UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

Facultad de Ingeniería

Ingeniería en Ciencias de la Computación

SIMULACIÓN DE SISTEMAS

2.1.4.1 Práctica. Lectura de voltaje y Control de PWM

Trabajo de:

- ADRIAN (ADORA) GONZÁLEZ DOMÍNGUEZ *[359834]*
- JOSÉ ANGEL ORTÍZ MERAZ *[353195]*

Asesor: OSCAR RAMSES RUIZ VARELA

10 de octubre de 2024

Introducción

La lectura de valores analógicos resulta muy útil en ocasiones para poder realizar la lectura de un voltaje que procede por ejemplo de un potenciómetro y utilizarlo para activar determinadas salidas en función del valor que tome este regulador. El uso de microcontroladores como el ESP32 nos permite implementar técnicas avanzadas para el control de sistemas basados en senales analogicas, como lo es la Modulacion por Ancho de Pulso (PWM), que es utilizada en sistemas que requieren el ajuste de intensidad de LEDs, velocidad en motores y control de servomotores, entre otros.

En esta práctica se utilizará un microcontrolador ESP32 para leer un voltaje analógico generado por un potenciómetro conectado en serie con una resistencia. La lectura del voltaje será realizada a través de un canal ADC (Convertidor Analógico a Digital) permitiendo convertir el voltaje en un valor digital el cual se procesa para controlar la salida de una señal PWM. El valor PWM lo utilizaremos para ajustar la intensidad o brillo de un diodo LED. De esta manera exploramos la capacidad y uso de utilizar señales analógicas para controlar salidas de dispositivos en un sistema digital.

Metodología

Utilizando el microcontrolador ESP32, conectar a una entrada ADC un arreglo de una resistencia de 330 ohms y un potenciómetro de 10k ohms, en serie, para variar el voltaje en la resistencia. (semejante al circuito 1.5.1). Alimentar el arreglo con 3.3v. Variar el potenciómetro y comprobar con un multímetro que el voltaje en la resistencia varía. De un mínimo a 3.3v. Hacer un programa para que el ESP32 lea ese valor, y lo envié por el puerto serie como un número representativo de ese voltaje, no como un voltaje. En el programa, variar la salida de uno de los canales de PWM para que varíe de un mínimo (aprox 1/30) a un máximo, (100 %) del ciclo de trabajo. Conectar a ese canal de PWM un LED, y observar que su iluminación es controlada por la posición del potenciómetro.

Explicación del funcionamiento del ADC en el ESP32

El **ADC (Convertidor Analógico a Digital)** es una característica fundamental de los microcontroladores, incluida en el ESP32. Su principal función es transformar señales analógicas en valores digitales, lo que permite que el microcontrolador pueda procesar y analizar señales provenientes de sensores que generan voltajes variables, como sensores de temperatura, luz, presión, etc.

- ## Resolución

El ADC del ESP32 tiene una resolución de **12 bits**, lo que significa que puede convertir una señal analógica en uno de **4096 (2^12) valores posibles**, que van desde **0** hasta **4095**. Estos valores digitales corresponden a un rango de voltajes entre **0V y 3.3V**, siendo 0 para 0V y 4095 para 3.3V. De esta manera, cada paso del ADC representa una fracción de este rango de voltajes:

$$Resolución = 3.3V/4095 = 0.0008V$$

Esto significa que el ADC puede detectar cambios en el voltaje de aproximadamente **0.8 mV**.

- ## Canales del ADC

El ESP32 cuenta con varios canales ADC distribuidos en dos ADCs internos llamados **ADC1** y **ADC2**. Cada uno de estos ADCs tiene diferentes pines asignados:

  - **ADC1** tiene **8 canales** que corresponden a los pines GPIO 32 a 39.
  - **ADC2** tiene **10 canales** que van desde GPIO 0 hasta GPIO 15.

Estos canales se pueden utilizar para conectar múltiples sensores o dispositivos analógicos que generen señales de voltaje.

- ## Configuraciones del ADC

A través del código, se puede configurar diferentes aspectos del ADC, como su **ancho de resolución** o el **rango de voltaje** que puede medir. Por defecto se usa el rango de 0V a 3.3V, sin embargo, este rango puede modificarse para ajustarse a las necesidades del proyecto.

- **Resolución de los canales:** Aunque la resolución máxima es de 12 bits, puede reducirse si no se necesita tanta precisión. Por ejemplo, se podría configurar una resolución de 10 bits para obtener un rango de 0 a 1023, lo que puede ser útil para optimizar la velocidad de la conversión.
- **Rango de conversión:** Si la señal de entrada nunca excede un determinado valor (por ejemplo, 2.5V), se puede reducir el rango ADC para que el valor máximo digital (4095) corresponda a este voltaje más bajo, logrando así una mayor precisión.

Explicación del funcionamiento del PWM en el ESP32

PWM son las siglas de ***Pulse Width Modulation*** **(*modulación por ancho de pulso*)**, es una una señal de voltaje de ***pulso rectangular*** y ***periódico*** (los pulsos se repiten a intervalos de tiempo fijo) y se utiliza para enviar información o para modificar la cantidad de energía que se envía a una carga (emulando una señal analógica).

Es muy común utilizarlas para:

- Regular la intensidad de LEDs.
- Controlar el funcionamiento de servomotores.
- Controlar velocidad de motores de corriente continua.
- Controlar motores eléctricos de inducción o asincrónicos.

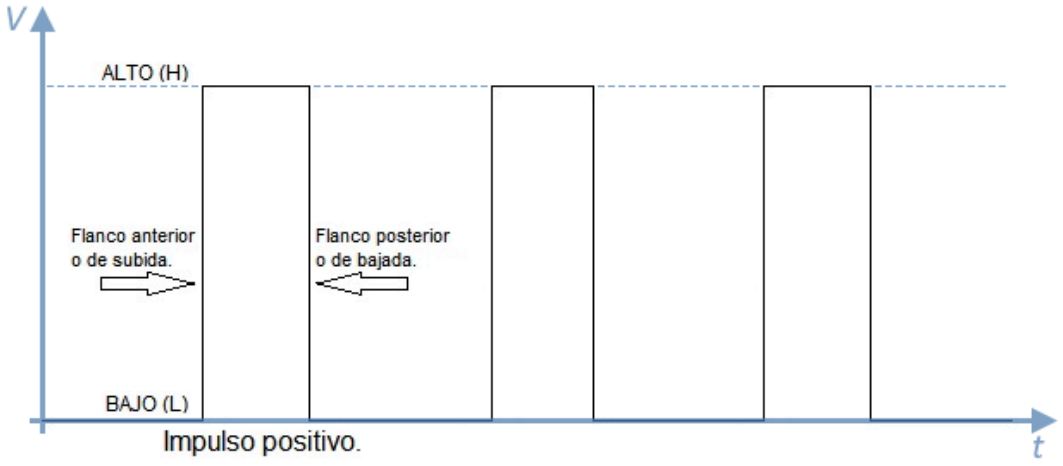
El controlador ESP32 LED PWM cuenta con 16 canales independientes que pueden configurarse para generar señales PWM con diferentes propiedades. Todos los pines que pueden actuar como salidas se pueden usar como pines PWM.

Para producir una señal PWM, es necesario definir a través del código lo siguiente:

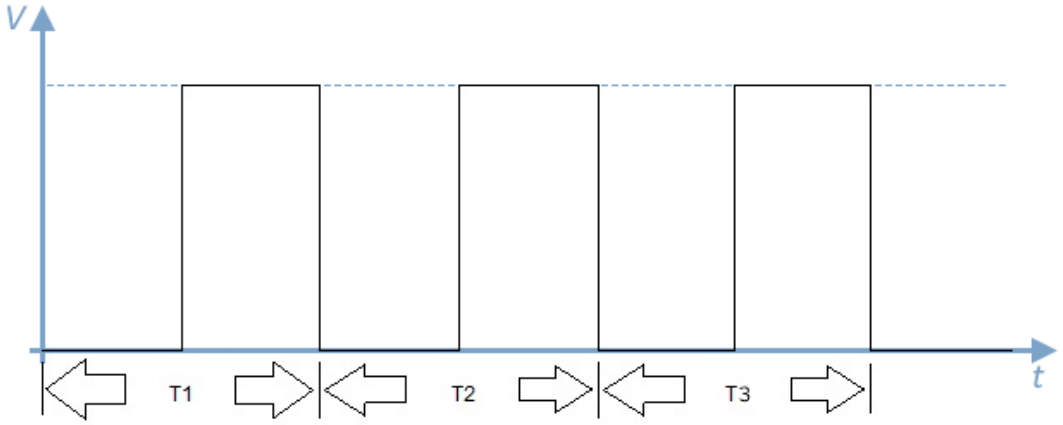
- Frecuencia de la señal
- Ciclo de trabajo
- Canal PWM
- GPIO que emite la señal.

Una señal PWM se se caracteriza por:

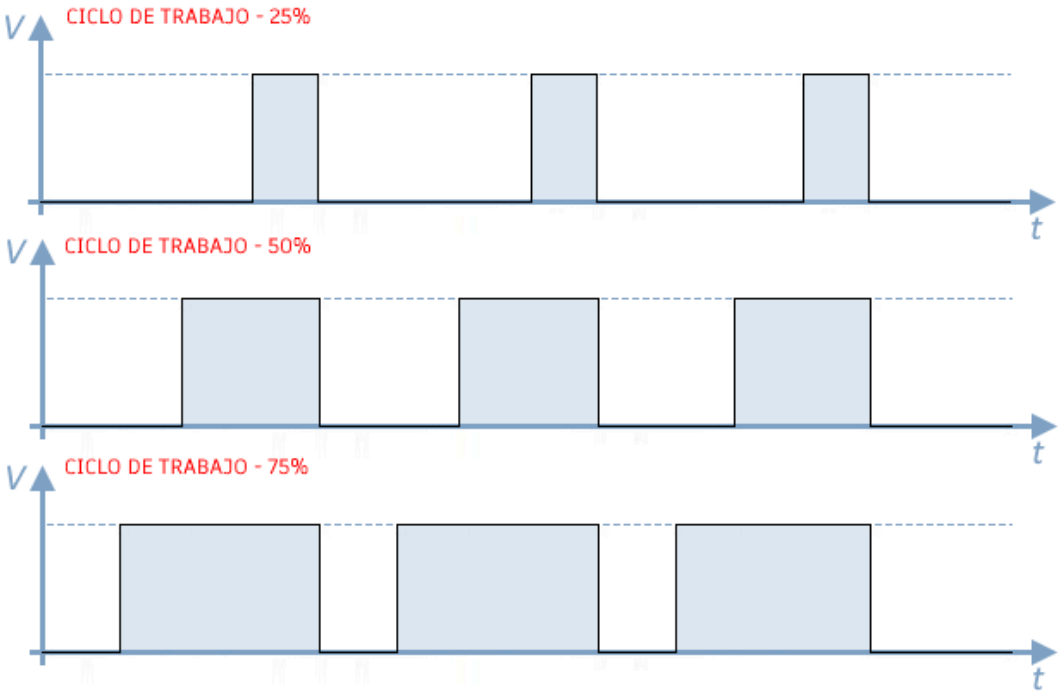
- Sus valores **alto** y **bajo** de tensión: son los valores mínimo –*bajo*– y máximo –*alto*– de tensión que alcanza el pulso. Se corresponden con los dos estados lógicos *bajo=0* y *alto =1*.



- Su **frecuencia** –*freq*-: es la inversa del intervalo de tiempo que tarda en realizarse un ciclo completo ***freq=1/T***, se mide en *hercios* (*Hz = ciclos por segundo*). ***T=T1=T2=T3…***, es decir el tiempo que tarda cada ciclo se denomina **periodo** .



- Su **ciclo de trabajo** –*duty cicle*-: indica el porcentaje de tiempo que el ***pulso*** está a nivel alto.



Los PWM en un *microcontrolador* ESP32 tienen las siguientes características:

- Unos valores ***bajo*** y ***alto*** de tensión son 0V y 3.3V respectivamente.
- La **frecuencia** puede tomar valores entre 1 y 40.000.000 Hz.
- El **ciclo de trabajo** (*duty cicle*) puede tomar valores entre 0 (0%) y 1023 (100%).

Cálculos de los voltajes que se obtienen del arreglo de resistencia y potenciómetro

Para calcular el voltaje intermedio del circuito, debemos calcular el voltaje que se cae en el potenciómetro, para ello, lo realizamos con la siguiente fórmula:

$$V_P = \frac{R_P \times V_I}{R_T}$$

En el caso de que el potenciómetro tenga su mayor valor de resistencia:

$$V_P = \frac{10k\Omega \times 3.3V}{10.3k\Omega} = 3.2038V$$

En el caso de que el potenciómetro tenga su menor valor de resistencia:

$$V_P = \frac{0\Omega \times 3.3V}{300\Omega} = 0V$$

Programa que lee la entrada de voltaje y varía el ciclo de trabajo del PWM de acuerdo al voltaje de entrada

```
#include <Arduino.h>
#include <esp32-hal-gpio.h>

const int sensorPin = GPIO_NUM_15;
const int ledPin = GPIO_NUM_2;
const int ledChannel = 0;
const int frequency = 5000;
const int resolution = 8;

void setup()
{
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);

  ledcSetup(ledChannel, frequency, resolution);
  ledcAttachPin(ledPin, ledChannel);
}
```

```
void loop()
{

  // Leer el valor del sensor en GPIO15 (ADC2_3)
  int sensorValue = analogRead(sensorPin);

  // Calcular el voltaje
  float voltage = (sensorValue / 4095.0) * 3.3;

  Serial.print(sensorValue);
  Serial.print(", ");
  Serial.println(voltage, 2);

  // Establecer el ciclo de trabajo según el valor leído por
el adc
  float dutyCycle = sensorValue / 400;
  ledcWrite(ledChannel, dutyCycle);
}
```

Explicación del código del programa

Necesitaremos conectarnos a 2 pines GPIO. El primero de ellos [el cual es el pin 15], debe ser pin de ADC ya que será en el que leeremos el voltaje intermedio del arreglo de resistencia y potenciómetro (es decir, el voltaje que varía según la posición del potenciómetro) [y corresponde al cable naranja en la imagen del circuito].

El segundo pin a usar [que es el pin 2], será el que se configure con PWM a fin de poder entregar una intensidad que controle la iluminación del LED. Este será el voltaje de entrada para el circuito de LED y resistencia [y corresponde al cable blanco en la imagen del circuito].

Entonces, en la función setup se configura el puerto serial para poder imprimir los valores en la consola, se configura el pin 15 para lectura analógica, se configura el pin 2 con el canal 0 del PWM que permitirá modular la salida a la frecuencia y resolución establecidas, con el ciclo de trabajo que se calculará en el ciclo.

En la función loop se realiza la lectura analógica, y se calcula el voltaje. Por último, el valor leído lo dividiremos entre 160 para obtener el ciclo de trabajo (duty cycle) en el que se establecerá el canal 0 del PWM, así es como variamos la luminosidad del LED.

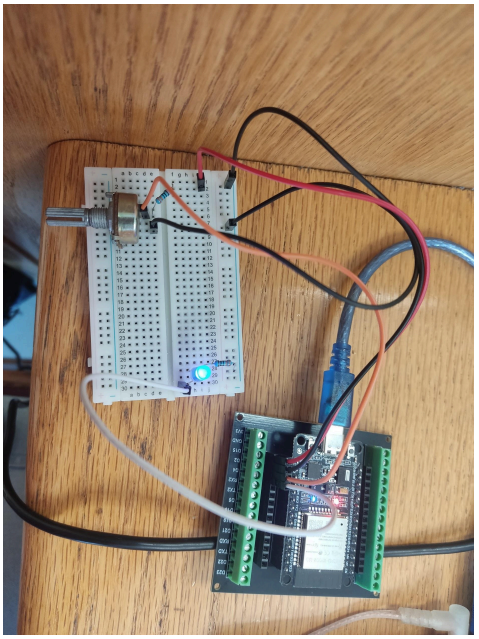
Cálculos de los valores numéricos usados en el programa

Hay 2 valores numéricos para los que se requiere hacer cálculos.
El cálculo del voltaje se realiza dividiendo el valor leído entre el mayor valor leíble que es 4095 (por los 12 bits del ADC), y se multiplica por el voltaje de entrada del circuito que es 3.3V.

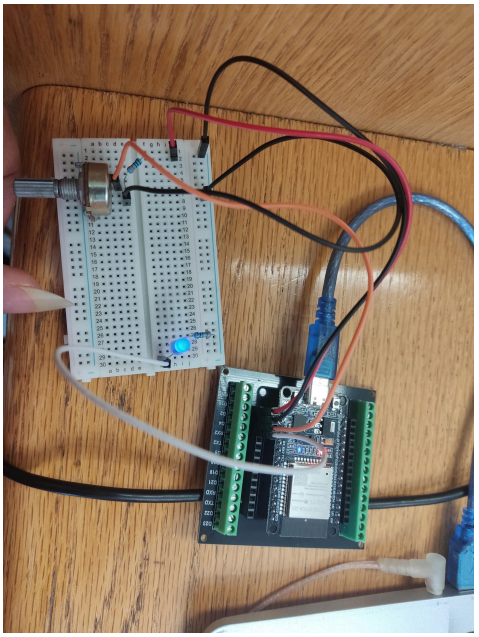
El segundo valor calculado es el 160 el cual divide el valor leído por el ADC para obtener el ciclo de trabajo. Para calcularlo el profesor nos recomendó un duty cycle máximo de 25 que calculó por prueba y error; para obtener este duty cycle, el mayor valor del ADC, es decir 4095, debe dividirse entre un número para dar 25, y ese número es 160.

Fotos del circuito armado, y de su operación

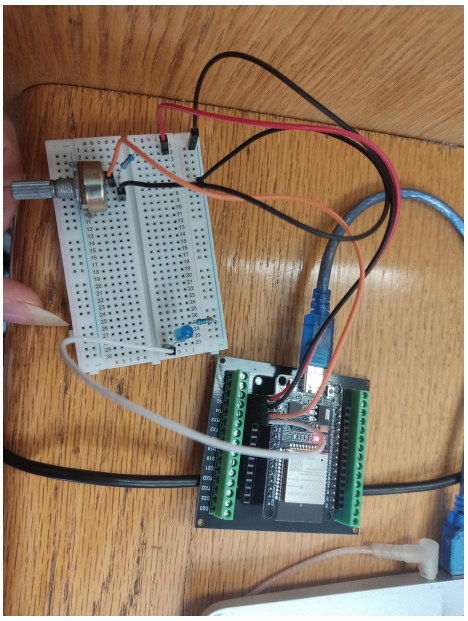
Potenciómetro en su valor mayor.



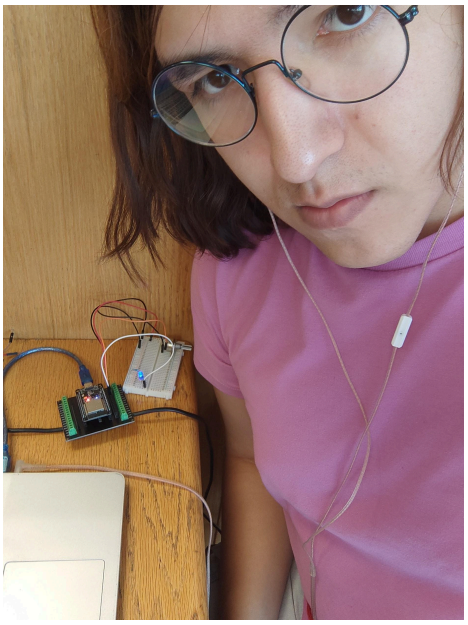
Potenciómetro en su valor menor.



Potenciómetro en un valor intermedio.



Evidencia de realización.



Bibliografía

Pinillos, J. M. (2021, 1 noviembre). Básicos ESP32: Lectura de valor analógico. Tecnotizate. Recuperado 9 de octubre de 2024, de https://tecnotizate.es/basicos-esp32-lectura-de-valor-analogico/

Pinillos, J. M. (2021b, noviembre 1). Básicos ESP32: Mapeo de pines y sensores internos. Recuperado 10 de octubre de 2024, de Tecnotizate. https://tecnotizate.es/esp32-mapeo-de-pines-y-sensores-internos/

Carranza, S. (2022, 17 septiembre). CONOCIENDO AL ESP32. TodoMaker. Recuperado 10 de octubre de 2024, de https://todomaker.com/blog/conociendo-al-esp32/

No, D. (s. f.). MICROPYTHON ESP32 – Modulación por ancho de pulsos PWM (Pulse Width Modulation). ESPloradores -Ayuda y Consejos Para DIY Makers-. Recuperado 10 de octubre de 2024, de https://www.esploradores.com/micropython_pwm/