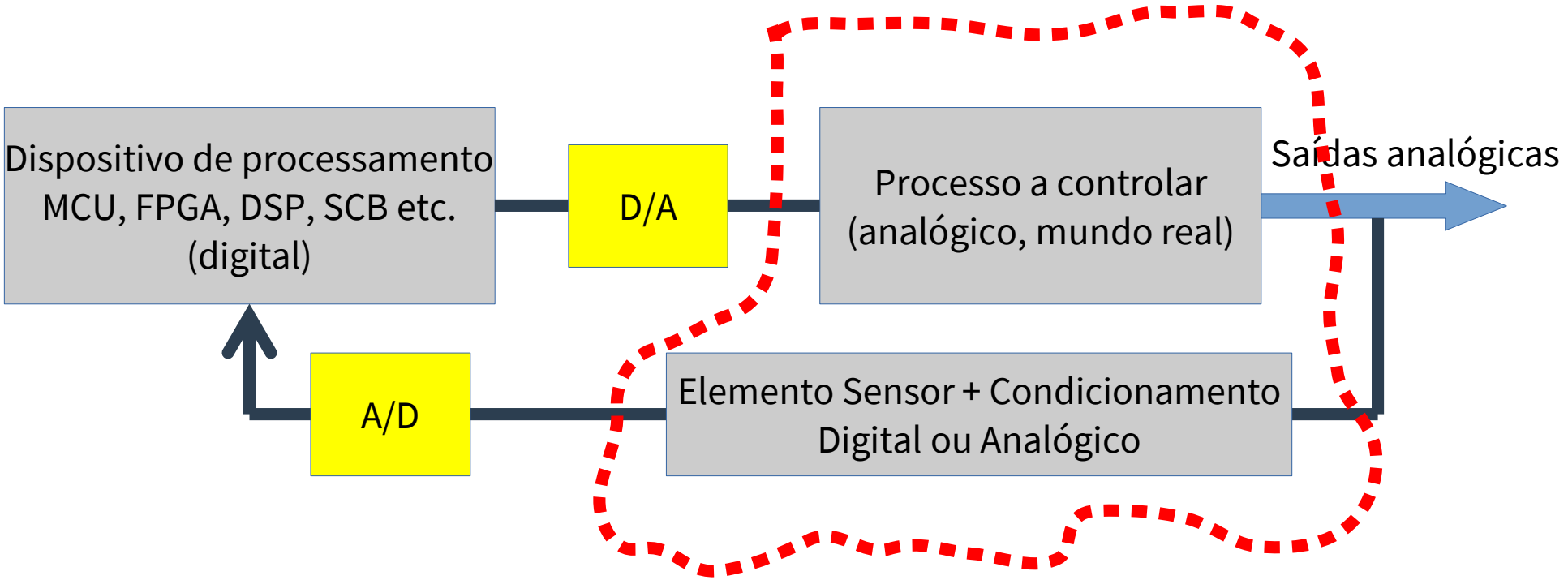
SISTEMAS EMBARCADOS

PROF. JOSENALDE OLIVEIRA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

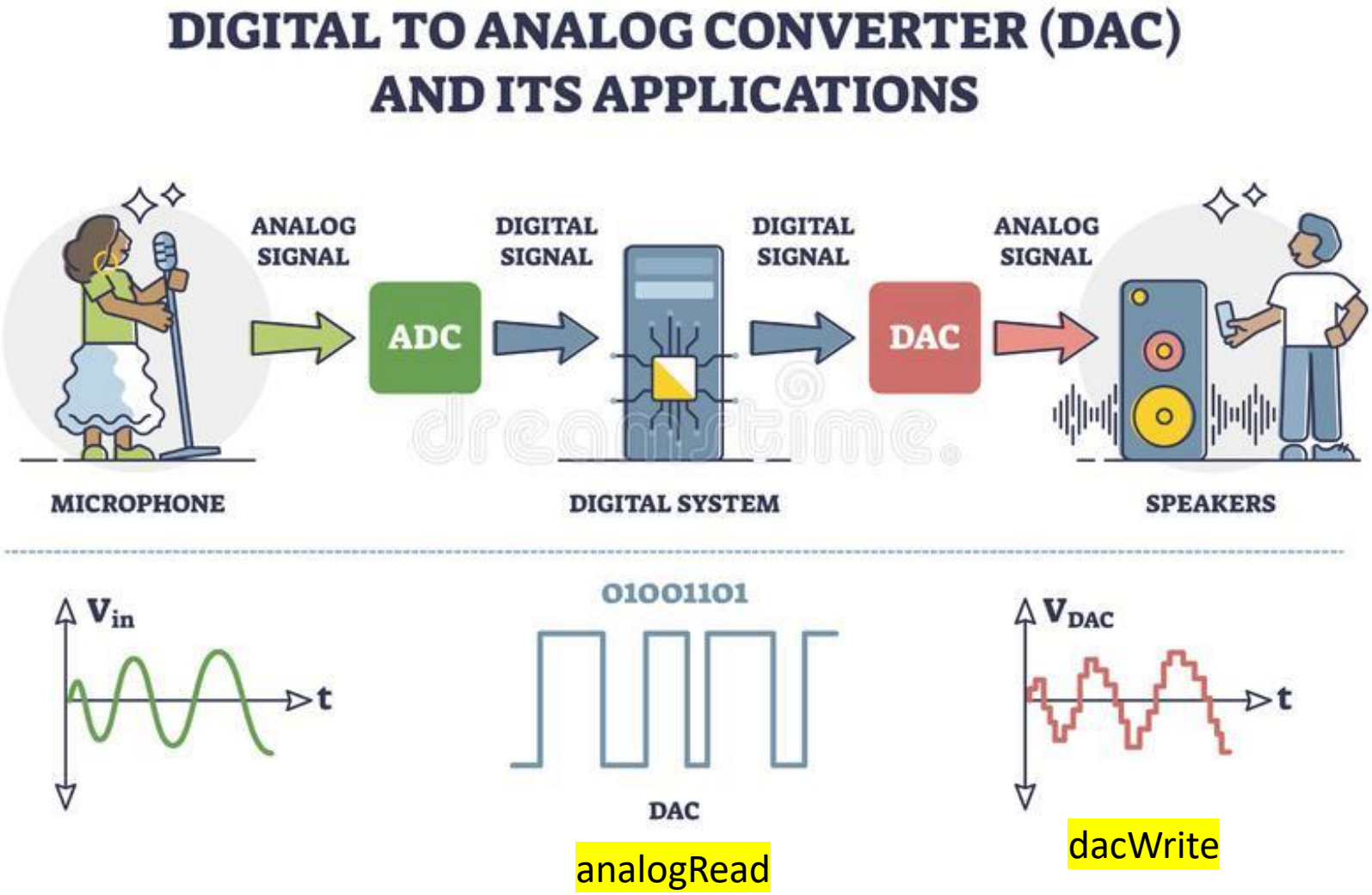
Aquisição de Dados em Sistemas Embarcados



Conversor Analógico-Digital: representar grandeza analógica por número associado para manipulação digital

Conversão Digital-Analógico: PWM é uma possibilidade, mas existem placas com tensão DC equivalente em faixas: -10V :10V

Aquisição de Dados em Sistemas Embarcados

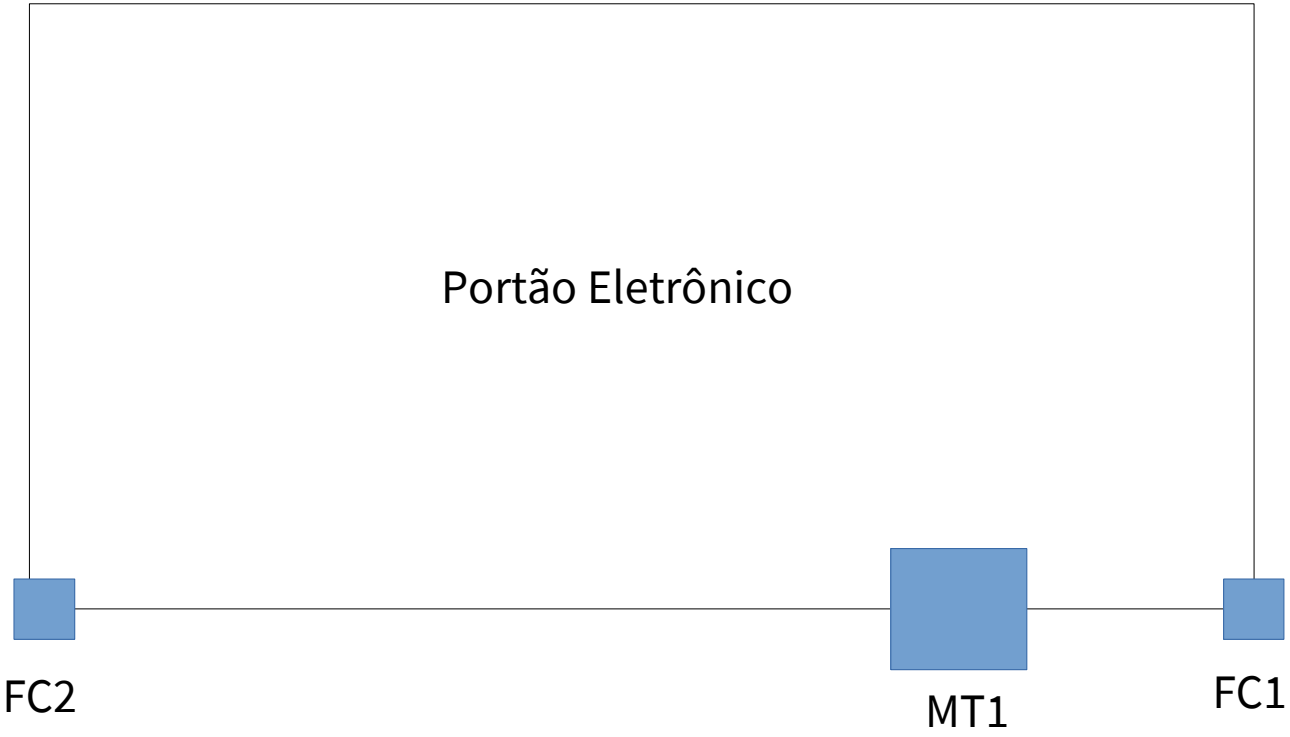


analogRead

dacWrite

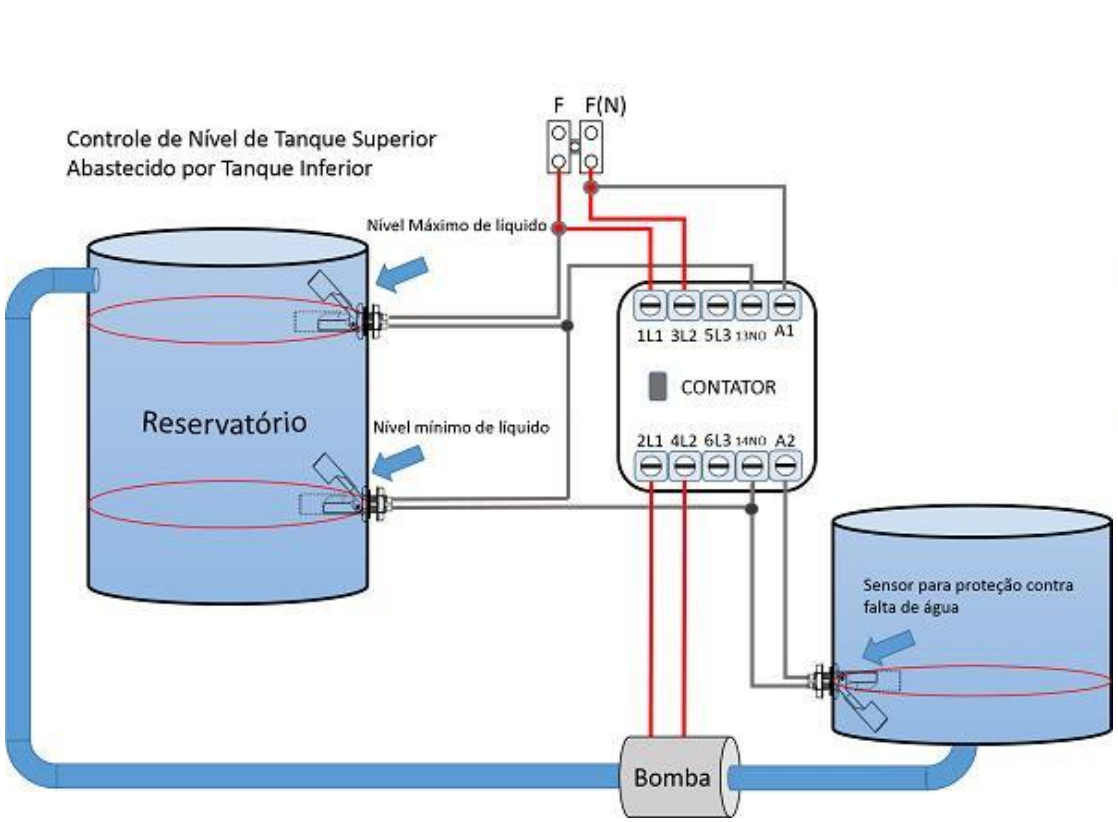
Aquisição de Dados em Sistemas Embarcados

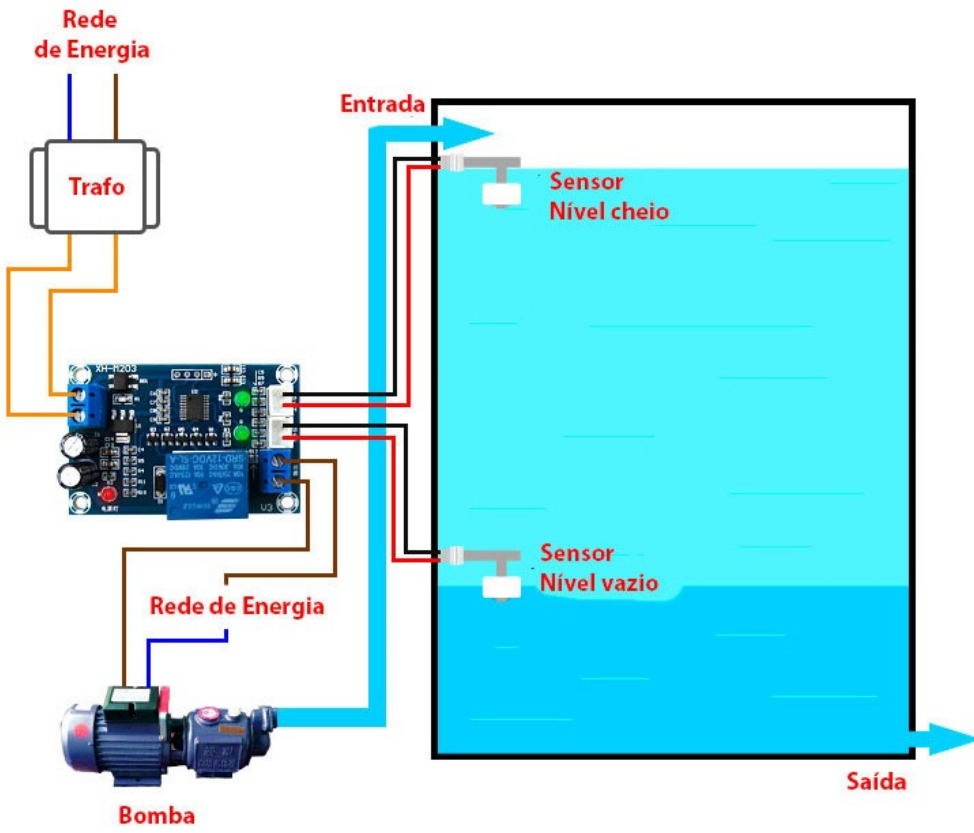
Exemplo : Entradas e Saídas Digitais



Aquisição de Dados em Sistemas Embarcados

Exemplo : Entradas e Saídas Digitais: Controle de Nível

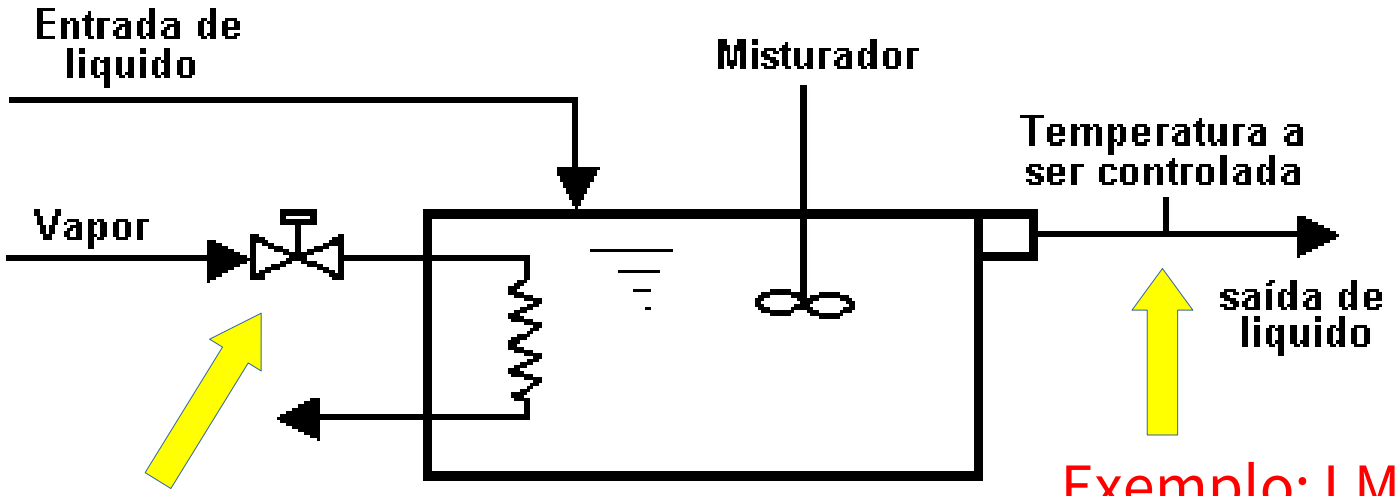




https://www.mundodaeletrica.com.br/controle-de-nivel-de-tanque-superior-abastecido-por-tanque-inferior/

Aquisição de Dados em Sistemas Embarcados

Exemplo : E se além de ligar/desligar, controlar nível, desejar também Controlar temperatura do líquido no tanque?



Exemplo: LM35

Atuador: e variável manipulada (Potência)

Normalmente sensores e atuadores com variáveis analógicas: válvula 4 a 20 mA, sensor de nível e temperatura (0 – 10 V), resistência (0 - 10V)

http://www.ebah.com.br/content/ABAAAA_TUAI/teoria-controle

Aquisição de Dados em Sistemas Embarcados

Variáveis analógicas comuns: nível, temperatura, pressão, posição, luminosidade, velocidade, pH, EC, distância, massa, ...

Para cada variável, tipos de sensores diferentes (magnéticos, capacitivos, Indutivos, resistivos...), com faixas de leituras diferentes, precisão, tipo de saída, Faixa de alimentação, linearidade, frequência de operação tempo de resposta

TRANSDUTOR: dispositivo que trabalha junto com o sensor e converte/ tranforma/adequa o valor lido para compatibilidade com a unidade de Processamento e ou meio de transmissão: ex: 1mV/°C

http://www.ebah.com.br/content/ABAAAA_TUAI/teoria-controle

Aquisição de Dados em Sistemas Embarcados

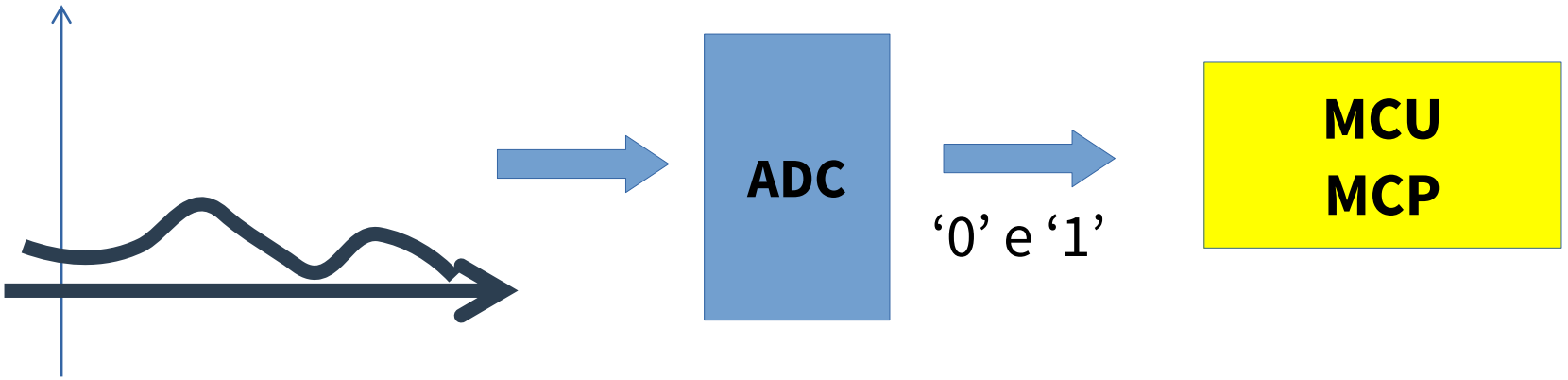
Comercialmente existem placas de aquisição e envio de dados (AD-DA): National Instruments, Advantech etc.

1.638,00 Euros

16 Canais Analógicos de Entrada, 2MSamples/s = 2MHz, 16 bits resolução, 24 entradas digitais bi-dir 02 canais analógicos de saída (-10, 10V), 04 temporizadores/contadores

Soluções low-cost para AD e processo AD

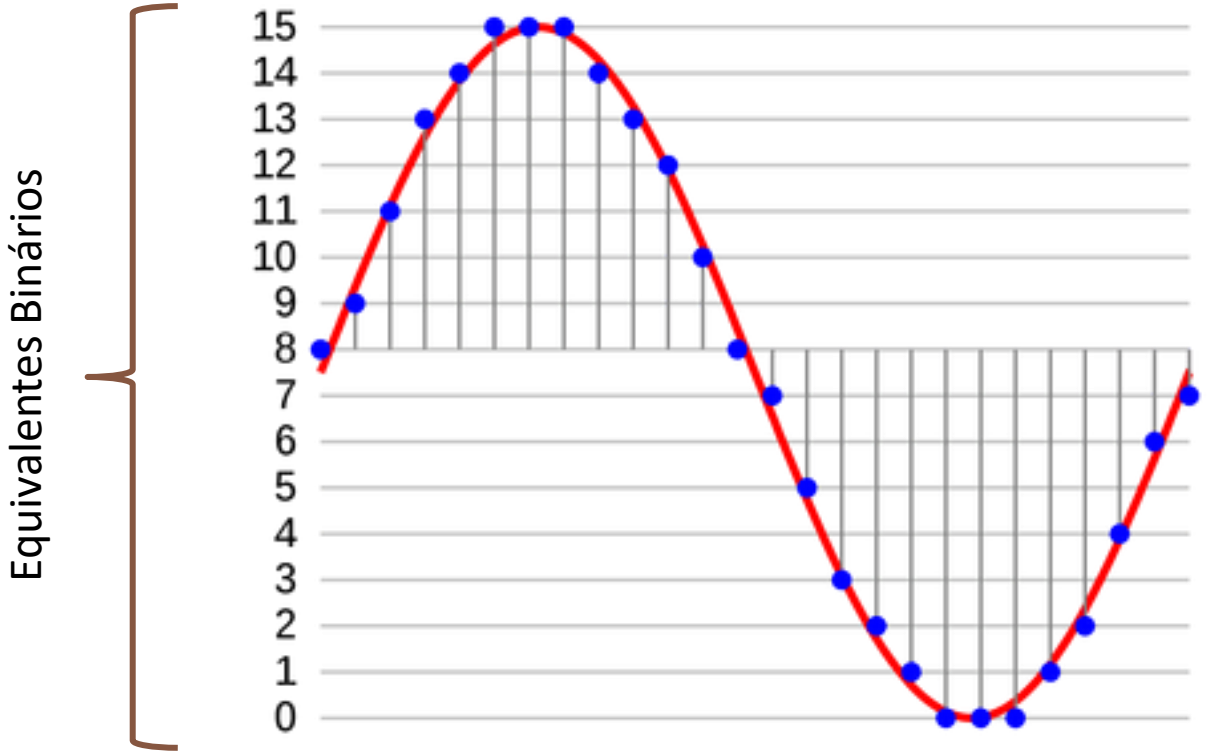
- ATmega328 (06 canais no Arduino UNO: A0-A5): AD de 10 bits de resolução
- ATmega2560 (16 canais no Arduino Mega), mesma resolução



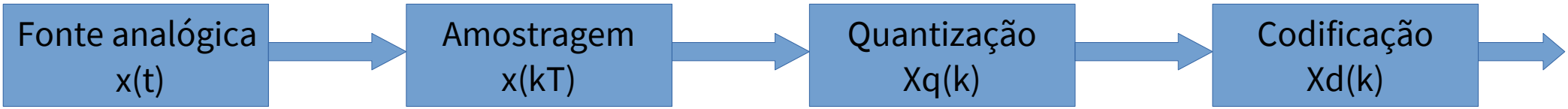
Existem várias técnicas de ADC: rampa, ***aproximações sucessivas,*** *flash, sigma-delta etc.*

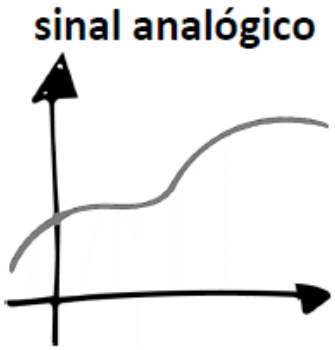
Soluções low-cost para AD e processo AD

- ADC converte sinal analógico no tempo (x(t)) em sinal amostrado no tempo x(kT), onde T é o tempo de amostragem. Estes valores são quantizados em conjuntos de valores inteiros e depois codificados em valores digitais.

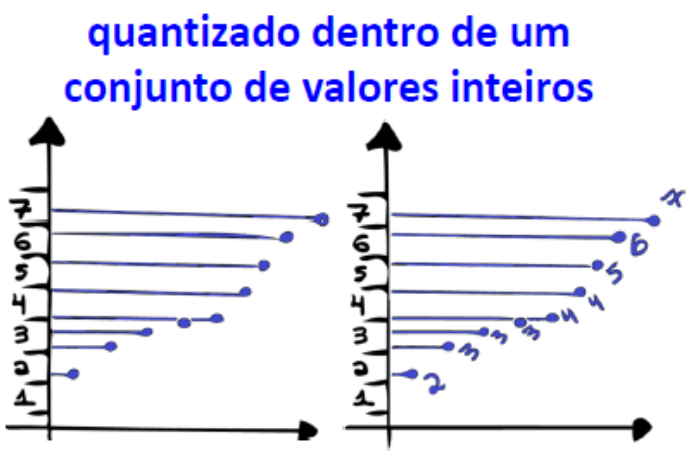


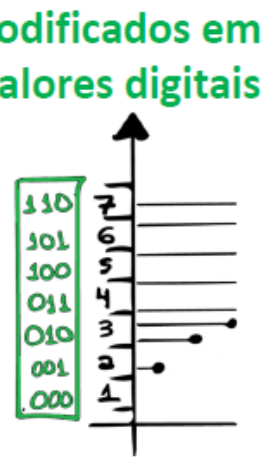
Soluções low-cost para AD e processo AD





No ATMEGA328 e 2560: 10 bits
Q = 2^10 = 1024 níveis
Cada nível: VCC/1024: 4,88mV

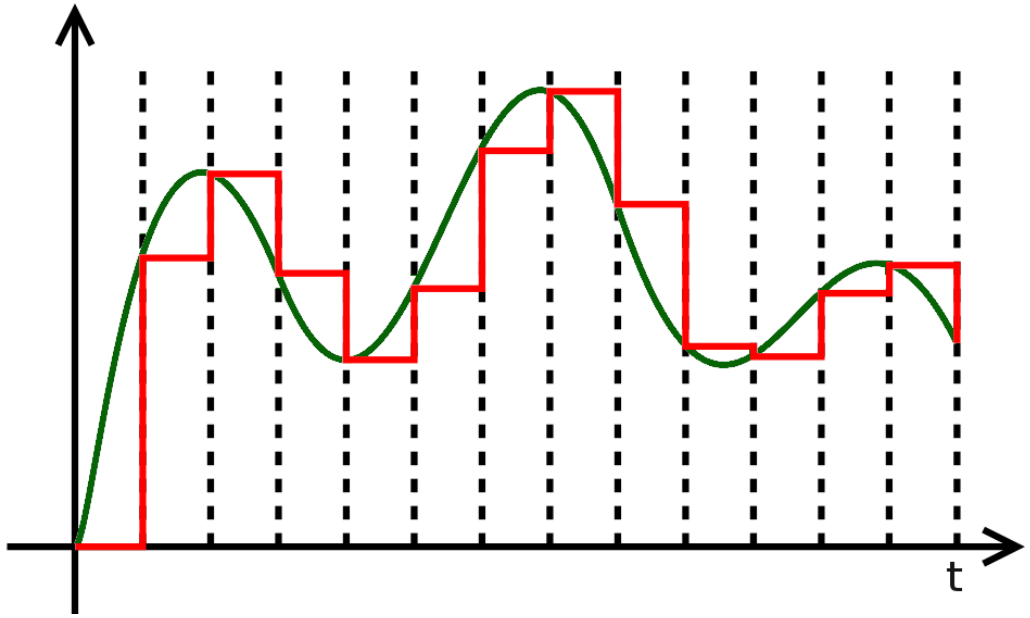




Fonte: Slides DCA0119: Sistemas Digitais. Prof. Heitor Medeiros Florencio

Soluções low-cost para AD e processo AD

- A amostragem utiliza técnica de sample-hold
- Entre um instante de amostragem e o próximo, mantém valor constante

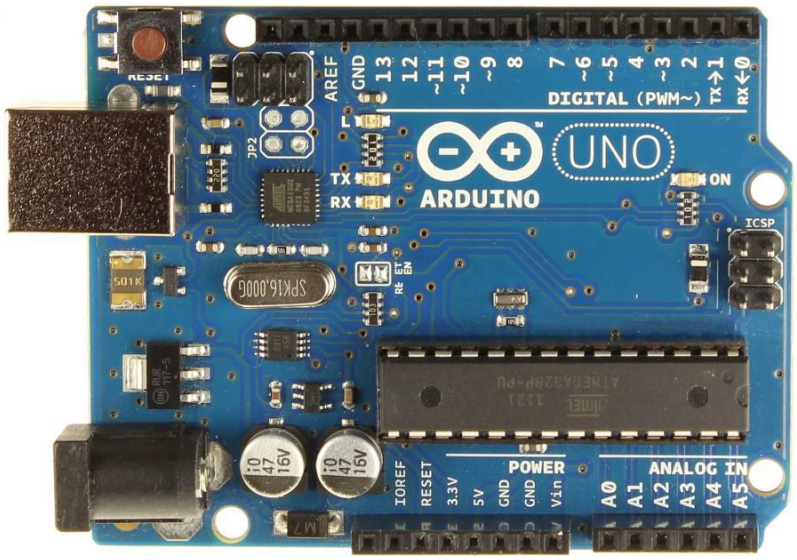


Fonte: https://www.embarcados.com.br/conversor-a-d/

Soluções low-cost para AD e processo AD

Fonte: https://www.embarcados.com.br/conversor-a-d/

- Configuração padrão do Arduino UNO – ATMEGA328P
  - Clock nominal interno: 16 MHz
  - Pre-scaler: 128 (divisor padrão)
  - Logo: 16.000.000/128 = 125 kHz (**fADC**)
  - Utiliza +-13 ciclos de clock para fazer uma conversão
  - Portanto: **+- 9,6 kHz (**frequência máxima de entrada **Fs** a ser amostrado)
  - Freq. de amostragem:



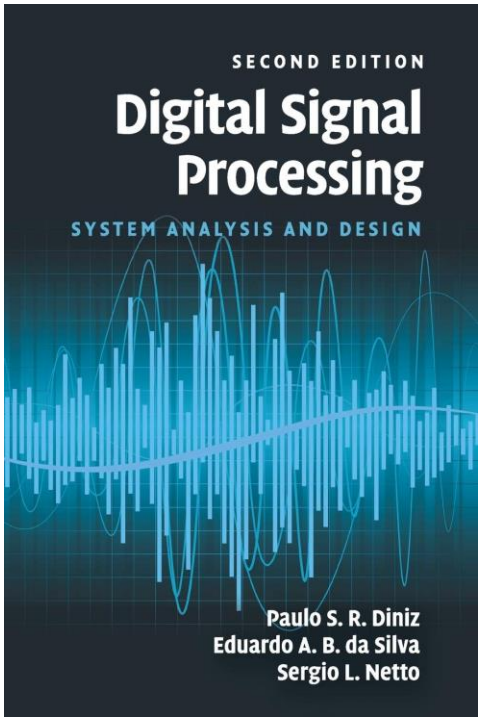
Fs >= 2 x F_in, ou seja,

F_in <= Fs/2 (Frequência de Nyquist) <= 4,8 kHz

Ou seja, com este Pre-scaler consegue-se amostrar e posteriormente reconstruir um sinal com até 4,8 kHz de frequência.

Tópicos de processamento digital de sinais (dsp)

- Ciência que estuda as regras que governam sinais discretos e os dispositivos que os processam
- Primeiro o sinal precisa ser discretizado
- https://github.com/josenalde/computing-fundamentals/blob/master/src/analog_discrete.py



- Tipos:
  - OFFLINE: não há restrição de tempo; dados a serem processados já estão armazenados;
    Permite processar sistemas não causais

CAUSAL: saída presente depende de entradas presentes e passadas
implementável tempo real

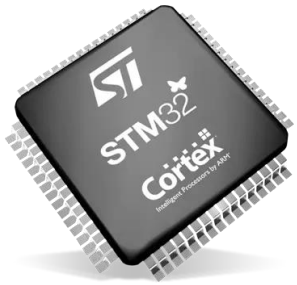
NÃO CAUSAL: saída presente depende de entradas presentes e futuras
com dados armazenados é implementável

- Tipos:
  - ONLINE: os dados são apresentados ao processador, mas o mesmo não precisa terminar o processamento do dado antes que um novo chegue.
  - REAL TIME: tempo de processamento crítico; o processamento de um dado termina para que novo chegue

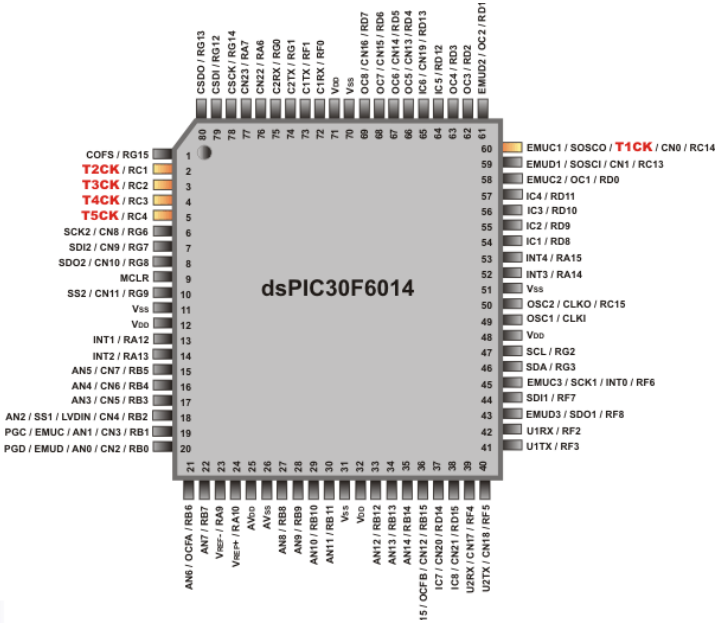
Aplicações: áudio, vídeo, telecom, imagens etc.

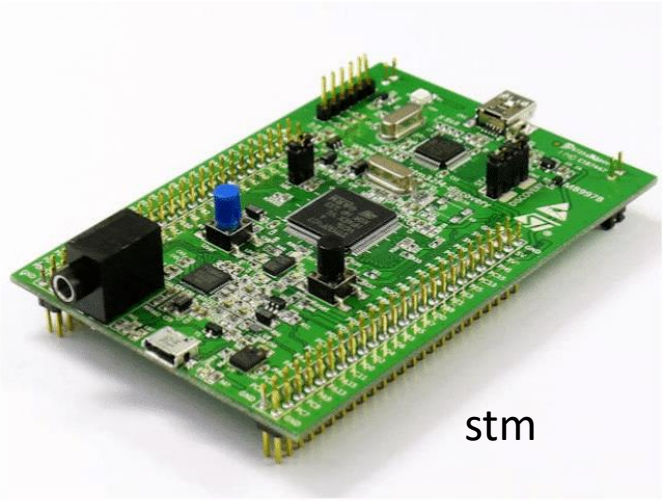
Tópicos de processamento digital de sinais (dsp)

Algoritmos: filtros, Fourier (FFT) etc.

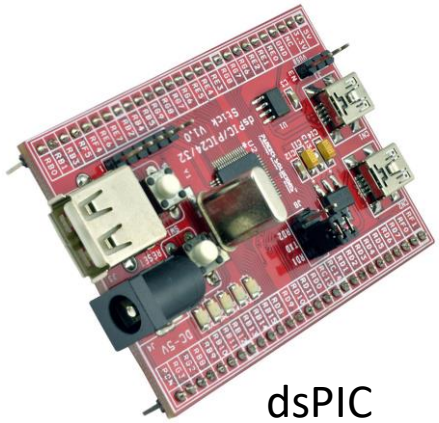


FPGA





stm



dsPIC

Soluções low-cost para AD e processo AD

Faixa de tensão para conversão

- VCC
- Internal (1,1 V)
- Referência (AREF) : max VCC 5V!
- O clock recomendado para o AD em 10 bits é entre 50 e 200 kHz
- Com cristal de 12 MHz / 64: 187 kHz / 13 = 14384 amostras/s

16 MHz / 2 = 8 MHz
16 MHz / 4 = 4 MHz
16 MHz / 8 = 2 MHz
16 MHz / 16 = 1 MHz
16 MHz / 32 = 500 kHz
16 MHz / 64 = 250 kHz
16 MHz / 128 = 125 kHz

Fonte: https://www.embarcados.com.br/conversor-a-d/

Soluções low-cost para AD e processo AD

Existem outros conversores

- ADS1115 (16 bits) e ADS1015 (12 bits): I2C (SDA, SCL)
- No PIC 18F4550: 10 bits
- No ESP8266/nodeMCU: 01 canal A0 de 10 bits

- Pode expandir com MCP3008 (SPI): 08 canais 10 bits ou MCP3208 de 12 bits e 08 canais (SPI): MOSI/MISO

MCP3008 - Conversor Analógico/Digital

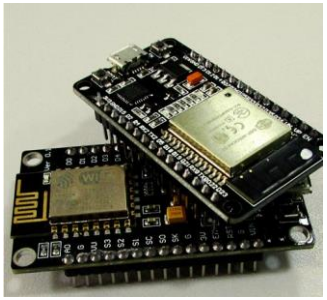
Código: MCP3008DIP

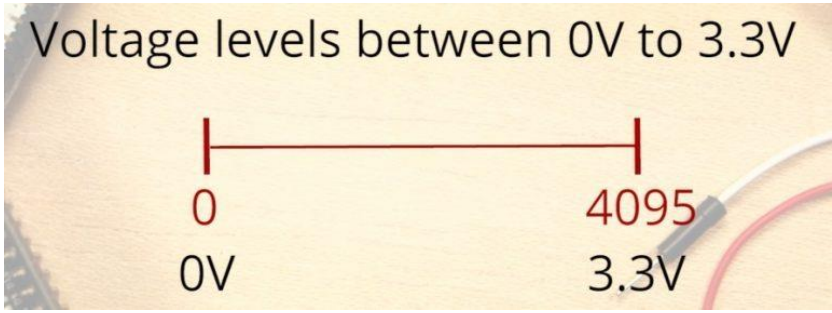
R$ 21,90

até **3x** de **R$ 7,30** sem juros
ou **R$ 20,80** via depósito

Autocore robotica

Soluções low-cost para AD e processo AD

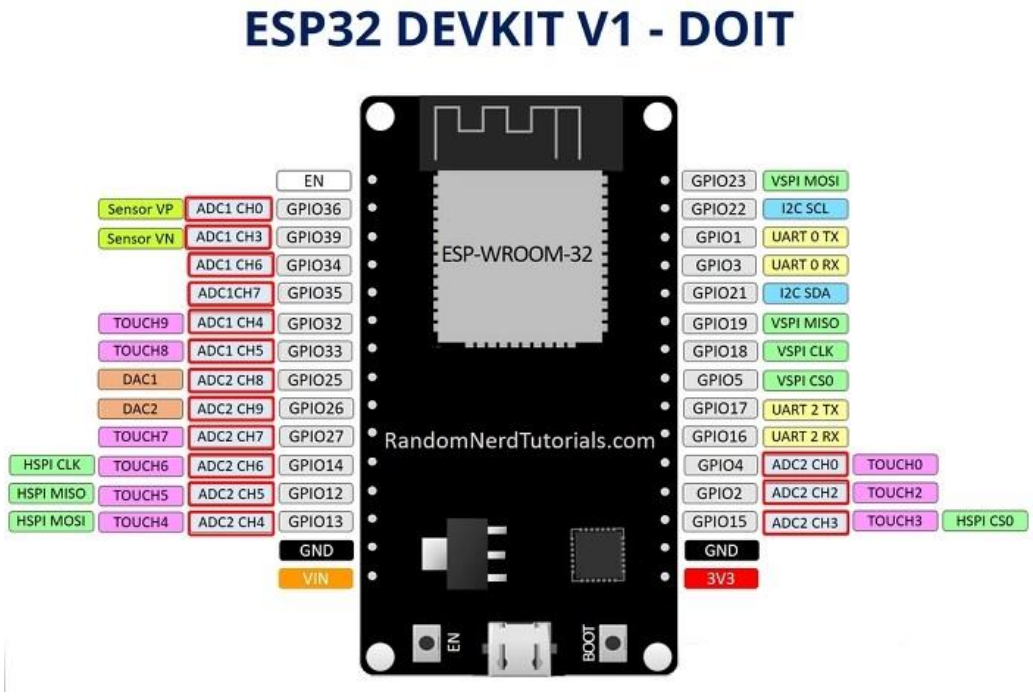




No ESP32: 15 canais

- Porta 1 (ADC1) com 06 canais (CH0,CH3,CH4,CH5,CH6,CH7)
- Porta 2 (ADC2) com 09 canais (CH0,CH2,CH3,...,CH9)
- Padrão de resolução: 12 bits (0-4095)
- Ciclos por conversão no ESP32, padrão 8
- https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/

Especificações	ESP8266	ESP32
MCU	xtensa® single 32-bit l106	xtensa® dual-core 32-bit lx6 600 DMIPS
802.11 b/g/n Wi-Fi	HT20	HT40
Bluetooth	Não	Bluetooth 4.2 Le
Frequência	80 MHz	160 MHz
SRAM	160 Kbytes	512 Kbytes
Flash	SPI Flash, 16 Mbytes	SPI Flash, 16 Mbytes
GPIO	17	36
Hardware/ Software PWM	Não/ 8 canais	1 / 16 canais
SPI/ I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	Não	1
Interface Ethernet Mac	Não	1
Sensor Capacitico	Não	Sim
Sensor de Temperatura	Não	Sim
Temperatura de Trabalho	-40°C a 125°C	-40°C a 125°C



Soluções low-cost para AD e processo AD

Exemplo de código para medir tempo de conversão Configuração padrão

- Entre 112us e 116us
- Com scaler de 64,
- 60 us (dobra freq)
- em torno de 16kHz

```
unsigned long tempo_inicio;
unsigned long tempo_fim;
unsigned long valor;

void setup() {
 Serial.begin(115200);    //inicia a comunicação serial
}

void loop() {
   // leitura
   tempo_inicio = micros(); //marca tempo de inicio de leitura
   valor = analogRead(0);  //le valor convertido
   tempo_fim = micros();   //le tempo no fim da conversão

   //exibe valor lido e tempo de conversão
   Serial.print("Valor = ");
   Serial.print(valor);
   Serial.print(" -- Tempo leitura = ");
   Serial.print(tempo_fim - tempo_inicio);
   Serial.println(" us");
   delay(500);
}
```

Soluções low-cost para AD e processo AD

Exemplo de código para medir tempo de conversão Configuração padrão

- Entre 112us e 116us
- Com scaler de 64,
- 60 us (dobra freq)
- em torno de 16kHz

```
// Variáveia para armazenar os resultados

unsigned long tempo_inicio;
unsigned long tempo_fim;
unsigned long valor;
double Ts;

// constante para configuração do prescaler ADPS2 = 2: 0010, ADPS1 = 1: 0001, ADPS = 0: 0000
const unsigned char PS_16 = (1 << ADPS2); // 2^ADPS2 = 4 = 100
const unsigned char PS_32 = (1 << ADPS2) | (1 << ADPS0); // 2^ADPS2 = 100 | 2^ADPS0 = 001, 100 | 001 = 101
const unsigned char PS_64 = (1 << ADPS2) | (1 << ADPS1); // 100 | 010 = 110
const unsigned char PS_128 = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // 111

void setup() {
 Serial.begin(115200);
 Serial.println(ADPS2);
 Serial.println(ADPS1);
 Serial.println(ADPS0);

 // configura o preescaler do ADC
 ADCSRA &= ~PS_128;  //limpa configuração da biblioteca do arduino ADCSRA = ADCSRA & NOT(111) = x x x x x x x x
                     //                                                                        & 1 1 1 1 1 0 0 0
                     //                                                                          x x x x x 0 0 0

 // valores possiveis de prescaler só deixar a linha com prescaler desejado
 // PS_16, PS_32, PS_64 or PS_128
 //ADCSRA |= PS_128; // 64 prescaler
 ADCSRA |= PS_64; // 64 prescaler ADCSRA = ADCSRA OR 00000110
 //ADCSRA |= PS_32; // 32 prescaler
 //ADCSRA |= PS_16; // 16 prescaler
}
```

Soluções low-cost para AD e processo AD

Exemplo de código para medir tempo de conversão Configuração padrão

- Entre 112us e 116us
- Com scaler de 64,
- 60 us (dobra freq)
- em torno de 16kHz

```
void loop() {
 // leitura
   tempo_inicio = micros(); //marca tempo de inicio de leitura
   valor = analogRead(0);  //le valor convertido
   tempo_fim = micros();   //le tempo no fim da conversão
   Ts = (1.0/(tempo_fim - tempo_inicio))*1000;

   //exibe valor lido e tempo de conversão
   Serial.print("Valor = ");
   Serial.print(valor);
   Serial.print(" -- Ts = ");
   Serial.print(Ts, 2);
   Serial.println(" kS/s");
   delay(500);
}
```

Soluções low-cost para AD e processo AD

Em testes no embarcados.com.br, conseguiu manter os 10 bits até 1 Mhz (pre scaler 16), com Fs de 50kHz!

ADCSRA – ADC Control and Status Registe A

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 2:0 – ADPS2:0: ADC Prescaler Select Bits

Configura o fator de divisão entre o clock do sistema e a entrada de clock do ADC. Os valores possíveis são exibidos na tabela abaixo:

ADC Prescaler Selections

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