

Práctica 3: Diseño de interfaz gráfica para el control de salidas en Controllino

I. OBJETIVOS

- Controlar el brillo de un foco LED mediante modulación por ancho de pulso (PWM) utilizando una interfaz gráfica en una pantalla HMI (Human-Machine Interface)

II. MATERIALES

- Tablero de control con Controllino Mega y HMI integrado.
- Fuente de alimentación del tablero.
- Cable USB tipo A a B 2.0.
- Cable USB tipo A a A.
- PC con Arduino IDE instalado y configurado para Controllino.
- Software Stone Designer GUI.

III. DESCRIPCIÓN DEL FUNCIONAMIENTO

Se tienen dos widget tipo SpinBox en la interfaz final del HMI. Estos cumplen con lo siguiente:

- **sp1**: controla el duty cycle (porcentaje de ciclo de trabajo PWM) del primer led.
- **sp2**: controla el duty cycle (porcentaje de ciclo de trabajo PWM) del segundo led.

Por otra parte, en el tablero se deben configurar dos botones físicos para cumplir con lo siguiente:

- El primer botón físico (**I_16**) controlará el encendido/apagado del primer LED.
- El segundo botón físico (**I_17**) controlará el encendido/apagado del segundo LED.

Cada SpinBox deberá estar asociado de manera independiente a su respectivo LED, de forma que, al mover el SpinBox, se ajuste la intensidad del brillo del LED correspondiente, sin afectar al otro LED.

La acción de los botones físicos debe ser independiente del valor del SpinBox:

- El botón únicamente habilitará o deshabilitará el encendido del LED, pero el brillo será determinado por el valor actual del SpinBox asociado.
- El sistema debe garantizar que, si un LED está apagado por el botón físico, no se active aunque se modifique el SpinBox correspondiente (hasta que el botón vuelva a presionarse).

IV. DESARROLLO

IV-1. Diseño de la interfaz en el HMI:

1. Instalar y abrir el software **STONE Designer GUI**.
2. Crear un nuevo proyecto desde el menú **Project**, asignando el nombre, carpeta de destino, baudios, dimensiones de la pantalla (800x480 px) y nivel de brillo correspondientes.
3. En la ventana principal *home_page*, hacer clic derecho y seleccionar **Add Widget → Label** para insertar una etiqueta que muestre información en pantalla.
4. Configurar el tamaño y tipo de fuente del widget **label1**, ajustando su tamaño en la ventana de propiedades para una correcta visualización.
5. Agregar un **widget tipo spin_box** desde *home_page* → **Add Widget → spin_box**, este tendrá como identificador **spin_box1**.
6. Configurar sus propiedades: fuente, límites de valores (0–100), tipo de dato **int**, y alineación del texto.
7. Repetir los puntos 3 y 6 para obtener un **label2** y **spin_box2**.
8. Organizar los widgets en pantalla de acuerdo con la distribución mostrada en la Figura 1.
9. Guardar el proyecto antes de proceder con la carga al HMI.

PWM para controlar el brillo de dos Leds

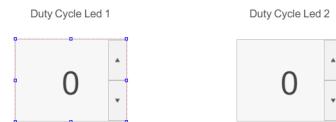


Figura 1. Interfaz Final de HMI

IV-A. Explicación del Código

El programa permite controlar el brillo de dos LEDs mediante valores de *duty cycle* (0–100 %) enviados desde una HMI (widgets `spin_box sp1` y `sp2`), convirtiéndolos a

niveles PWM (0–255). Cada LED se habilita o deshabilita con un botón físico (modo *toggle*) usando banderas.

IV-A1. Asignación de pines y variables:: Necesarias para el desarrollo de código.

```
const int led = CONTROLLINO_D0; // Salida digital D0
const int led2 = CONTROLLINO_D6; // Salida digital
    ↪ D6
const int boton1 = CONTROLLINO_I16; // Entrada
    ↪ Digital I16
const int boton2 = CONTROLLINO_I17; // Entrada
    ↪ Digital I17
int pwmValue = 0; // valor convertido (0-255) led 1
int pwmValue2 = 0; // valor convertido (0-255) led 2
float dutyCyclePercent = 0; // valor en porcentaje
    ↪ (0-100) led 1
float dutyCyclePercent2 = 0; // valor en porcentaje
    ↪ (0-100) led 2
int bandera1 = 0; // bandera para boton 1
int bandera2 = 0; // bandera para boton 2
int valor1 = 0; // estado del boton 1
int valor2 = 0; // estado del boton 2
```

■ Salidas PWM: Leds del tablero

- led = CONTROLLINO_D0
- led2 = CONTROLLINO_D6

■ Entradas: Botones del tablero

- boton1 = CONTROLLINO_I16
- boton2 = CONTROLLINO_I17

■ Estados:

- dutyCyclePercent y dutyCyclePercent2: toman valores de 0–100 % que definen el Duty Cycle.
- pwmValue y pwmValue2: toman valores de 0–100 % que definen los valores de encendido de leds.
- bandera1 y bandera2: toman valores de 0 o 1, que funcionan como banderas.
- valor1 y valor2: toman valores de 0 o 1, que guardan el estado de las banderas.

IV-A2. setup(): Inicializa variables y estados antes del bucle principal.

```
Serial.begin(115200); // Comunicacion serial con
    ↪ el PC
Serial2.begin(115200); // Comunicacion serial con
    ↪ el HMI
pinMode(led, OUTPUT); // led1 como salida
pinMode(led2, OUTPUT); // led2 como salida
pinMode(boton1, INPUT); // boton 1 como entrada
pinMode(boton2, INPUT); // boton 2 como entrada
HMI_init(); // Inicializa el sistema de colas para
    ↪ las respuestas el HMI
Stone_HMI_Set_Value(spin_box, sp1, NULL, 0); //
    ↪ Pone en 0 el valor del spin box en el HMI.
Stone_HMI_Set_Value(spin_box, sp2, NULL, 0); //
    ↪ Pone en 0 el valor del spin box en el HMI.
```

- Configura UART de PC (Serial) y HMI (Serial2) a 115200 bps.
- Define direcciones de pines (pinMode).
- HMI_init() y puesta a cero de sp1 y sp2 para un estado inicial conocido.

IV-B. loop(): lectura y control:

Define la lógica principal para el encendido de los leds.

```
void loop() {
dutyCyclePercent=HMI_get_value(spin_box, sp1); //
    ↪ Obtiene el valor del spin_box1
dutyCyclePercent2=HMI_get_value(spin_box, sp2); //
    ↪ Obtiene el valor del spin_box2

valor1 = digitalRead(boton1); //lectura digital
    ↪ del boton 1
valor2 = digitalRead(boton2); //lectura digital
    ↪ del boton 2

if (valor1 == HIGH){ //cada que aplasta el boton1
    ↪ se suma 1 a la bandera1
bandera1 += 1; //bandera 1 = 1 es para prender el
    ↪ led 1
if (bandera1 == 2){ //si aplasta de nuevo se pone
    ↪ a valor 0 la bandera1
bandera1 = 0;
} //bandera1 = 0 significa que el led 1 se apaga
}

if (valor2 == HIGH){ //cada que aplasta el boton2
    ↪ se suma 1 a la bandera2
bandera2 += 1; //bandera 2 = 1 es para prender el
    ↪ led 2
if (bandera2 == 2){ //si aplasta de nuevo se pone
    ↪ a valor 0 la bandera2
bandera2 = 0;
} //bandera2 = 0 significa que el led 2 se apaga
}

// ***** PARA LED 1 *****
if (dutyCyclePercent >= 0 && dutyCyclePercent
    ↪ <=100){
pwmValue = map(dutyCyclePercent, 0, 100, 0, 255);
    ↪ // Mapea el valor de duty cycle 1 en
    ↪ porcentaje a valores de 0 a 255
Serial.print(Duty cycle (%) led 1: );
Serial.print( -> PWM value: );
Serial.println(pwmValue);

if (bandera1 == 1) { // Solo aqui se prende
    analogWrite(led, pwmValue);
} else {
    analogWrite(led, 0); // Si no cumple se apaga
}
} else {
    Serial.println(Ingresa un valor entre 0 y 100.);
}

// ***** PARA LED 2 *****
if (dutyCyclePercent2 >= 0 && dutyCyclePercent2
    ↪ <=100){
pwmValue2 = map(dutyCyclePercent2, 0, 100, 0,
    ↪ 255); // Mapea el valor de duty cycle 1 en
    ↪ porcentaje a valores de 0 a 255
Serial.print(Duty cycle (%) led 2: );
Serial.print( -> PWM value: );
Serial.println(pwmValue2);

if (bandera2 == 1) { // Solo aqui se prende
    analogWrite(led2, pwmValue2);
} else {
    analogWrite(led2, 0); // Si no cumple se apaga
}
} else {
    Serial.println(Ingresa un valor entre 0 y 100.);
}
```

1. **Lee HMI:** sp1 y sp2 entregan el *duty* (0–100 %).
2. **Lee botones:** digitalRead en I16 e I17.
3. **Toggle de banderas:** cada lectura en HIGH alterna 0/1 para habilitar el LED correspondiente.
4. **PWM:** si el *duty* está en rango, se mapea (0–100) → (0–255) y se aplica con analogWrite *sólo* si la bandera = 1; en caso contrario, se apaga (valor 0).

IV-C. Resultados

En la figura 2 se puede observar que el led 1 se encuentra encendido al nivel **PWM** correspondiente al **Duty Cycle** colocado en la interfaz del tablero. Se debe mencionar que el led 1 se encuentra encendido solamente porque el usuario presionó el primer botón.

Por esta razón, el segundo led se encuentra apagado, ya que el usuario no presionó el botón 2 correspondiente al led 2.



Figura 2. Led 1 encendido

Por otra parte, en la figura 3 el segundo led se encuentra encendido ya que se sigue la misma lógica presentada anteriormente,



Figura 3. Led 2 encendido

Finalmente, se prueba un nivel diferente de **Duty Cycle** para el segundo led, verificando así su funcionamiento. Referirse a la figura 4.



Figura 4. Segundo Led con Duty Cycle menor

V. CONCLUSIONES

- Se logró establecer la comunicación entre el Controllino Mega y la HMI Stone, permitiendo controlar el brillo de los LEDs mediante la modulación PWM.
- El uso de widgets *spin_box* facilitó la interacción entre el usuario y el sistema, demostrando la eficacia del control gráfico en entornos de automatización.
- La integración de botones físicos permitió implementar un sistema de control combinado (manual y digital), mejorando la comprensión de los principios de entrada/salida en PLC.
- Se evidenció la importancia del procesamiento correcto de datos del HMI y del manejo de estados mediante banderas para evitar errores lógicos.
- La práctica permitió reforzar conocimientos sobre comunicación serial, control de PWM y diseño de interfaces HMI aplicadas a sistemas embebidos.

VI. ANEXOS

VI-A. Link para Repositorio de la Práctica

<https://github.com/MateoDutan/ControlDigital>