

# 🎓 Conversores A/D

Unidade 4 | Capítulo 8 - Aula síncrona (10/02/2024)  
Prof. Wilton Lacerda Silva

Executores:



Coordenação:



Iniciativa:



# Jornada até aqui...

Executores:



Coordenação:



Iniciativa:



# Sumário

- Objetivos
- Conversão Analógico Digital
- Tipos de conversores ADC
- Uso do periférico ADC do RP2040
- Exemplos de Código
- Conclusão

# Pré-requisitos

- Familiaridade com a IDE do VS Code para desenvolvimento de software para microcontroladores.
- Controlar os pinos de entrada e saída (GPIO).
- Saber converter um número de decimal para binário.
- Dominar o conceito de tensão elétrica.
- Compreender a modulação PWM.



# Objetivos

- Compreender a diferença entre sinais analógicos e digitais.
- Reconhecer exemplos de sinais analógicos e digitais no cotidiano.
- Entender por que a conversão de analógico para digital é importante.
- Compreender o funcionamento do módulo A/D.
- Conhecer os diferentes modos de configuração do módulo conversor A/D.
- Configurar e implementar o uso do módulo A/D, inclusive em conjunto com o módulo PWM.

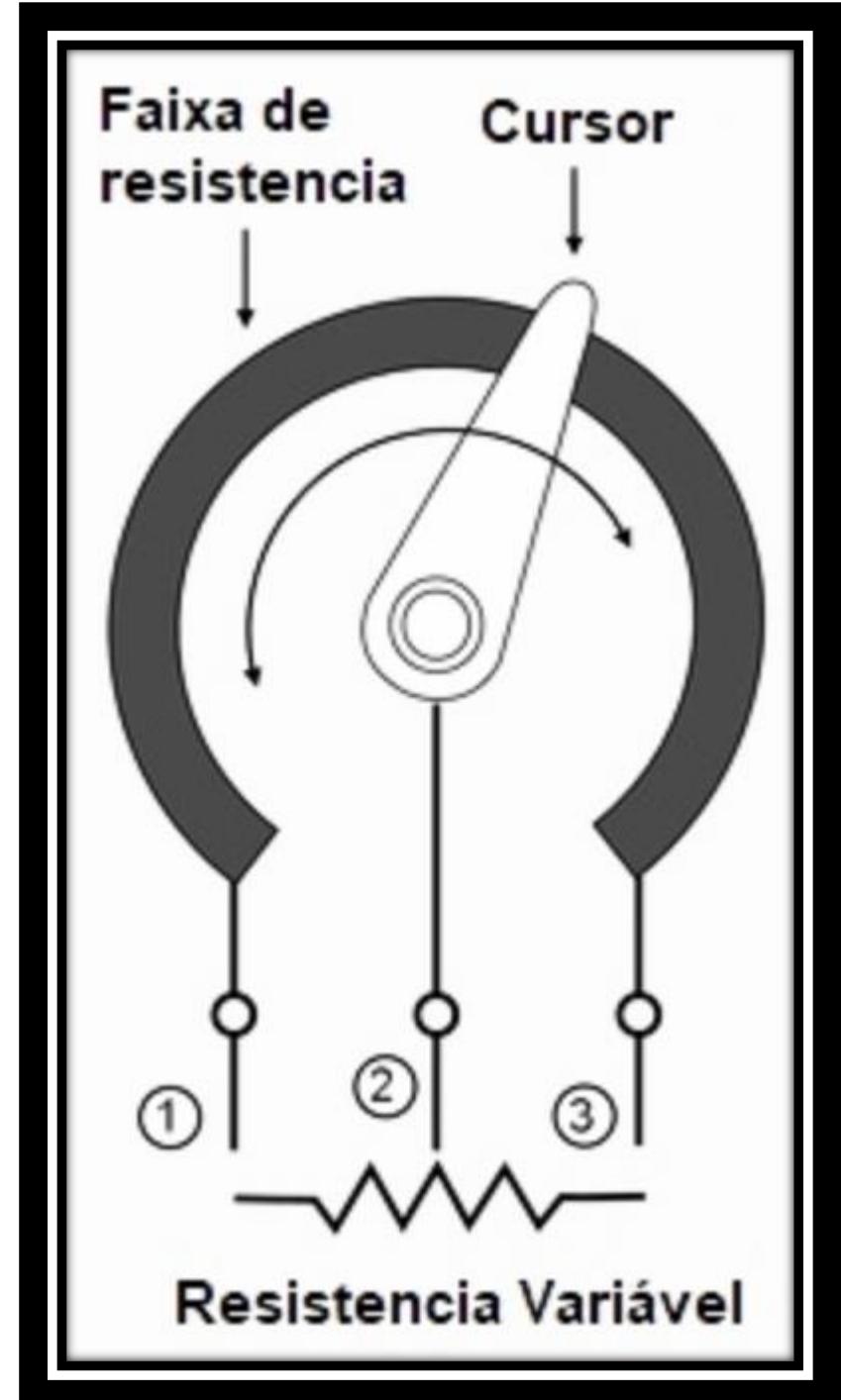
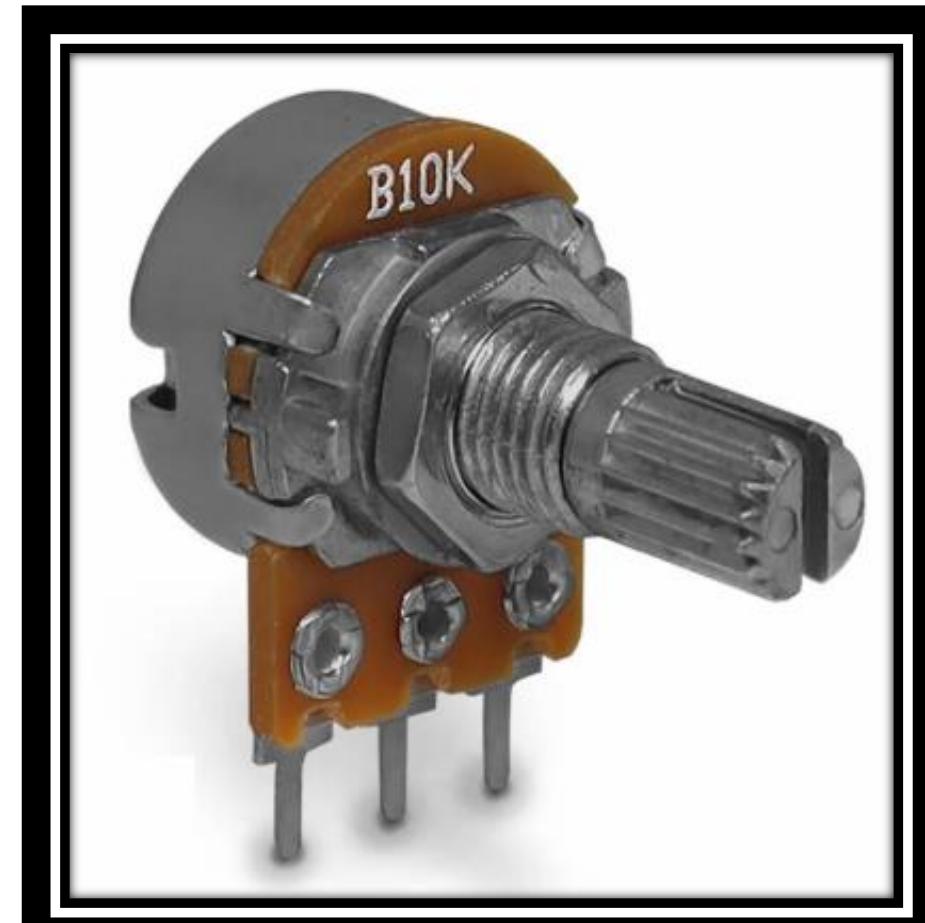
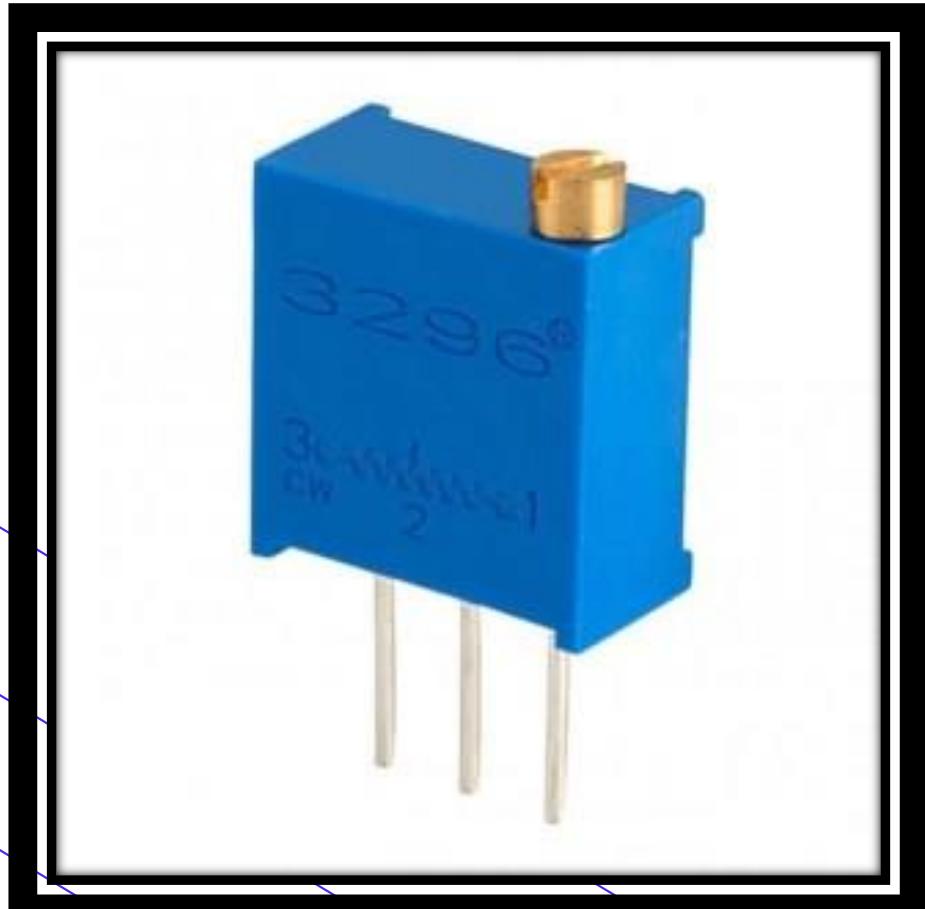


# Revisão

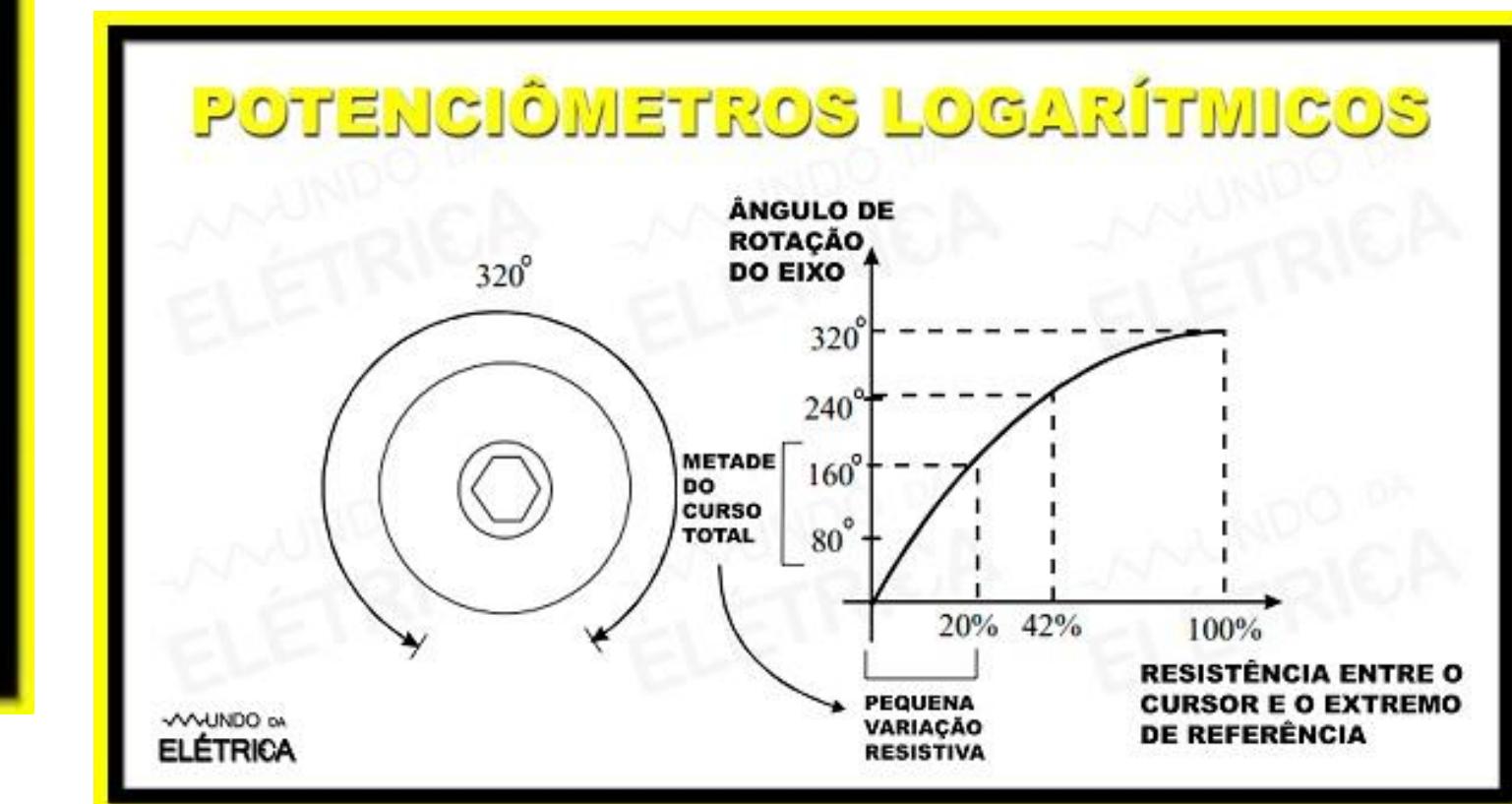
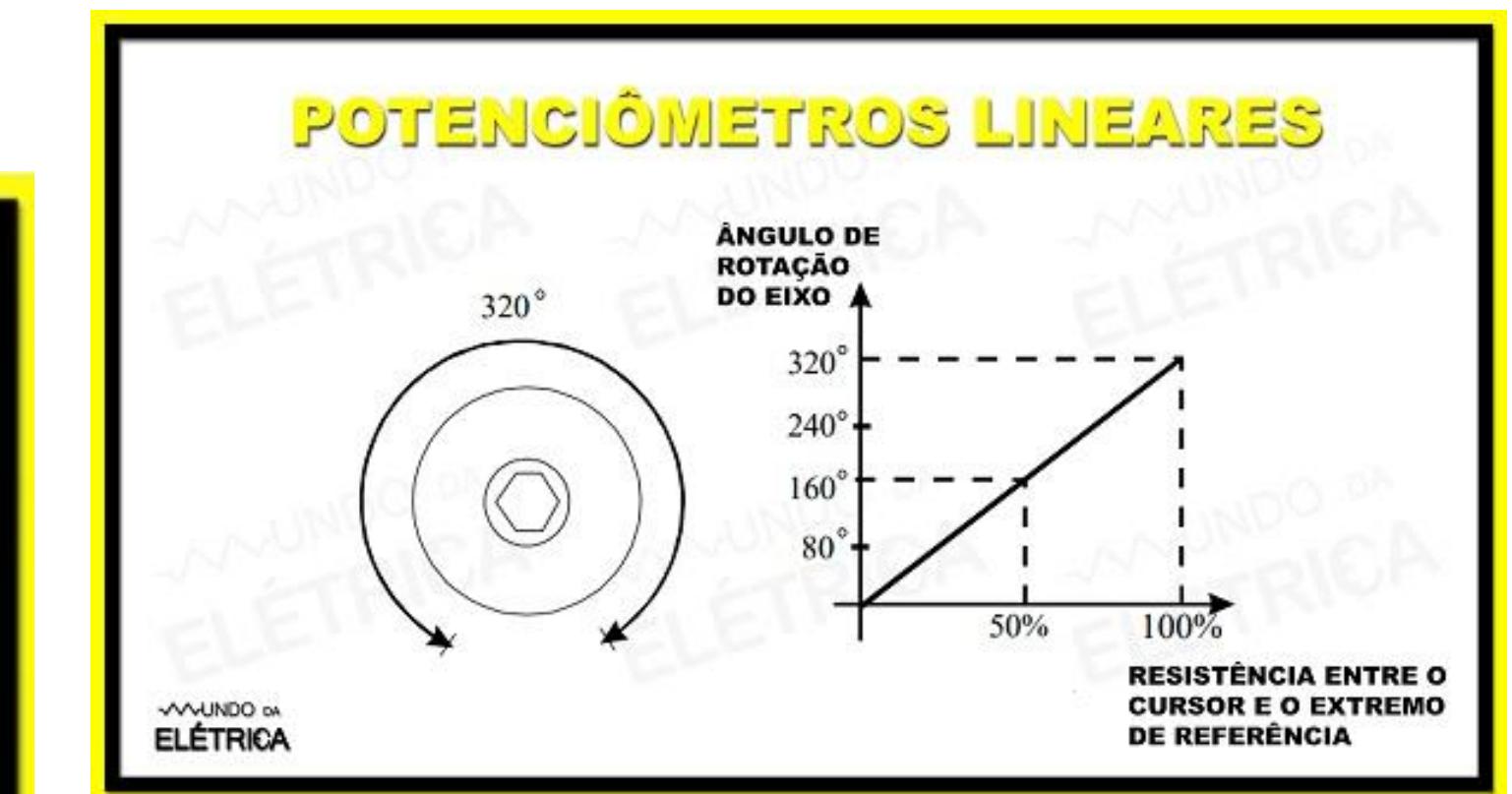
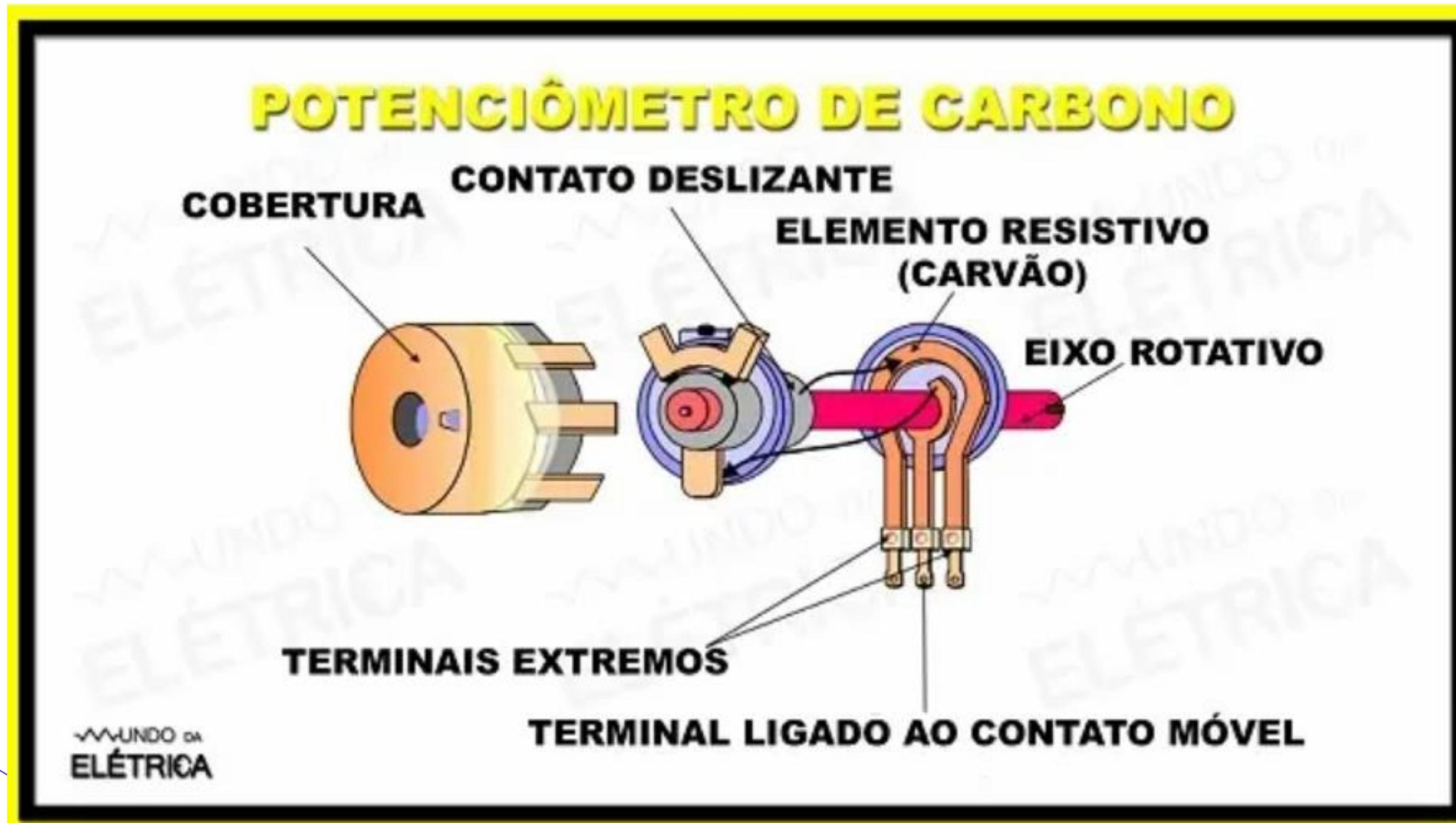
## Potenciômetro

É um componente eletrônico que funciona como um **resistor variável**, permitindo ajustar a resistência elétrica manualmente.

Ele é composto por três terminais: dois fixos (geralmente conectados aos extremos de um material resistivo) e um terminal móvel (cursor ou contato deslizante), que se move ao longo do material resistivo, alterando a resistência entre o terminal central e os extremos.



# Revisão



# Revisão

## Potenciômetro

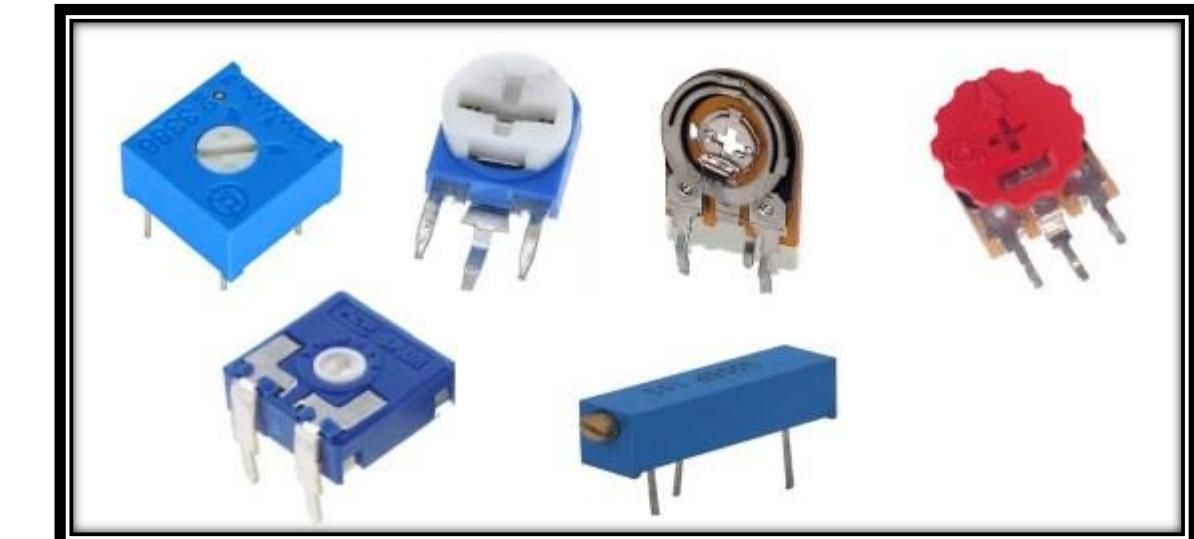
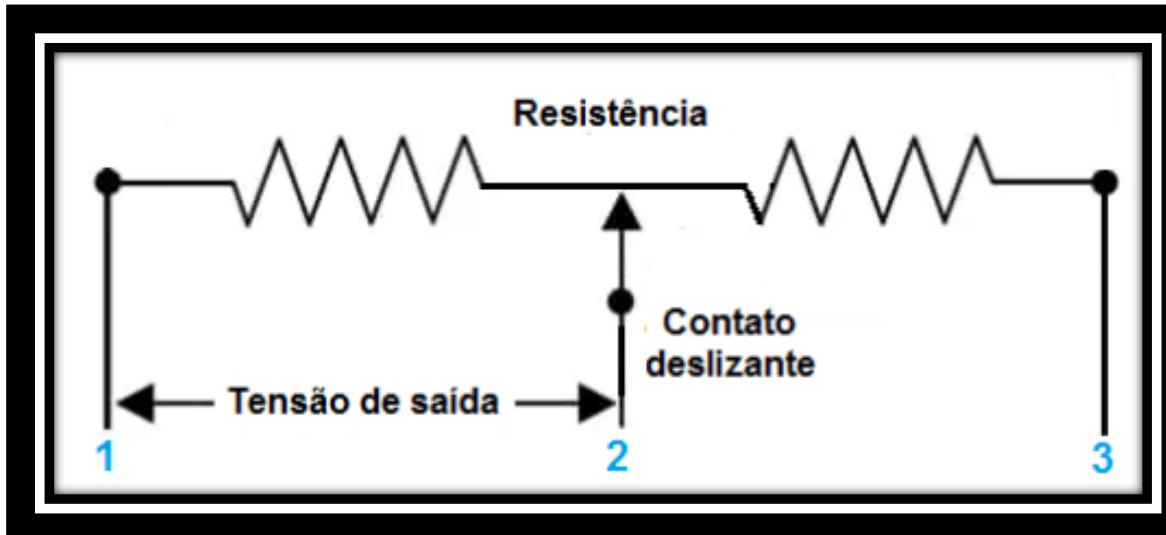
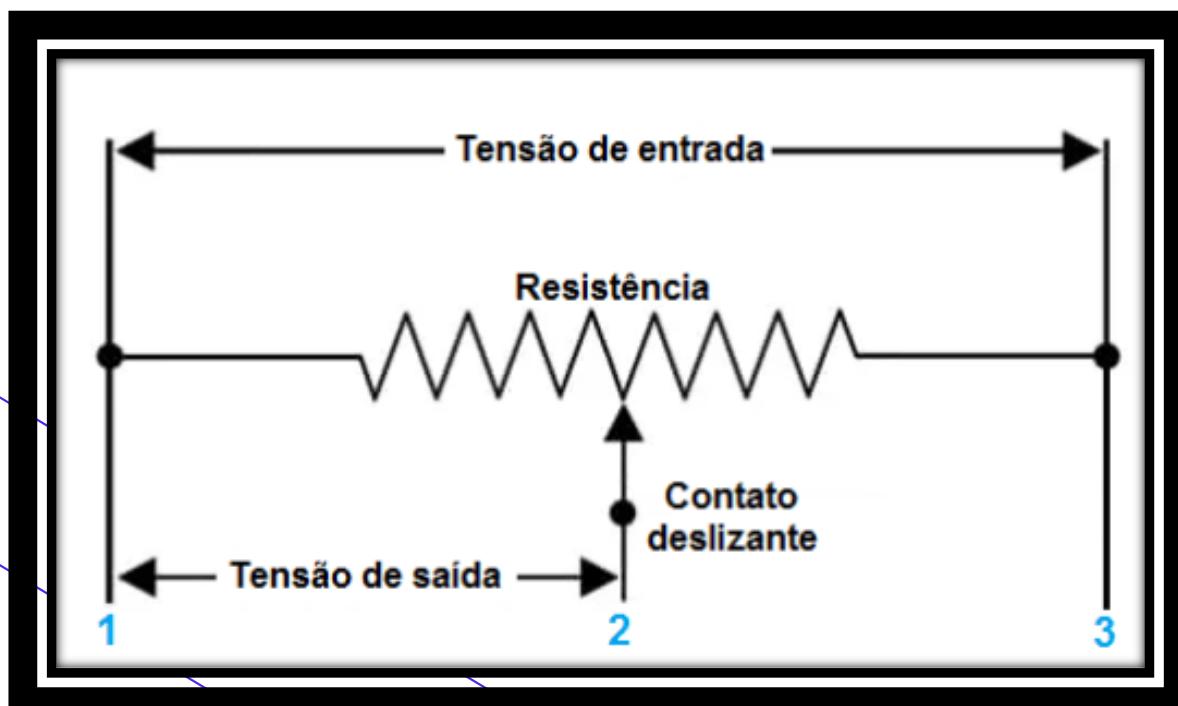
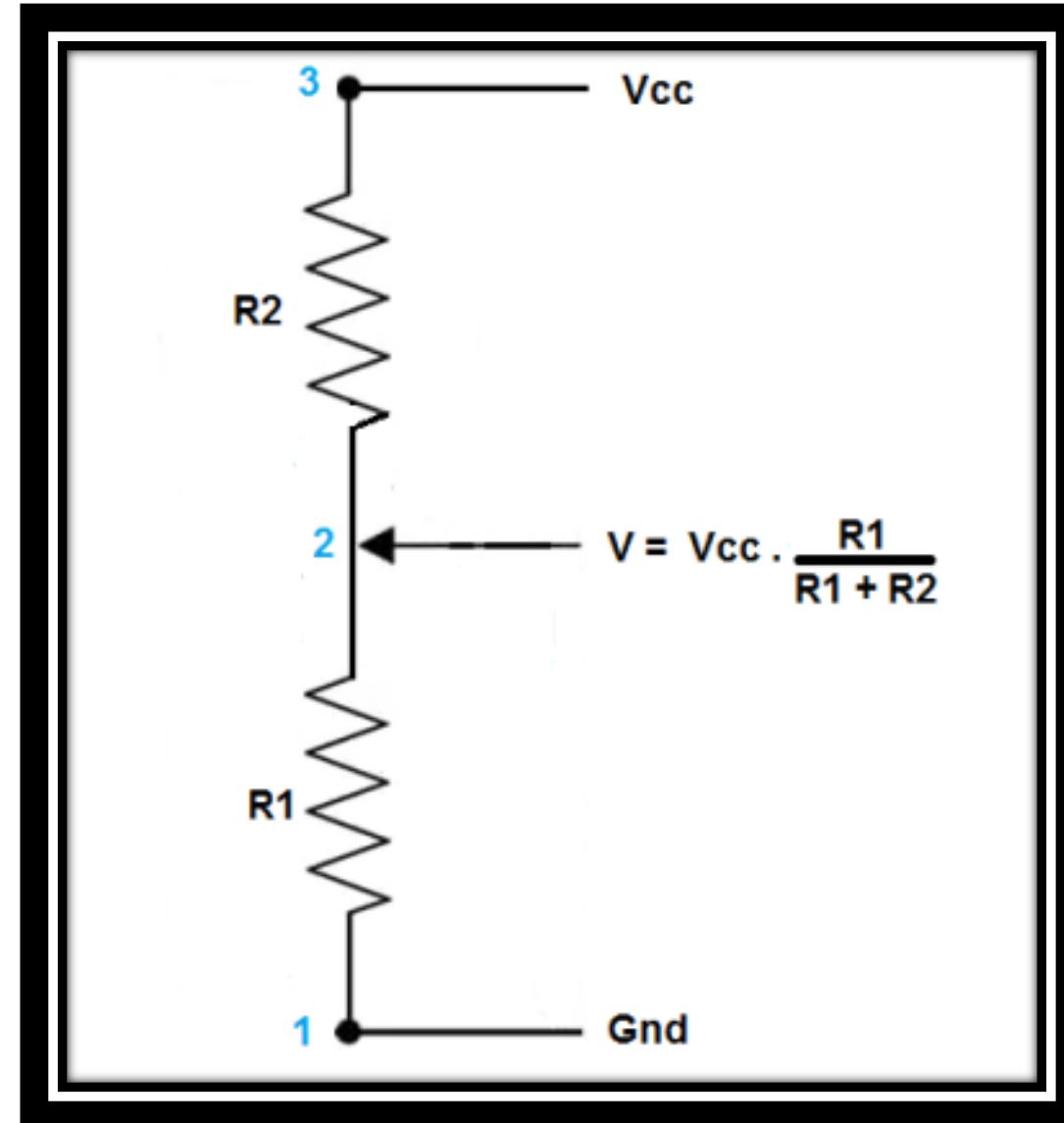
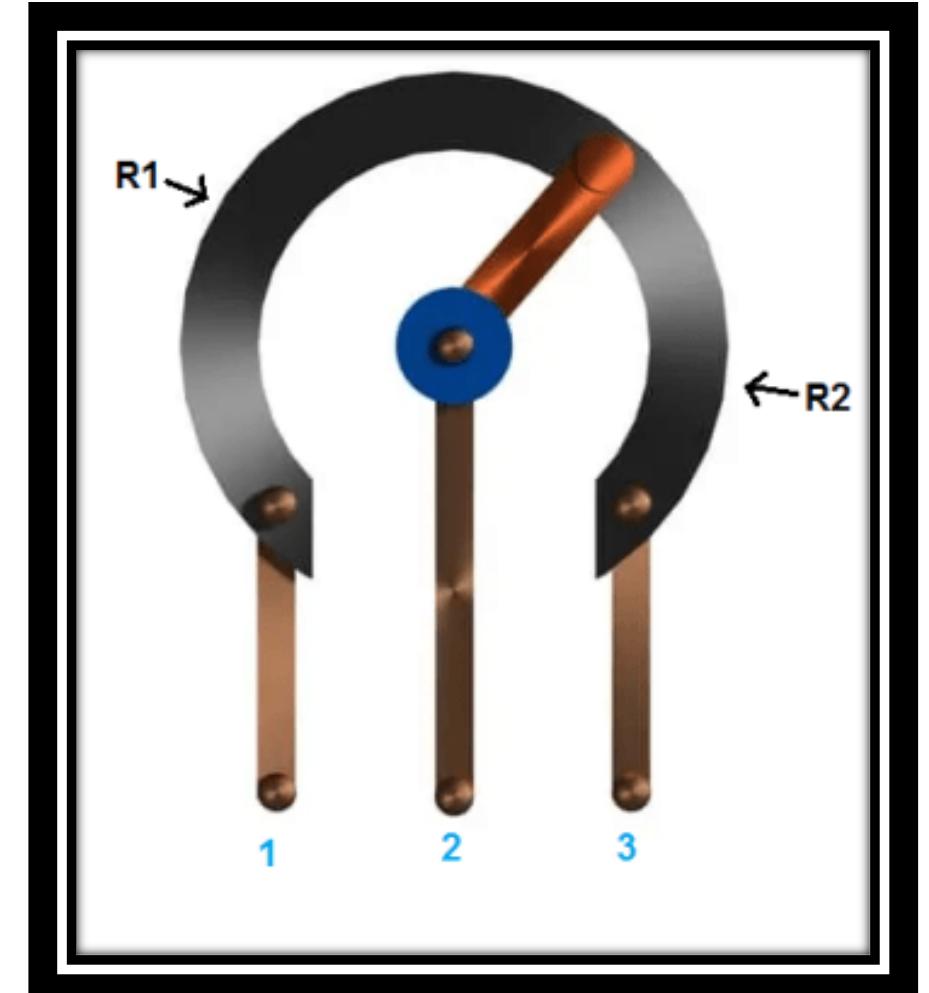
Sua principal função é atuar como um divisor de tensão.

Podem ser:

- "B" ou "Lin": Potenciômetro linear.
- "A" ou "Log": Potenciômetro logarítmico.

Aplicações comuns:

1. Controle de volume em equipamentos de áudio.
2. Ajuste de brilho em displays.
3. Controle de velocidade em motores.
4. Sensores de posição em dispositivos mecânicos.



## Sinais Analógicos

Um sinal analógico é uma representação contínua de uma grandeza física, que varia suavemente ao longo do tempo.

### Características:

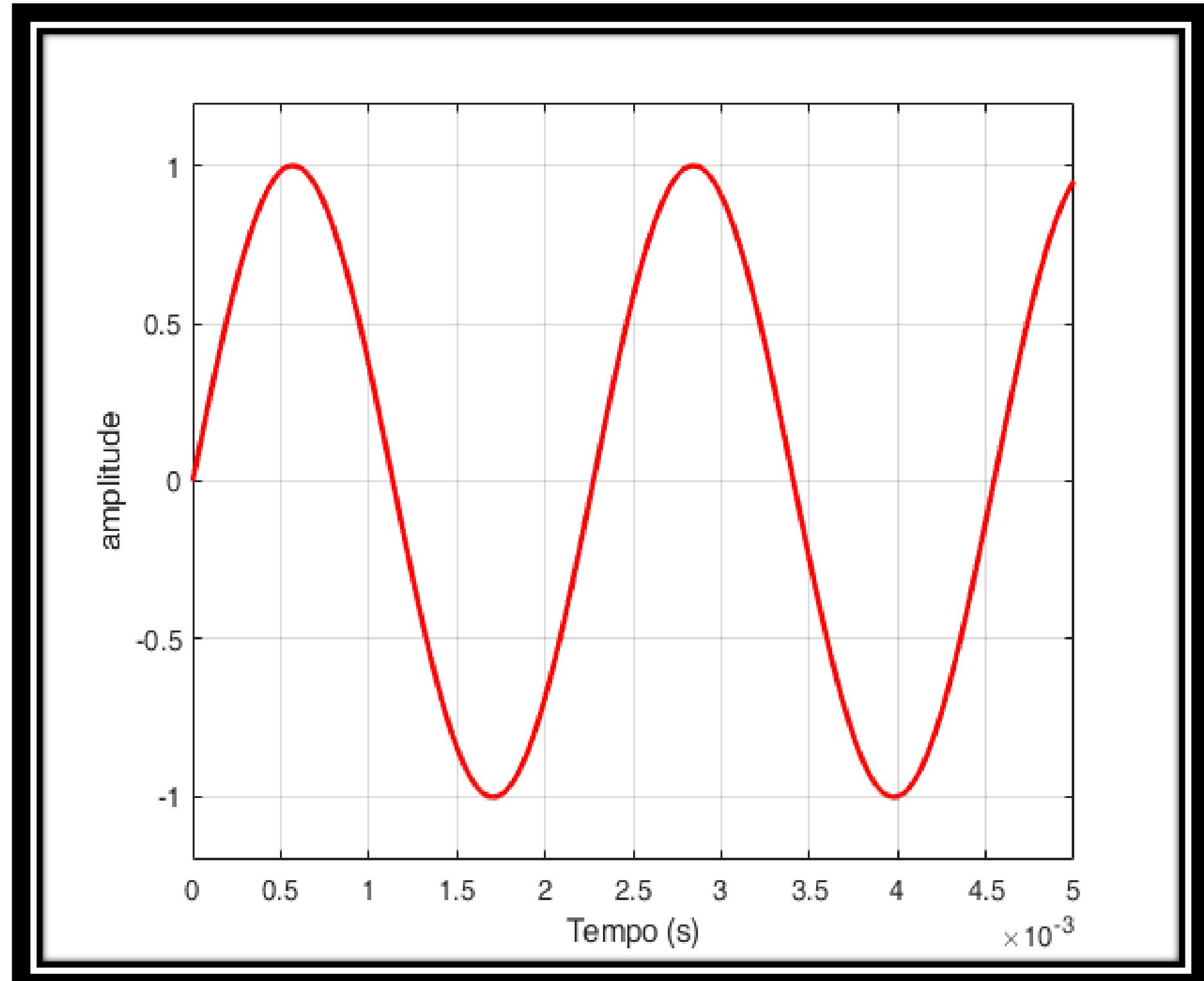
- Contínuo no tempo e na amplitude.
- Pode assumir **infinitos valores** dentro de um intervalo.

### Exemplos:

Som (onda sonora), temperatura, luz, tensão, corrente e potência elétrica em um circuito.

### Representação Gráfica:

Uma onda senoidal é exemplo clássico de sinal analógico. Veja o seu gráfico Eixo X (tempo) vs. Eixo Y (amplitude).



## Sinais Digitais

Um sinal digital é uma representação discreta de uma grandeza física, composta por valores específicos em intervalos de tempo definidos.

### Características:

- Discreto no tempo e na amplitude.
- Assume um número **finito de valores**.  
(geralmente representados em binário: 0 e 1).

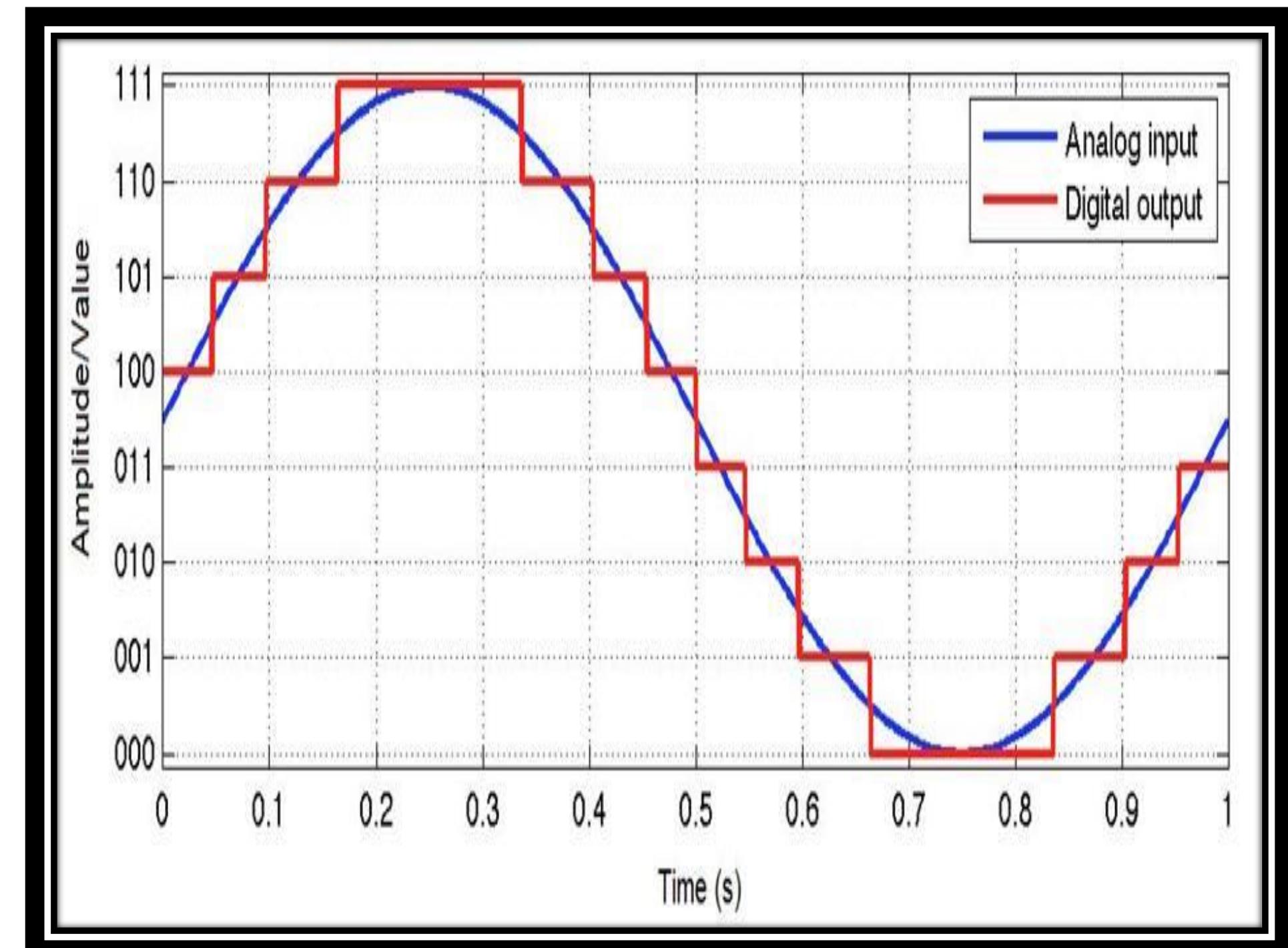
### Exemplos:

Dados em um computador, mensagens de texto, imagens digitais.

### Representação Gráfica:

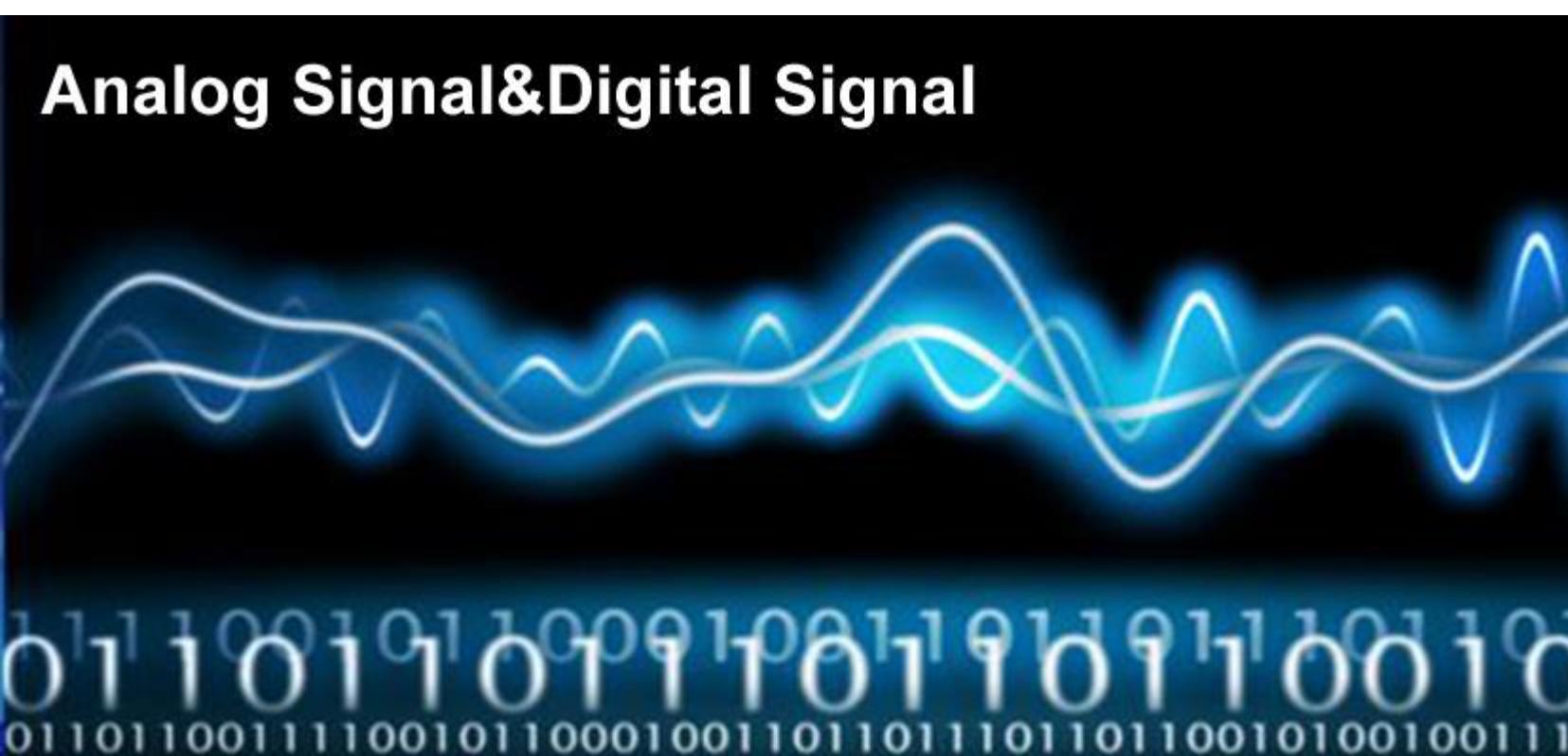
Uma onda senoidal discretizada.

Veja o seu gráfico Eixo X (tempo) vs. Eixo Y (amplitude).



## Comparação entre Sinais Analógicos e Digitais

Característica	Sinal Analógico	Sinal Digital
Natureza	Contínuo	Discreto
Valores Possíveis	Infinitos	Finitos (Dependente da resolução)
Exemplos	Som, temperatura, pressão, tensão	Dados em computadores, smartphones
Ruídos	Mais suscetível a ruídos	Menos suscetível a ruídos
Processamento	Mais complexo	Mais simples e robusto



## Por que Converter Sinais Analógicos para Digitais?

### Sinais Analógicos no Mundo Real:

- A maioria dos fenômenos naturais é analógica (som, luz, temperatura, etc.).
- Sensores capturam esses fenômenos e os transformam em sinais elétricos analógicos.

### Limitações dos Sinais Analógicos:

- Suscetibilidade a ruídos e interferências.
- Dificuldade de armazenamento e transmissão sem perda de qualidade.
- Complexidade no processamento e manipulação.

### Vantagens dos Sinais Digitais:

- Robustez contra ruídos (sinais digitais podem ser corrigidos usando técnicas de correção de erro).
- Facilidade de armazenamento (em dispositivos de memória como HDs, SSDs, etc.).
- Transmissão eficiente (através de redes digitais como internet, Wi-Fi, etc.).
- Processamento simples e preciso (usando computadores e microcontroladores).

## Aplicações Práticas da Conversão Analógico-Digital

### Áudio e Vídeo:

- Gravação e reprodução de música (ex.: CDs, streaming de áudio).
- Câmeras digitais e vídeos (ex.: fotos, filmes digitais).

### Telecomunicações:

- Transmissão de voz e dados em redes celulares e internet.

### Medicina:

- Eletrocardiogramas (ECG), monitores de pressão arterial, imagens médicas (ressonância magnética, ultrassom).

### Controle Industrial:

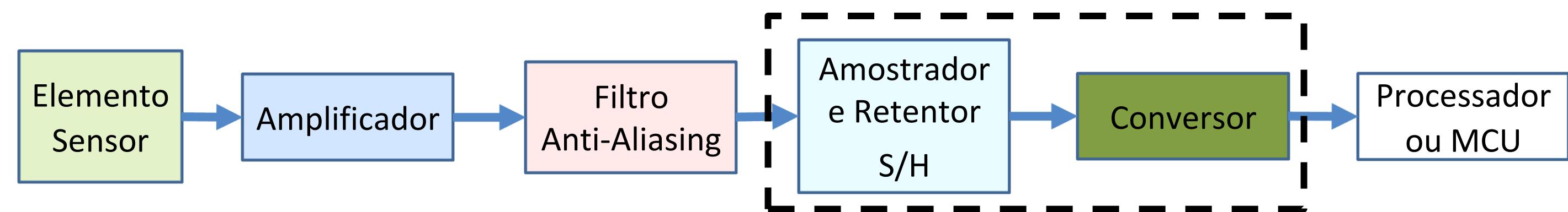
- Sensores de temperatura, pressão, umidade, etc.

### Eletrônica de Consumo:

- Smartphones, tablets, TVs digitais.

## Visão Geral do Sistema de Conversão

Um sistema de conversão analógico-digital (ADC) é composto por vários estágios, cada um com uma função específica. Esses estágios garantem que o sinal analógico seja convertido em digital de forma precisa e eficiente.



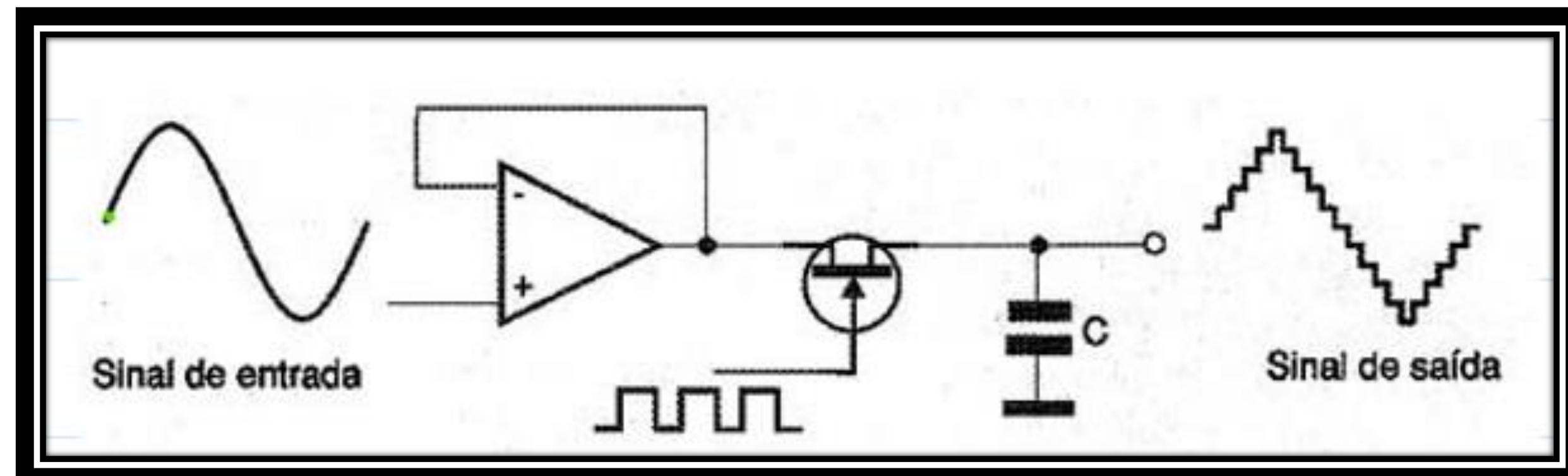
## Sample and Hold

### Amostragem (Sample):

O circuito mede o valor da tensão analógica em um instante específico. Essa amostragem deve ocorrer rapidamente para evitar distorções causadas por variações no sinal.

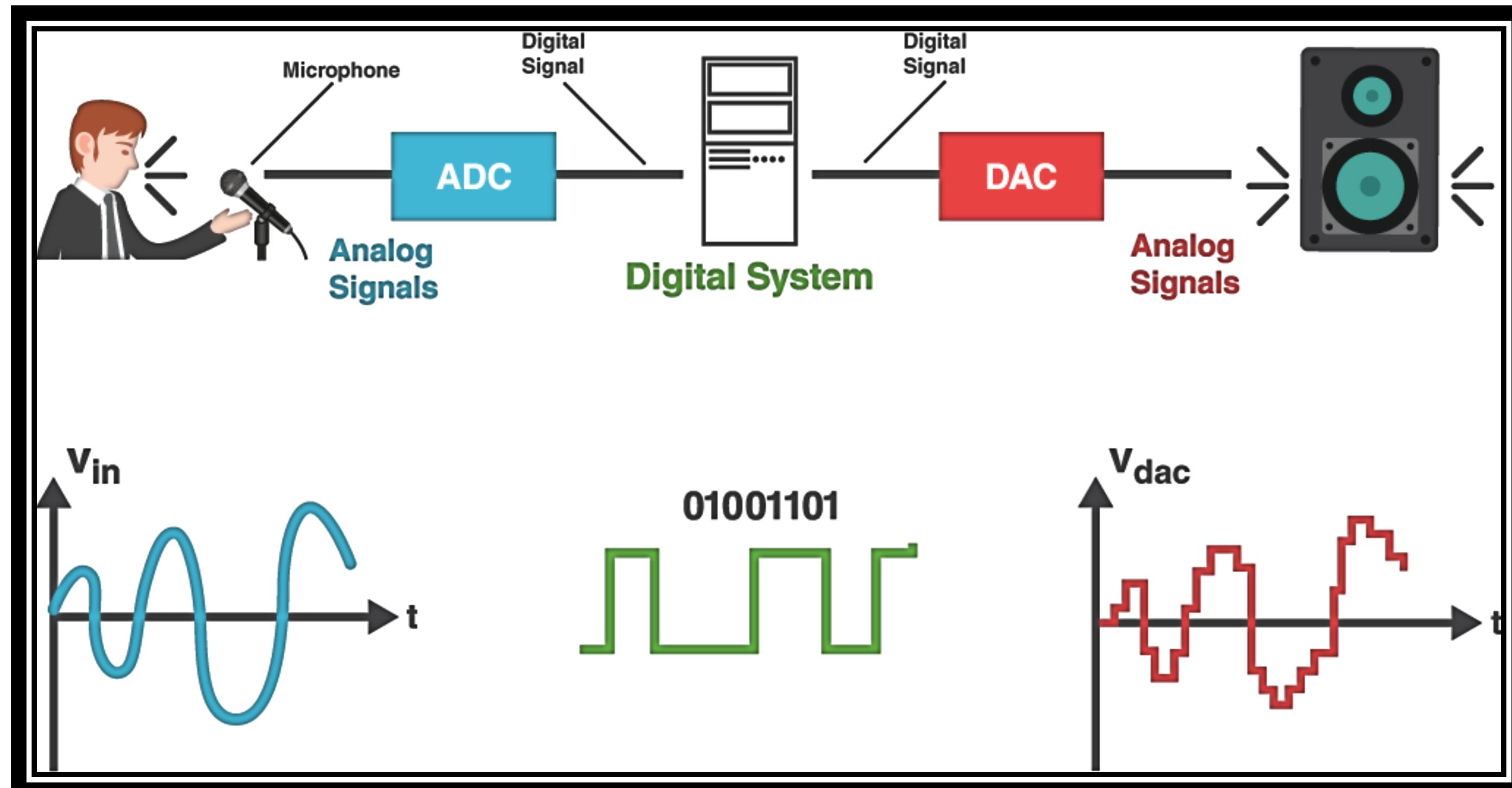
### Retenção (Hold):

O valor amostrado é mantido constante enquanto o ADC realiza a conversão para digital. Isso é importante porque o processo de conversão leva um certo tempo e, sem a retenção, o sinal poderia mudar antes da conclusão da conversão.



# ADC Teoria

Conversão A/D, processamento e conversão D/A.



## Teorema da Amostragem de Nyquist-Shannon

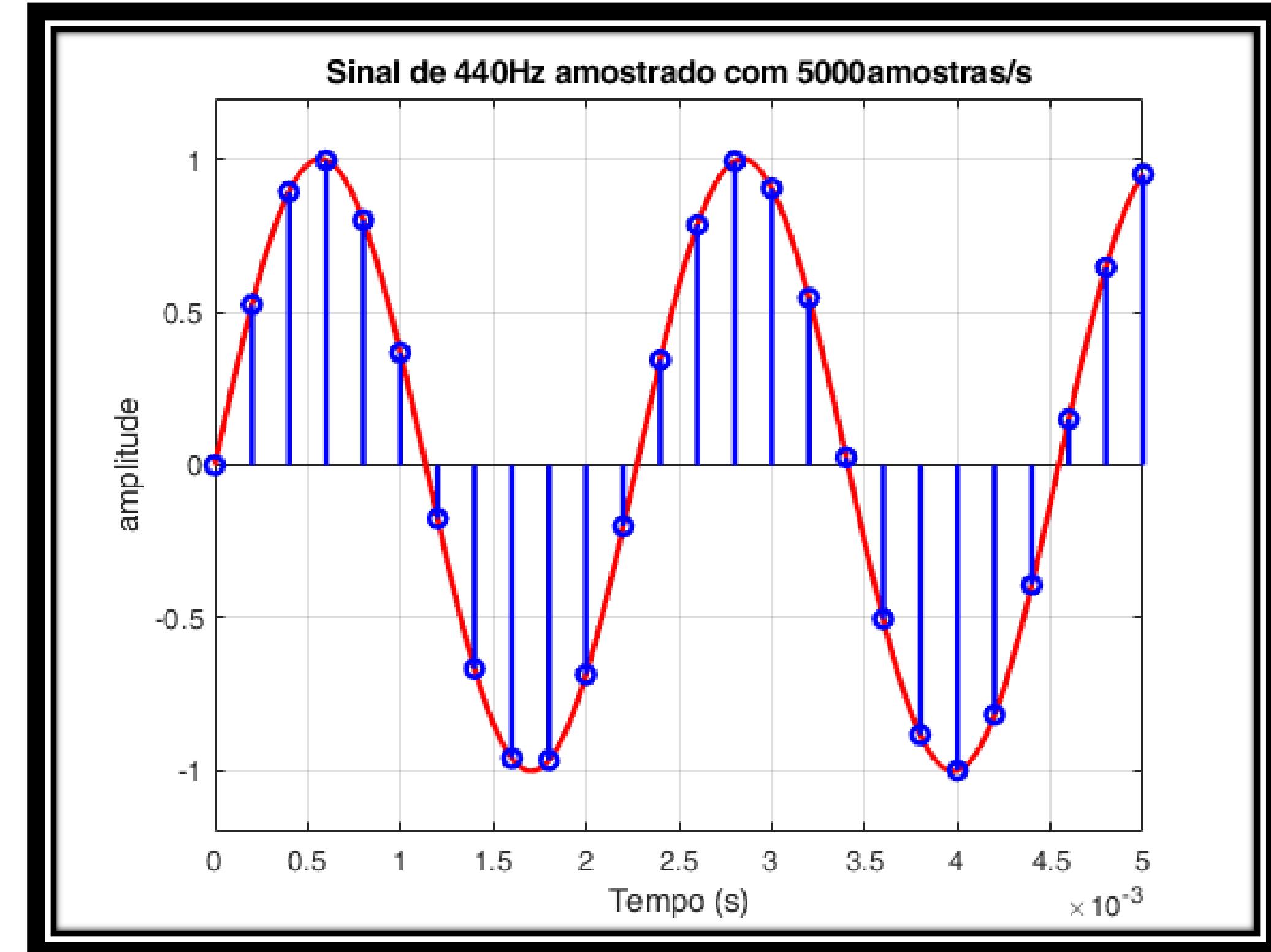
O Teorema da Amostragem de **Nyquist-Shannon** estabelece que, para reconstruir um sinal analógico a partir de suas amostras, a taxa de amostragem deve ser pelo menos o dobro da maior frequência presente no sinal.

• **Fórmula:**  $fs \geq 2f_{max}$  , onde:

- $fs$ : Taxa de amostragem.
- $f_{max}$ : Maior frequência presente no sinal.

• **Frequência de Nyquist:** Metade da taxa de amostragem ( $fs/2$ ).

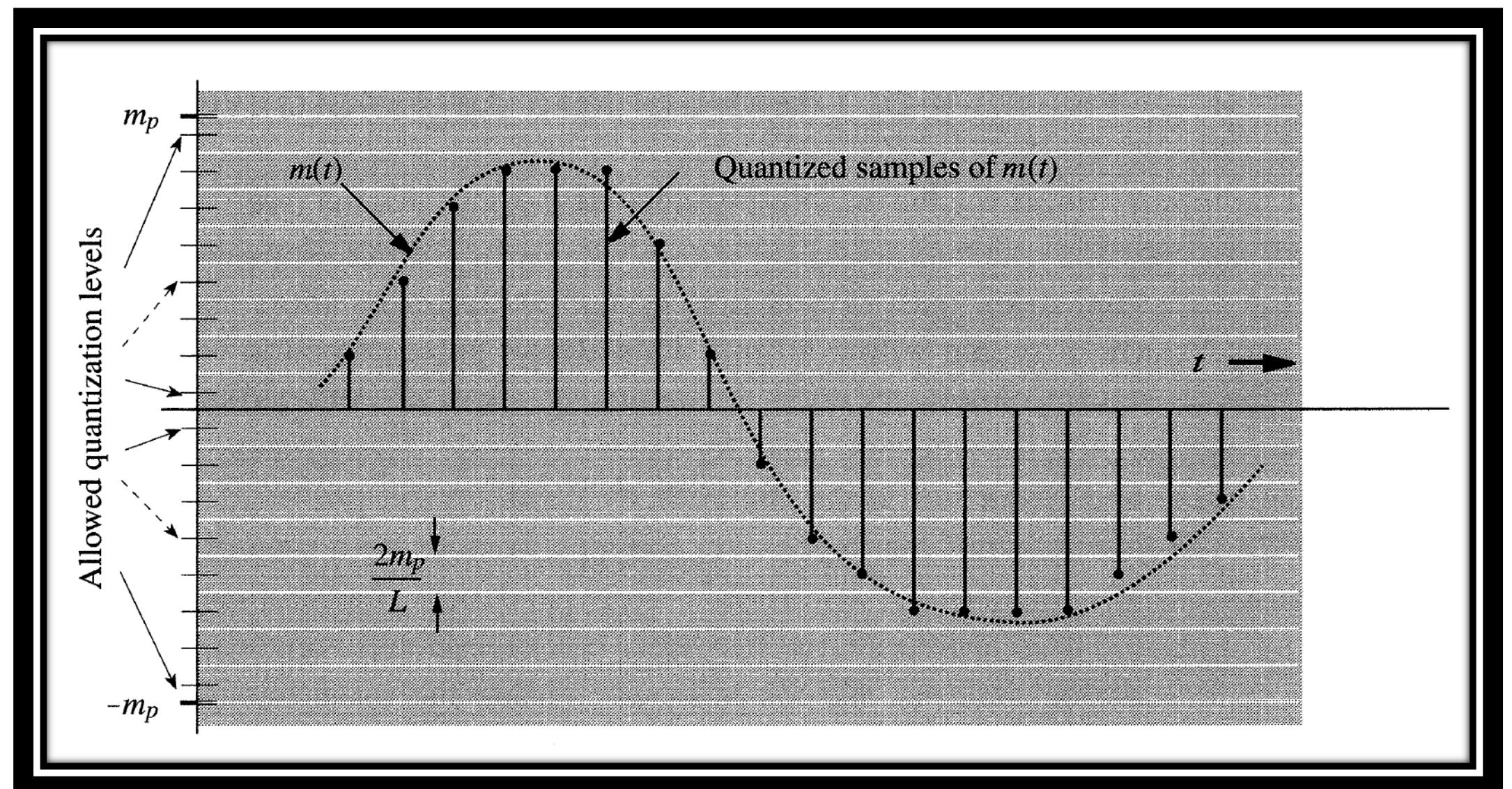
## Teorema da Amostragem de Nyquist-Shannon



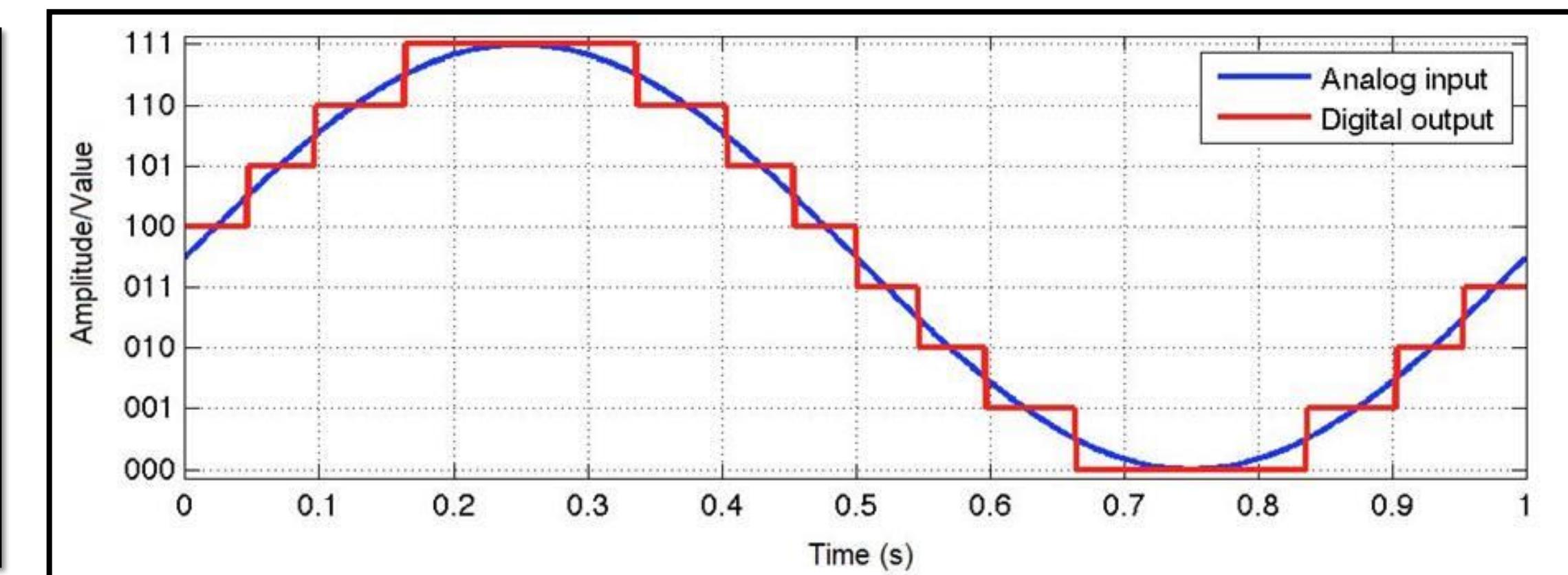
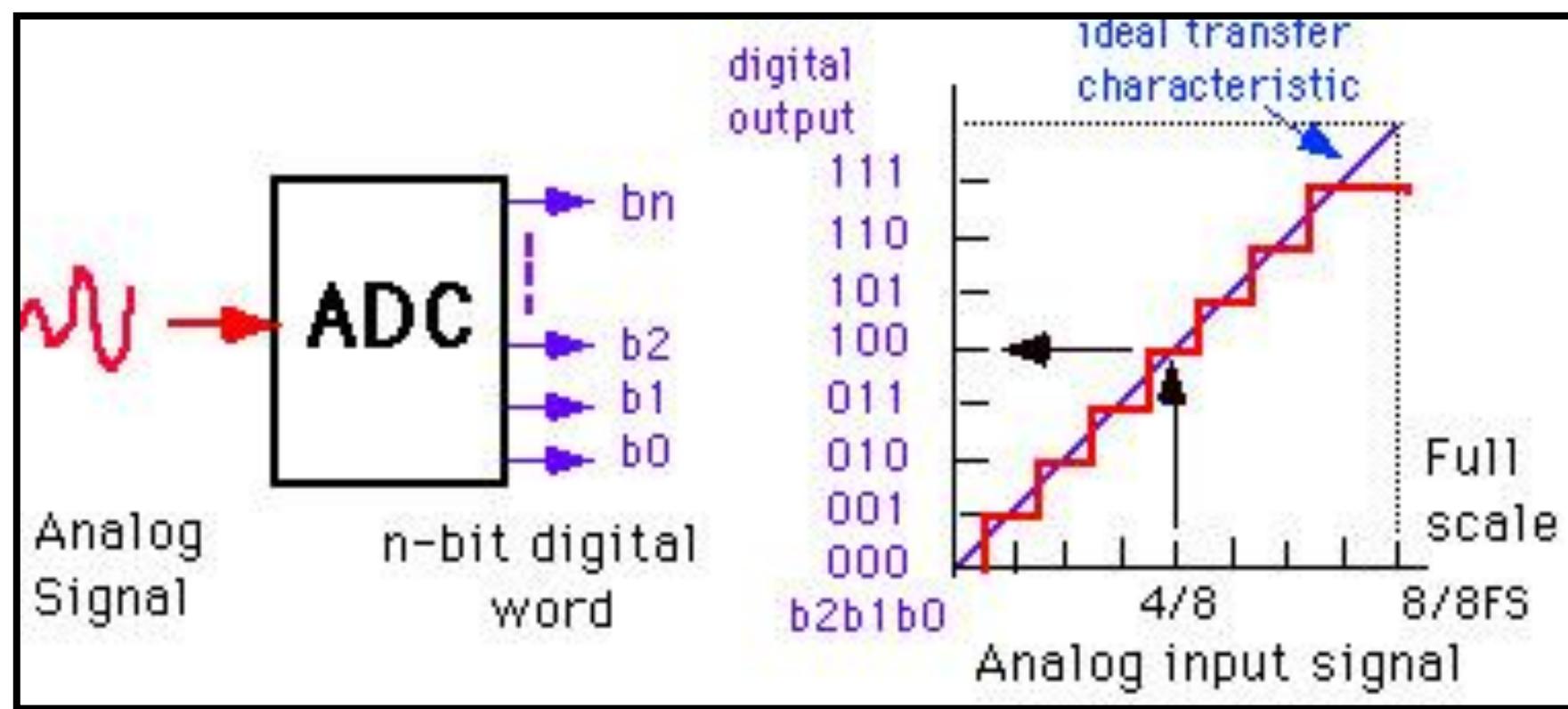
## Quantização

A quantização é a etapa em que o sinal analógico, após ser amostrado, é mapeado para um número finito de níveis de amplitude. Esses níveis são determinados pela resolução do conversor analógico-digital (ADC), que é especificada em bits.

Por exemplo, um ADC de 8 bits pode representar 256 níveis de amplitude distintos.



## Resolução do ADC



Conversor de 3bits

## Resolução do ADC

A resolução de um ADC refere-se ao número de bits usados para representar o sinal digital. Quanto maior a resolução, mais preciso é o sinal digital.

Fórmula da Resolução:

$$\text{Resolução} = \frac{V_{\text{ref}}}{2^n}$$

- Vref: Tensão de referência do ADC. Ex. RP2: 3,3V; Arduino Uno: 5,0V
- n: Número de bits do ADC. Ex. RP2 12bits; Arduino Uno 10bits.

Ex.  $\text{Resolução} = \frac{V_{\text{ref}}}{2^n} = \frac{3,3V}{2^{12}} = \frac{3,3V}{4096} = 805,6 \times 10^{-6} V/bit$

## Menor Valor Detectável (LSB)

O LSB (Least Significant Bit) é a menor mudança que o ADC pode detectar no sinal analógico.

### Fórmula do LSB:

$$\text{LSB} = \frac{V_{\text{ref}}}{2^n}$$

### Exemplo:

- Para um ADC de 8 bits com  $V_{\text{ref}} = 5 \text{ V}$ :

$$\text{LSB} = \frac{5 \text{ V}}{256} \approx 19,53 \text{ mV}$$

- ADC de 12 bits com  $V_{\text{ref}} = 5 \text{ V}$ :

$$\text{LSB} = \frac{5 \text{ V}}{4096} \approx 1,22 \text{ mV}$$

## Menor Valor Detectável (LSB)

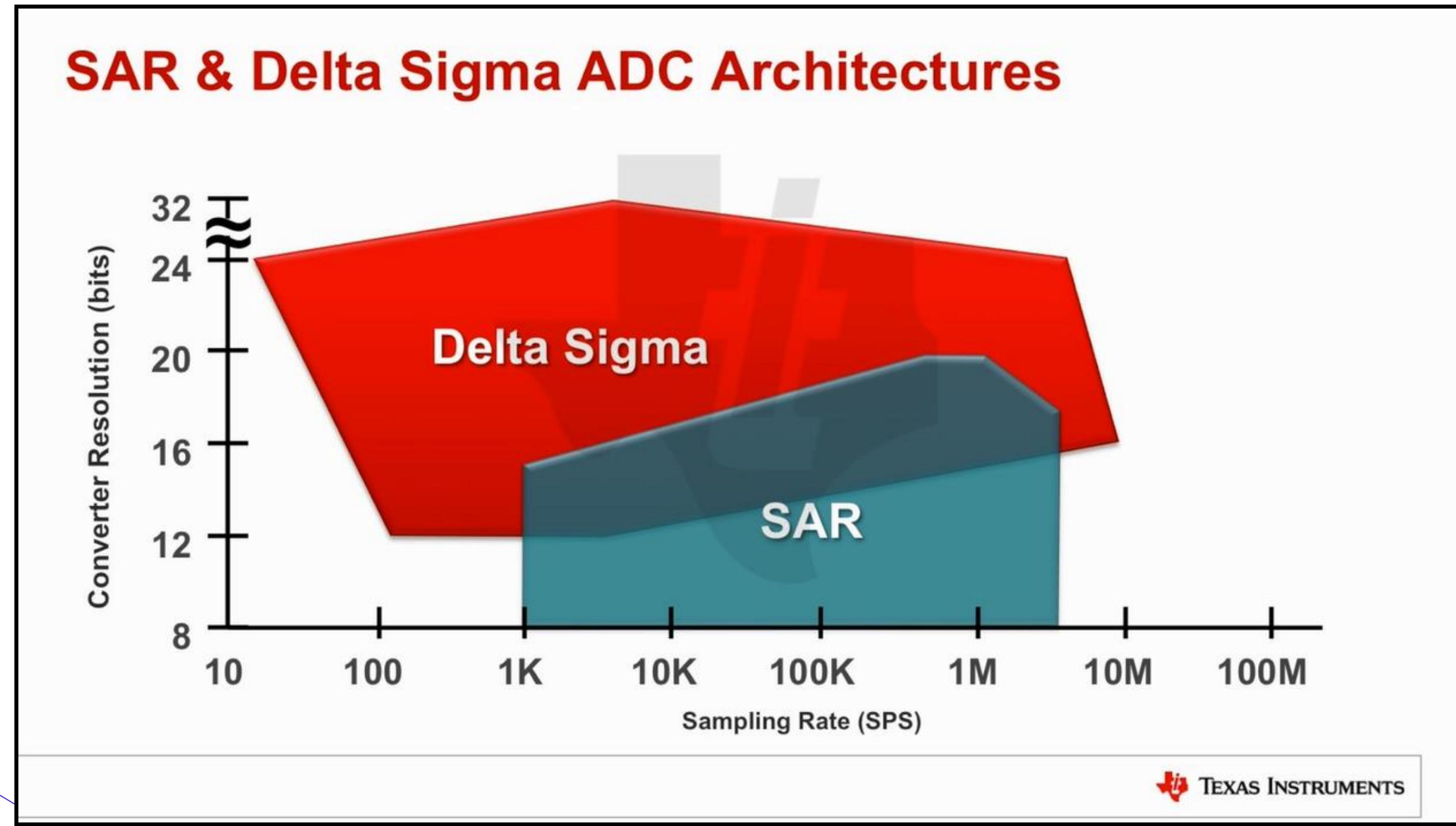
Número de bits	Níveis	Largura do passo (3,3V)
8-bits	256	12,89 mV
10-bits	1024	3,22 mV
12-bits	4096	805 uV
16-bits	65536	50,3 uV
18-bits	262144	12,6 uV
20-bits	1048576	3,15 uV
24-bits	16777216	197 nV

## Tipos de Conversores Analógico-Digital

Existem várias arquiteturas de ADCs, cada uma com características específicas que as tornam adequadas para determinadas aplicações. Podemos destacar:

- **ADC Flash.**
- **ADC de Aproximação Sucessiva (SAR).**
- **ADC Sigma-Delta ( $\Delta\Sigma$ ).**
- **ADC de Integração (Dual-Slope).**
- **ADC Pipeline.**

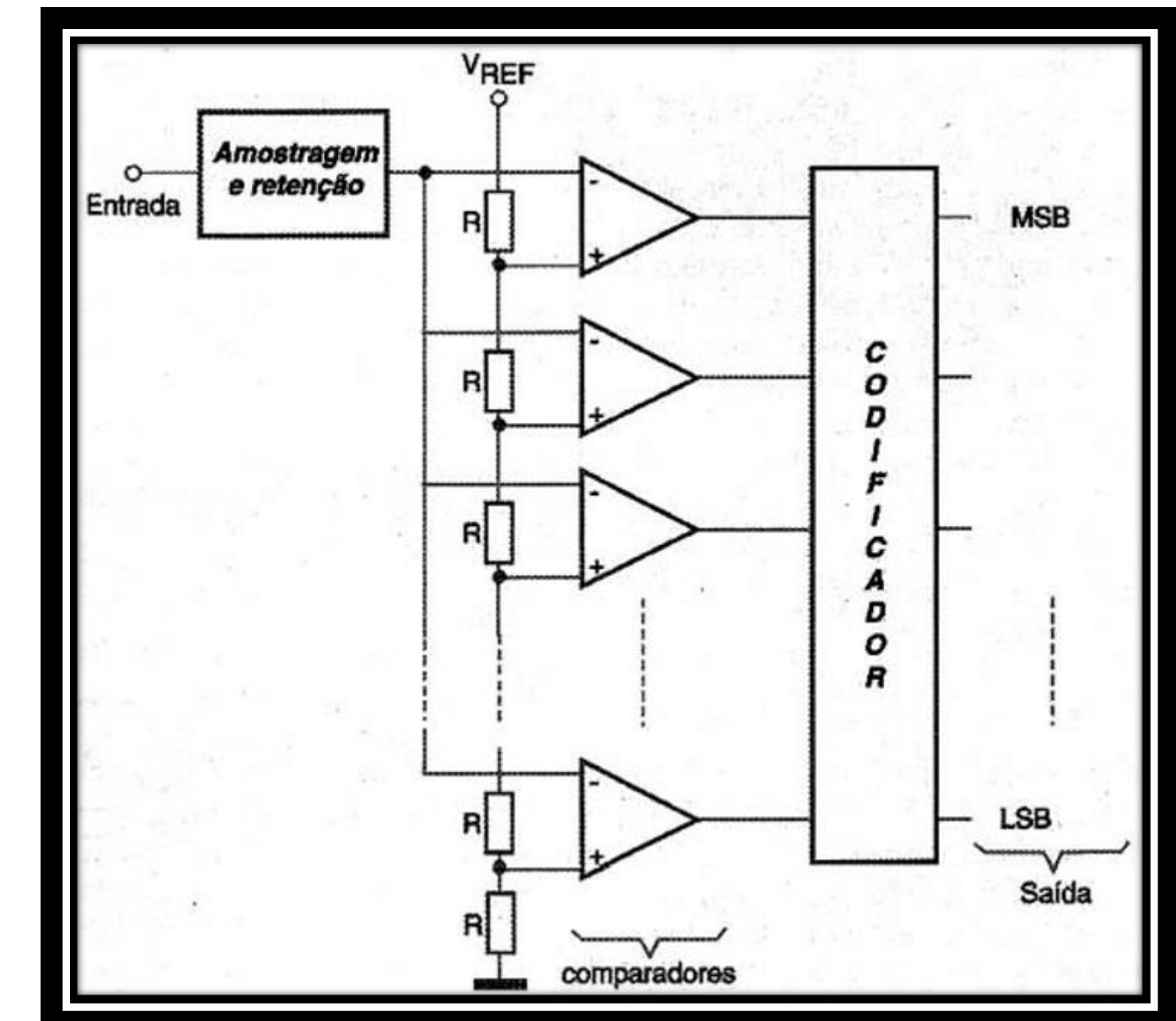
## Comparação entre tecnologias mais utilizadas atualmente



## ADC Flash

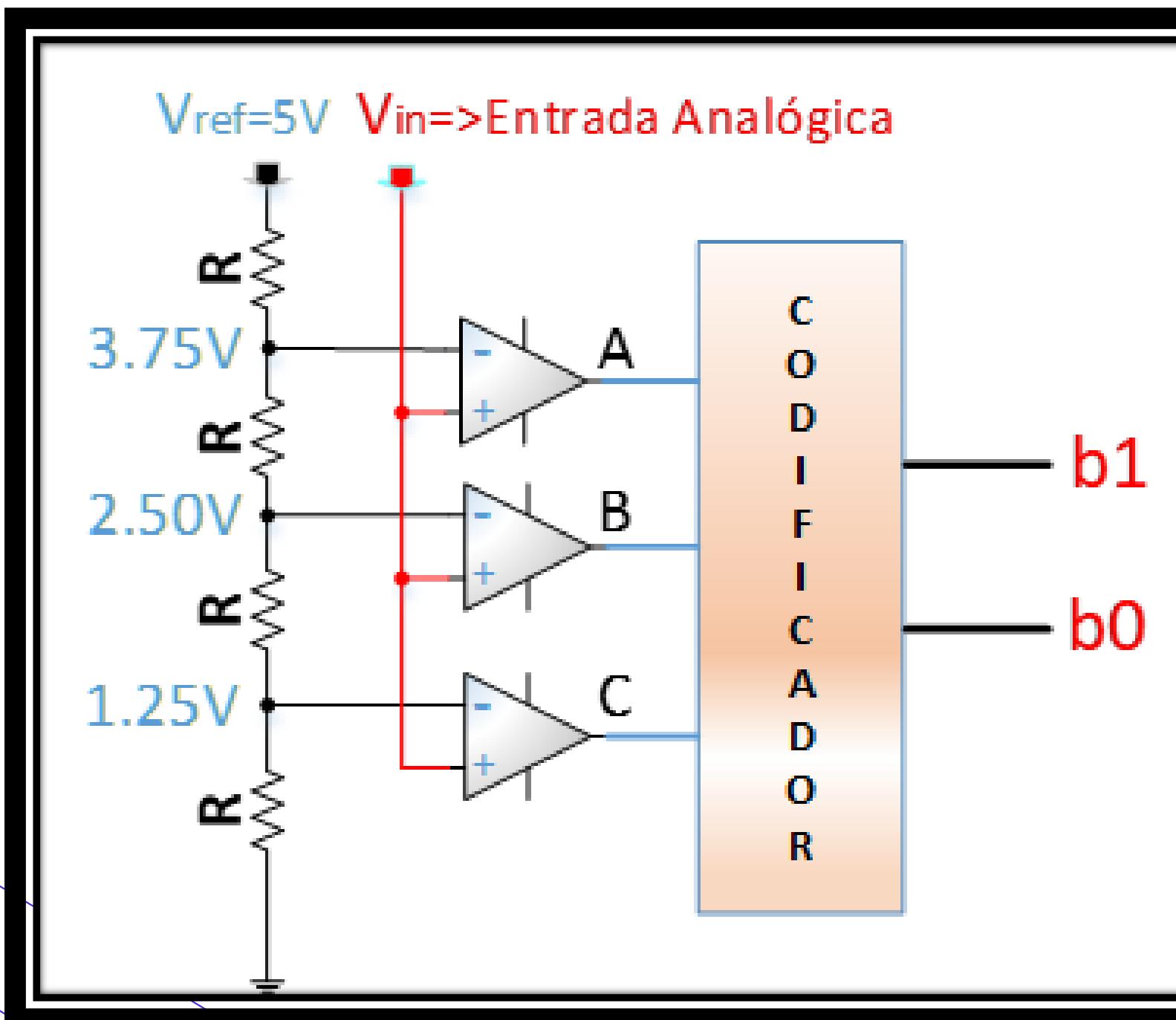
O conversor ADC tipo Flash (também conhecido como conversor analógico-digital paralelo) é um dos tipos mais rápidos de conversores analógico-digital (ADC) disponíveis.

Ele é amplamente utilizado em aplicações que exigem alta velocidade de conversão, como em sistemas de comunicação, processamento de sinais em tempo real e instrumentação de alta frequência.



# ADC Teoria

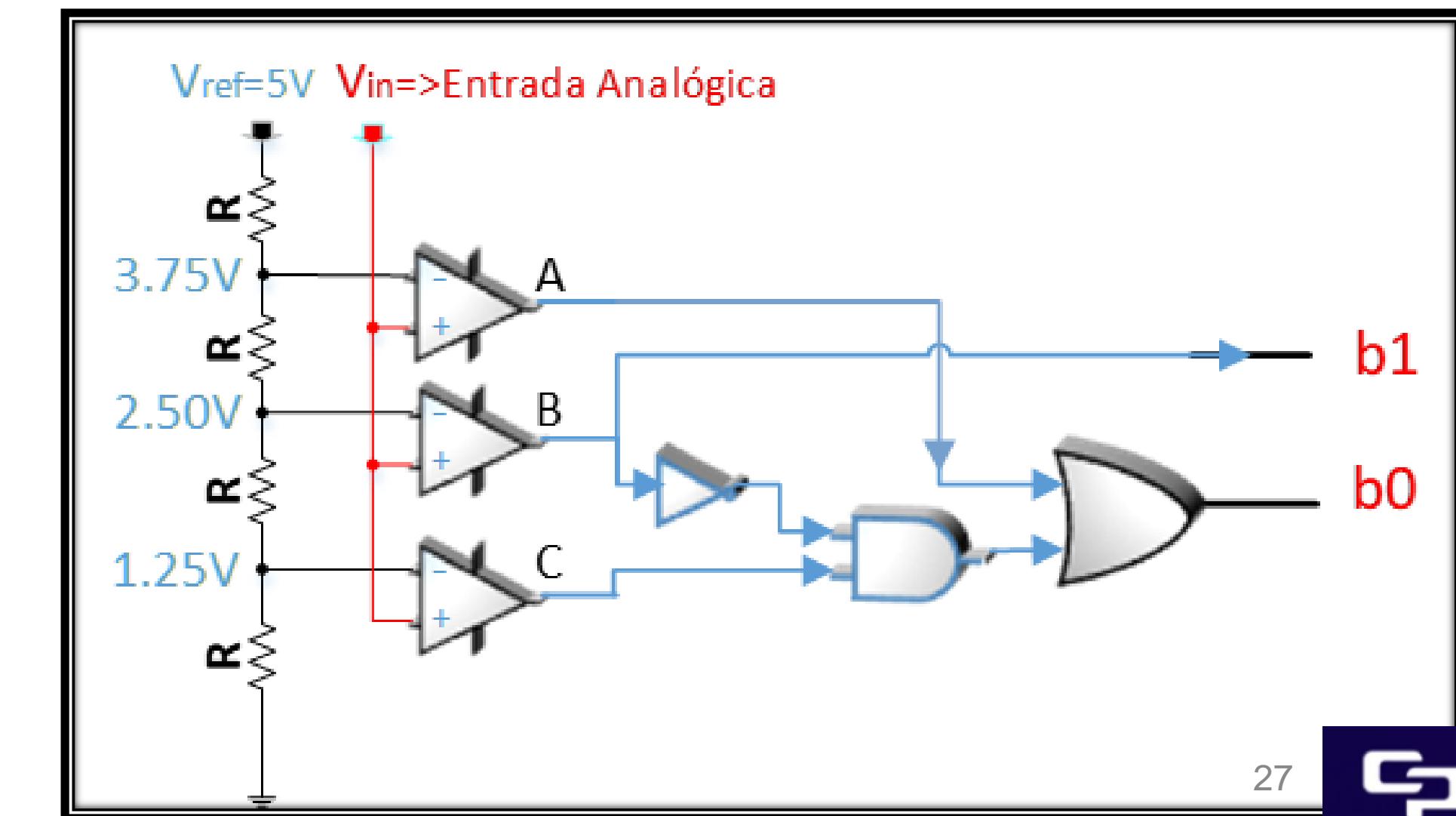
## ADC Flash



A	B	C	b1	b0	Significado
0	0	0	0	0	$0 < V_{in} < 1.25V$
0	0	1	0	1	$1.25 < V_{in} < 2.50V$
0	1	1	1	0	$2.50 < V_{in} < 3.75V$
1	1	1	1	1	$V_{in} > 3.75V$

b0	B	B	$\overline{B}$	$\overline{B}$
A	x	1	x	x
$\overline{A}$	x	0	1	0
$\overline{C}$	C	C	$\overline{C}$	$\overline{C}$

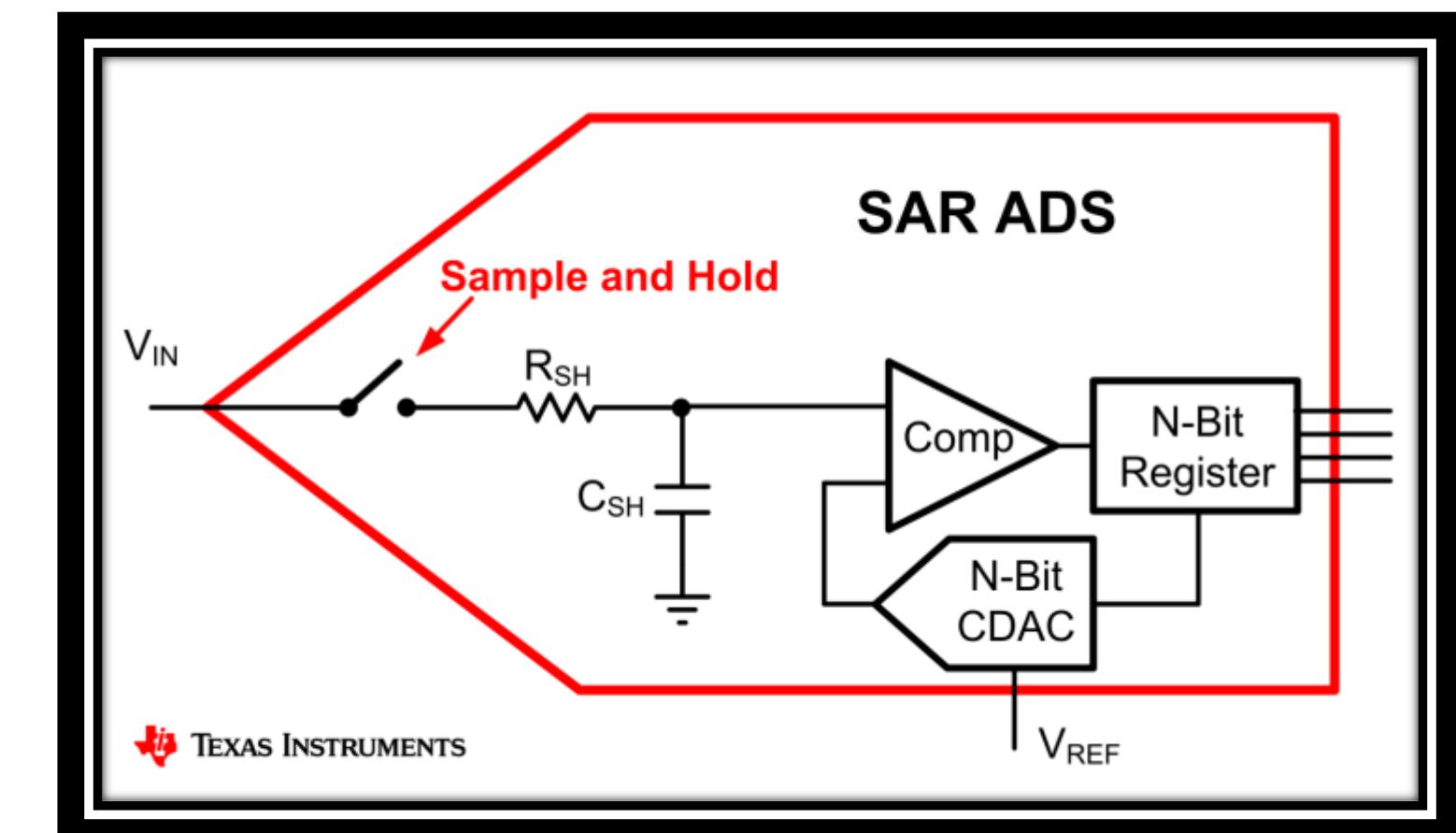
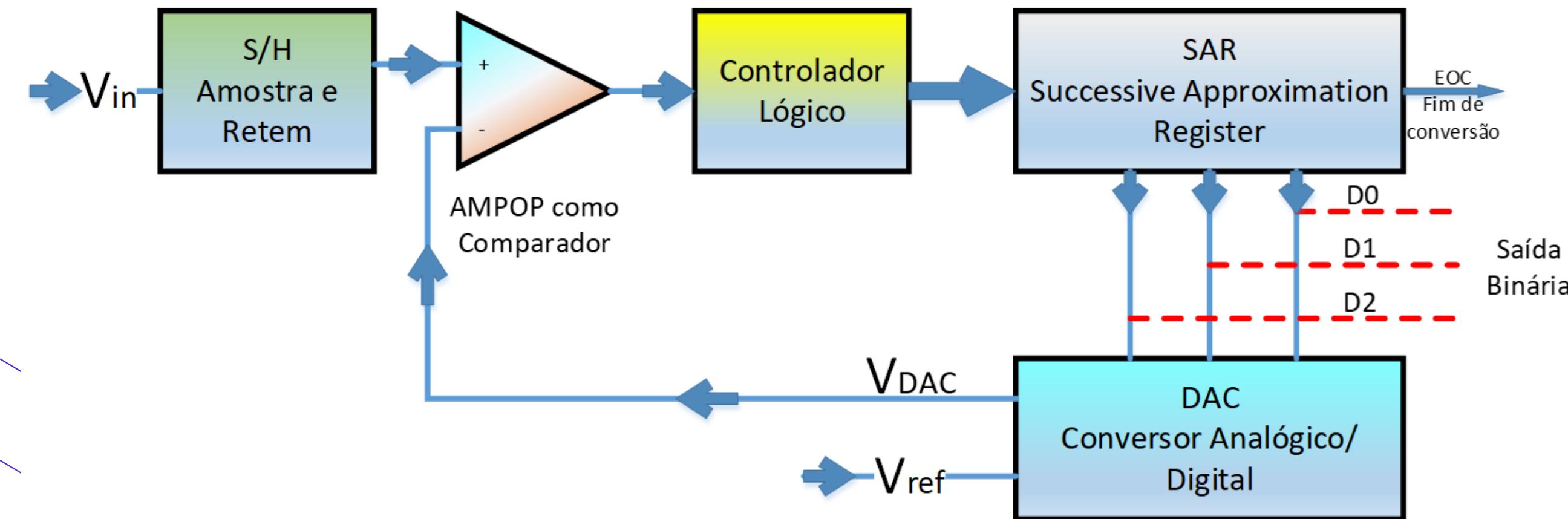
Portanto:  $b0 = A + \overline{B}C$  e  $b1 = B$



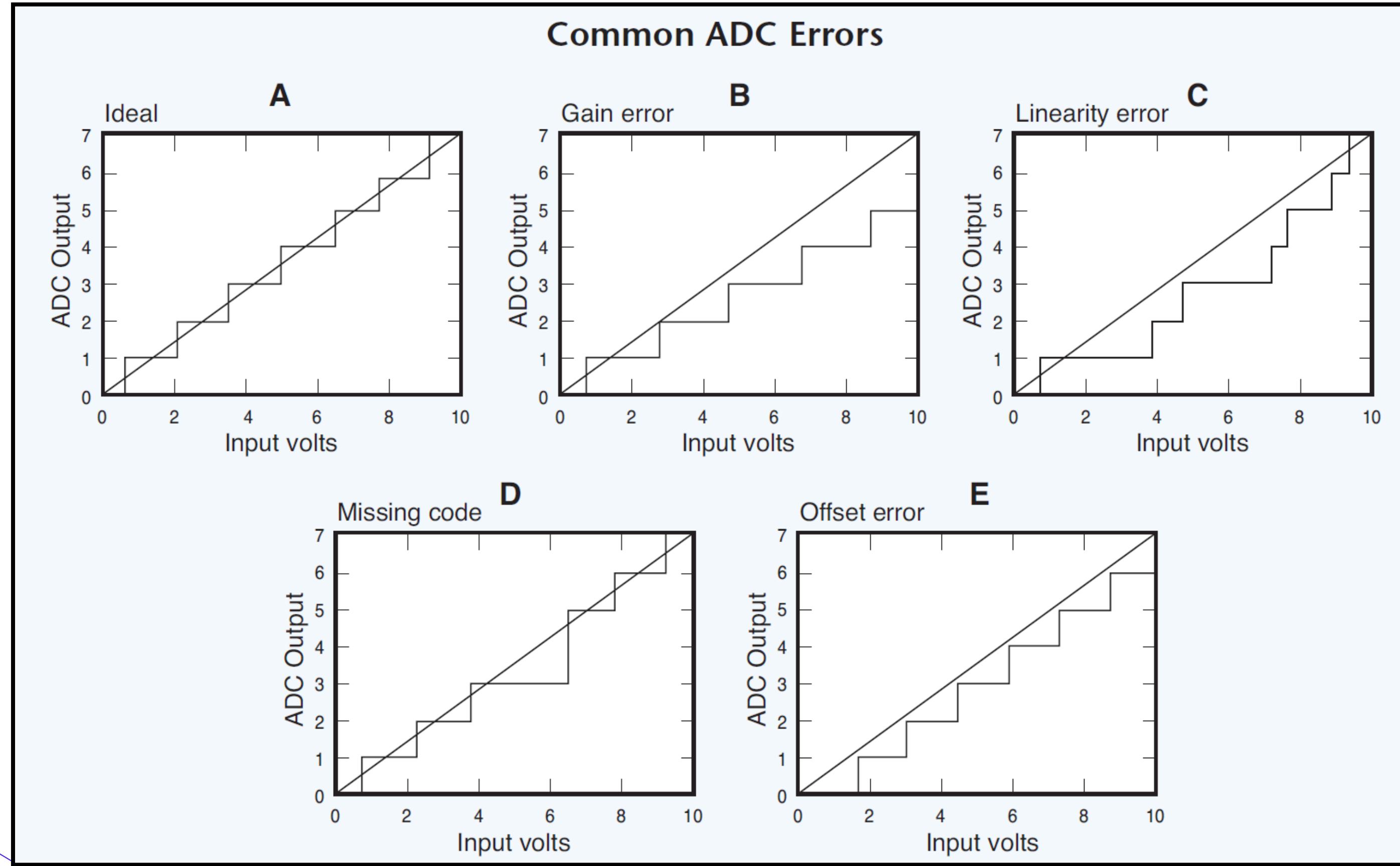
## ADC de Aproximação Sucessiva - SAR

O **SAR** (Successive Approximation Register) é um tipo de conversor analógico-digital (ADC) amplamente utilizado devido ao seu equilíbrio entre velocidade, resolução e consumo de energia.

O **SAR ADC** opera usando um método de aproximação sucessiva para determinar o valor digital correspondente ao sinal analógico de entrada.



## Erros comuns nos ADCs



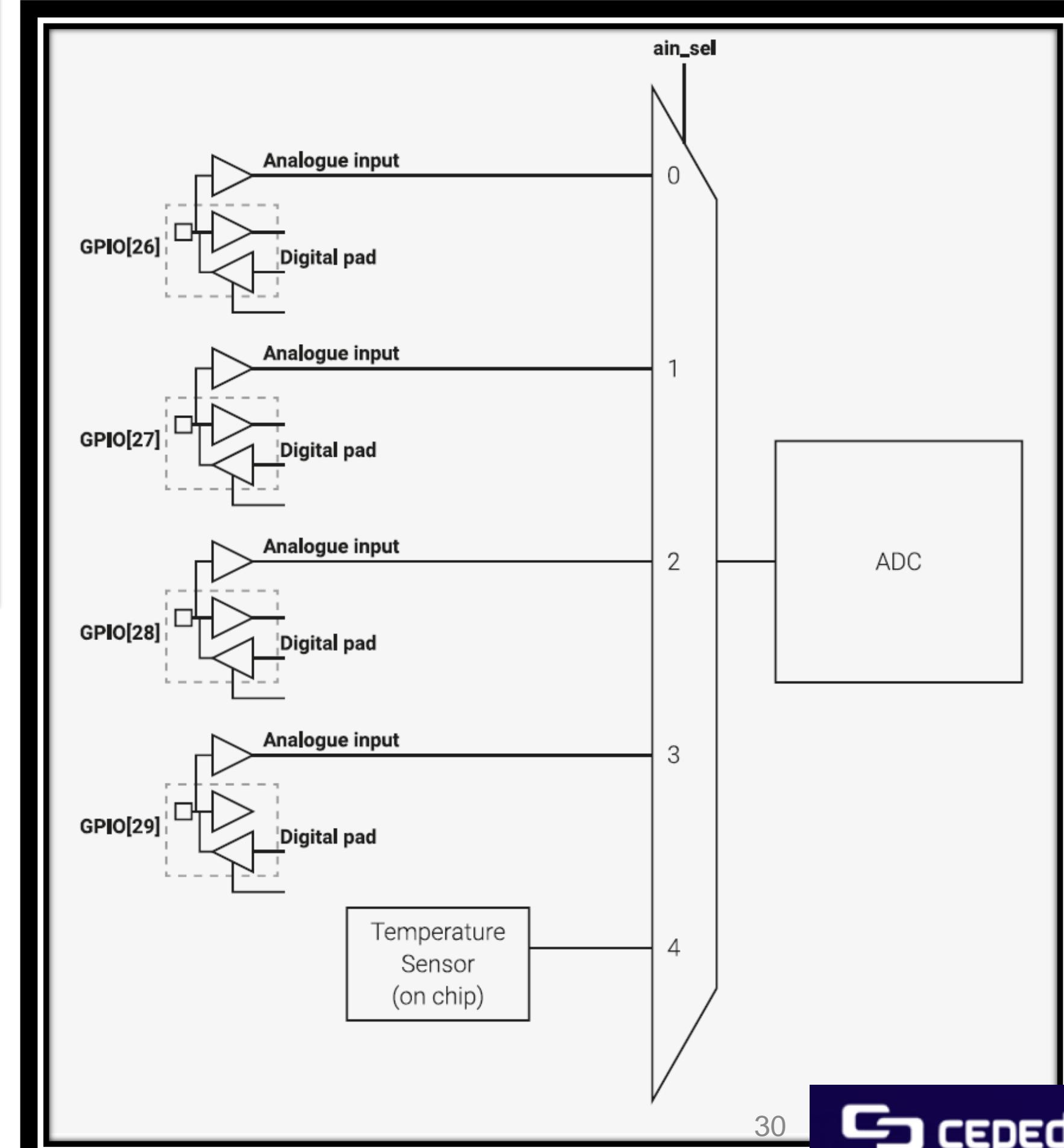
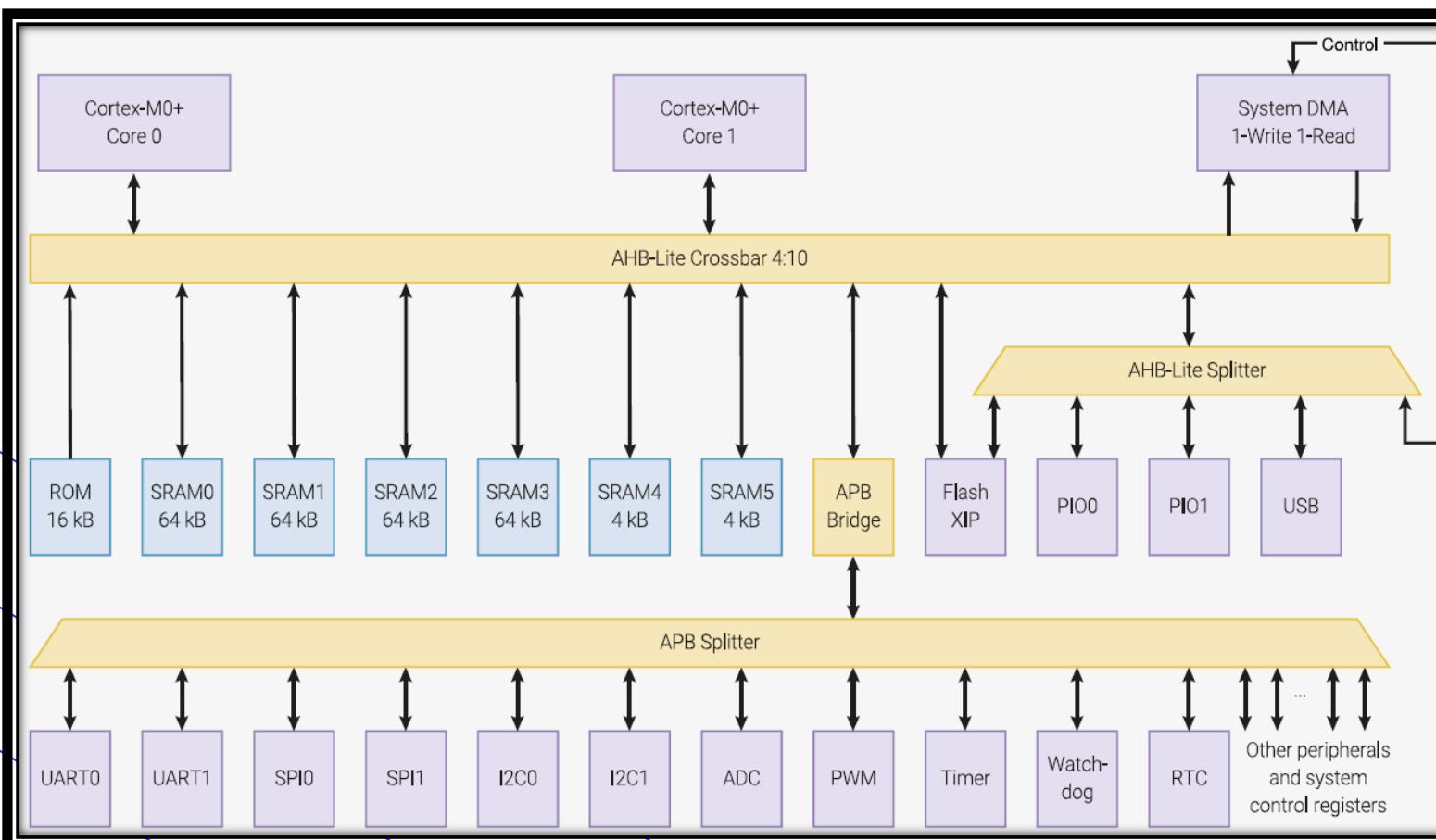
# ADC no RP2040

## 4.9. ADC and Temperature Sensor

(informação completa no datasheet)

O RP2040 conta com um conversor analogico-digital internamente, com as seguintes características:

- SAR ADC (Seção 4.9.2 do datasheet)
- 500ksps (usando um clock independente de 48MHz)
- 12-bit com uma ENOB de 8.7 bits (Seção 4.9.3)\*
- Conta com um multiplexador de cinco entradas, onde:
  - Quatro entradas estão disponíveis nos pinos do chip GPIO[26:29]
  - Uma entrada é dedicada para um sensor interno de temperatura (seção 4.9.5)
- Geração de interrupção
- Interface DMA (Seção 4.9.2.5)
- A referência de tensão do ADC é a alimentação de 3,3V do chip (VREF = VDD)



# ADC no RP2040 (Observações)

## Sobre o ENOB de 8.7 bits:

Essa especificação significa que o conversor analógico-digital (ADC) do RP2040 possui uma resolução nominal de **12 bits**, mas sua **Efetiva Quantidade de Bits** (ENOB - Effective Number of Bits) é **8,7 bits**. Ou seja, na prática, devido a ruído, distorções e não linearidades, o ADC funciona de forma equivalente a um conversor de aproximadamente 8,7 bits reais.

## Sobre a referência do ADC\_VREFs:

Como o ADC usa a alimentação de 3,3V como referência, variações nessa tensão afetam diretamente as leituras.

## Sobre sensor de temperatura:

A leitura do sensor requer conversão usando a fórmula fornecida na documentação oficial. Normalmente é necessário calibração.

## Sobre GPIO29:

GPIO29 (usado na placa Raspberry Pi Pico para monitorar a tensão da fonte de alimentação VSYS).

## Sobre Acesso Direto à Memória (DMA):

O ADC do RP2040 pode ser utilizado com DMA, permitindo a leitura contínua de dados sem a necessidade de intervenção da CPU.

O DMA pode ser configurado para armazenar automaticamente os valores do ADC em um buffer de memória, tornando o processamento mais eficiente para aplicações como aquisição de sinais, filtros digitais e leitura rápida de sensores.

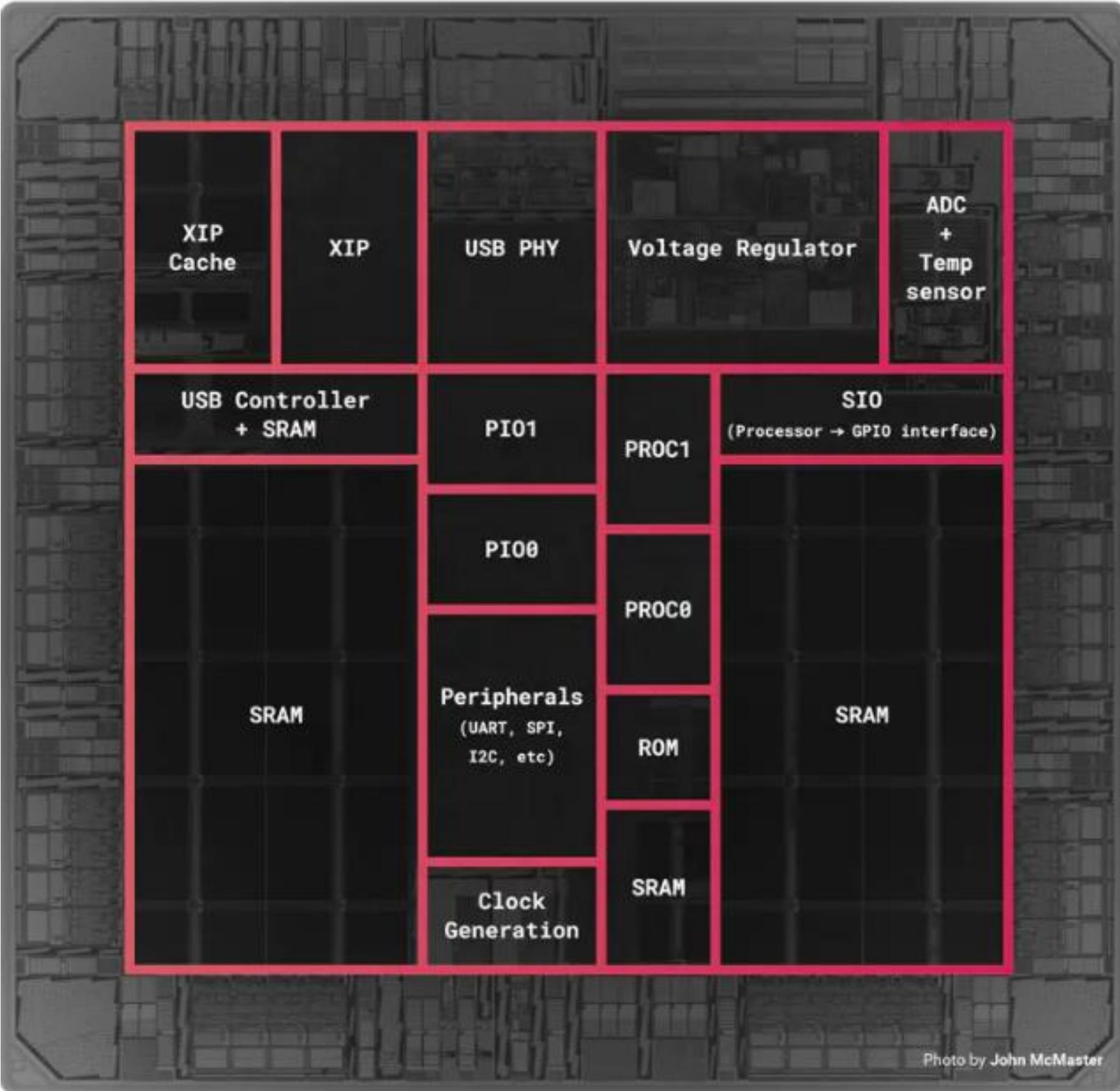


Photo by John McMaster

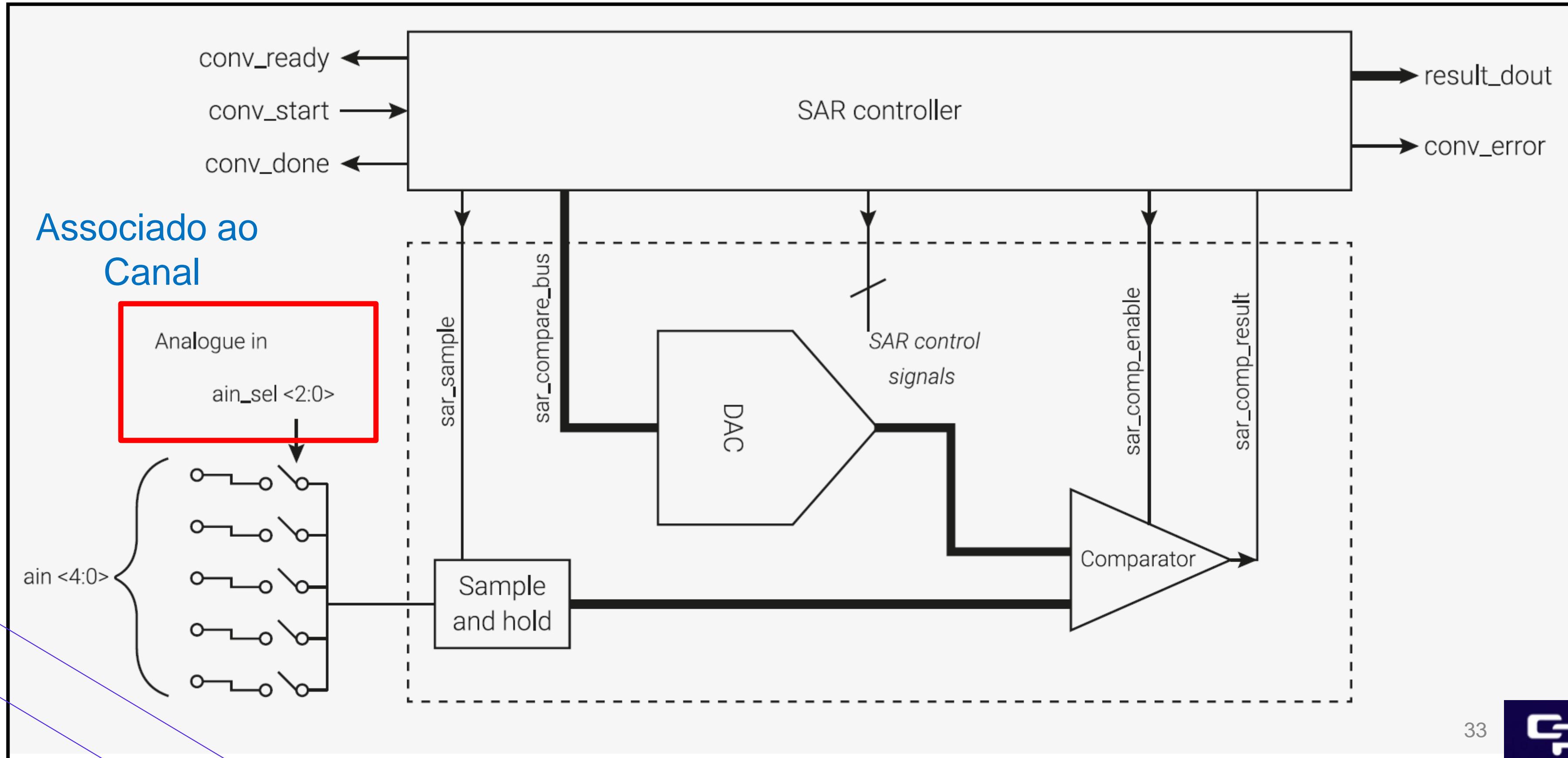
# ADC no RP2040 (Observações)

## Comparação entre os ENOBs de diversos ADCs

Microcontrolador/ADC	Resolução Nominal	ENOB Aproximado	Observações
ATmega328P (Arduino Uno)	10 bits	8.5 bits	Depende da referência de tensão e filtragem.
RP2040	12 bits	8.7 bits	Valor informado no datasheet.
ESP32	12 bits	6.2 bits	Desempenho ruim devido a não-linearidades.
ADS1115	16 bits	15 bits	ADC externo de alta precisão.

# ADC no RP2040

The ADC requires a 48MHz clock (`clk_adc`), which could come from the USB PLL. Capturing a sample takes 96 clock cycles ( $96 \times 1/48\text{MHz} = 2\mu\text{s}$  per sample (**500ksps**)). The clock must be set up correctly before enabling the ADC.



# ADC no RP2040

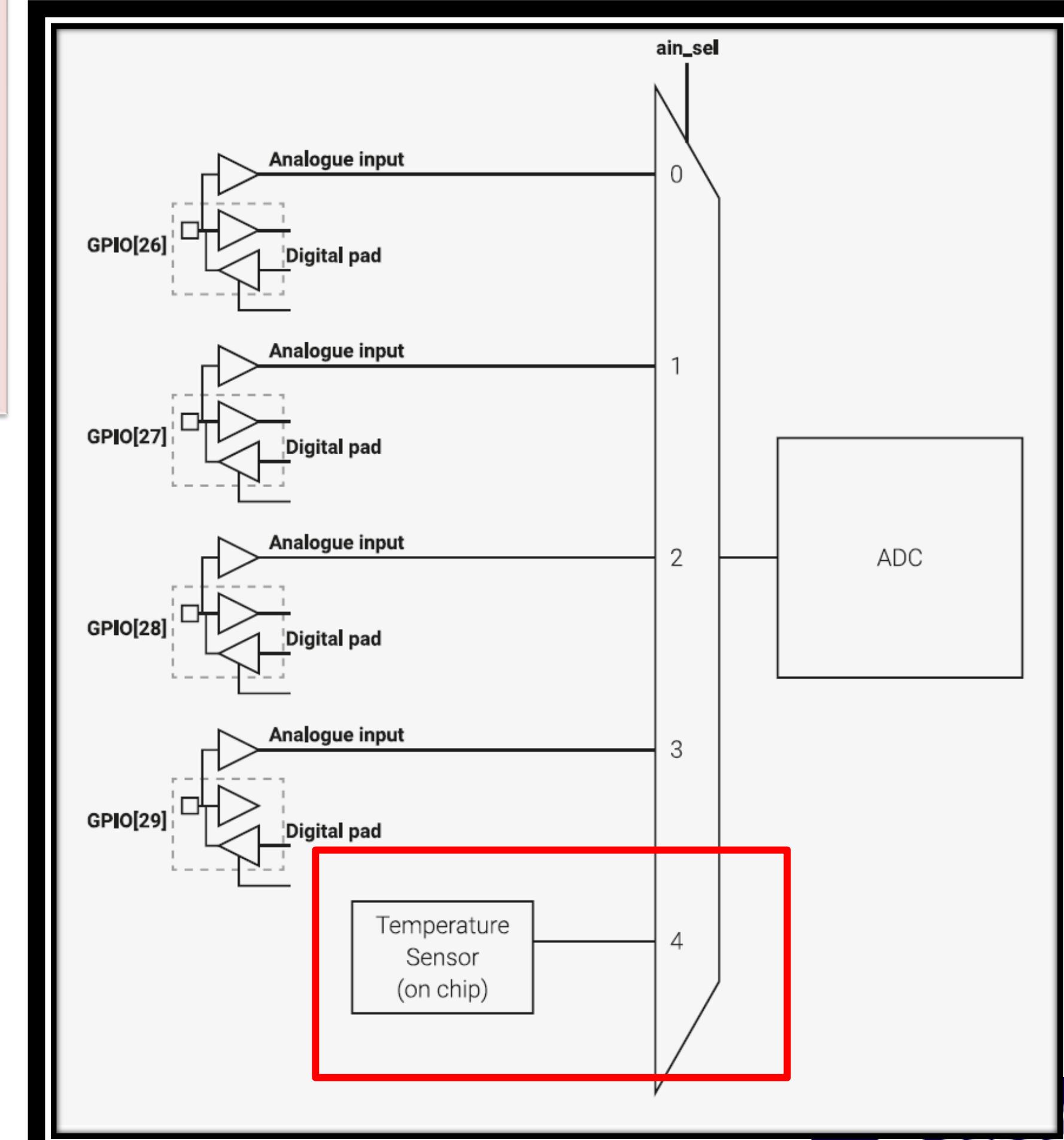
## Sensor de Temperatura

- Ele é um canal separado dentro do ADC do RP2040, acessado diretamente por software (**CANAL 4**).
- Não possui um pino externo e sua leitura ocorre via o próprio periférico ADC.
- Ele só pode ser usado para medir a temperatura interna do chip.
- Sua leitura retorna um valor proporcional à temperatura, mas precisa ser convertida para graus Celsius usando a seguinte equação:

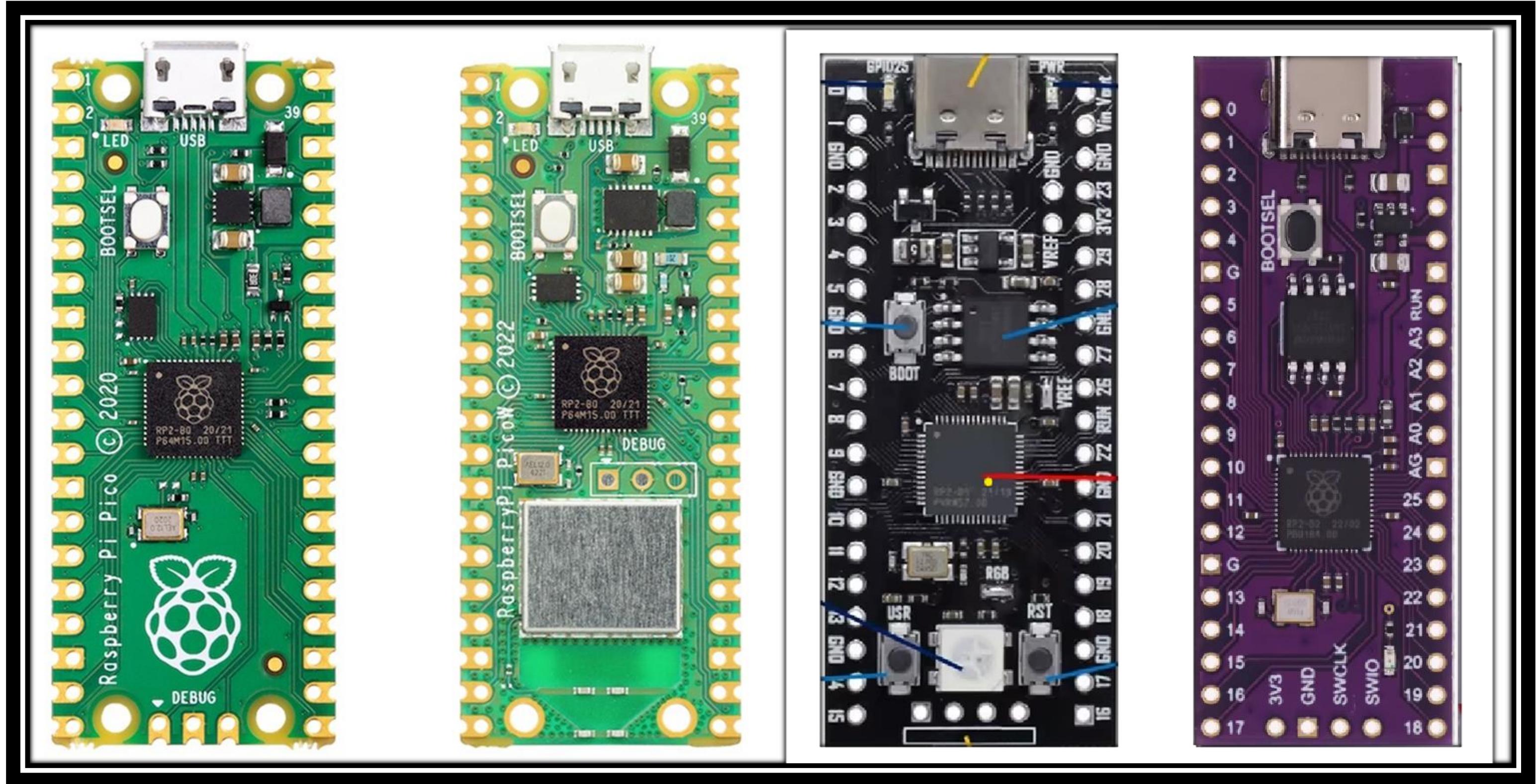
$$T = 27 - \frac{(V_{sense} - 0.706)}{0.001721}$$

Onde:

- $T$  é a temperatura em graus Celsius.
- $V_{sense}$  é a tensão lida no ADC3.
- 0.706V é a tensão típica do sensor a 27°C.
- 0.001721 V/°C é o coeficiente de variação da tensão com a temperatura.



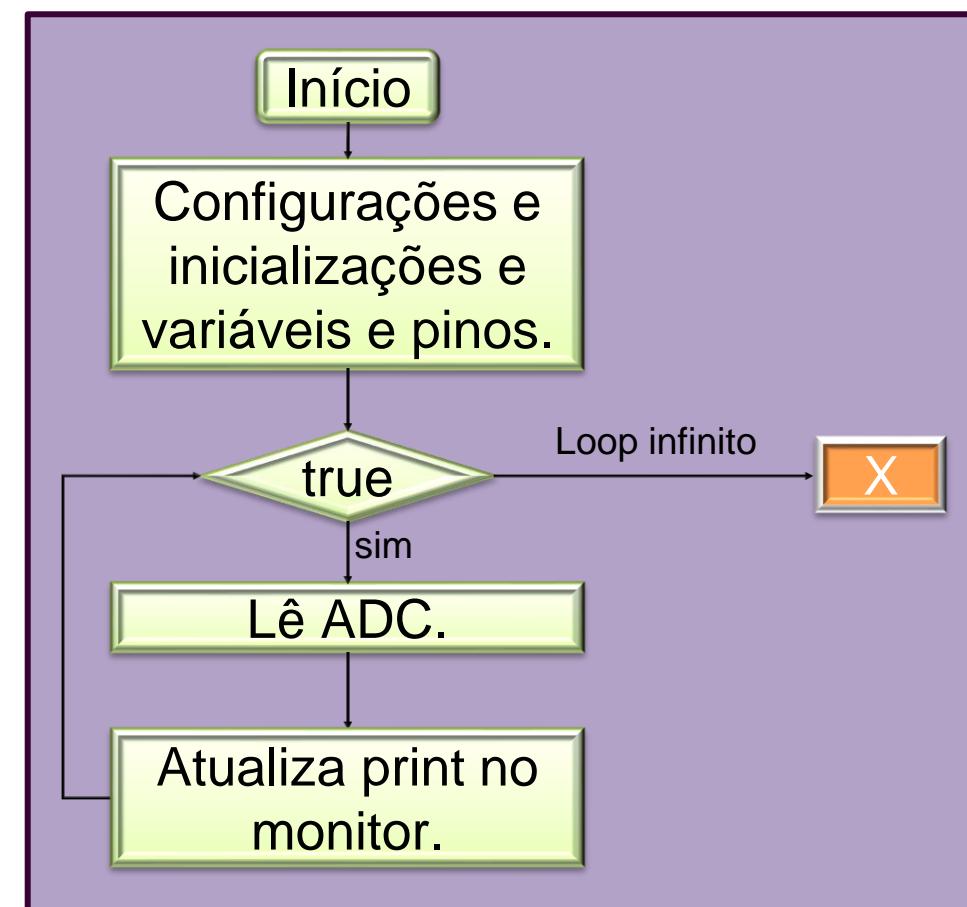
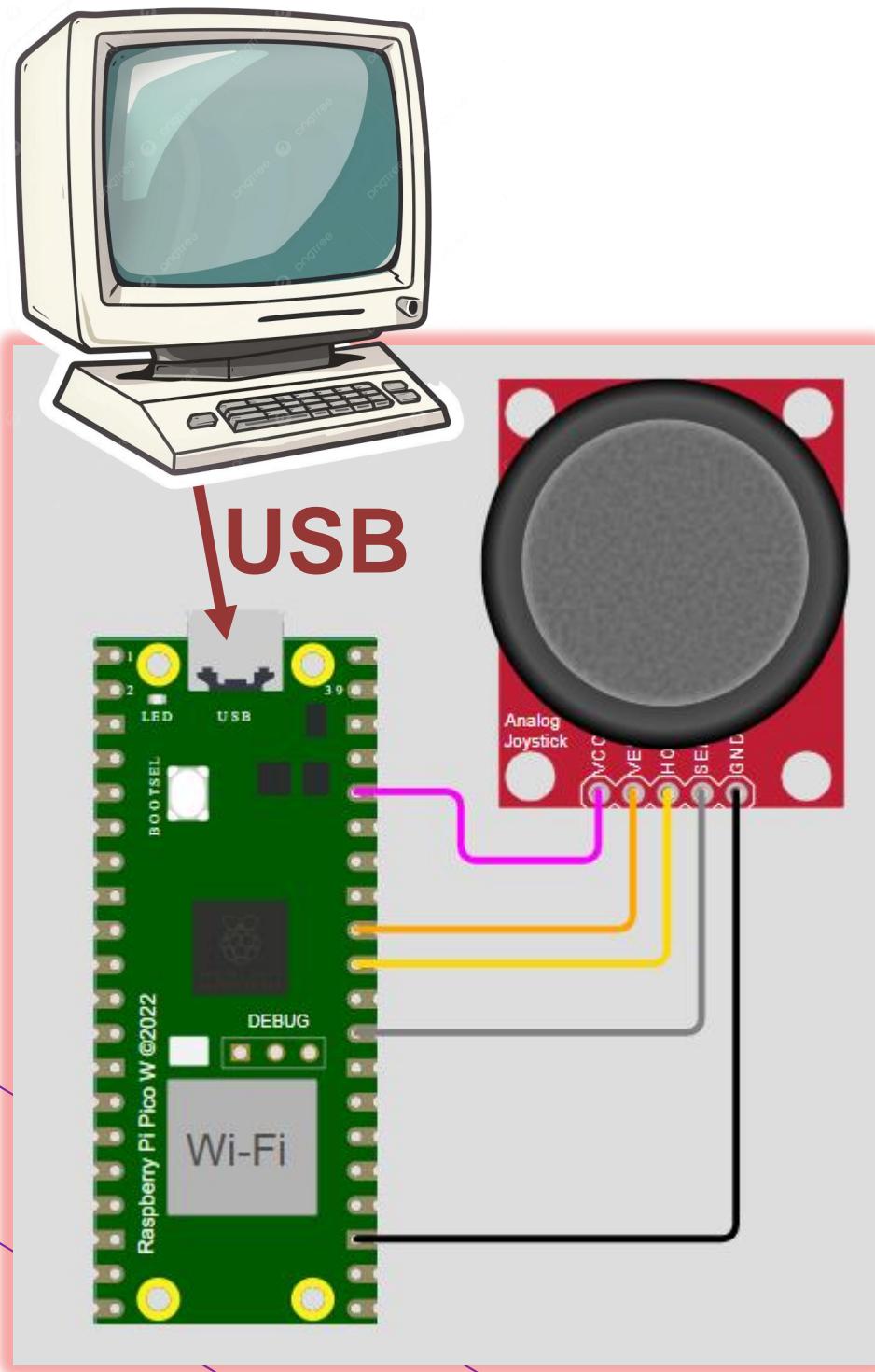
# ADC no RP2040 Exemplos



Vá no: [github.com/wiltonlacerda](https://github.com/wiltonlacerda)  
git clone <https://github.com/wiltonlacerda/EmbarcaTechU4C8.git>

# Conversor Analógico/Digital exemplo 1

Neste exemplo o valor “bruto” capturado pelo ADC será impresso no monitor via conexão serial (USB).



```
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/adc.h"

int main() {
    stdio_init_all();
    adc_init();
    adc_gpio_init(27);
    adc_select_input(1);

    while (true) {
        uint16_t adc_value = adc_read();
        printf("Valor do ADC: %u\n", adc_value);
        sleep_ms(500);
    }
    return 0;
}
```

Pasta:  
**CAP8\_EX1**

# Conversor Analógico/Digital

É correto dizer que o ADC do RP2040 é essencialmente um **medidor de tensão** elétrica. Ele foi projetado para converter sinais analógicos de **tensão** em valores digitais, que podem ser processados pelo microcontrolador.

**1. Entrada analógica:** O ADC do RP2040 aceita sinais de tensão analógica em suas entradas.

**2. Faixa de tensão:** O ADC opera com uma faixa de tensão de **0V a 3,3V** (tensão de referência padrão). Qualquer tensão acima de 3,3V pode danificar o chip.

**3. Resolução:** O ADC do RP2040 tem uma resolução de **12 bits**, o que significa que ele pode representar uma tensão analógica em 4096 ( $2^{12}$ ) níveis discretos.

**4. Conversão:** O ADC converte a tensão analógica em um valor digital proporcional. Por exemplo:

- 0V corresponde ao valor digital **0**.
- 3,3V corresponde ao valor digital **4095**.
- Uma tensão intermediária, como 1,65V, corresponderia a aproximadamente **2048**.

## Perguntas?

Como converter o valor de níveis (0 a 4095) para uma valor que represente uma medida?

Como converter o dado bruto em tensão?

Como converter o dado bruto em temperatura?

# Conversor Analógico/Digital

Vamos mostrar um exemplo de cálculo para converter o valor digital lido pelo ADC do RP2040 (que tem resolução de 12 bits e tensão de referência de 3,3V) em uma tensão elétrica em volts.

## Fórmula básica:

A fórmula utilizada para converter o valor digital do ADC em tensão elétrica é:

$$V_{\text{medida}} = \left( \frac{\text{Valor digital do ADC}}{\text{Resolução máxima do ADC}} \right) \times V_{\text{ref}}$$

Onde:

- $V_{\text{medida}}$  = Tensão medida (em volts).
- Valor digital do ADC = Valor lido pelo ADC (entre 0 e 4095).
- Resolução máxima do ADC =  $2^{12} - 1 = 4095$  (para 12 bits).
- $V_{\text{ref}}$  = Tensão de referência do ADC (3,3V para o RP2040).

Suponha que o ADC leu um valor de **3072**. Vamos calcular a tensão correspondente:

1. Aplique a fórmula:

$$V_{\text{medida}} = \left( \frac{3072}{4095} \right) \times 3,3$$

2. Calcule a fração:

$$\frac{3072}{4095} \approx 0,75$$

3. Multiplique pela tensão de referência:

$$V_{\text{medida}} = 0,75 \times 3,3 = 2,475 \text{ V}$$

## Generalização:

Para qualquer valor digital  $N$  lido pelo ADC, a tensão correspondente pode ser calculada usando a fórmula:

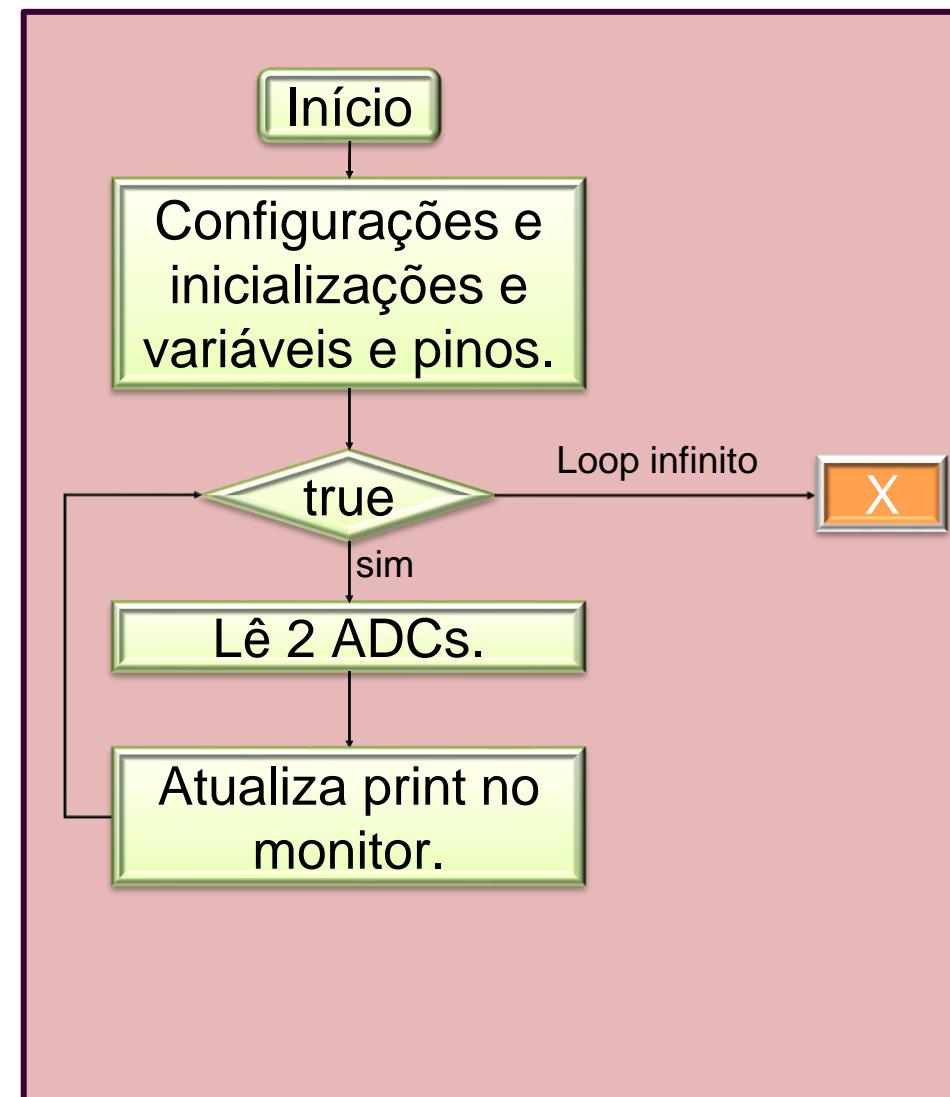
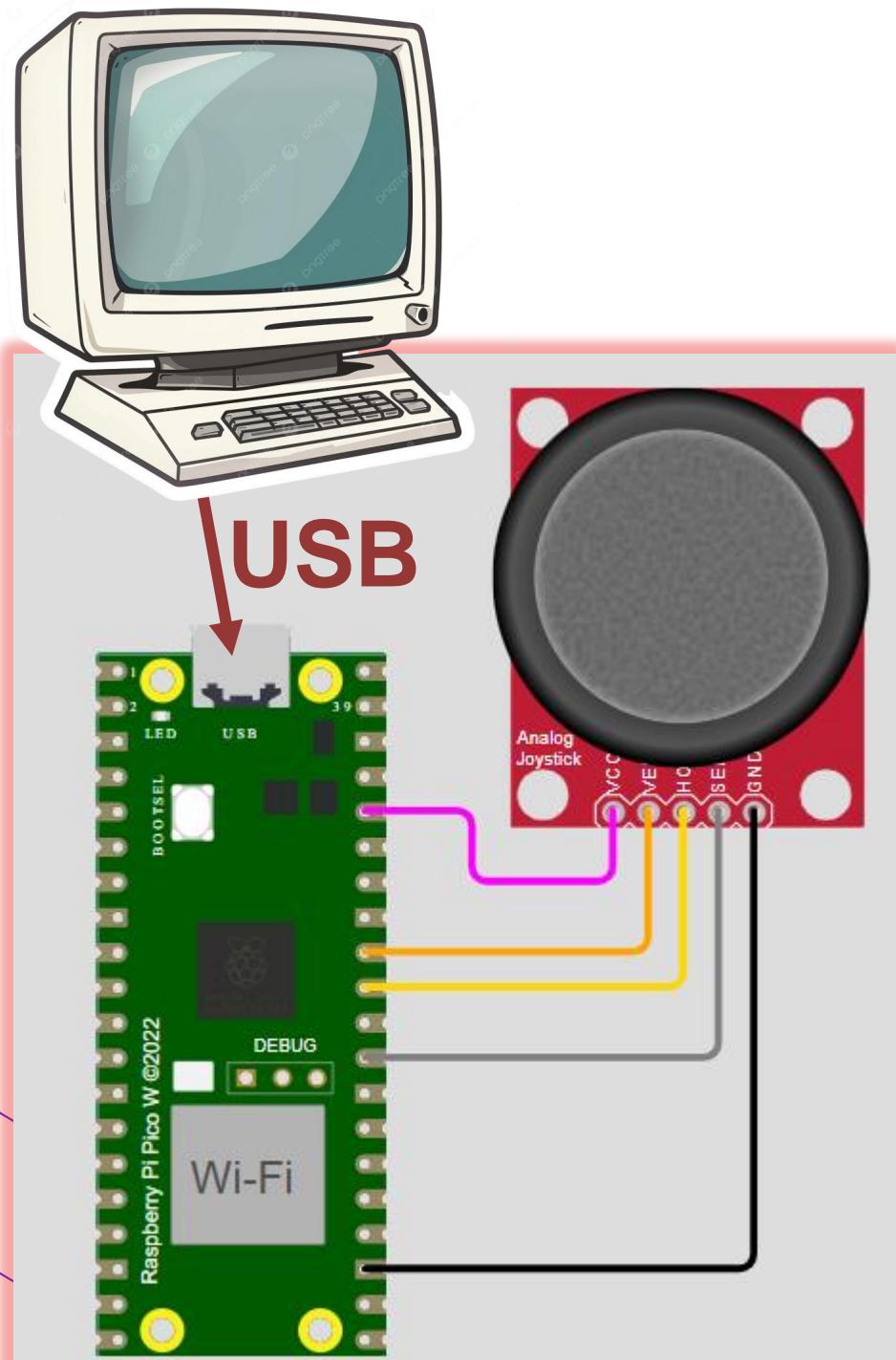
$$V_{\text{medida}} = \left( \frac{N}{4095} \right) \times 3,3$$

Onde:

- $N$  = Valor digital do ADC (0 a 4095).
- 4095 = Valor máximo para um ADC de 12 bits.
- 3,3 = Tensão de referência do ADC (em volts).

# Conversor Analógico/Digital exemplo 2

Neste exemplo o valor “bruto” capturado pelo ADC será impresso no monitor via conexão serial (USB).



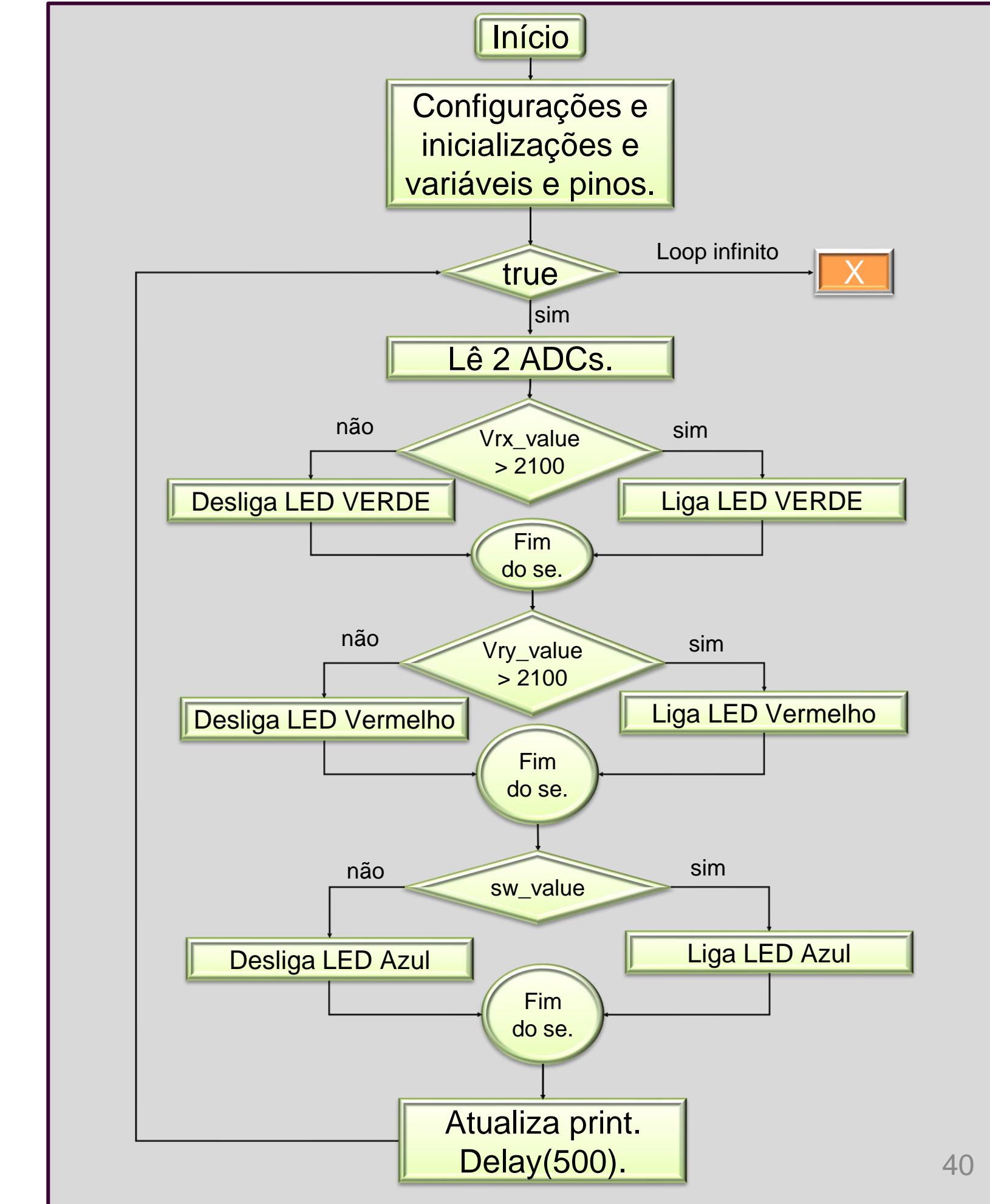
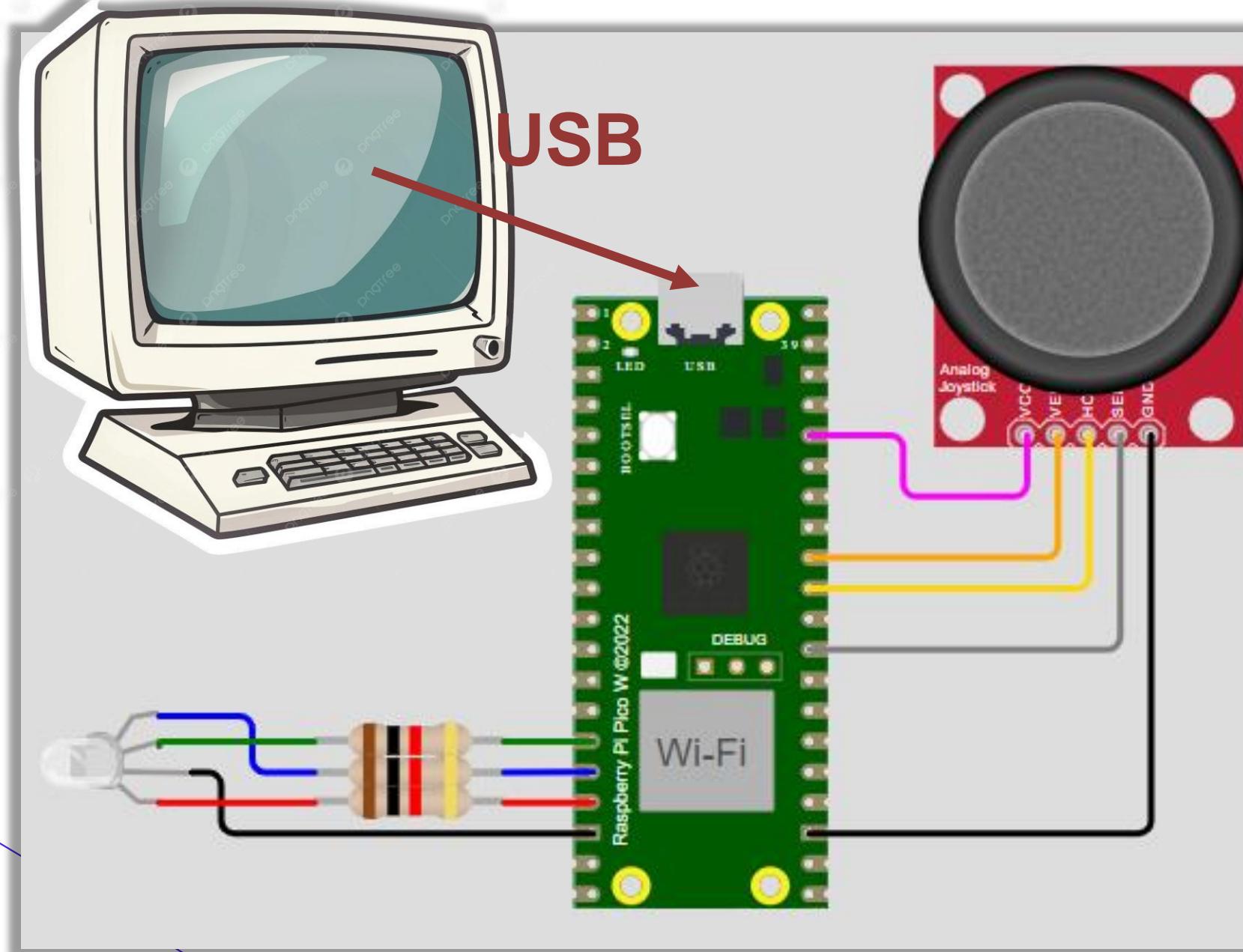
**Observação:**  
O VRX\_PIN é o 27  
O VRY\_PIN é o 26

Pasta:  
**CAP8\_EX2**

```
#define VRX_PIN 26
#define VRY_PIN 27
#define SW_PIN 22
int main()
{
    stdio_init_all();
    adc_init();
    adc_gpio_init(VRX_PIN);
    adc_gpio_init(VRY_PIN);
    gpio_init(SW_PIN);
    gpio_set_dir(SW_PIN, GPIO_IN);
    gpio_pull_up(SW_PIN);
    while (true)
    {
        adc_select_input(0);
        uint16_t vrx_value = adc_read();
        adc_select_input(1);
        uint16_t vry_value = adc_read();
        bool sw_value = gpio_get(SW_PIN) == 0;
        printf("VRX: %u, VRY: %u, SW: %d\n", vrx_value,
vry_value, sw_value);
        sleep_ms(500);
    }
}
```

# Conversor Analógico/Digital exemplo 3

Neste exemplo, os valores brutos dos ADCs capturados não apenas são exibidos no monitor, mas também controlam os LEDs. Além disso, o push-button do joystick é utilizado.



# Conversor Analógico/Digital exemplo 3

```
#include <stdio.h>
#include "pico/stdc.h"
#include "hardware/adc.h"

#define VRX_PIN 26
#define VRY_PIN 27
#define SW_PIN 22
#define LED1_PIN 11
#define LED2_PIN 13
#define LED3_PIN 12

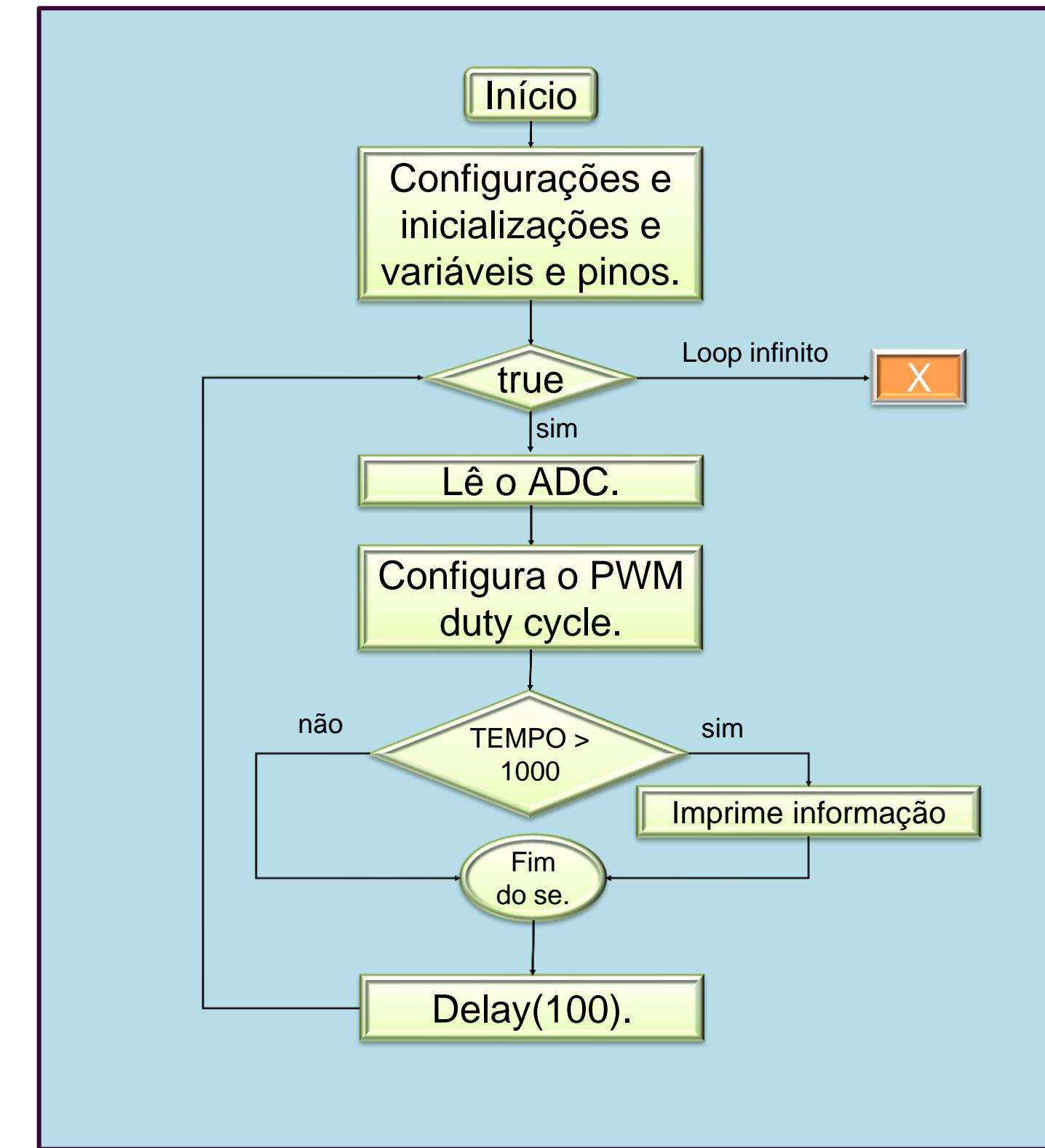
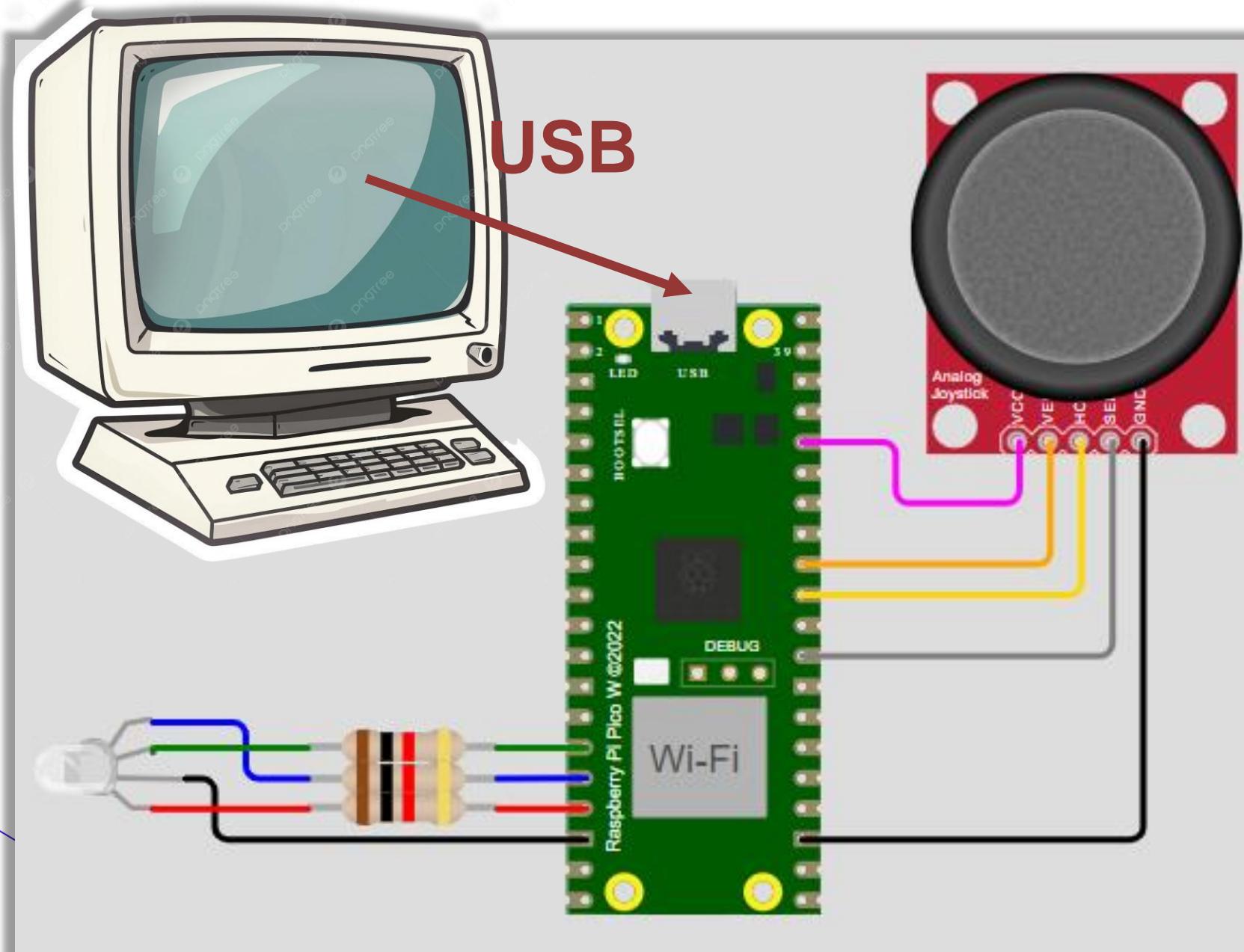
int main() {
    stdio_init_all();
    adc_init();
    adc_gpio_init(VRX_PIN);
    adc_gpio_init(VRY_PIN);
    gpio_init(SW_PIN);
    gpio_set_dir(SW_PIN, GPIO_IN);
    gpio_pull_up(SW_PIN);
    gpio_init(LED1_PIN);
    gpio_set_dir(LED1_PIN, GPIO_OUT);
    gpio_put(LED1_PIN, false);
    gpio_init(LED2_PIN);
    gpio_set_dir(LED2_PIN, GPIO_OUT);
    gpio_put(LED2_PIN, false);
    gpio_init(LED3_PIN);
    gpio_set_dir(LED3_PIN, GPIO_OUT);
    gpio_put(LED3_PIN, false);
```

```
while (true) {
    adc_select_input(0);
    uint16_t vrx_value = adc_read();
    adc_select_input(1);
    uint16_t vry_value = adc_read();
    bool sw_value = gpio_get(SW_PIN) == 0;
    if (vrx_value > 2100) {
        gpio_put(LED1_PIN, true);
    } else {
        gpio_put(LED1_PIN, false);
    }
    if (vry_value > 2100) {
        gpio_put(LED2_PIN, true);
    } else {
        gpio_put(LED2_PIN, false);
    }
    if (sw_value) {
        gpio_put(LED3_PIN, true);
    } else {
        gpio_put(LED3_PIN, false);
    }
    printf("VRX: %u, VRY: %u, SW: %d\n", vrx_value, vry_value, sw_value);
    sleep_ms(500);
} }
```

**Pasta:  
CAP8\_EX3**

# Conversor Analógico/Digital exemplo 4

Neste exemplo, o valor lido no ADC é passado como parâmetro de configuração para o DUTY CYCLE do sinal PWM que controlará o LED.



# Conversor Analógico/Digital exemplo 4

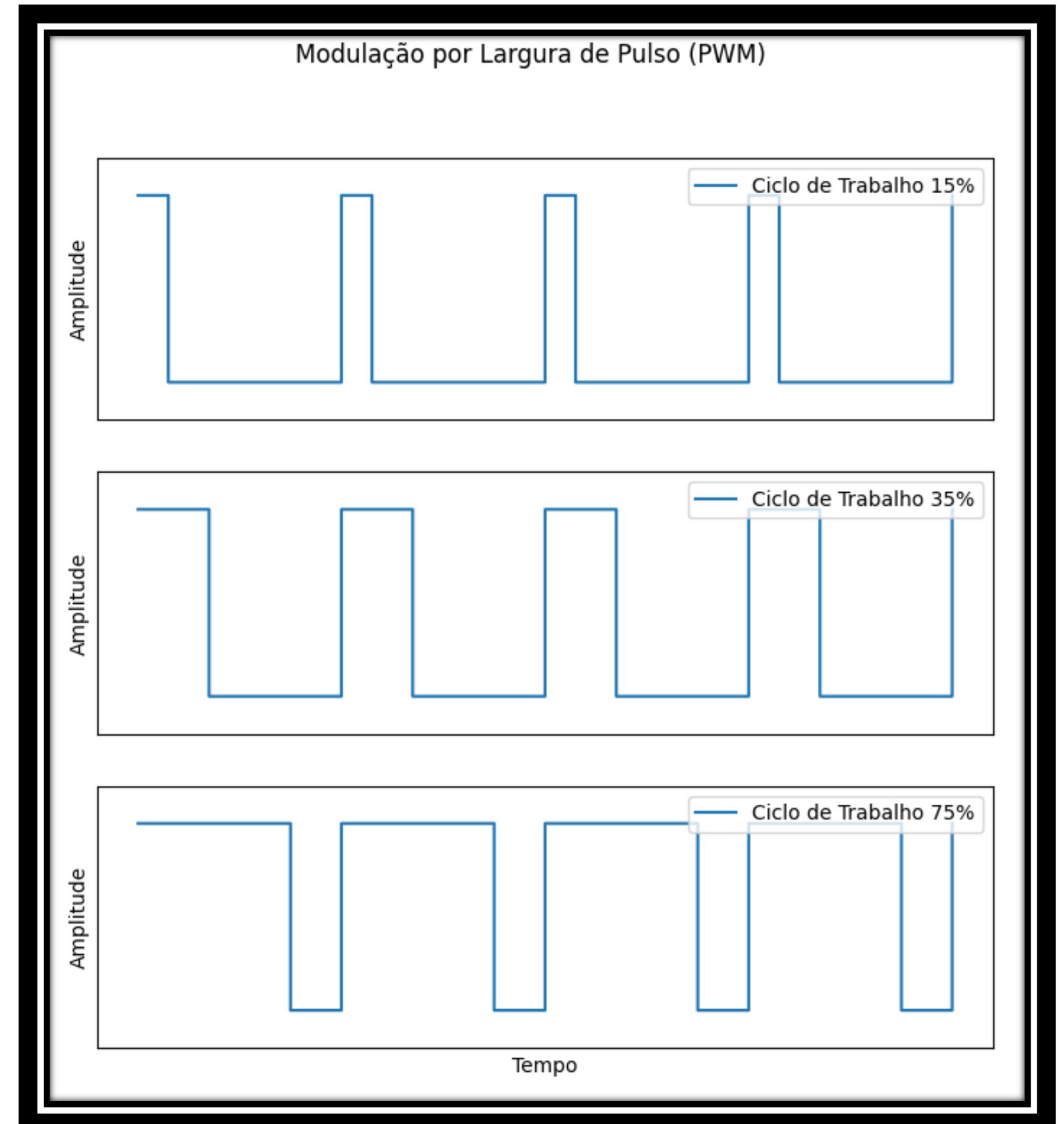
Modulação por Largura de Pulso é uma técnica para **simular uma tensão analógica** usando sinais digitais.

## Analise dos ciclos de trabalho representados no gráfico:

Ciclo de Trabalho de 15%:

Descrição: O sinal está ativo (em alta amplitude) por 15% do tempo do período total e inativo (em baixa amplitude) por 85% do tempo.

Aplicação: Este ciclo de trabalho é útil quando se necessita de baixa potência média, como em aplicações de controle fino de motores ou LEDs de baixa intensidade.



# Conversor Analógico/Digital exemplo 4

```
#include <stdio.h>
#include "pico/stlolib.h"
#include "hardware/adc.h"
#include "hardware/pwm.h"

#define VRX_PIN 26
#define LED_PIN 12

uint pwm_init_gpio(uint gpio, uint wrap) {
    gpio_set_function(gpio, GPIO_FUNC_PWM);
    uint slice_num= pwm_gpio_to_slice_num(gpio);
    pwm_set_wrap(slice_num, wrap);
    pwm_set_enabled(slice_num, true);
    return slice_num;
}

int main() {
    stdio_init_all();
    adc_init();
    adc_gpio_init(VRX_PIN);
    uint pwm_wrap = 4096;
    uint pwm_slice = pwm_init_gpio(LED_PIN, pwm_wrap);
    uint32_t last_print_time = 0;
```

**Pasta:**  
**CAP8\_EX4**

```
while (true) {
    adc_select_input(0);
    uint16_t vrx_value = adc_read();
    pwm_set_gpio_level(LED_PIN, vrx_value);

    float duty_cycle = (vrx_value / 4095.0) * 100;

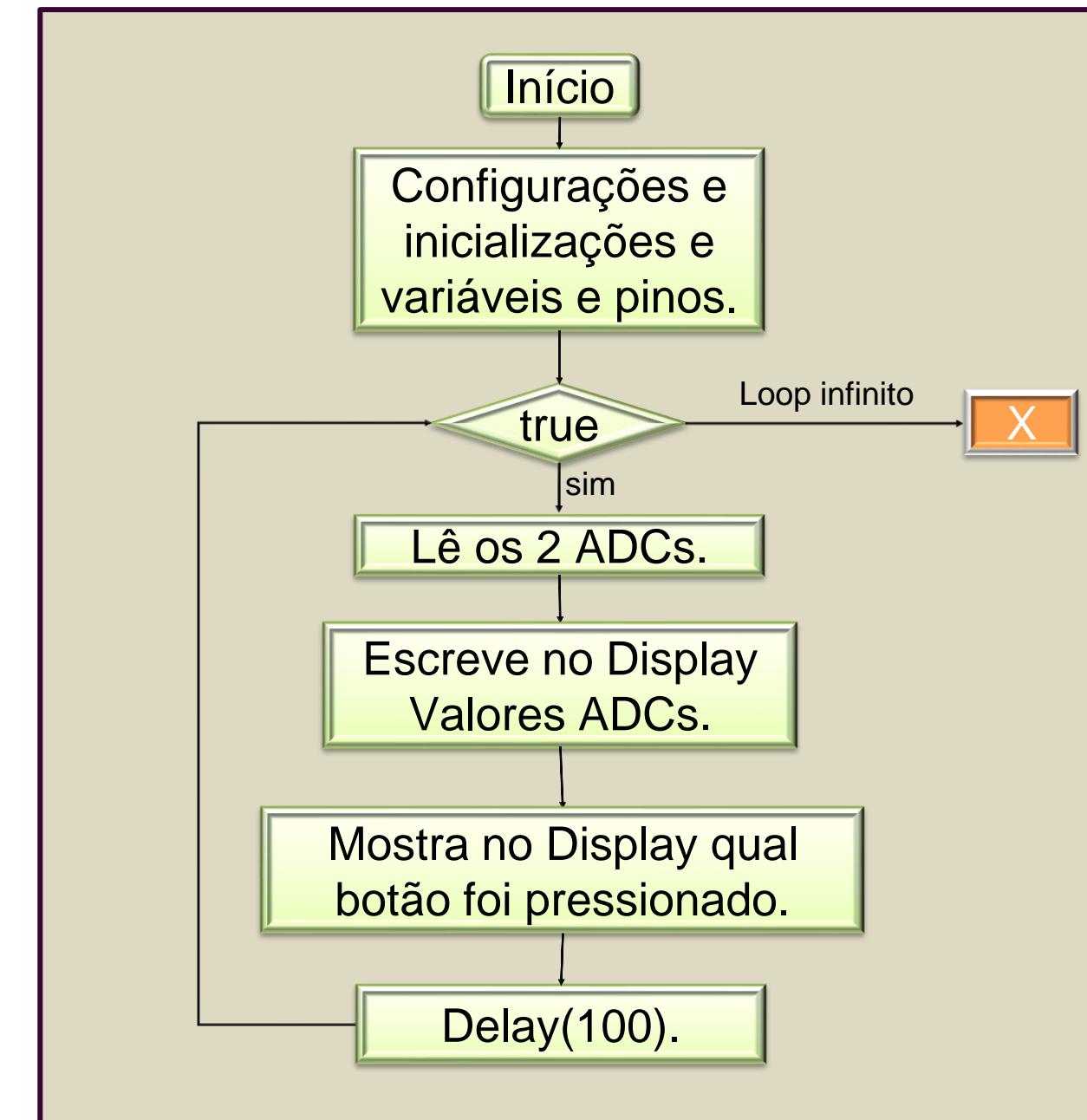
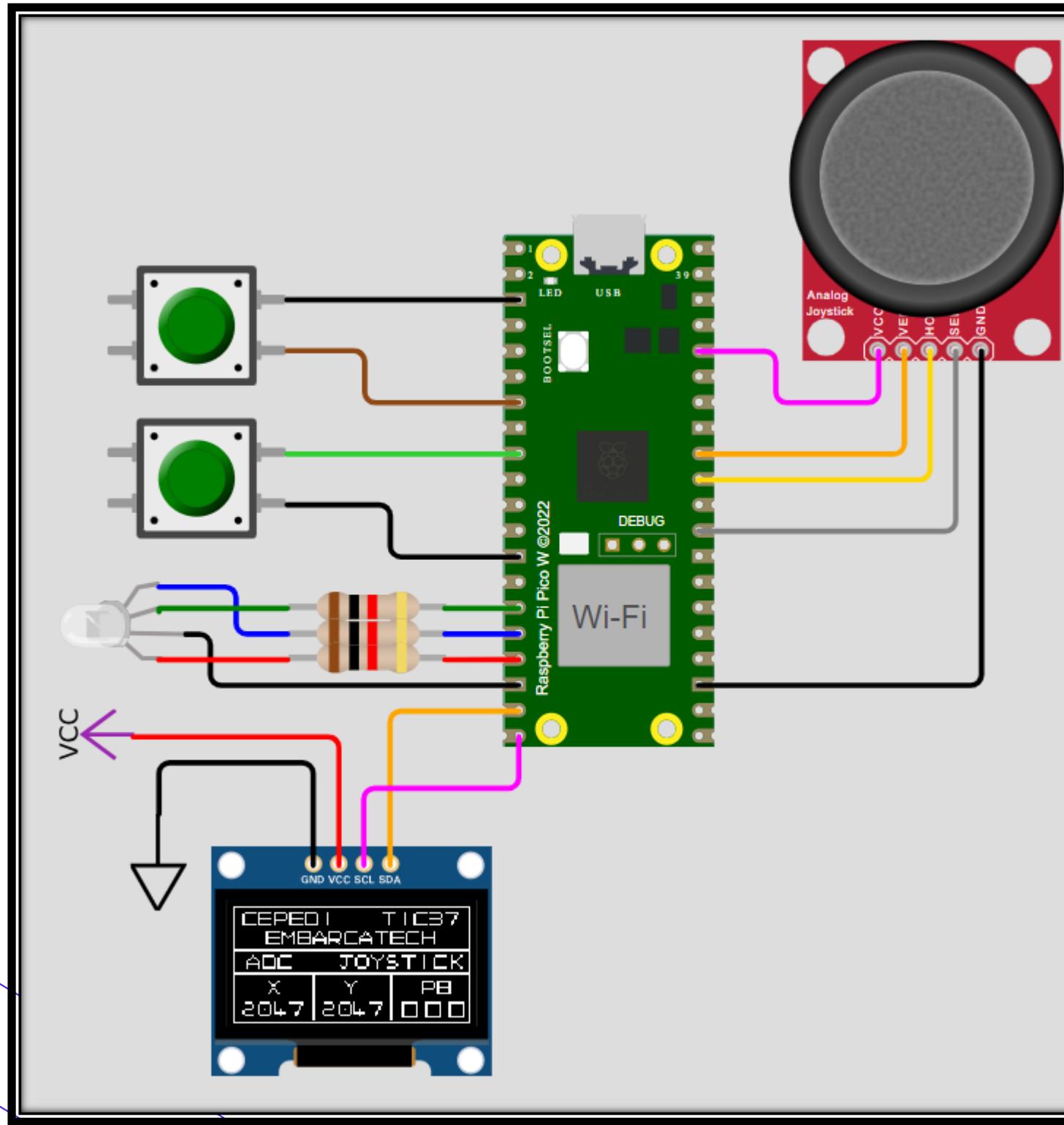
    uint32_t current_time =
                    to_ms_since_boot(get_absolute_time());

    if (current_time - last_print_time >= 1000) {
        printf("VRX: %u\n", vrx_value);
        printf("Duty Cycle LED: %.2f%%\n", duty_cycle);
        last_print_time = current_time;
    }

    sleep_ms(100);
}
```

# Conversor Analógico/Digital exemplo 5

Neste exemplo, os valores lidos nos ADCs são apresentados no Display OLED. Também é mostrado a leitura dos botões do Joystick e do botão A.



# Conversor Analógico/Digital exemplo 5

```
/* Configurações iniciais */  
  
adc_init();  
adc_gpio_init(JOYSTICK_X_PIN);  
adc_gpio_init(JOYSTICK_Y_PIN);  
  
uint16_t adc_value_x;  
uint16_t adc_value_y;  
char str_x[5]; // Buffer para armazenar a string  
char str_y[5]; // Buffer para armazenar a string  
  
bool cor = true;  
while (true)  
{  
    adc_select_input(0); // Seleciona o ADC para eixo X.  
    adc_value_x = adc_read();  
    adc_select_input(1); // Seleciona o ADC para eixo Y.  
    adc_value_y = adc_read();  
    sprintf(str_x, "%d", adc_value_x); //Converte int em string  
    sprintf(str_y, "%d", adc_value_y); //Converte int em string
```

**Pasta:  
ADC\_DISPYEMC**

```
ssd1306_fill(&ssd, !cor);  
ssd1306_rect(&ssd, 3, 3, 122, 60, cor, !cor);  
ssd1306_line(&ssd, 3, 25, 123, 25, cor);  
ssd1306_line(&ssd, 3, 37, 123, 37, cor);  
ssd1306_draw_string(&ssd, "CEPEDI TIC37", 8, 6);  
ssd1306_draw_string(&ssd, "EMBARCATECH", 20, 16);  
ssd1306_draw_string(&ssd, "ADC JOYSTICK", 10, 28);  
ssd1306_draw_string(&ssd, "X Y PB", 20, 41);  
ssd1306_line(&ssd, 44, 37, 44, 60, cor);  
ssd1306_draw_string(&ssd, str_x, 8, 52);  
ssd1306_line(&ssd, 84, 37, 84, 60, cor);  
ssd1306_draw_string(&ssd, str_y, 49, 52);  
ssd1306_rect(&ssd, 52, 90, 8, 8, cor, !gpio_get(JOYSTICK_PB));  
ssd1306_rect(&ssd, 52, 102, 8, 8, cor, !gpio_get(Botao_A));  
ssd1306_rect(&ssd, 52, 114, 8, 8, cor, !cor);  
ssd1306_send_data(&ssd); // Atualiza o display  
sleep_ms(100);  
}
```

# Conversor Analógico/Digital exemplos

Estão disponibilizados um exemplo de como capturar dados do microfone embutido na BitDogLab, bem como um exemplo da leitura da temperatura interna do RP2040.

Comentar a tarefa da semana

# ADCs: Conclusões

## Finalizando

O ADC do RP2040 possui uma resolução de 12 bits, permitindo representar sinais analógicos com 4096 níveis diferentes. A precisão real pode ser afetada por ruído e variações na alimentação.

Possui quatro canais de entrada analógica nos pinos GPIO26 a GPIO29, entretanto na placa Raspberry Pi Pico, o GPIO29 é utilizado como VSYS.

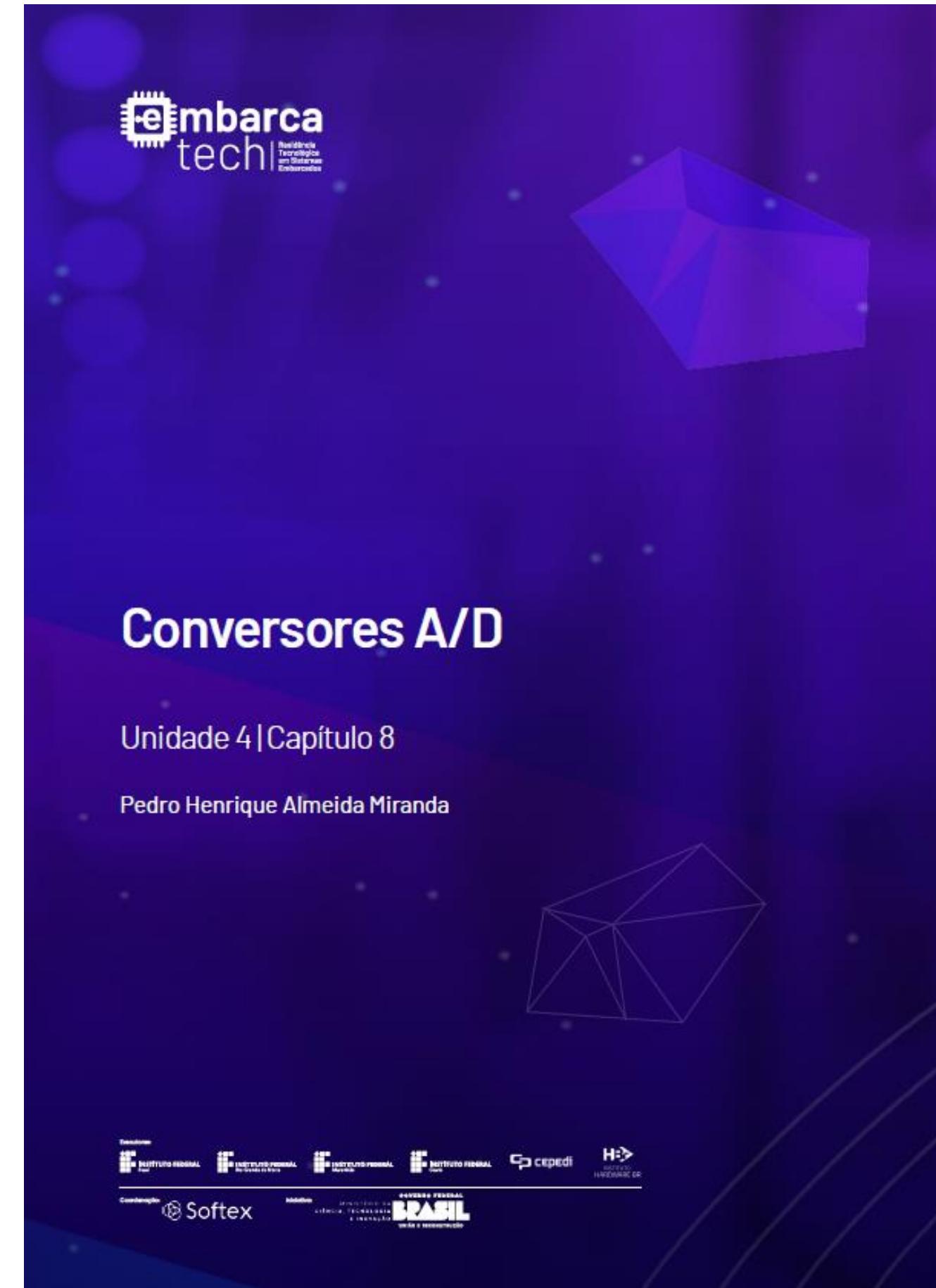
O ADC4 mede a tensão de um diodo interno cujo valor varia conforme a temperatura do chip.

A referência padrão de tensão do ADC é a alimentação de 3,3V. O valor lido pode ser convertido para a tensão correspondente. Variação na tensão de alimentação comprometerá o resultado da conversão.

A taxa de amostragem pode atingir cerca de 500 kSPS.

# Conversor Analógico/Digital

## Referências



# Obrigado!

Executores:



Coordenação:



Iniciativa:

