ESCUELA: Universidad Politécnica de Chiapas.

CARRERA: Ingeniería en Software.

TEMA: Práctica 4 - ESP32 – LED DIMMER.

ASIGNATURA: Electricidad y Magnetismo.

GRADO Y GRUPO: “4° B”

INTEGRANTES:

Sujey Calderón Martínez. 233291

Hannia Paola De Los Santos Bautista. 233273

Victor Fabricio Pérez Constantino. 233394

Ameth De Jesús Méndez Toledo. 233363

Joaquín Esaú Pérez Diaz. 233412

PROFESOR:

Juan Manuel Martínez Constantino

FECHA DE ENTREGA: viernes 8 de noviembre de 2024.

CONTENIDO

ANTECEDENTES

Este proyecto utiliza un microcontrolador ESP32 para controlar la intensidad de un LED mediante Modulación de Ancho de Pulso (PWM). El objetivo es comprender cómo se regula la intensidad de luz en aplicaciones de iluminación inteligente, destacando su potencial para el ahorro energético y la personalización de ambientes. Esta práctica sienta las bases para sistemas de control de iluminación más complejos.

INTRODUCCIÓN

En el contexto de la electrónica y la automatización del hogar, los sistemas de control de iluminación permiten regular la intensidad de luces LED, mejorando tanto el ambiente como la eficiencia energética. El ESP32, un microcontrolador popular en el Internet de las Cosas (IoT), soporta técnicas como PWM que son útiles para aplicaciones de dimming. Este proyecto explora cómo configurar y controlar un LED mediante el ESP32, mostrando el proceso de modulación y destacando la practicidad de esta función en la domótica.

DESARROLLO

MARCO TEÓRICO

- ESP32: El ESP32 es un microcontrolador de doble núcleo ampliamente utilizado por sus capacidades de procesamiento y conectividad inalámbrica (Wi-Fi y Bluetooth). Es ideal para aplicaciones de IoT, ofreciendo un entorno flexible para proyectos de control y automatización.
- PWM (Modulación de Ancho de Pulso): La PWM es una técnica que permite variar la potencia suministrada a una carga al alternar el encendido y apagado de una señal digital. En el contexto de este proyecto, la PWM permite regular la intensidad del LED ajustando el ciclo de trabajo.

- Control de Intensidad de Luz: La capacidad de controlar la intensidad de una fuente de luz es fundamental en sistemas de iluminación ajustable, permitiendo no solo ahorrar energía, sino también crear ambientes personalizados.

PROCEDIMIENTOS

Primero, descargamos y configuramos el IDE de Arduino en la computadora. Luego, conectamos el cable de datos USB al ESP32 y a la computadora para iniciar la programación. Una vez en el IDE de Arduino, agregamos el soporte para la placa ESP32 desde las preferencias, instalando las librerías correspondientes. Posteriormente, seleccionamos el modelo de ESP32 en la configuración del IDE y configuramos el puerto COM correcto para poder cargar el código.

A continuación, conectamos el LED al pin 2 del ESP32. Colocamos una resistencia entre el cátodo del LED y la conexión a tierra (GND) para limitar la corriente y proteger el LED. Ahora estamos listos para cargar el código.

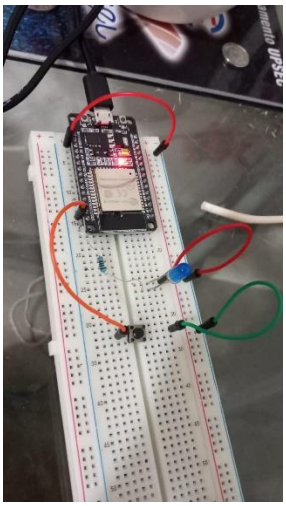
Para el código, escribimos un programa básico que usa PWM para controlar la intensidad del LED. Este código incrementa y disminuye el ciclo de trabajo (duty cycle) del PWM, creando un efecto de aumento y disminución gradual en la intensidad de la luz del LED. Luego, ajustamos el intervalo de tiempo entre cambios en el ciclo de trabajo para modificar la velocidad de cambio en la intensidad del LED.

Finalmente, cargamos el código en el ESP32 desde el IDE de Arduino. Observamos que el LED cambia su intensidad de manera gradual, creando un efecto de "fade in" y "fade out", cumpliendo con el objetivo de la práctica de regular la intensidad de luz del LED usando PWM.

LISTA DE MATERIALES

- Placa ESP32
- LED
- Resistencia de 300 ohmios
- Cables de conexión
- Laptop
- Cable de datos USB
- Pushbutton
- Protoboard

RESULTADOS DE LA PRÁCTICA



La foto muestra el momento en que el LED está encendido, como resultado del código de control de intensidad cargado en el ESP32. Este montaje demuestra que la placa está funcionando correctamente, permitiendo ajustar la intensidad del LED mediante las instrucciones del código y la interacción con el botón. Cada vez que se presiona el botón, el código en la ESP32 modifica el brillo del LED, simulando un efecto de "dimmer".

CÓDIGO

```
int LED = 23;
int buttonPin = 22;
int buttonState = 0;
int lastButtonState = 0;
int LEDState = LOW;
int pressCount = 0;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  digitalWrite(LED, LOW);
}

void loop() {
  int reading = digitalRead(buttonPin);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
```

DECLARACIÓN DE VARIABLES:

- LED y buttonPin son los pines a los que están conectados el LED (pin 23) y el botón (pin 22), respectivamente.
- buttonState y lastButtonState mantienen el estado actual y el estado anterior del botón, inicializados en 0.
- LEDState controla el estado del LED (inicialmente apagado con valor LOW).
- pressCount es un contador para registrar el número de veces que se presiona el botón.
- lastDebounceTime y debounceDelay son utilizados para evitar el efecto de rebote del botón (un pequeño retraso de 50 ms).

FUNCIÓN SETUP():

- Serial.begin(115200); inicializa la comunicación serial a 115200 baudios, lo cual permite monitorear el comportamiento desde la consola.

- pinMode(LED, OUTPUT); y pinMode(buttonPin, INPUT_PULLUP); configuran los pines del LED como salida y el pin del botón como entrada con resistencia de "pull-up" interna.
- digitalWrite(LED, LOW); apaga el LED al inicio.

FUNCIÓN LOOP()**:**

- reading = digitalRead(buttonPin); lee el estado actual del botón.
- El código usa un retraso para "debounce" (eliminar el ruido al presionar el botón): Si el tiempo transcurrido desde la última pulsación es mayor a debounceDelay (50 ms), el código continúa.

CONDICIONES PARA DETECTAR PRESIONES DEL BOTÓN:

- Si reading == LOW && buttonState == HIGH, significa que el botón fue presionado (de HIGH a LOW). Se incrementa el contador pressCount, y se imprime el número de pulsaciones.
- Si pressCount == 2, el LED se enciende (LEDState = HIGH), y se imprime "LED Encendido" en la consola.
- Si pressCount == 3, el LED se apaga (LEDState = LOW), se reinicia pressCount a 0, y se imprime "LED Apagado" en la consola.

Finalmente, buttonState = reading; y lastButtonState = reading; actualizan los estados del botón para la siguiente iteración del bucle.

RESUMEN: Este programa controla un LED que se enciende al presionar el botón dos veces y se apaga con una tercera pulsación.

CONCLUSIONES

Este proyecto demostró cómo una placa ESP32 puede ser utilizado para controlar la intensidad de un LED mediante PWM. Esta práctica es útil para aplicaciones donde el ajuste de intensidad de luz es fundamental, tanto para mejorar el ambiente como para reducir el consumo energético. Personalmente, considero que el uso del ESP32 para aplicaciones de control en IoT es altamente eficiente, y esta práctica sentó una excelente base para explorar proyectos más complejos de domótica.

REFERENCIAS

- Espressif Systems. (2023). *ESP32 Technical Reference Manual*. Recuperado de https://www.espressif.com
- Arduino Documentation. (2023). *PWM*. Recuperado de https://www.arduino.cc