

Ressumo AC.

Níveis de Abstração:

Mecanismos de conversão entre níveis de utilização

Compilador: convertem programa para nível inferior

Interpretador: Executam programa de nível superior

Assembleia: convertem o programa para código máquina

Linker: ligam diversos módulos de 1 programa para 1 executável.

Loader: Carregam programa para o RAM.

Sistemas Embutidos:

São sistemas informáticos com apenas 1 tarefa, que não precisam da interacção dos utilizadores.

Representação de dados:

• Propriedades:

- Domínio - Valores que um tipo de dado pode assumir
- Gama de Variação - N.º de Valores que um dado pode assumir
- Precisão - Distância entre dois valores consecutivos
- Operações - Operações permitidas sobre dados

Classificação:



MSB - Bit mais significativo (o da esquerda)

LSB - Bit menos significativo (o da direita)

Códigos Binários

Propriedades

N.º Bits	Gama de variação	Nome
N	0.. $2^N - 1$	
8	0..255	Byte
16	0.. 65535	Word
32	0.. 4294967295	dWord

Representação de números inteiros negativos

Códigos bipolarares mais comuns são:

- Sinal e Valor absoluto
- Complemento para 1
- Complemento para 2

Sinal e Valor absoluto

Bit de Sinal + Valor absoluto

$$19 \Rightarrow 0010011$$

$$+19 \Rightarrow 0 + 0010011$$

$$-19 \quad 1 + 0010011$$

Complemento para 1

Para converter um valor para complemento para 1, basta trocar o "1" por "0" e o "0" por "1".

Exemplo

$$79 \rightarrow 00010011$$

$$-79 (\text{complemento para } 1) \rightarrow 11101100$$

Soma Algebra em complemento para 1:

$$(10) + (-2) = 8$$

$$10 \rightarrow 0101$$

$$2 \rightarrow 0010$$

$$-2 (\text{complemento para } 1) \rightarrow 1101$$

$$\begin{array}{r} 0 \\ 100101 \\ -2 + 11101 \\ \hline 100111 \\ + \\ \hline 01000 = 8 \end{array}$$

Existem situações em que o resultado da soma algebra não é representável com o mesmo número de bits utilizado para os operandos. Nesta situação dizemos que ocorreu **overflow**.

Complemento para 2

Para converter um número para complemento para 2, temos que copiar o menor da direita para a esquerda o 1: "1" depois inverter o resto.

$$79 \rightarrow 00010011$$

$$-79 (\text{complemento para } 2) \rightarrow 11101101$$

Soma Algebra em Complemento para 2:

$$8 + (-3)$$

$$8 \rightarrow 1000$$

$$3 \rightarrow 11$$

$$-3 \rightarrow 101$$

$$\begin{array}{r} 0 \\ 01000 \\ + 11101 \\ \hline \cancel{1}00101 \\ \text{Despreza} \end{array} \quad 00101 = 5$$

Organização Funcional

Modelo de Arquitetura de Von Neumann

CPU - Unidade Central de Processamento

- Leitura de Instruções da memória
- Execução de Instruções
- Leitura de Dados
- Escrita de Resultados

Memória Principal

- Dispositivo com capacidade para armazenar informação digital binária, organizada em blocos de 8 bits
- Armazena dados, instruções e resultados
- Dividida em dois tipos: RAM - Random Access Memory e ROM - Read Only Memory

Unidades de I/O

- São utilizadas para comunicar com o exterior (periféricos)

Bus de Sistema

- Conjunto de linhas (ligações) que transportam a informação digital binária
- Permitem comunicação entre o CPU, a memória e as Unidades de I/O
- Este Bus é constituído pelo Bus de Dados, Bus de Endereço e Bus de Controlo

CLK - clock

- Define a frequência de operação do sistema. É utilizado para sincronizar as operações

Bus de Dados

- Conjunto de linhas (ligações físicas) por onde se transporta a informação digital binária (instruções, dados, resultados) entre o CPU, memória e I/O (bidirecional).
- A largura do Bus de dados é dada pelo nº de bits do Bus ou pelo nº de bits do microprocessador.

Bus de Endereços

- Conjunto de ligações físicas que transportam o endereço dos celhos de memória ou das portas de I/O (unidirecional)
- A largura do bus de endereços é dada pelo nº de linhas do bus ou pelo nº de bits, e define a capacidade de endereçamento.

Bus de Controlo

- Contém os sinais necessários para o protocolo de implementação
- M/I/O - controla o acesso à memória ou unidades de I/O
- RD - Operações de Leitura
- WA - Operações de Escrita

Organização funcional

- A organização é em celhos, constituídos por 8 bits, cada celho tem 1 endereço.
- A palavra mínima do computador é igual à unidade de memória mínima endereçável.
- O arranjoamento de dados é baseado no modelo "Little Endian".

Little Endian \rightarrow bytes menos significativos não correspondem aos endereços menos significativos.

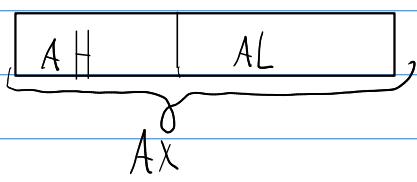
Lectura de celhos com endereço X:

- CPU ativa linha de controlo MEHA
- CPU coloca o valor X nas linhas A19-A0 do Bus de Endereços
- A memória decodifica o endereço X, e ativa o celho de memória correspondente
- Como o MEHA está ativado, o conteúdo do celho é colocado no registo de dados.

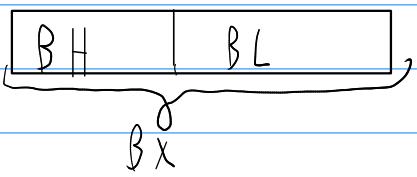
Escrita de celul com endereço X:

- O CPU ativa o MENW
- O CPU coloca o valor X nas linhas A19-A0 do BUS de Endereços
- A memória desodifica o endereço X, e ativa o celul correspondente.
- Como o MENW foi ativado o conteúdo no Registro de Dados é colocado no celul ativo.

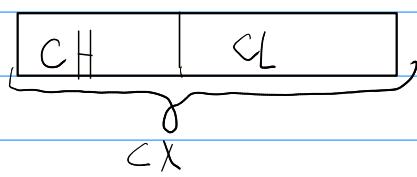
Registros de uso genérico



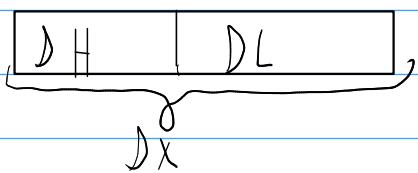
- AX - Registro de 16 bits
- AH - Registro de 8 bits (+ significativo)
- AL - Registro de 8 bits (- significativo)
- Registro acumulador
- Implicito em algumas instruções.



- BX - Registro de 16 bits
- BH - Registro de 8 bits (+ significativo)
- BL - Registro de 8 bits (- significativo)
- Registro de base
- Utilizado para endereçar variáveis em memória.



- CX - Registro de 16 bits
- CH - Registro de 8 bits (+ significativo)
- CL - Registro de 8 bits (- significativo)
- Registro acumulador
- Implicito em algumas instruções como contador



- DX - Registro de 16 bits

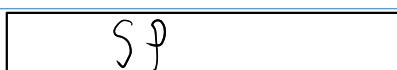
- DH - Registro de 8 bits (+ sinal)

- DL - Registro de 8 bits (- sinal)

- Registro de Dados

- Utilizado em algumas operações aritméticas

- Utilizado em instruções de I/O



- SP, Registro de 16 bits

- Stack Pointer

- Utilizado para referenciar variáveis na pilha do sistema



- BP, Registro de 16 bits

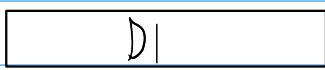
- Base Pointer

- Utilizado para referenciar parâmetros e variáveis locais em subrotinas



- SI, Registro de 16 bits

- Source Index



- DI, Registro de 16 bits

- Destination Index

IP

- Instruction Pointer (IP)
- Registro de 16 bits
- Contém o Endereço da 1^a instrução a ser executada

Registers de Segmentos

Existem 4 registros de Segmentos:

- CS - "Code Segment" - Armazena as instruções do programa
 $CS: [IP]$ - Ponteiro para a próxima instrução a ser executada

- DS - "Data Segment" - Armazena dados/resultados relativos a variáveis
 $DS: <deslocamento>$

- SS - "Stack Segment" - Suporte à programação orientada a subprogramas; passagem de parâmetros
 $SS: <deslocamento>$

- ES - "Extra Segment" - Segmento auxiliar usado na manipulação de outros endereços de variáveis

Segmentos de memória

Endereço logico:

(Segmento) : [Deslocamento]
(Segmento) : [Offset]
(Segmento) : [Desplacamento]

Endereço físico / linear

Segmento \times 16 ($\times 16h =$ associar com 0)
+ deslocamento
Endereço físico / linear

Endereço Efetivo

Instruções Aritméticas

Instruções	Parâmetros	Descrição
ADD	a, b	$a := a + b$
ADC	a, b	$a := a + b + \text{carry}$
NEG	a	$a := -a$
SUB	a, b	$a := a - b$
SUBB	a, b	$a := a - b - \text{carry}$
MUL	b	$Ax := Ax \times b$
IMUL	b	$Ax := Ax \times b$
DIV	b	$Ax := Ax / b$
IDIV	b	$Ax := Ax / b$
INC	a	$a := a + 1$
DEC	a	$a := a - 1$

Instruções Lógicas

Existem 4 instruções lógicas:

• AND - Se quando ambos valores são 1 é 1, senão é 0

• OR - Se quando ambos valores são 0 é 0, senão é 1

• XOR - Quando ambos valores são iguais é 0, senão é 1

• NOT - complemento o valor, se era 1 para 0, se era 0 para 1

tablas

Unidimensional

C

ASSEMBLY

$i=3$

$\text{TAB}[i] = 50$

MOV $i, 3$

MOV $Ax, _$ // Somatorio de los elementos

MUL i

MOV Bx, Ax

MOV Word PTR [TAB[Bx]], 50

bidimensional

Unigned char TAB[4][3]

C

$\text{TAB}[1][2] = 50$

ASSEMBLY

MOV B91E PTR [TAB[1][3][2]], 50

1º Celda

Tomando

Nº Elemento Celda

Nº Columna

//Supiendo que TAB es [0044]

MOV B91E PTR [0044][1][2], 50

MOV B91E PTR [0044][5], 50

MOV B91E PTR [0044], 50

Unigned char TAB[4][3]

Unigned int i,j;

ASSEMBLY

C

MOV $i, 1$

MOV $j, 2$

MOV $Ax, 1$ // Tomando de los elementos

MOV $Bx, 3$ // N° elementos

MUL Bx // $Ax = Ax \times Bx$

MUL i

MOV Bx, Ax

$i=1;$

MOV $Ax, 1$ // Tomando de los elementos

$j=2;$

MUL j // $Ax = Ax \times j$

$\text{TAB}[i][j] = 50$

ADD Bx, Ax // $Bx = Bx + Ax$

MOV B91E PTR [TAB[Bx]], 50

ADICÃO com 32 BITS

long Lxx, Lyy, Lzz

C

$$Lxx = Lyy + Lzz$$

ASSIGNMENT

MOV AX, WORD PTR Lzz

ADD AX, WORD PTR Lyy

MOV WORD PTR Lxx, AX

MOV WORD PTR Lzz[0]

ADC AX, Word PTR Lyy[0]

MOV WORD PTR Lxx[0], AX

Divisão inteira

$$X = q \% Z$$

MOV AL, Y

MOV AH, 0 → armazena para 16 bits

DIV Z

MOV X, AH → Resto

$$X = q / Z$$

MOV AL, Y

MOV AH, 0

DIV Z

MOV X, AL → Quociente da divisão inteira

Deslocamento

O Deslocamento lógico preenche a nova posição com 0 (SHR e SHL)

O Deslocamento aritmético preenche a nova posição com uma cópia do bit de sinal. (SHR e SAL)

SHL e SAL

Aplicação: Multiplicação rápida e eficiente, deslocar 1 bit para o esquerda é o mesmo que multiplicar por 2

Deslocar para o esquerda n bits é o mesmo que multiplicar por 2^n

SHR e SAR

Aplicação: Divisão rápida e eficiente

Deslocar para o direito n bits é o mesmo que dividir por 2^n

ROL

O ROL desloca todos os bits para o esquerda, o bit mais significativo vai para o carry e o carry para o bit menos significativo

ROR

O ROR desloca todos os bits para o direito, o bit menos significativo vai para o carry, e o carry vai para o bit mais significativo

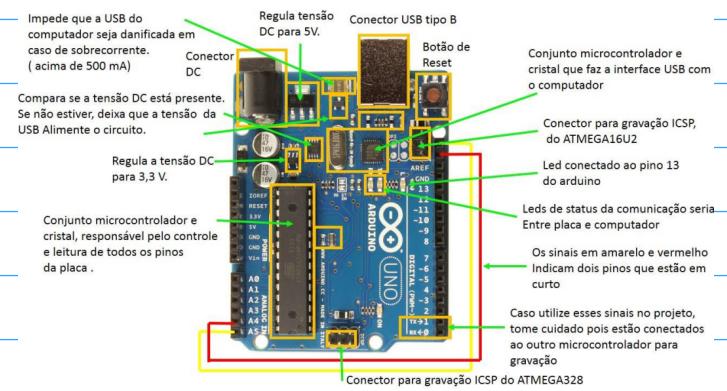
RCL (Rotate Left)

A instrução RCL desloca cada bit para a esquerda e copia o flag do zero para o bit menor significativo, e o bit mais significativo para o flag do carry

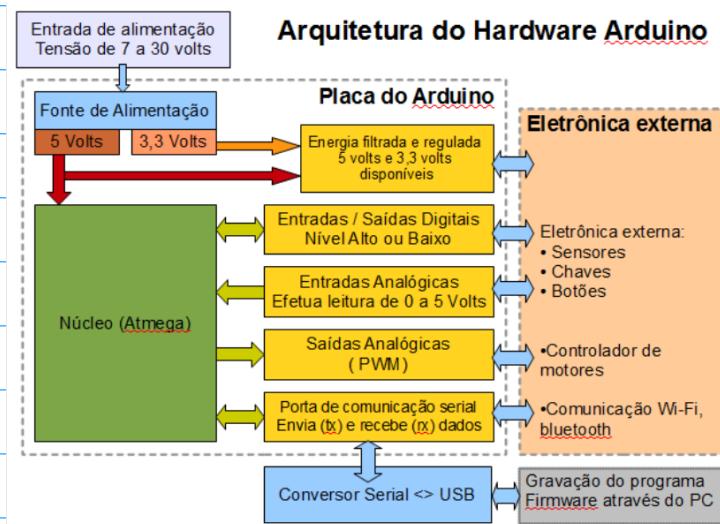
RCR (Rotate Right)

A instrução RCR desloca cada bit para a direita e copia o flag do carry para o bit mais significativo, e o bit menos significativo para o flag do carry

Arduino



Arquitetura do Arduino



Microcontrolador

Um microcontrolador é desenvolvido de forma a integrar diversos componentes em um único circuito integrado

Características da Arduino Uno

Microcontrolador: ATmega328

Tensão de operação: 5V

Tensão recomendada: 7V - 12V

Límite da tensão de entrada: 6-20V

Pinos digitais: 14

Entradas analógicas: 6 pinos

Corrente contínua por pino: 40 mA

Corrente para o pino de 3.3V = 50 mA

Quantidade de memória FLASH: 32 kB (0.5 kB pra o bootloader)

Quantidade de memória SRAm: 2 kB

Quantidade de memória EEPROM: 1 kB

Velocidade de clock: 16 MHz

Arduino Uno - Alimentações

O arduino é alimentado com 5V

Pode ser alimentado por uma porta USB, ou entrada "Power jack" com fonte externa DC

A fonte externa tem que ter entre 7-12V, e ligada na conexão da fonte ou nos pinos Vin e Gnd

3.3V - Fornece 3.3V para alimentação de shield e módulos externos, corrente max de 50 mA

5V - Fornece tensão de 5V para alimentação de shields e módulos externos

GND - Pinos de referência, terra.

VIN - Pino de alimentação por bateria ou shield. Se o pino é alimentado pelo conector jack, a tensão da fonte não muda pra.

Arduino - Funções base

Para o programação em Arduino existem 2 funções fundamentais:

- void setup() - É executada 1 vez e serve para a inicialização de variáveis, bibliotecas, e definição de pinos (como input ou output)
- void loop() - como o próprio nome indica, faz um loop seletivo, o que permite a sucessiva leitura de portas e parâmetros de sensores externos

Arduino constantes

True - valor lógico verdadeiro

False - valor é falso

HIGH - Porta ativada, a 5V

LOW - Porta desativada, a 0V

INPUT - Entrada de dados

OUTPUT - Saída de dados

Arduino - Portas digitais

As portas digitais trabalham com valores definidos, ou seja, 0V ou 5V.

Escrita em uma porta digital: digitalWrite(pino, estado);

Lectura em uma porta digital: digitalRead(pino);

Sobre PWM

3 pinos com PWM pôr: 3, 5, 6, 9, 10, 11.

Q) Que é PWM

PWM (Pulse Width Modulation) é uma técnica usada para controlar a intensidade média de um sinal, em forma de onda.

Portas Analogicas

Existem 6 portas analogicas no arduino Uno (A0, A1, A2, A3, A4, A5).

Para ler o valor de uma porta analogica usamos analogRead(pino);

Operadores no arduino

	Aritméticos	Relacionais	
+	Adição	>	Maior
-	Subtração	<	Menor
*	Multiplicação	\geq	Maior ou igual
/	Divisão	\leq	Menor ou igual
%	Resto da divisão inteira	$= =$	Igual
		$!=$	Diferente

& &	e (and)
	ou (or)
	não (not)

Operadores Compostos

$++$	Incremento
$--$	Decremento
$+=$	Adição com atribuição
$-=$	Subtração com atribuição
$*=$	Multiplicação com atribuição
$/=$	Divisão com atribuição

Características Fundamentais

Finitude - um algoritmo deve ter sempre um fim

Definição - As opções têm que ser bem definidas

Entrada - O ou + entradas, quantidades que não fornecidas antes do algoritmo iniciar

Saída - 1 ou + saídas, quantidades que têm uma relação específica com as entradas

Eficiência - Ser eficiente

Representações de Algoritmos

Linguagem Natural - Expresso em linguagem natural (Português)

Pseudo-código - Linguagem intermediária entre o natural e a de programação

Fluxograma - Representação gráfica que usa formas geométricas para indicar muitas ações e decisões que devem ser executadas para resolver o problema.

Interrupções

As interrupções param o programa e fazem com que ele execute uma outra função. Para isso acontecer, usamos a função attachInterrupt.

attachInterrupt (digitalPinToInterrupt(yina), função, CHANGE);

Classificação de sistemas embutidos

Primeira geração

- Microprocessadores de 8 bits (8085, Z80, etc.) e microcontroladores de 8 bits
- Hardware simples com firmware desenvolvido em assembly

Segunda geração

- Microprocessadores de 16 bits e microcontroladores de 8 ou 16 bits
- Alguns contêm sistemas operativos embutidos
- Processadores como Intel, Pentium ou Motorola apareceram
- O conjunto de instruções dos processadores tornar-se mais complexo e poderoso

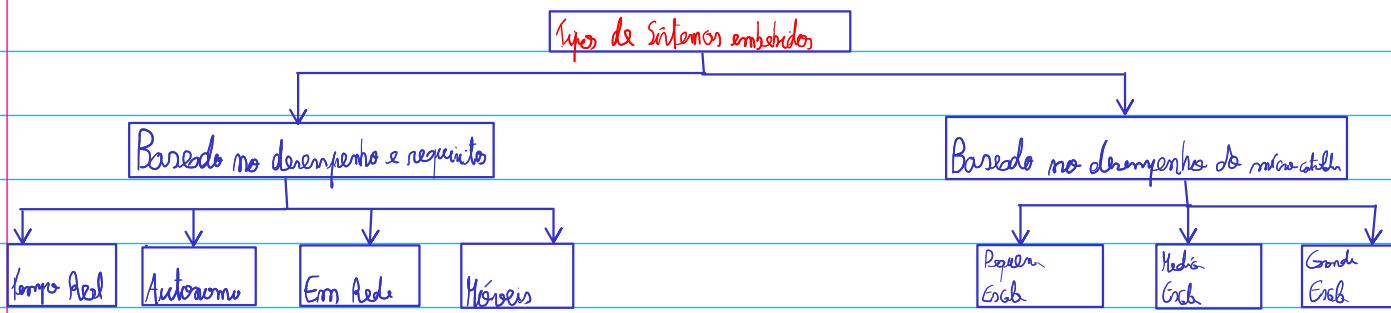
Terceira geração

- Microprocessadores de 32 bits e microcontroladores de 16 bits
- Novo conceito de processadores específicos de aplicação
- Pipelining de instruções
- S.O. embutido em tempo real
- Controlo de Robótica e Processos Industriais
- Mercado de microprocessadores mais competitivo e amplo.

Quarta geração

- Processadores reconfiguráveis e processadores multi-core
- S.O. embutido em tempo real de alto desempenho
- Conceito de sistemas em chips

Classificações de sistemas embutidos



Tempo Real

Existem 3 tipos de sistemas tempo-real

Forte (hard): O não cumprimento de um tempo de resposta a um evento conduz a uma falha do sistema.

Fmeye (firm): Tolerada uma baixa probabilidade no não cumprimento de um tempo de resposta conduz a uma falha.

Frouxo (soft): O não cumprimento de um tempo de resposta a um evento conduz a uma degradação do desempenho.

Autônomo

Os sistemas independentes são simples. A entrada é recebida pelos pinos de I/O, podendo ser analógica ou digital.

O MCU processa a entrada, toma a decisão e executa a saída.

Em Rede

Este tipo de sistema está ligado a uma rede, comunicando com um servidor ou não. A comunicação pode usar LAN, WAN ou outros protocolos.

Móvel

São sistemas portáteis, que têm limitações tais como memória.

Pequena Escala

São sistemas simples, tendo microprocessadores e microcontroladores de baixa performance e custo.

Média Escala

São sistemas ligeiramente complexos, tendo microprocessadores e microcontroladores de performance média e baixo custo, normalmente têm um S.O.

Grande escala

Sistema com hardware e software complexos, com alta performance. Tem um S.O. de alta performance em Tempo Real. Estes sistemas têm interfaces gráficas, portas de comunicação com I_DC, CAN, RS₄₈₅, Ethernet, USB

Vantagens de usar Microprocessadores no domínio de um SE:

Maior velocidade de clock e tamanho da palavra

Maior capacidade de processamento

Operações Básicas

Usar o ALU para executar operações aritméticas

Mover dados de um local de memória para outro

Tomar decisões e saltar para a execução de um novo conjunto de instruções

Tipos de memória comuns em Sistemas Embutidos

RAM:

- DRAM: Preciso de Refreshamento periódico, usada na memória principal
- SRAM: Preciso de Refreshamento periódico, Mais rápida, pequena e cara. Menor consumo de energia

ROM:

- PROM: Programada quando fabricada
- EPROM: Pode ser programada pelo utilizador (apenas programar 1 vez)
- EEPROM: Pode ser programada muitas vezes (eletronicamente) e é apagada por ligar 0.V.

Híbrido:

NVRAM: Memória tipo RAM, guarda informação mesmo sem alimentação

EEPROM: Programada eletronicamente, escrita lenta, apaga eletronicamente, apaga byte a byte em um chip todo.

FLASH: Programada eletronicamente, apaga eletronicamente, apaga em bloco em chip, Grande durabilidade, baixo custo.

Transdutor:

Transforma 1 tipo de energia num outro

Sensor: INPUT, tem um transdutor que permite obter info.

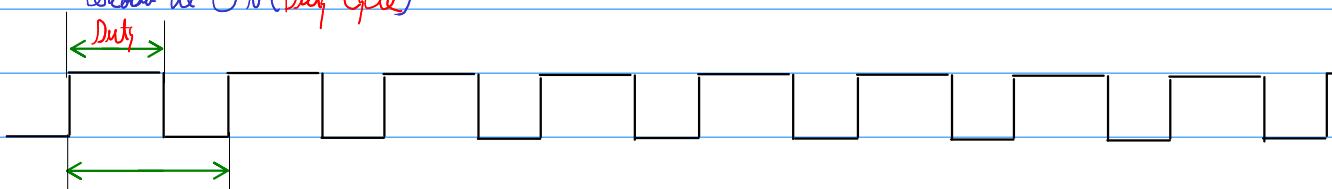
Atuador: Output, tem um transdutor que converte energia elétrica num outro tipo de energia.

PWM

Técnica digital que permite obter resultados analógicos a partir de dados digitais.

Tem um período fundamental (Period Cycle)

Período de ON (Duty Cycle)



Variacão de pulso ON (Duty Cycle) varia entre 0 e 100%

Em Arduino utiliza-se a função analogWrite para gerar um sinal PWM com um determinado período fundamental pré-programado e com resolução de 8 bits.

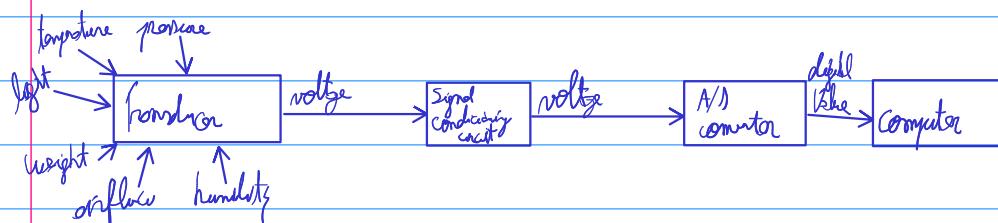
0% → analogWrite(0)
100% → analogWrite(255)

Conversão A/D

Um transdutor é necessário para converter uma grandeza física num sinal elétrico.

Um sistema de aquisição de dados é normalmente referido como um sistema que executa conversões A/D.

A conversão de tensões elétricas para valores digitais é chamada de conversão Analógico-Digital.

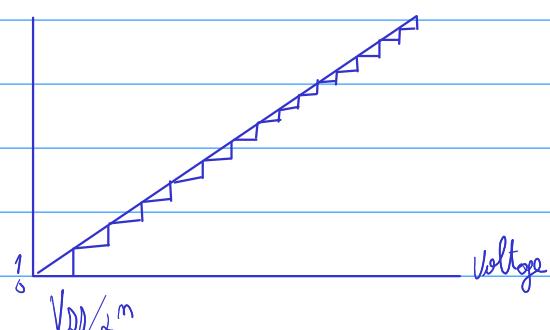


Característica ideal de um conversor A/D de m bits:

A resolução deste conversor é $V_{DD}/2^m$

O seu conversor A/D não é sempre linear

$$2^{m-1}$$



n -bits - Número de bits do conversor A/D

Gama Dinâmica - (N^o de Steps) - Relação entre a máxima e a mínima amplitude possível de medir que representa o n^o de níveis possíveis de representar, 2^m onde m é o n^o de bits do módulo A/D

Resolução - (Step size), é a alteração mínima que pode ser discriminada.

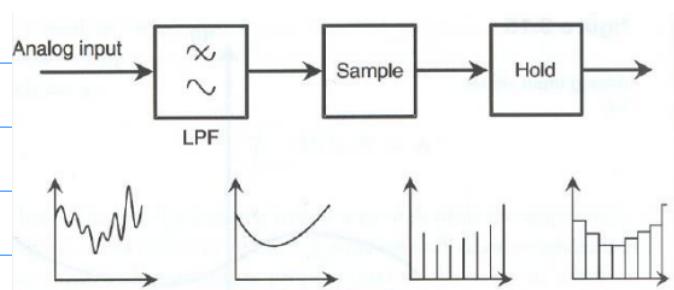
n -bits	Gama Dinâmica	Resolução (mV)
8	$2^8 = 256$	$5V/256 = 19.53$
10	$2^{10} = 1024$	$5V/1024 = 4.88$
12	$2^{12} = 4096$	$5V/4096 = 1.2$
16	$2^{16} = 65536$	$5V/65536 = 7.63e-2$
24	$2^{24} = 16777216$	$5V/16777216 = 2.98e-5$

Frequência de Amostragem

- Um sinal analógico pode ser convertido num sinal digitalizado
- Ao ritmo a que os valores são amostrados a partir do sinal analógico chama-se frequência de amostragem.
- A reprodução fiel de um sinal a partir de uma sequência digital não é garantida se a frequência de amostragem utilizada for de pelo menos 2 vezes superior à maior frequência existente no sinal original - Teorema de Nyquist-Shannon

Teorema da Amostragem de Nyquist-Shannon

- Para uma digitalização fiel, a frequência de amostragem deve ser pelo menos a metade da largura de banda do sinal, $f_s \geq 2f_{\max}$
- No entanto, não é possível fazer uma conversão instantânea, o valor de entrada deve ser mantido constante durante o tempo em que o conversor executa sua conversão
- Existe um circuito de entrada chamado Sample-and-hold que é responsável por este trabalho.

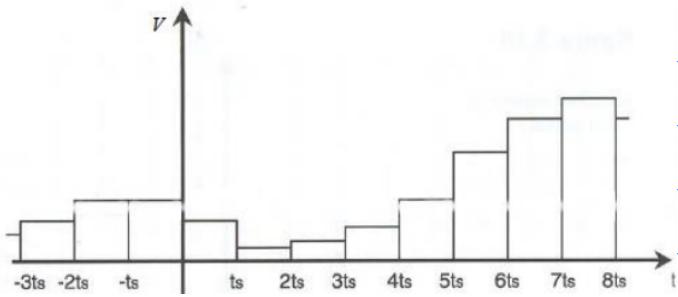


Quantificação

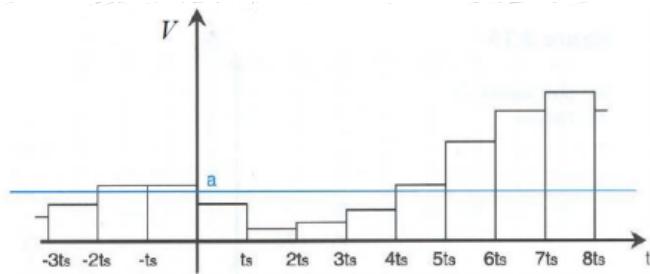
Da operação de amostragem resulta um nível em degrau.

A etapa que se segue à amostragem, converte em quantifica os valores dos amostras do nível analógico.

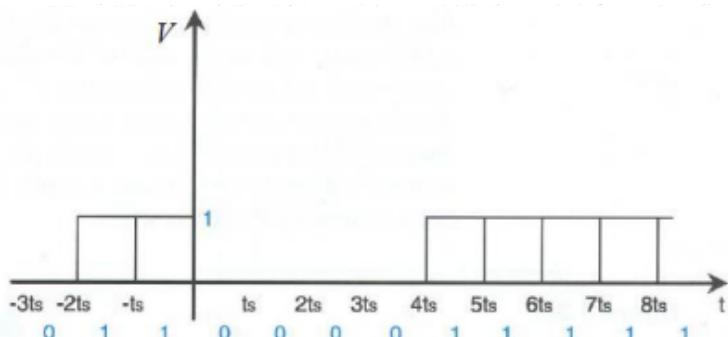
A operação de quantificação é realizada por 1 conversor analógico para digital (analog to digital converter - ADC)



A Quantificação pode ser interpretada como uma classificação dos amostras do nível, relativamente a um nível de tensão pre-determinado. Por exemplo, para classificar os amostras da figura anterior em 2 grupos, pode escolher-se um nível " α ".

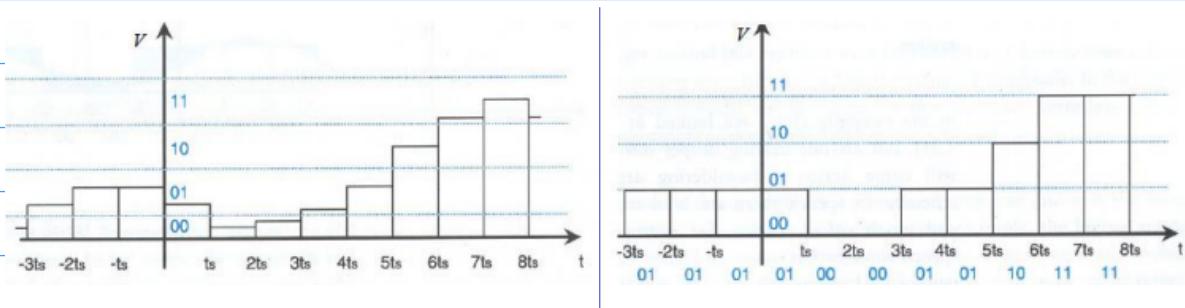


Assim, todos os amostras acima do nível de ' α ' pertencem, por hipótese, ao "grau 1" e aquelas cujo valor seja abaixo de ' α ' pertencem ao "grau 0". Com base neste critério, os amostras geram a sequência de zeros (0_s) e uns (1_s), que constitui um exemplo da quantificação das amostras utilizando 1 bit.



Se forem usados 2 bits para quantificar os níveis do sinal analógico, terão de definir 4 níveis de tensão, que podem ser igualmente espaçados, para quantificar os níveis da amostra.

O sinal pode ser quantificado em 4 grupos (00, 01, 10, 11)



Comparando os níveis quantificados com o sinal analógico original, pode verificar-se o erro introduzido pela conversão A/D. Estes erros produzem um efeito chamado **ruido de quantificação**.

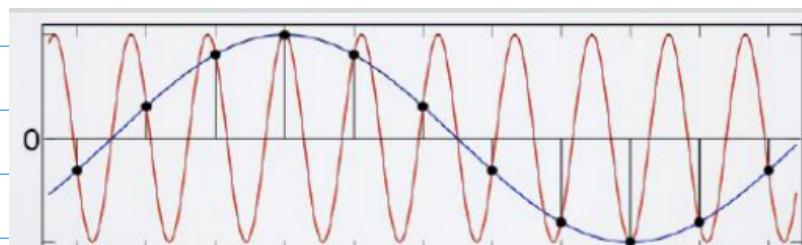
Aleatoriedade

Uma conversão A/D baseia-se na amostragem da entrada em intervalos fixos de tempo, sendo a saída uma regravação incompleta do comportamento de entrada.

Se o sinal de entrada estiver a variar a um ritmo muito superior à frequência de amostragem, pode ser obtido um sinal com outro comportamento, nomeada chama **alias**.

A frequência do sinal alias é a diferença entre a freq. do sinal e a freq. de amostragem.

Por exemplo, uma onda sinusoidal de 9 Hz a ser amostrada a 3 Hz seria reconhecida como sendo uma onda sinusoidal de 1 Hz. Isto é **aliasing**.



Precisão de conversores A/D

Existem 2 maneiras de melhorar a precisão da conversão A/D:

Aumentar a Resolução que melhora a precisão na medição da amplitude do sinal analógico.

Aumentar a Taxa de Amostragem que aumenta a frequência máxima que pode ser medida.

Discretização

Exemplo

- 0-10V nível de entrada
- Definir 8 intervalos de 1,25V

Output States	Discrete Voltage Ranges (V)
0	0.00-1.25
1	1.25-2.50
2	2.50-3.75
3	3.75-5.00
4	5.00-6.25
5	6.25-7.50
6	7.50-8.75
7	8.75-10.0

Quantificação

O número de estados possíveis que o conversor pode gerar:

$$N = 2^m$$

onde m é o n.º de bits no conversor AD

Exemplo: Para um AD de 3 bits

$$N = 2^3 = 8$$

Tamanho da quantização analógica

$$Q = (V_{\max} - V_{\min}) / m = (10V - 0V) / 8 = 1.25V$$

Codificação

Attribuir o valor digital (nº binário) a cada estado

Discrete Voltage Ranges (V)	Output Binary Equivalent
0.00-1.25	000
1.25-2.50	001
2.50-3.75	010
3.75-5.00	011
5.00-6.25	100
6.25-7.50	101
7.50-8.75	110
8.75-10.0	111

Resolução

Resolução de um ADC tem com o nº de valores discrete que o conversor pode produzir
Um ADC com 8 bits pode codificar um sinal analógico entre 0 e 15.

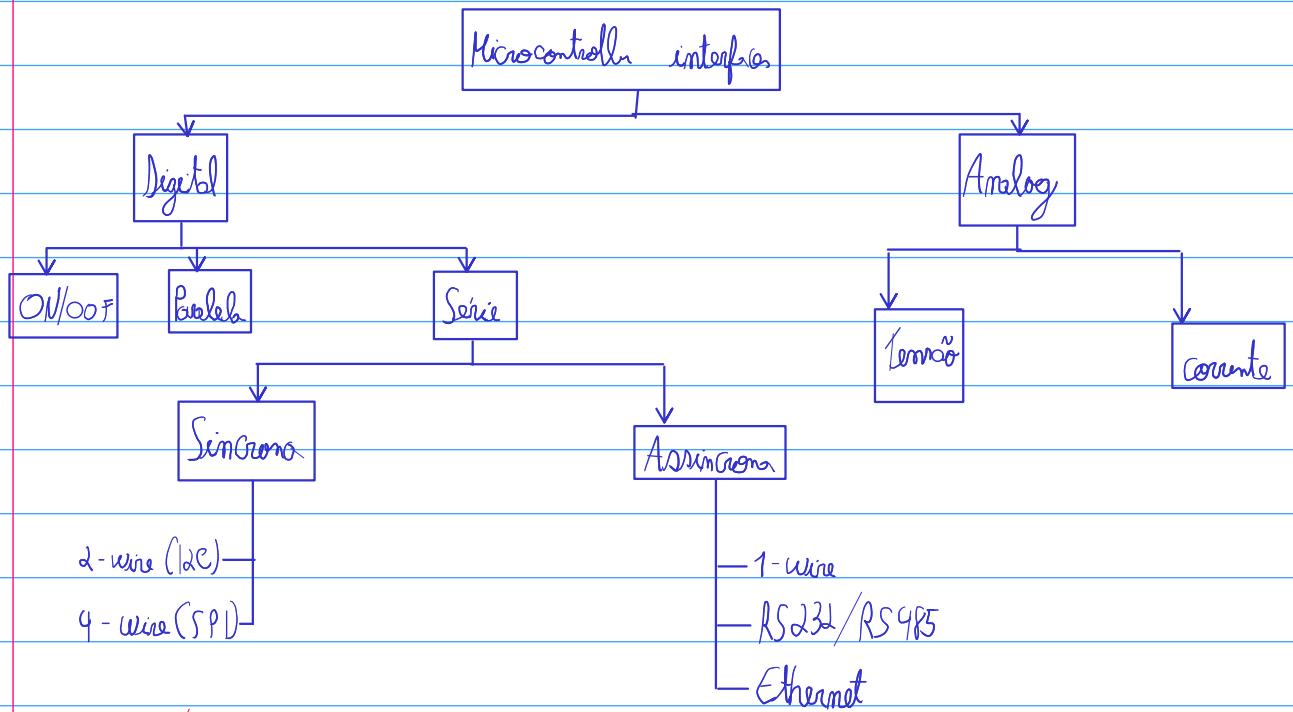
A resolução de voltagem ADC tem o nome de **Intervalo de Quantização analógica (Q)**

$Q = V_{range} / 2^m$, onde V_{range} é o intervalo de voltagem analógico, que podem ser representados

No exemplo anterior: $Q = 1.25 \text{ V}$

Se usarmos somente 2 bits, então a resolução será menor $10/2^2 = 2.50 \text{ V}$.

Técnicas de interface de comunicação



Entrada / Saída Analógica

Gamas típicas para entrada / saída Analógica baseado em tensão

- 0 to 2.5V
- 0 to 4V
- 0 to 5V
- +/- 2.5V
- +/- 4V
- +/- 5V

Gamas típicas para entrada / saída Analógica baseado em corrente

- 0-20 mA
- 4-20 mA

(On/off) entrada digital

- Interface mais simples
- Menor custo de implementação
- Alta velocidade
- Baixa sobrecarga (complexidade) de programação

Comunicação em série

Porquê usar uma comunicação série?

- I/O paralelo utiliza muitos pinos de sinal
- Com o aumento da distância de ligação aumentam os problemas de sincronismo
- Custo do cabo superior
- Grande parte das aplicações não precisam de taxas de transferência de dados elevadas

Exemplos de I/O Série

USART

SPI

I2C

CAN bus

Comunicação Síncrona: O receptor tem um mecanismo de sincronização baseado num sinal de relogio com o emissor. É possível transmitir blocos de dados de cada vez.

Comunicação Asíncrona: O receptor não apresenta nenhum mecanismo de sincronização. Neste caso as sequências de bits emitidos têm de conter a informação de inicio e fim de cada "grupo" de dados.

Sentido

Simplex - A comunicação é feita num sentido, da emissão para 1 ou + receptores.

Half-Duplex - A comunicação pode ser feita nos 2 sentidos, mas 1 de cada vez.

Full-Duplex - A comunicação pode ser feita nos 2 sentidos em simultâneo.

Comunicação em sinal Assíncrono

Dados transmitidos escritos a escrita, cada caractere entre um start e um stop bit, taxa de transmissão em bps.
O receptor usa um CLK com mesma frequência para ler o nível de entrada.
Nos equipamentos mais recentes é usado apenas o stop bit

Duração de bit

Cada bit é transmitido por 1 período fixo

transmissão sincronizada por emissor e receptor

Baud Rate (f) determina a duração (T)

Baud Rate é o n.º de transições / seg

medido em bps

$$T = \frac{1}{f}$$

Throughput vs Baud

Toda a transmissão envolve o envio de bits de冗余 (Start, Stop Bits)

Taxa de transferência de dados é inferior à taxa de transmissão, os bits de冗余 têm de ser enviados

Exemplo 8 bits/bits | 1 parity bit | baud rate = 9600

Precisa de 11 bits para enviar 8

$$\text{Data throughput} = 9600 \times 0.73 = 6981.8 \text{ bps}$$

$$\text{Eficiência} = \frac{8}{11} = 73\%$$

Comunicação Síncrona

Receptor tem mecanismo de sincronização baseado num nível de relogio sincronizado com o emissor.
Start e Stop bit não são necessários

Serial Peripheral Interface

SPI é um protocolo de comunicação síncrona síncrona de dados, utilizado entre um dispositivo master e outros slave.

Estrutura:

Interface de 4 fios: clock, data in, data out, slave select

Protocolo síncrono

Protocolo Master-Slave: Master controla CLK e comunica com slave

Protocolo de Data Exchange: Ente Master/Slave tem 2 linhas de dados, 1 pra cima, outra pra baixo.

Comunicação serial no Arduino

Todos os arquivos tem 1 ou 2 portas serial

Arduino Uno comunica nos pins 0(RX) e 1(TX) para receber e enviar dados

Via DSB com o PC

IoT

A IoT é a rede de coisas incorporadas em componentes eletrônicos, software, sensora e conectividade de rede, que permite a essas objetos extra e trocar dados.

Síntesis

Rede das coisas

Planeta inteligente

Internet industrial

Sistemas ciber-físicos

Estrutura do IoT

RFID	Sensor	Tecn.	Fam.
Identifica e rastreia os objetos Tagging things	Obter e processar os dados para detectar as mudanças no status físico dos objetos Feeling things	A inteligência incorporada nos dispositivos estávols dos sensores formam a rede como internet Thinking things	A microtecnologia provoca a possibilidade de coisas monitorar interagir e se conectar com outros ou dispositivos intelectuais Shrinking things

Segurança na IoT

Segurança em todos os níveis

Mecanismos de segurança existentes não são suficientes em IoT

Nova: Pode ser o protocolo utilizado no desenrolhamento com mecanismo de segurança forte

TLS e DTLS não são protocolos muito usados no enlace de transporte

Análise de Big Data

Baixo o custo

Tempo de decisão, mais rápido e melhor