UNIVERSIDAD DEL VALLE DE GUATEMALA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA, MECATRÓNICA

Nombre: María Alejandra Rodríguez Roldán
Carné: 21620
Curso: Electrónica Digital 2
Sección: 10
Catedrático: Pablo Mazariegos y José Morales
Fecha: 6/08/2023

Laboratorio 4 – Señales de PWM

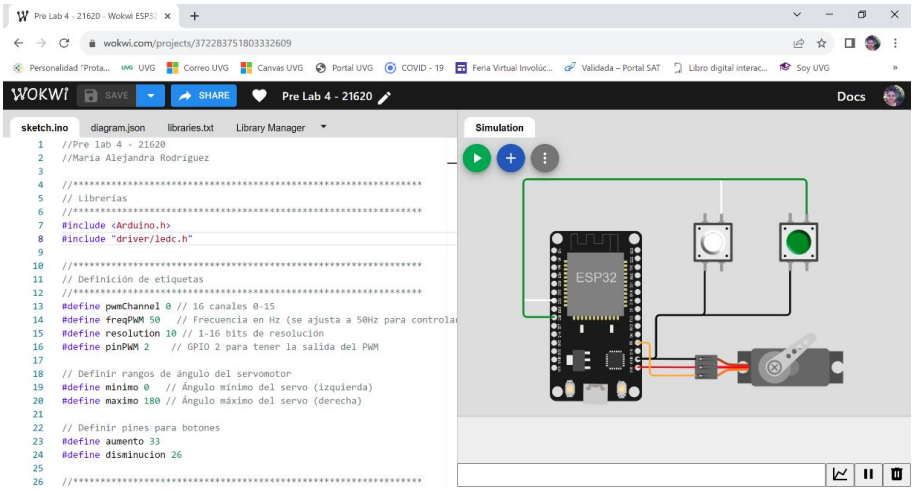
Utilizando el microcontrolador ESP32

Pre-lab (20%) – Físico/Simulado

Se sube antes del inicio del laboratorio en canvas en formato *.zip con el nombre prelab.

Diseñe e implemente una señal PWM para controlar un servomotor utilizando dos botones. El primer botón (B1) será para aumentar el movimiento para el lado derecho y el otro botón (B2) para mover el movimiento de regreso a la izquierda. Debe ser capaz de mover la posición del servo entre su rango máximo sin dañar el Servo, REVISAR que el servo motor no esté haciendo ruidos raros.

- ✓ Link de simulación en Wokwi: https://wokwi.com/projects/372283751803332609



- ✓ **Código utilizado:**

```
//Pre lab 4 - 21620
//María Alejandra Rodríguez

//*****************************************************************
// Librerías
//*****************************************************************
#include <Arduino.h>
#include "driver/ledc.h"

//*****************************************************************
// Definición de etiquetas
//*****************************************************************
```

```
#define pwmChannel 0 // 16 canales 0-15
#define freqPWM 50   // Frecuencia en Hz (se ajusta a 50Hz para controlar
el servomotor)
#define resolution 10 // 1-16 bits de resolución
#define pinPWM 2    // GPIO 2 para tener la salida del PWM

// Definir rangos de ángulo del servomotor
#define minimo 0   // Ángulo mínimo del servo (izquierda)
#define maximo 180 // Ángulo máximo del servo (derecha)

// Definir pines para botones
#define aumento 33
#define disminucion 26

//*****************************************************************
// Variables Globales
//*****************************************************************
int inicial = 90; // Ángulo inicial del servo

//*****************************************************************
// Configuración
//*****************************************************************
void setup() {
  pinMode(aumento, INPUT_PULLUP); // Botón B1 habilitado con resistencia
pull-up interna
  pinMode(disminucion, INPUT_PULLUP); // Botón B2 habilitado con
resistencia pull-up interna

  // Paso 1: Configurar el módulo PWM
  ledcSetup(pwmChannel, freqPWM, resolution);

  // Paso 2: seleccionar en qué GPIO tendremos nuestra señal PWM
  ledcAttachPin(pinPWM, pwmChannel);

  //Paso 3: Establecer la posición del servo motor inicialmente
  ledcWrite(0,map(180,0,180,0, 1023));
}

//*****************************************************************
// Loop Principal
//*****************************************************************
void loop() {
  if (digitalRead(aumento) == LOW){
    inicial += 5; // Aumentar el ángulo en 5 grados
```

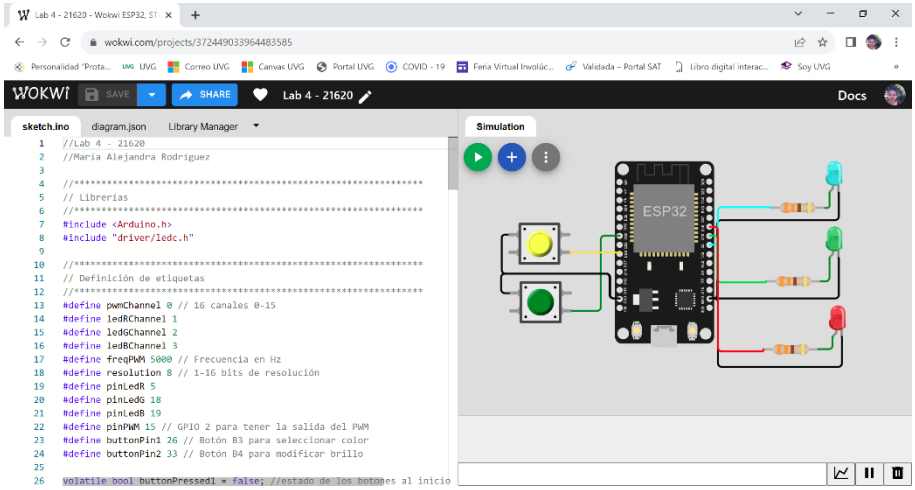
```
    if (inicial > maximo) {
      inicial = maximo;
    }
  }

  if (digitalRead(disminucion) == LOW) {
    inicial -= 5; // Disminuir el ángulo en 5 grados
    if (inicial < minimo) {
      inicial = minimo;
    }
  }

  ledcWrite(pwmChannel, map(inicial, 0, 180, 30, 115)); // Mapear el
ángulo a la escala del servomotor (ajustar según sea necesario)
  delay(100); // Añadir un pequeño retraso para evitar lecturas de
botones rápidas
}
```

Lab (30%) - Físico

Deberá mostrarlo al catedrático o auxiliar durante el tiempo del laboratorio.

Diseñe e implemente tres señales PWM para controlar un LED RGB o en su defecto 3 leds de color Rojo, Verde y Azul utilizando dos botones. El primer botón (B3) nos servirá para seleccionar cuál de los tres colores queremos modificar. Con el otro botón (B4) puedo modificar el brillo del LED seleccionado, al presionar debe aumentar el brillo, cuando llegue al brillo máximo, si se vuelve a presionar empieza desde apagado y se va incrementando de nuevo con cada vez que se presione el botón (B4).

- ✓ Link de simulación en Wokwi: https://wokwi.com/projects/372449033964483585



- ✓ **Código utilizado:**

```
//Lab 4 - 21620
//María Alejandra Rodríguez

//*************************************************************
// Librerías
//*************************************************************
#include <Arduino.h>
#include "driver/ledc.h"

//*************************************************************
// Definición de etiquetas
//*************************************************************
#define pwmChannel 0 // 16 canales 0-15
#define ledRChannel 1
#define ledGChannel 2
#define ledBChannel 3
#define freqPWM 5000 // Frecuencia en Hz
#define resolution 8 // 1-16 bits de resolución
#define pinLedR 5
#define pinLedG 18
#define pinLedB 19
#define pinPWM 15 // GPIO 2 para tener la salida del PWM
#define buttonPin1 26 // Botón B3 para seleccionar color
#define buttonPin2 33 // Botón B4 para modificar brillo

volatile bool buttonPressed1 = false; //estado de los botones al inicio
volatile bool buttonPressed2 = false;

unsigned long lastDebounceTime1 = 0; //variables para los debounce
(antirrebotes)
unsigned long lastDebounceTime2 = 0;

const unsigned long debounceDelay = 50;

//*************************************************************
// Prototipos de funciones
//*************************************************************
void configurarPWM(void);

//*************************************************************
// Variables Globales
//*************************************************************
```

```
int colorSelected = 0; // Variable para guardar el color seleccionado (0: Rojo, 1: Verde, 2: Azul)
int brightness = 0;    // Variable para guardar el brillo actual (0 a 255)

//************************************************************
// Configuración
//************************************************************
void setup() {
  pinMode(buttonPin1, INPUT_PULLUP); // Configurar botón B3 como entrada con resistencia pull-up interna
  pinMode(buttonPin2, INPUT_PULLUP); // Configurar botón B4 como entrada con resistencia pull-up interna
  configurarPWM();

  Serial.begin(115200);
}

//************************************************************
// Loop Principal
//************************************************************
void loop() {
  // Leer el estado de los botones
  bool button1State = digitalRead(buttonPin1);
  bool button2State = digitalRead(buttonPin2);

  if (millis() - lastDebounceTime1 >= debounceDelay) {
    lastDebounceTime1 = millis();
    buttonPressed1 = true;
    // Si el botón B3 (buttonPin1) está presionado, cambiar el color seleccionado
    if (button1State == LOW) {
      colorSelected = (colorSelected + 1) % 3; // Cambiar entre Rojo, Verde y Azul
      delay(200); // Debounce para evitar múltiples cambios rápidos al presionar el botón
      }
  }

  if (millis() - lastDebounceTime2 >= debounceDelay) {
    lastDebounceTime2 = millis();
    buttonPressed2 = true;
    // Si el botón B4 (buttonPin2) está presionado, modificar el brillo del color seleccionado
    if (button2State == LOW) {
```

```
      brightness += 16;// Incrementar el brillo de 0 a 255
      if (brightness >= 255) {
        brightness = 0;
        }
        delay(200); // Debounce para evitar múltiples incrementos rápidos al
presionar el boton
      }
  }

  // Establecer los valores de brillo para los LEDs según el color
seleccionado
  switch (colorSelected) {
    case 0: // Rojo
      ledcWrite(ledRChannel, brightness);
      ledcWrite(ledGChannel, 0);
      ledcWrite(ledBChannel, 0);
      break;
    case 1: // Verde
      ledcWrite(ledRChannel, 0);
      ledcWrite(ledGChannel, brightness);
      ledcWrite(ledBChannel, 0);
      break;
    case 2: // Azul
      ledcWrite(ledRChannel, 0);
      ledcWrite(ledGChannel, 0);
      ledcWrite(ledBChannel, brightness);
      break;
  }

  // Establecer el mismo valor de brillo para todos los LEDs
  ledcWrite(pwmChannel, brightness);
}

//*****************************************************************
// Función para configurar módulo PWM
//*****************************************************************
void configurarPWM(void) {
  // Paso 1: Configurar el módulo PWM
  ledcSetup(pwmChannel, freqPWM, resolution);
  ledcSetup(ledRChannel, freqPWM, resolution);
  ledcSetup(ledGChannel, freqPWM, resolution);
  ledcSetup(ledBChannel, freqPWM, resolution);

  // Paso 2: seleccionar en qué GPIO tendremos nuestra señal PWM
```

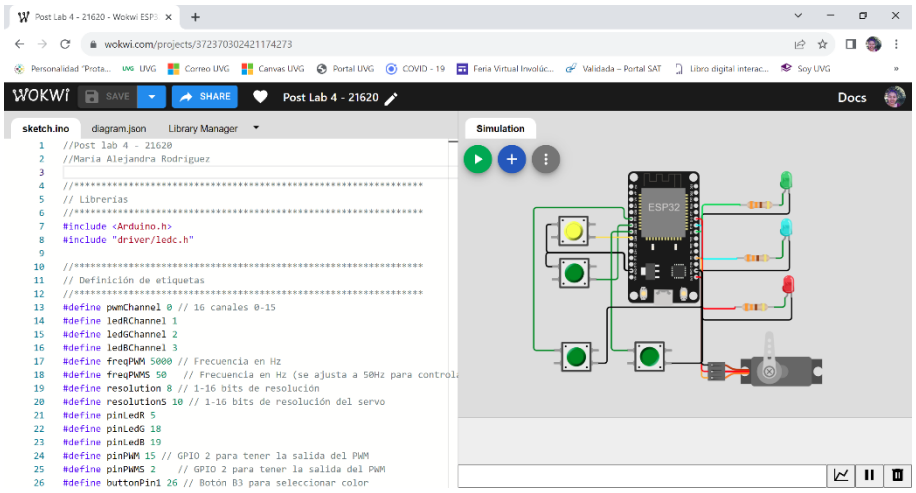
```
  ledcAttachPin(pinPWM, pwmChannel);
  ledcAttachPin(pinLedR, ledRChannel);
  ledcAttachPin(pinLedG, ledGChannel);
  ledcAttachPin(pinLedB, ledBChannel);
}
```

Post-lab (40%) - Físico

Se entrega después del tiempo de laboratorio según el portal. Deberá subir los entregables en formato *.zip con el nombre entregables.

Implemente una rutina la cual se seleccione con el mismo botón (B3), en donde dependiendo de la posición del servo cambie el color del LED RGB, como se muestra en la siguiente figura.

- ✓ Link de simulación en Wokwi: https://wokwi.com/projects/372370302421174273



- ✓ **Código utilizado:**

```
//Post lab 4 - 21620
//María Alejandra Rodríguez

//*************************************************************
// Librerías
//*************************************************************
#include <Arduino.h>
#include "driver/ledc.h"

//*************************************************************
// Definición de etiquetas
```

```
//*************************************************************
#define pwmChannel 0 // 16 canales 0-15
#define ledRChannel 1
#define ledGChannel 2
#define ledBChannel 3
#define freqPWM 5000 // Frecuencia en Hz
#define freqPWMS 50   // Frecuencia en Hz (se ajusta a 50Hz para
controlar el servomotor)
#define resolution 8 // 1-16 bits de resolución
#define resolutionS 10 // 1-16 bits de resolución del servo
#define pinLedR 5
#define pinLedG 18
#define pinLedB 19
#define pinPWM 15 // GPIO 2 para tener la salida del PWM
#define pinPWMS 2    // GPIO 2 para tener la salida del PWM
#define buttonPin1 26 // Botón B3 para seleccionar color
#define buttonPin2 33 // Botón B4 para modificar brillo

// Definir rangos de ángulo del servomotor
#define minimo 0   // Ángulo mínimo del servo (izquierda)
#define maximo 180 // Ángulo máximo del servo (derecha)

// Definir pines para botones
#define aumento 25
#define disminucion 32

//Posición del servo al presionar el botón
#define servoPositionRed 135  // Posición del servo para el color rojo
#define servoPositionGreen 90  // Posición del servo para el color verde
#define servoPositionBlue 45  // Posición del servo para el color azul

volatile bool buttonPressed1 = false; //estado de los botones al inicio
volatile bool buttonPressed2 = false;

unsigned long lastDebounceTime1 = 0; //variables para los debounce
(antirrebotes)
unsigned long lastDebounceTime2 = 0;

const unsigned long debounceDelay = 50;

//*************************************************************
// Prototipos de funciones
//*************************************************************
void configurarPWM(void);
```

```
//************************************************************
// Variables Globales
//************************************************************
int colorSelected = 0; // Variable para guardar el color seleccionado (0: Rojo, 1: Verde, 2: Azul)
int brightness = 0;    // Variable para guardar el brillo actual (0 a 255)
int inicial = 90; // Ángulo inicial del servo
//************************************************************
// Configuración
//************************************************************
void setup() {
  pinMode(buttonPin1, INPUT_PULLUP); // Configurar botón B3 como entrada con resistencia pull-up interna
  pinMode(buttonPin2, INPUT_PULLUP); // Configurar botón B4 como entrada con resistencia pull-up interna
  pinMode(aumento, INPUT_PULLUP); // Botón B1 habilitado con resistencia pull-up interna
  pinMode(disminucion, INPUT_PULLUP); // Botón B2 habilitado con resistencia pull-up interna
  configurarPWM();

  // Paso 1: Configurar el módulo PWM
  ledcSetup(pwmChannel, freqPWMS, resolutionS);

  // Paso 2: seleccionar en qué GPIO tendremos nuestra señal PWM
  ledcAttachPin(pinPWMS, pwmChannel);

  //Paso 3: Establecer la posición del servo motor inicialmente
  ledcWrite(0, map(180,0,180,0, 1023));

  Serial.begin(115200);
}

//************************************************************
// Loop Principal
//************************************************************
void loop() {
  // Leer el estado de los botones
  bool button1State = digitalRead(buttonPin1);
  bool button2State = digitalRead(buttonPin2);

  if (millis() - lastDebounceTime1 >= debounceDelay) {
```

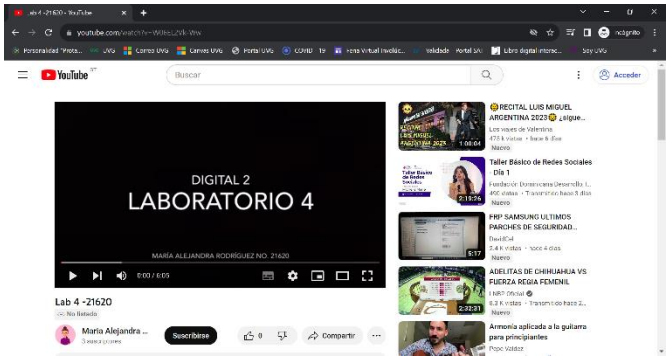
```
    lastDebounceTime1 = millis();
    buttonPressed1 = true;
    // Si el botón B3 (buttonPin1) está presionado, cambiar el color
seleccionado
    if (button1State == LOW) {
      colorSelected = (colorSelected + 1) % 4; // Cambiar entre Rojo,
Verde y Azul
      delay(200); // Debounce para evitar múltiples cambios rápidos al
presionar el botón
      }
  }

  if (millis() - lastDebounceTime2 >= debounceDelay) {
    lastDebounceTime2 = millis();
    buttonPressed2 = true;
    // Si el botón B4 (buttonPin2) está presionado, modificar el brillo
del color seleccionado
    if (button2State == LOW) {
      brightness += 16;// Incrementar el brillo de 0 a 255
      if (brightness >= 255) {
        brightness = 0;
        }
        delay(200); // Debounce para evitar múltiples incrementos rápidos
al presionar el boton
      }
  }

  // Establecer los valores de brillo para los LEDs según el color
seleccionado
  switch (colorSelected) {
    case 0: // Azul
      ledcWrite(ledRChannel, 0);
      ledcWrite(ledGChannel, 0);
      ledcWrite(ledBChannel, brightness);
      ledcWrite(pwmChannel, map(servoPositionBlue, 0, 180, 30, 115));
      break;
    case 1: // Verde
      ledcWrite(ledRChannel, 0);
      ledcWrite(ledGChannel, brightness);
      ledcWrite(ledBChannel, 0);
      ledcWrite(pwmChannel, map(servoPositionGreen, 0, 180, 30, 115));
      break;
    case 2: // Rojo
      ledcWrite(ledRChannel, brightness);
```

```
      ledcWrite(ledGChannel, 0);
      ledcWrite(ledBChannel, 0);
      ledcWrite(pwmChannel, map(servoPositionRed, 0, 180, 30, 115));
      break;
    case 3:
      ledcWrite(ledRChannel, 0);
      ledcWrite(ledGChannel, 0);
      ledcWrite(ledBChannel, 0);
      ledcWrite(pwmChannel, map(inicial, 0, 180, 30, 115)); // Mapear el
ángulo a la escala del servomotor (ajustar según sea necesario)
  }

  // Establecer el mismo valor de brillo para todos los LEDs
  //ledcWrite(pwmChannel, brightness);

  if (digitalRead(aumento) == LOW){
    inicial += 5; // Aumentar el ángulo en 5 grados
    if (inicial > maximo) {
      inicial = maximo;
    }
  }

  if (digitalRead(disminucion) == LOW) {
    inicial -= 5; // Disminuir el ángulo en 5 grados
    if (inicial < minimo) {
      inicial = minimo;
    }
  }


  delay(100); // Añadir un pequeño retraso para evitar lecturas de
botones rápidas
}

//*************************************************************
// Función para configurar módulo PWM
//*************************************************************
void configurarPWM(void) {
  // Paso 1: Configurar el módulo PWM
  ledcSetup(pwmChannel, freqPWM, resolution);
  ledcSetup(ledRChannel, freqPWM, resolution);
  ledcSetup(ledGChannel, freqPWM, resolution);
  ledcSetup(ledBChannel, freqPWM, resolution);
```

```
  // Paso 2: seleccionar en qué GPIO tendremos nuestra señal PWM
  ledcAttachPin(pinPWM, pwmChannel);
  ledcAttachPin(pinLedR, ledRChannel);
  ledcAttachPin(pinLedG, ledGChannel);
  ledcAttachPin(pinLedB, ledBChannel);
}
```

Entregables (10%)
Deberá entregar en Canvas un archivo comprimido que contenga:

- ✓ El folder completo de su proyecto
- ✓ Link a Video mostrando el funcionamiento y explicando su código (lo suben a YouTube/Odysee/Vimeo y lo ponen Unlisted o Público).
  https://youtu.be/W06EL2Vk-Ww



- ✓ Su código debe estar dividido en secciones y bien comentado
- ✓ Link a repositorio de GitHub:
  https://github.com/aler21620/Laboratorio-4---Digital-2---21620