Roteiro Nº 03	**Conversão Digital para Analógico no ESP32: Controle de Frequência em Inversores**	
Curso Engenharia Elétrica	**Disciplina** ELT 1119 – Redes e Aplicações IoT	**Professor** Carlos Alberto Vasconcelos Bezerra
Nome do Estudante		**Data**

1. Objetivos da Aprendizagem

Implementar um sistema no ESP32 que receba um valor de frequência (0 a 120 Hz) via teclado, converta esse valor para um sinal analógico utilizando o conversor digital-analógico (DAC) do ESP32 e gere uma tensão proporcional que pode ser utilizada no controle de um inversor de frequência.

2. Material Necessário

- Placa de desenvolvimento ESP32-DevKitC-V4;
- Computador com ambiente de desenvolvimento Thonny instalado;
- Cabo USB para comunicação entre o ESP32 e o computador;
- Fonte de alimentação 5V;
- Multímetro para aferição da saída analógica;
- Inversor de frequência.

3. Fundamentação Teórica

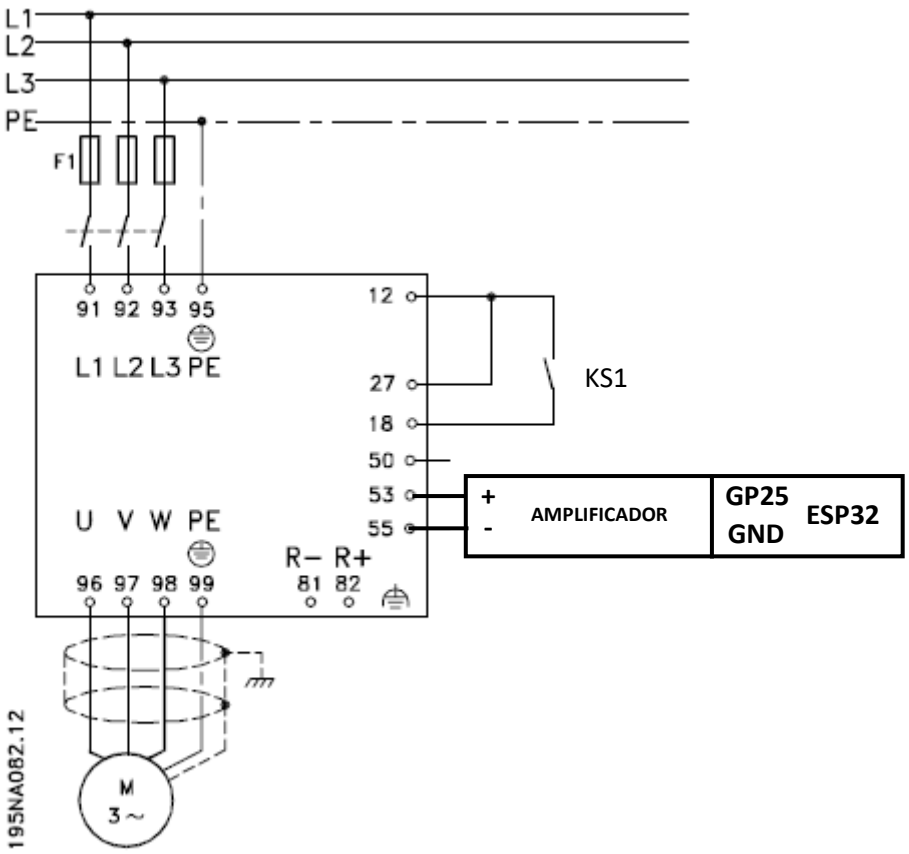
A conversão digital para analógico (DAC) permite que microcontroladores como o ESP32 convertam valores numéricos em contínuos, o que é essencial para o controle de dispositivos analógicos, como inversores de frequência. O ESP32 possui dois canais DAC embutidos (GPIO25 e GPIO26), permitindo a geração de sinais analógicos com resolução de 8 bits, variando de 0V a 3,3V. A relação entre a frequência de entrada e a tensão de saída pode ser dada por:

$$\boldsymbol{Vsaída = \frac{Fentrada}{120} * 3,3}$$

Onde: Fentrada é o valor de frequência inserido pelo usuário (de 0 a 120 Hz).
Esse controle pode ser utilizado para ajustar dinamicamente a frequência de saída de um inversor, permitindo o controle de velocidade dos motores AC.

4. Procedimento Experimental

4.1 Monte o circuito do inversor conforme diagrama abaixo. Conecte o pino DAC GP25 nos terminais 53 e 55 conforme indicado no diagrama.



4.2 Configuração do Ambiente:

- Conecte o ESP32 ao computador via cabo USB.
- Abra o Thonny e conecte-se à porta serial do ESP32.

4.3 Escrita do Código em MicroPython:

- Crie um novo script e implemente o seguinte código:

```
from machine import DAC, Pin
import time

# Configuração do canal DAC
dac = DAC(Pin(25))

while True:
    try:
        freq = int(input("Digite a frequência desejada (0-120 Hz): "))
        if 0 <= freq <= 120:
            # Conversão para nível de tensão (escala de 0 a 255)
            valor_dac = int((freq / 120) * 255)
            dac.write(valor_dac)
            print(f"Frequência digitada: {freq} Hz - Tensão de saída: {(valor_dac / 255) * 3.3:.2f} V")
        else:
            print("Frequência fora do intervalo. Digite um valor entre 0 e 120 Hz.")
    except ValueError:
        print("Entrada inválida. Digite um número inteiro.")
```

4.4 Ajustar os parâmetros do Inversor de frequência conforme sequência abaixo.

Programe os parâmetros do motor que estiverem na plaqueta de identificação

Potência do motor [kW]	Parâmetro 102
Tensão do motor [V]	Parâmetro 103
Freqüência do motor [Hz]	Parâmetro 104
Corrente do motor [A]	Parâmetro 105
Velocidade nominal do motor	Parâmetro 106

Programe o intervalo de referência

Referência mín., Ref_{MIN}	Parâmetro 204	**(0 Hz)**
Referência máx., Ref_{MAX}	Parâmetro 205	**(100 Hz)**

Programe o tempo de rampa

Tempo de aceleração [s]	Parâmetro 207	**(3 Seg)**
Tempo de desaceleração [s]	Parâmetro 208	**(3 Seg)**

No parâmetro 002, *Controle local/remoto,* o modo do conversor de freqüência pode ser selecionado como *Operação remota* [0] através dos terminais de controle ou *Local* [1] através da unidade de controle.

Ajustar o parâmetro 002 para (0)

4.5 Execução e Testes:

- Execute o código no ESP32.
- Digite diferentes valores de frequência e observe a tensão de saída usando um multímetro.

4.6 Análise dos Resultados:

- Compare os valores medidos com os esperados.
- Discuta possíveis imprecisões devido à resolução do DAC e como isso pode afetar o controle do inversor.

5. Exercícios

1. Qual a função principal do conversor digital para analógico (DAC) no ESP32? Explique sua importância no controle de dispositivos analógicos.
2. Considere que um usuário digitou uma frequência de 60 Hz. Qual será o valor da tensão de saída do DAC no ESP32? Justifique seu cálculo.
3. Explique como a resolução de 8 bits do DAC influencia a precisão da tensão de saída. Quais seriam as vantagens de uma resolução maior para essa aplicação?
4. O que aconteceria se um valor fora do intervalo (menor que 0 Hz ou maior que 120 Hz) fosse digitado no código implementado? Como o código trata essa situação?

6 Simulação

Simular os exercícios propostos utilizando o simulador wokwi no link:

https://wokwi.com/projects/new/esp32

Bibliografia

ASCHER, D.; LUTZ, M. Aprendendo Python. Porto Alegre: Bookman, 2007.

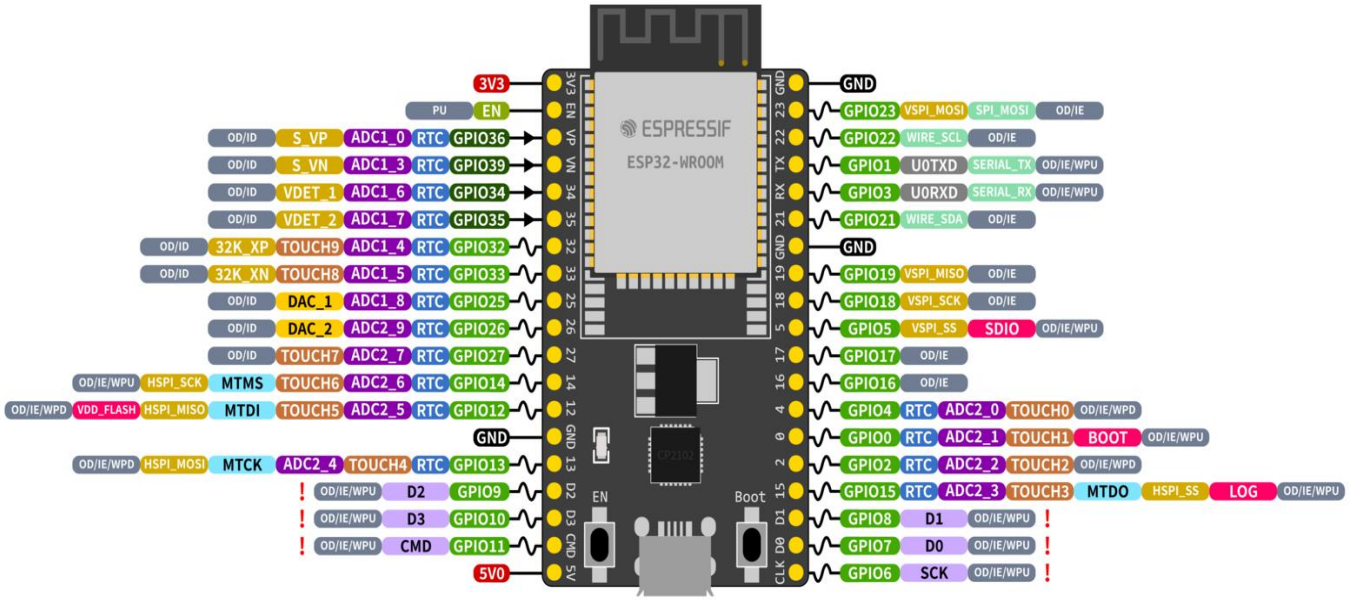
MENEZES, N. N. C. Introdução à programação com Python. São Paulo: Ed. Novatec, 2014.

OLIVEIRA, S. Internet das coisas com ESP8266, Arduino e Raspberry Pi. São Paulo: Ed. Novatec, 2017.

OLIVEIRA, C. L. V. IoT com Micropython e NodeMCU. São Paulo: Ed. Novatec, 2022.

ESP32-DevKitC

ESPRESSIF



ESP32 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
BLuetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
DAC_X Digital-to-Analog Converter
DEBUG JTAG for Debugging
FLASH External Flash Memory (SPI)
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
ARDUINO Arduino Related Functions
STRAP Strapping Pin Functions

RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)
! Pin Shared with the Flash Memory
Can't be used as regular GPIO

GPIO STATE
WPU: *Weak Pull-up (Internal)*
WPD: *Weak Pull-down (Internal)*
PU: *Pull-up (External)*
IE: *Input Enable (After Reset)*
ID: *Input Disabled (After Reset)*
OE: *Output Enable (After Reset)*
OD: *Output Disabled (After Reset)*