operación.
    - Según el modo seleccionado, el sistema ejecuta el proceso de lavado, verificando continuamente los parámetros establecidos.

Si se detecta un modo no válido o algún error, el sistema vuelve al estado de espera para garantizar la seguridad.

5. ANEXO

a. Presupuestos

Componente	Coste / Unidad	Unidades	Coste Total
Pulsador *PUL665PLN* *(pack de 10uds)*	1,10€	1	1,10€
Sensor *HC-SR04*	1,49€	2	2,98€
Servomotor *MG90S*	2,49€	2	4,98€
LED	0,07€	2	0,14€
Display *I2C*	8,95€	1	8,95€
Cables (pack de 100)	11,45€	1	11,45€

b. Problemas y Soluciones

- **PROBLEMA 1** → Detección de objetos mediante sensores ultrasónicos

    - DESCRIPCIÓN DEL PROBLEMA → Los sensores ultrasónicos para detectar objetos en el proceso de lavado no funcionaban de manera confiable debido a problemas de temporización en la generación y medición de los pulsos ECHO y TRIG. Esto provocaba lecturas inconsistentes de las distancias.

- SOLUCIÓN → Se mejoraron las rutinas de manejo de los sensores, utilizando temporizadores dedicados (TIM4 y TIM5) para controlar con precisión los pulsos ECHO y TRIG. Además, se implementaron funciones específicas para calcular la distancia, asegurando lecturas más fiables y consistentes. Esto mejoró significativamente la estabilidad del proceso de detección.

- **PROBLEMA 2** → Configuración de la temperatura de lavado mediante **potenciómetro**

  - DESCRIPCIÓN DEL PROBLEMA → Inicialmente, el sistema carecía de un mecanismo técnico adecuado para la selección de la temperatura de lavado a través de un componente que generase una señal analógica.

  - SOLUCIÓN → Se implementó un sistema basado en un potenciómetro analógico conectado a un conversor ADC (Convertidor Analógico-Digital) del microcontrolador. El valor analógico leído por el ADC es escalado a un rango definido de temperaturas y mostrado en una pantalla LCD mediante comunicación I2C. Además, se incorporó un botón de validación que, al ser presionado, guarda el valor seleccionado en una variable dedicada, asegurando la configuración estable del sistema. Este enfoque garantiza una calibración precisa de la temperatura en función del rango del potenciómetro y mejora la funcionalidad del control interno del sistema.

- **PROBLEMA 3** →**:** Correcta coordinación de las distintas funciones del programa para el funcionamiento adecuado

  - DESCRIPCIÓN DEL PROBLEMA → El sistema enfrentaba problemas de coordinación entre diferentes módulos funcionales, lo que resultaba en lecturas inexactas del potenciómetro y la imposibilidad de sincronizar correctamente los modos de operación y el arranque del sistema.

  - SOLUCIÓN → Se implementó un esquema basado en un bucle principal con una estructura switch-case para gestionar el sistema mediante estados finitos. Esto permitió coordinar de forma clara las funciones del programa: en el estado ESPERANDO, el sistema detecta el inicio; en el estado EJECUTANDO, se gestionan la lectura del potenciómetro y la selección de modos de operación. Este diseño simplificó las transiciones, corrigió las lecturas inexactas y garantizó la sincronización adecuada entre los módulos del sistema.

- **VARIOS PEQUEÑOS PROBLEMAS** → A lo largo del transcurso del trabajo surgieron pequeños problemas que conseguimos solucionar, pero nos hicieron perder tiempo.
    - EJEMPLO 1 → Algunos pines de nuestro micro estaban gastados, por ejemplo, el PD13, entonces varios quebraderos de cabeza vinieron porque no estaban conectados o se soltaban esos cables.
    - EJEMPLO 2 → Algunos componentes usados no funcionaban, por ejemplo algunos cables no conectaban bien, haciendo que la señal pasase cuando querían.

c. Librerías Adicionales

Librería "i2c-lcd.h"

La pantalla LCD se comunica con el microcontrolador a través del protocolo I2C, donde la pantalla actúa como esclavo y la placa como maestro. Para establecer esta comunicación, primero se selecciona la dirección de escritura del esclavo, lo cual corresponde a la configuración de pines [A0, A1, A2] según el datasheet. Posteriormente, se especifica el modo de funcionamiento y se inicializa el display en modo de 4 bits siguiendo las instrucciones indicadas.

El protocolo I2C opera con bloques de bytes, por lo que, para enviar comandos al LCD en modo de 4 bits, se realiza una división de los datos en dos conjuntos de 4 bits. Estos se combinan con las configuraciones de los pines de habilitación y escritura para formar bloques de información. De este modo, para transmitir un byte de información al LCD, se envían cuatro bloques que contienen los datos y las instrucciones de habilitación correspondientes.

Además, se describen funciones específicas para interactuar con el display, como **lcd_put_cur()**, que permite posicionar el cursor en una fila y columna seleccionada; **lcd_clear_row()** y **lcd_clear()**, que limpian una fila específica o todo el contenido de la pantalla; y funciones como **Display_Temp()**, utilizadas para mostrar datos temperatura en el display. Estas funciones realizan una limpieza previa del contenido antes de actualizar los valores, aunque esta acción genera un efecto visual no deseado debido a la frecuencia de refresco.

Por último, se menciona que la frecuencia de actualización podría optimizarse eliminando la limpieza constante antes de cada muestra, lo cual evitaría este problema visual.

6. BIBLIOGRAFÍA Y LINKS

Links

Enlace Vídeo

Enlace GitHub

Bibliografía

Serie de enlaces que nos han sido útiles para el proyecto

https://www.youtube.com/watch?v=WhGS10vynww

https://www.youtube.com/watch?v=1u2D-o4vnkU

https://theembeddedthings.com/stmp32/stm32-with-dht22-interfacing-temperature-and-humidity-sensor/

https://www.youtube.com/watch?v=0kHvCE_vD3o

https://www.youtube.com/watch?v=eg230lqdZYQ