Conversor AD

Introdução

Qualquer sistema que possua um microcontrolador, trabalha internamente apenas com dados digitais

As entradas digitais de um microcontrolador são “lidas” e interpretadas por apenas dois níveis lógicos: baixo e alto.
Como referências são usados os valores 0V e 5V.

Já, os dados lidos em uma entrada analógica podem assumir infinitos valores em uma faixa pré-estabelecida (normalmente entre 0V e 5V).

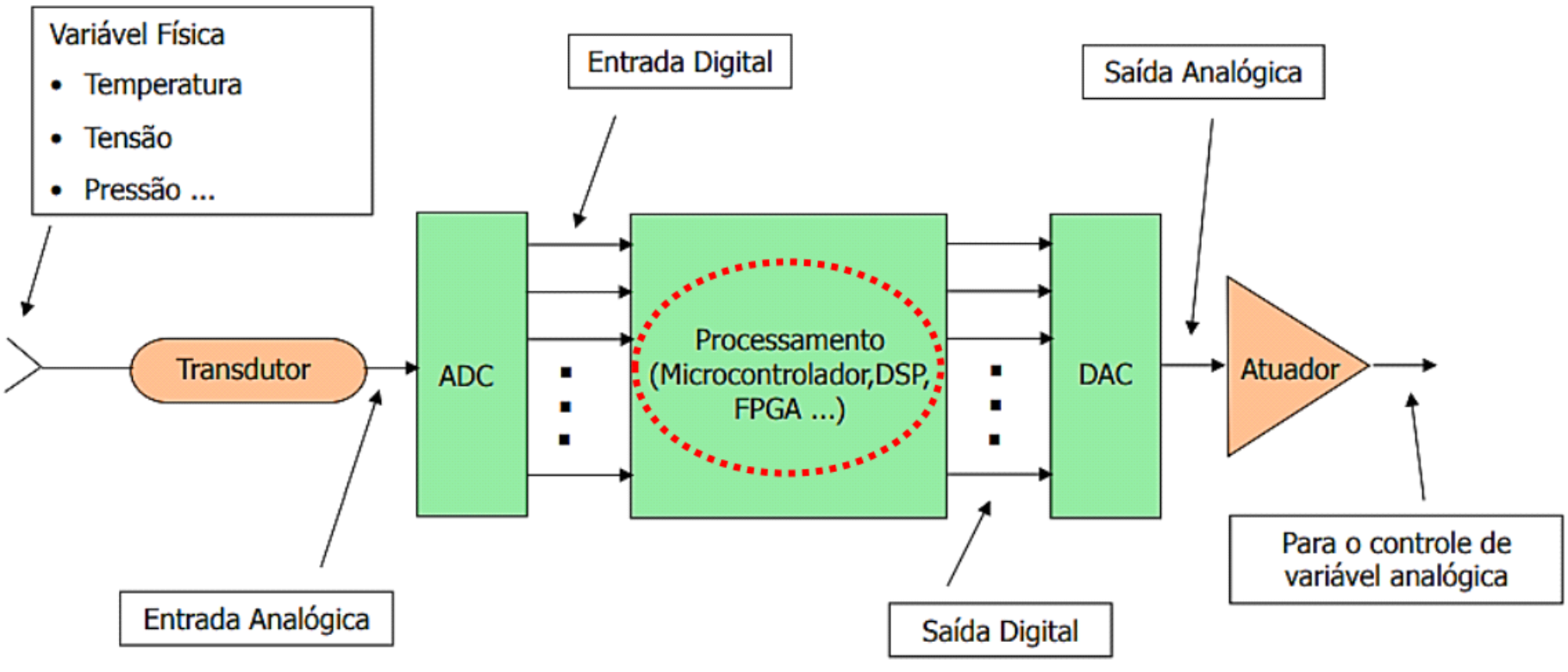
Variáveis analógicas são fundamentais pois representam medições de grandezas físicas, como tensão, temperatura, vazão, pressão, entre outras.

Valores analógicos lidos por um pino devem ser convertidos para um valor digital que o represente.

Este trabalho é feito por meio de um conversor Analógico - Digital (conversor A/D ou ADC).

Este tipo de conversor é encontrado tanto na forma de circuitos integrados quanto já embutidos dentro de microcontroladores, como é o caso do AVR.

Motivos para a Conversão



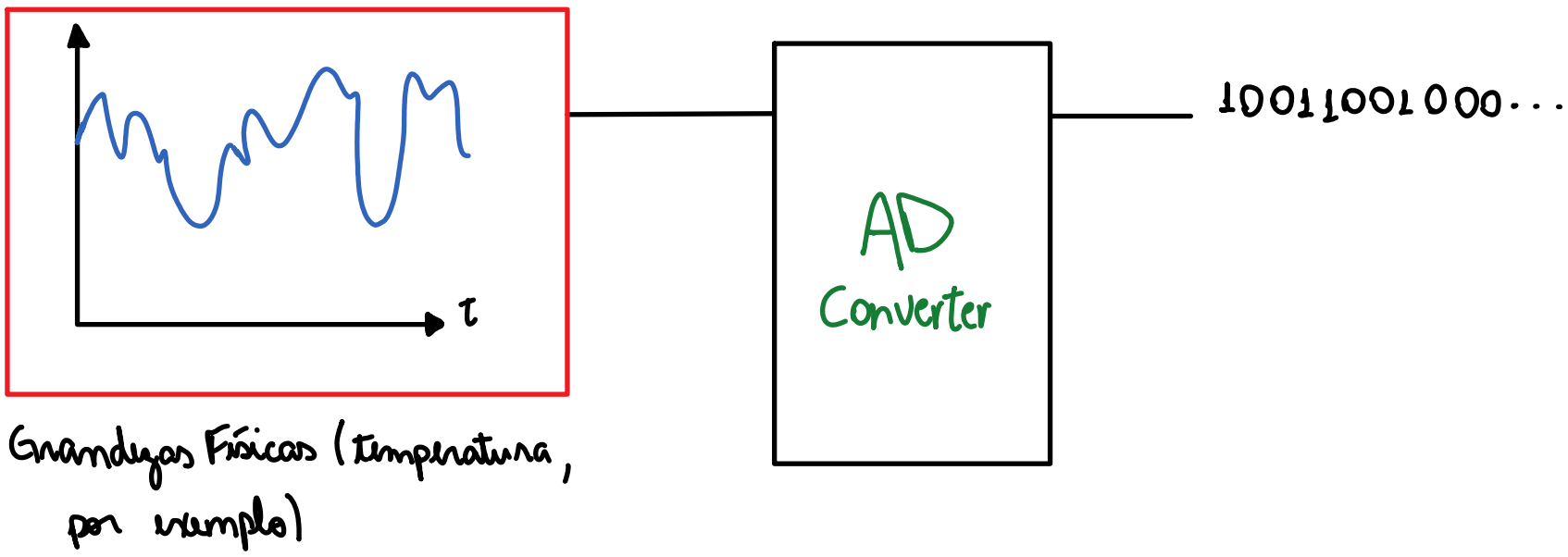
→ Os principais motivos são:

Armazenamento = Sinais analógicos são sinais infinitos, e sinais digitais são sinais discretos (finitos).

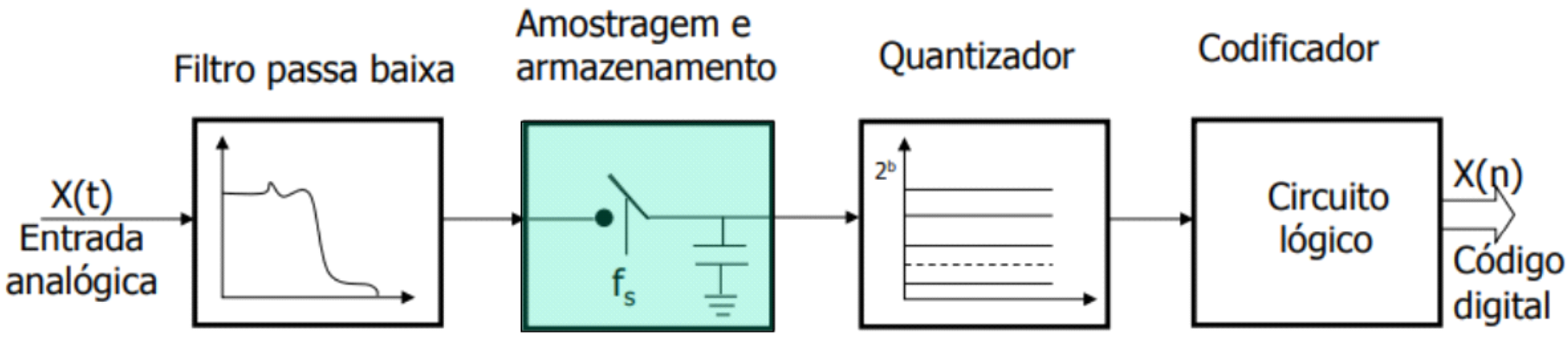
Compartilhamento (Transmissão) = É fácil de enviar e transmitir.

Processamento = Pela facilidade de processar áudios, imagens (apagar, tirar outra foto, vídeo, etc).

→ O esquema de conversão é o seguinte:



Processo de Conversão

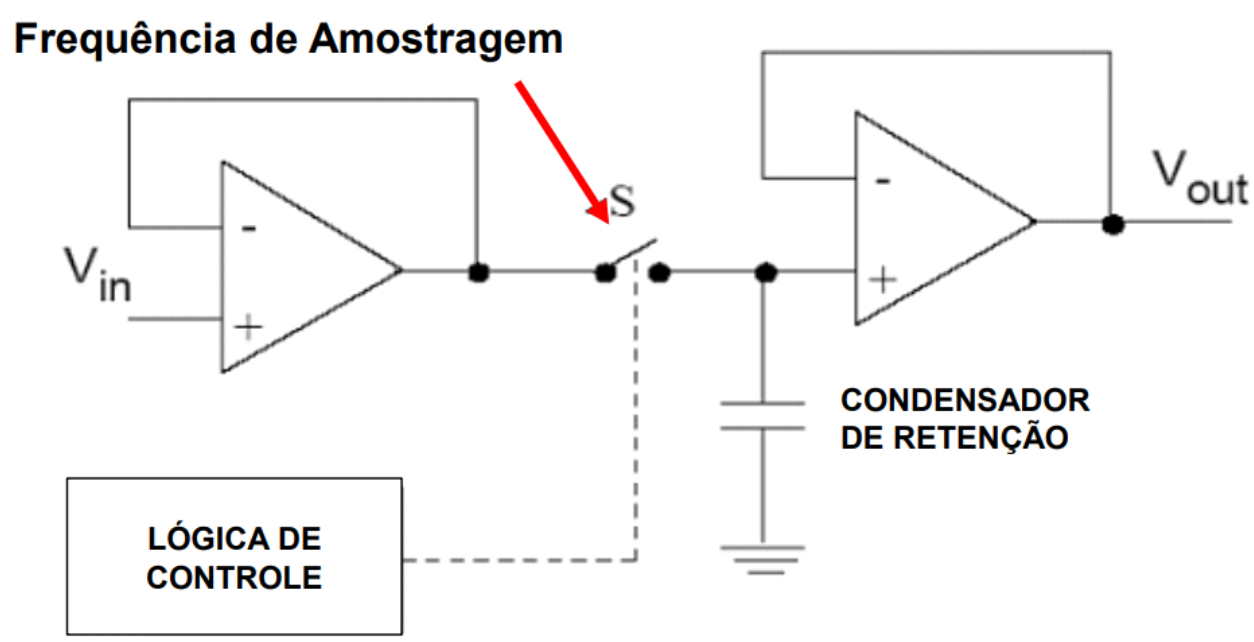


A fase de amostragem é realizada por meio de circuitos de amostragem e retenção (*sample and hold*), utilizados para retenção do valor analógico de entrada.

As amostras são coletadas em períodos de tempo fixos que definem o "período de amostragem".

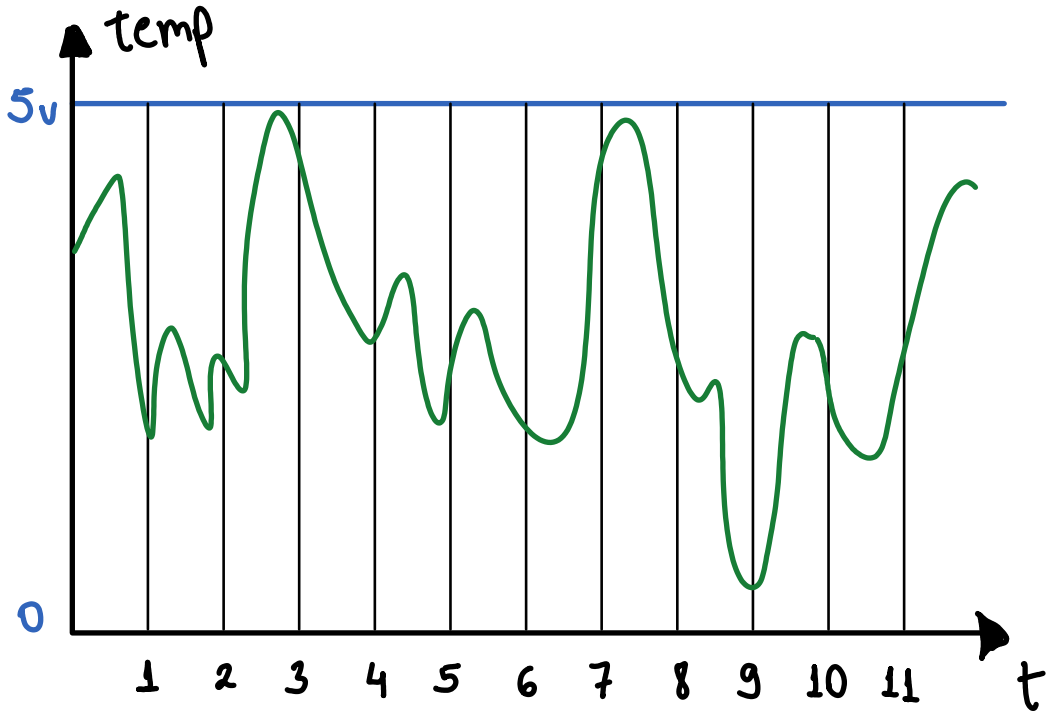
Teorema da amostragem (**Nyquist**): a frequência de amostragem (taxa de amostragem) deve ser maior ou igual a duas vezes a maior frequência do sinal.

Circuito Amostrador



SAÍDA = ENTRADA (estado Track)
SAÍDA = CONSTANTE (estado Hold)

• Quando a chave **S** está fechada, é feita a amostragem (*sample*). Neste, a saída do circuito é igual à entrada;
• Quando a chave **S** está aberta, o circuito encontra-se no estado de retenção (*hold*), ou seja, a saída é constante e seu valor corresponde ao amostrado no instante de tempo em que o estado *hold* foi iniciado.



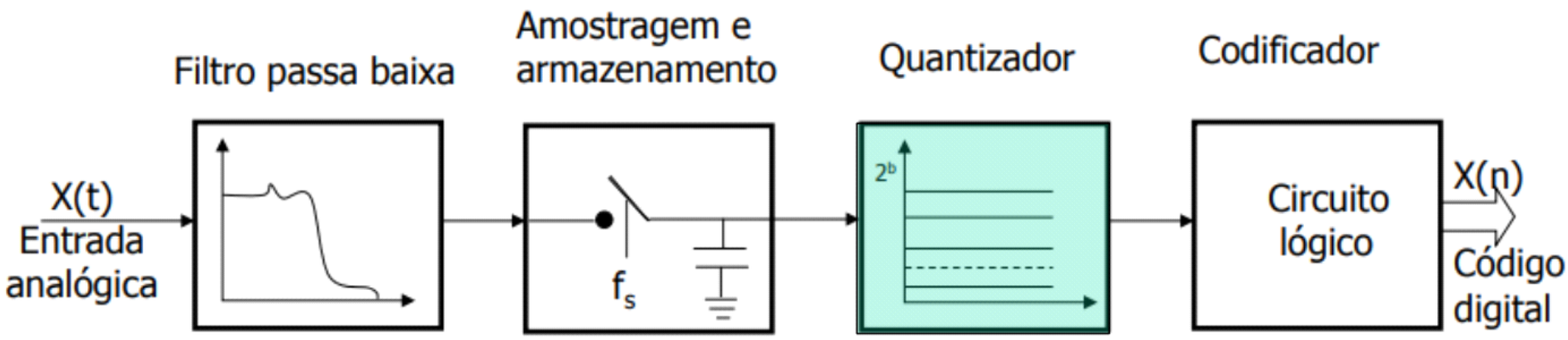
Amostragem

→ A cada 1 segundo, é coletada 1 amostra:

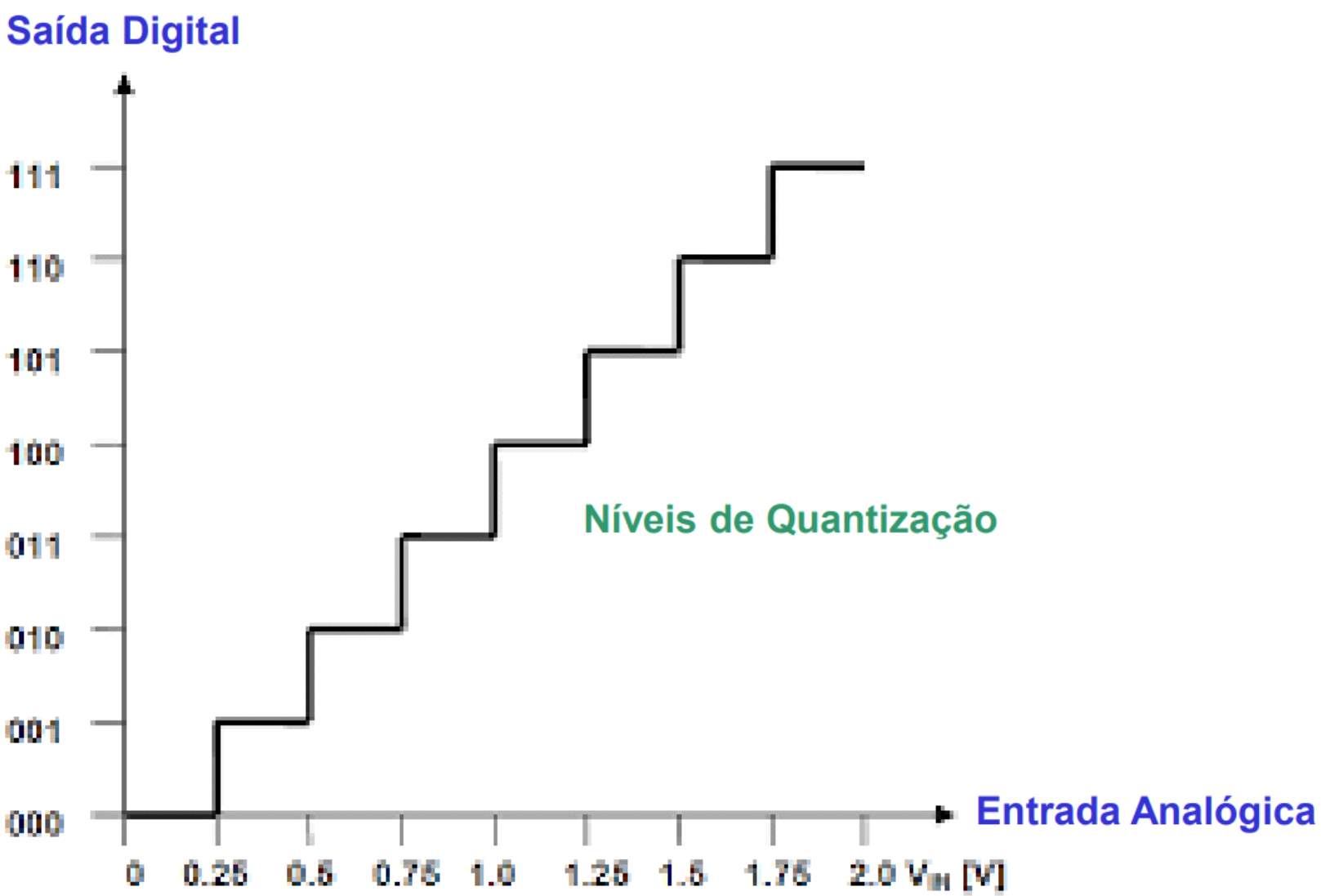
$t_a = 1\ s$

período de amostragem

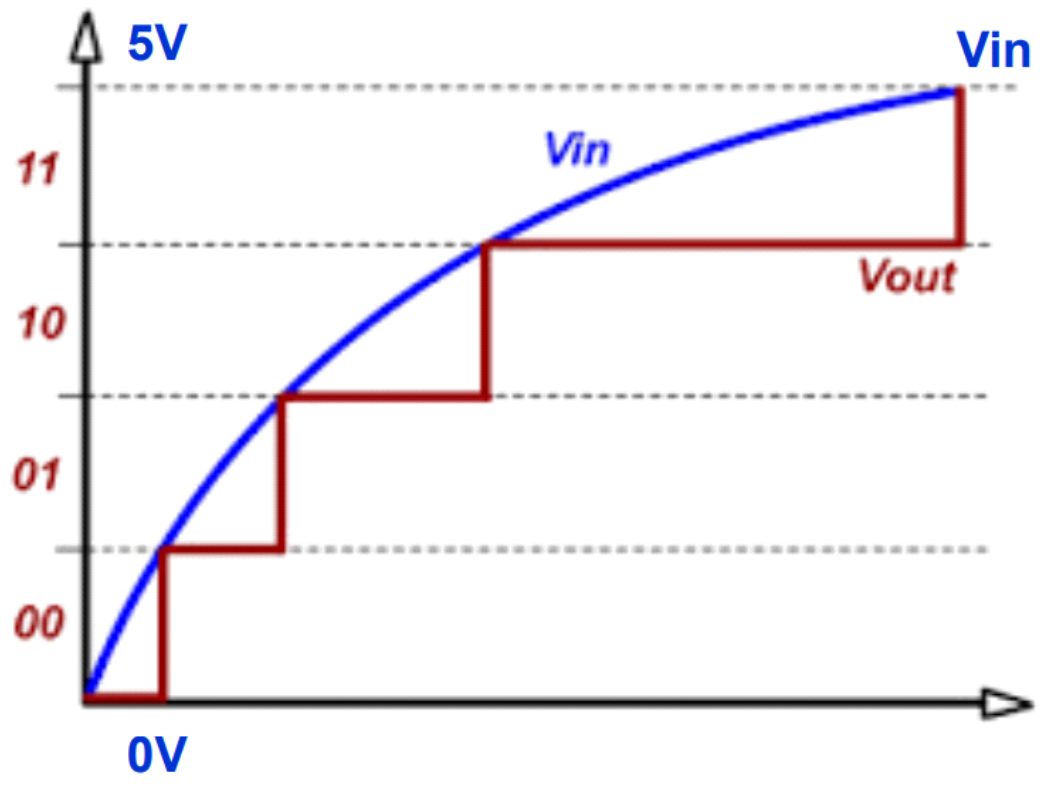
Quantizador



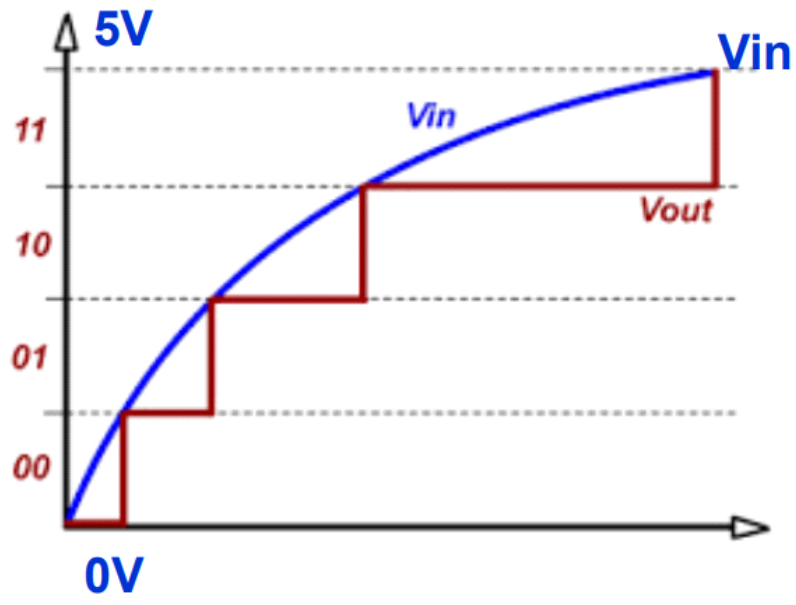
A quantização do valor armazenado é feita através de um circuito quantizador que aproxima o valor retido do nível de quantização mais próximo.



Suponha que a tensão fornecida ao conversor A/D varie de 0 a 5 volts e o este converta a tensão de entrada em um número binário de dois bits. Com dois bits, podemos exibir APENAS 4 opções diferentes: 00, 01, 10 e 11.

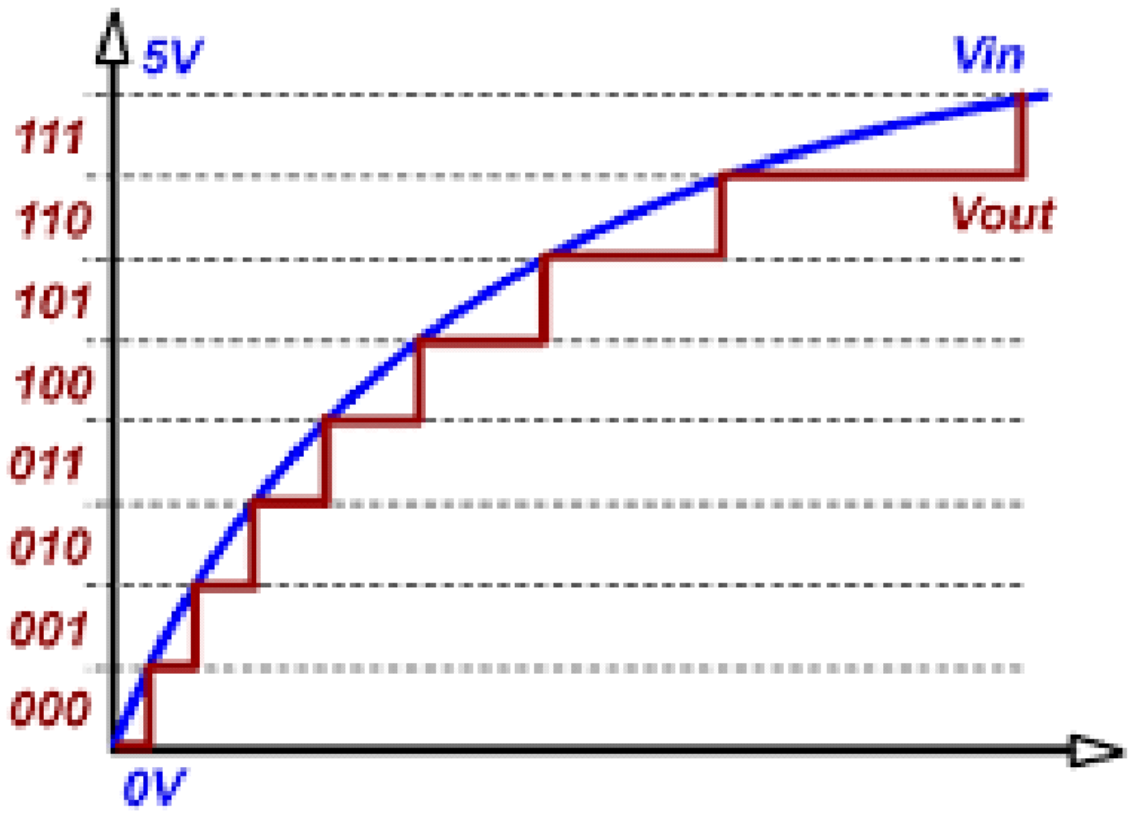


A linha AZUL descreve as mudanças na tensão de **entrada** do conversor A/D do microcontrolador.
A linha MARROM representa os níveis digitais na **saída** do mesmo.



Podemos ver que o sinal marrom está longe de ser ideal, ou seja, não está próximo o suficiente dos valores originais de tensão de entrada analógica. Assim, podemos dizer que o conversor A/D com dois bits tem **baixa resolução** e existe um grande *gap* entre o valor real da tensão de entrada analógica e os valores representados pelo mesmo.

Agora, suponha que o conversor A/D represente a tensão de entrada em um número binário de três bits.
Com três bits podemos obter 8 opções diferentes: 000, 001, 010, 011, 100, 101, 110, 111.



Podemos observar que a linha MARROM representa o sinal original “melhor” do que anteriormente.

A lacuna entre o sinal analógico e o sinal digital é menor agora.

A resolução do conversor A/D de 3 bits é maior do que o de 2 bits.

Quanto maior a quantidade de bits maior a resolução e melhor precisão na conversão de sinal analógico para sinal digital.

Um ponto importante é que a conversão em baixa resolução é mais rápida do que em alta resolução.

O ATMEGA328p dispõe de 6 pinos (PC0 – PC5) que podem realizar a leitura de uma tensão máxima de 5V.

Um conversor A/D quantifica o valor analógico conforme a quantidade de bits da sua resolução.

A resolução de um conversor A/D é dada pela seguinte equação:

$$Resolução = \frac{V_{REF}}{2^n - 1}$$

V_{REF} é a tensão de referência do conversor A/D (normalmente 5V)
n é o número de bits do conversor.

O conversor A/D do AVR possui 10 bits de resolução, a sua tensão de entrada pode variar de 0V até o valor de VCC. Dessa forma, o menor valor que pode ser lido será:

$$Resolução = \frac{5V}{2^{10} - 1} = 4{,}88\,mV$$

Este é o valor que representa o degrau para uma conversão em 10 bits com referência em 5V.

No diagrama em blocos pode-se observar que a entrada do conversor A/D é multiplexada.
Isto significa que é possível ligar até 6 entradas analógicas independentes.

A tensão de referência pode ser configurada como:

- Tensão de referência = Vcc;
- Tensão de referência = 1,1V (referência interna);
- Tensão de referência = Pino AREF.

Os registradores de controle das entradas analógicas são:

1. **ADMUX** (ADC Multiplexer Selection Bits): Responsável por determinar a tensão de referência e o canal de conversão:

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits REFS1:REFS0 – Determinam a referência do ADC de acordo com a tabela:

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Bit ADLAR – Ajusta a formatação do resultado:

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	

Bits MUX3:MUX0 – Determina o canal que está sendo lido pelo ADC, de acordo com a tabela:

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[(1)]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

ADCSRA (ADC Control and Status Register A): Responsável por configurar o *prescaler* e o controle de conversão.

O registrador **ADCSRB, não será utilizado nas conversões aqui realizadas.*

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit ADEN: Responsável por habilitar o ADC.
0 – Desliga o ADC;
1 – Liga o ADC;

Bit ADSC: Responsável por iniciar a conversão.
1 – Inicia a conversão;
0 – Indica que a conversão foi finalizada.

Após a finalização da conversão este bit vai para nível 0 automaticamente.

Bit ADATE: Utilizado para o modo de disparos automáticos do ADC (configurado no registrador ADCSRB);

Bit ADIF: Flag de interrupção do ADC;

Bit ADIE: Habilitação da interrupção do ADC;

Bits ADPS2:ADPS0: Configuração do *Prescaler*:

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADCL e ADCH: Registradores que armazenam o resultado da conversão.
** Em linguagem C pode ser utilizado apenas o nome "ADC", que é retornado o valor dos 10 bits da conversão.*

Exemplo 1: Apresentação do valor convertido do canal 0 no serial monitor.

```
void setup()
{
// inicializa os registros utilizados pelo comparador
ADMUX = 0x40;
ADCSRB = 0x0;
DIDR0 = 0x1;
Serial.begin (9600);
ADCSRA = 0xC7;  // dispara a aquisição
}
void loop()
{
      int nValue;
      while((ADCSRA && 0x40) == 0x40);  //espera conversão acabar
      nValue = ADCL;
      nValue |= (ADCH << 8);
      Serial.println (nValue);
      delay(1000);
      ADCSRA = 0xC7;  // dispara nova conversão
}
```

Exemplo 2: Apresentação do valor convertido do canal 0 nos LEDs do PORTD.

```
int main()
{
DDRD = 0b11111111; //PORTD inteiro como saída
//Habilita a referência de tensão interna do ADC e ativa o canal 0
//ADMUX = (1<<REFS0); //Outra maneira de manipular o registrador
ADMUX = 0b01000000;
//Habilita o ADC e configura o prescaler para 128
//ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
ADCSRA |= 0b10000111;
while(1){
//ADCSRA |= (1<<ADSC);
ADCSRA |= 0b01000000; //Inicia a conversão
//Aguarda o término da conversão
//while ( (ADCSRA & (1<<ADSC)));
while ((ADCSRA & 0b01000000));
//Descarta os dois bits menos significativos e apresenta o resultado no PORTD
PORTD = (ADC >> 2);
}
}
```