Parte II – Projeto de Sistemas Embarcados

Luciano Bertini Alessandro Copetti

ICT – RCM – UFF
Instituto de Ciência e Tecnologia
Departamento de Computação
Universidade Federal Fluminense

2018

Sumário

Introdução

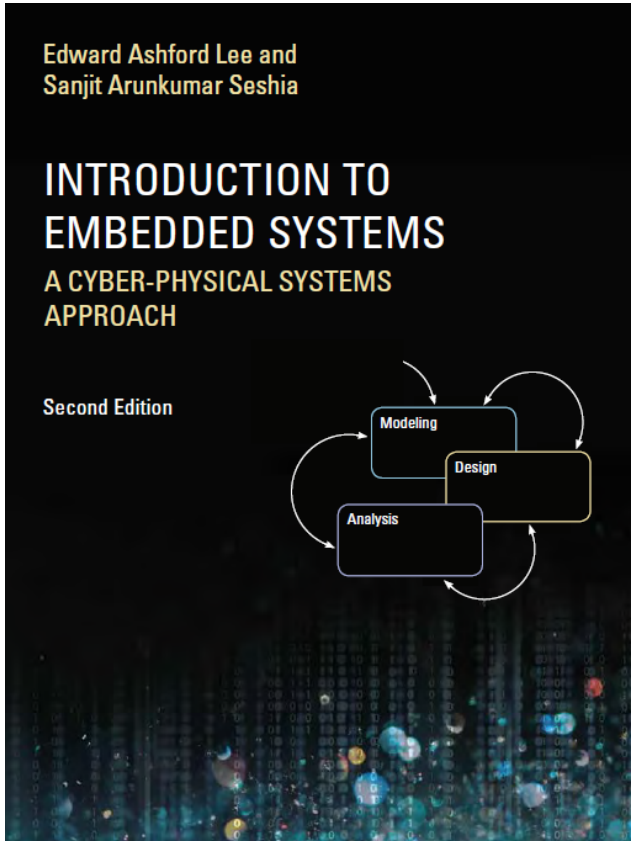
Parte II do livro

Técnicas usadas para construir software embarcado concorrente e de tempo real.

- sensores e atuadores
- processadores embarcados
- arquiteturas de memória
- mecanismos de I/O
- concorrência e escalonamento

Introdução

Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition, MIT Press, ISBN 978-0-262-53381-2, 2017. http://leeseshia.org



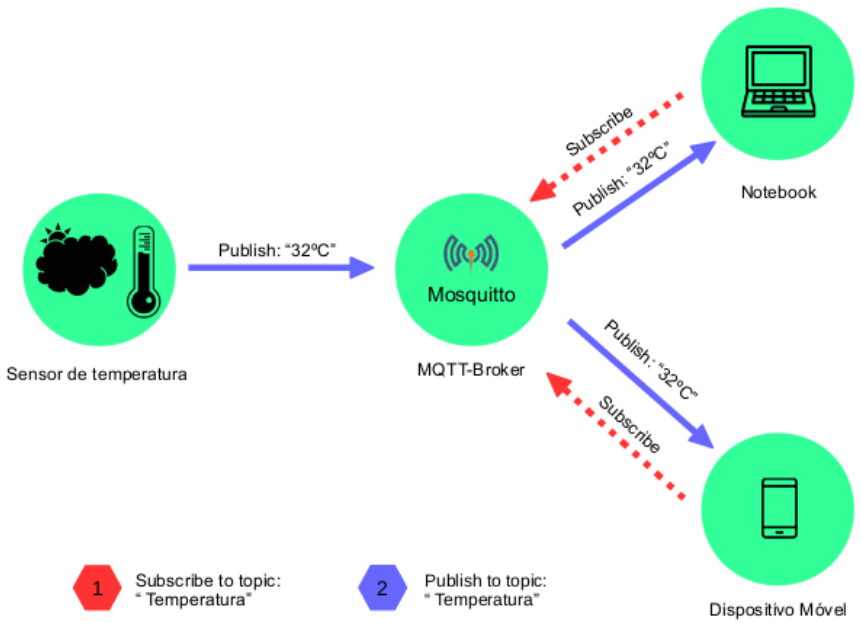
Sensores e Atuadores

Introdução

- Sensores e atuadores costumam ser empacotados com microprocessadores e interfaces de rede, permitindo que eles apareçam na Internet como serviços.
- A tendência é para uma tecnologia que conecte profundamente nosso mundo físico com o mundo da informação por meio de sensores e atuadores inteligentes.
- Este mundo integrado é variavelmente chamado de Internet of Things (IoT), Industry 4.0, the Industrial Internet, Machine-to-Machine (M2M), Internet of Everything, the Smarter Planet, TSensors (Trillion Sensors), ou The Fog (como a nuvem, porém de mais baixo nível).

Introdução

- Sensores e atuadores podem estar acessíveis usando Representational State Transfer (REST) ou através de protocolos como o MQTT (publish/subscribe)

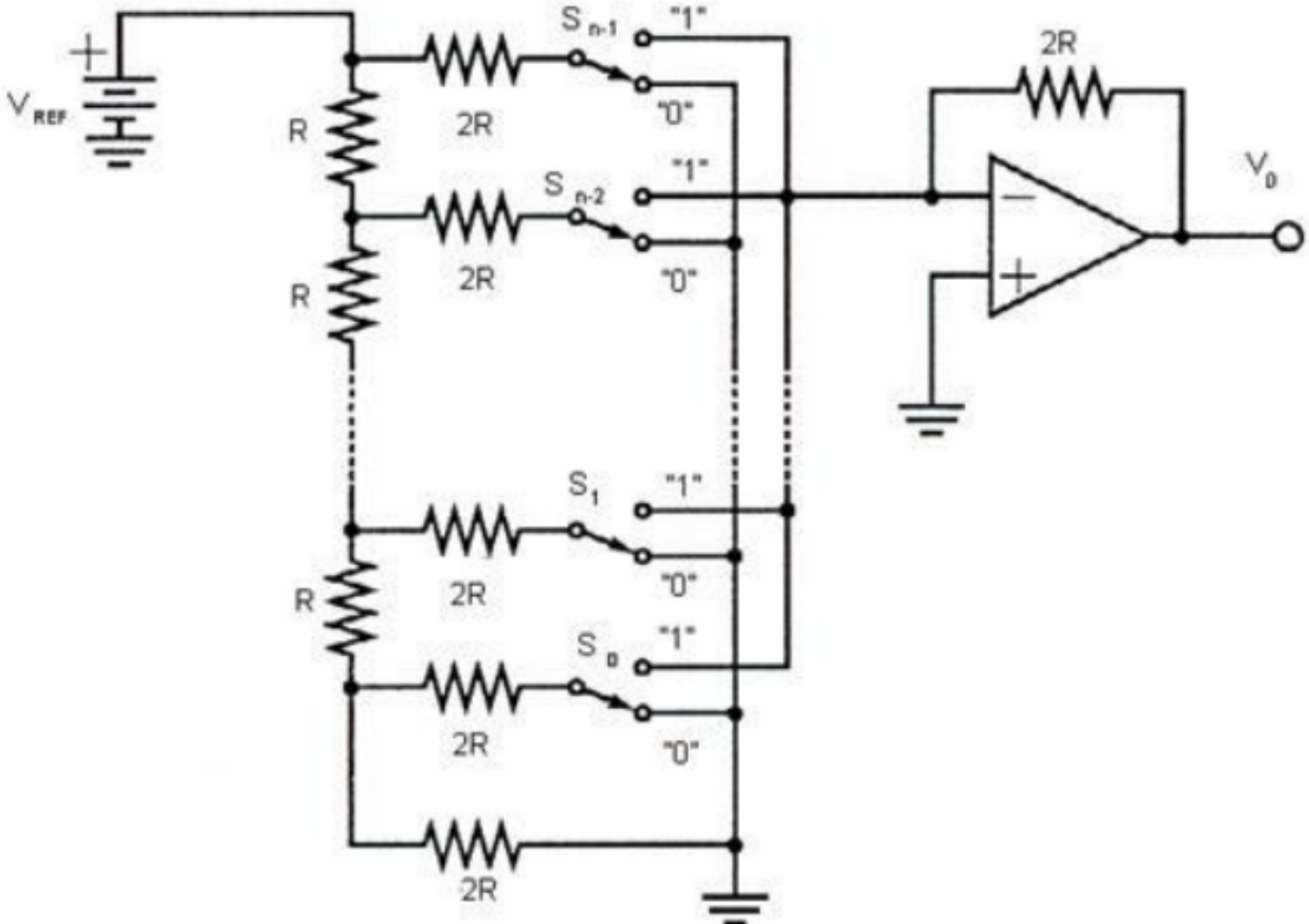


Fonte: TCC Marcelo Rocha

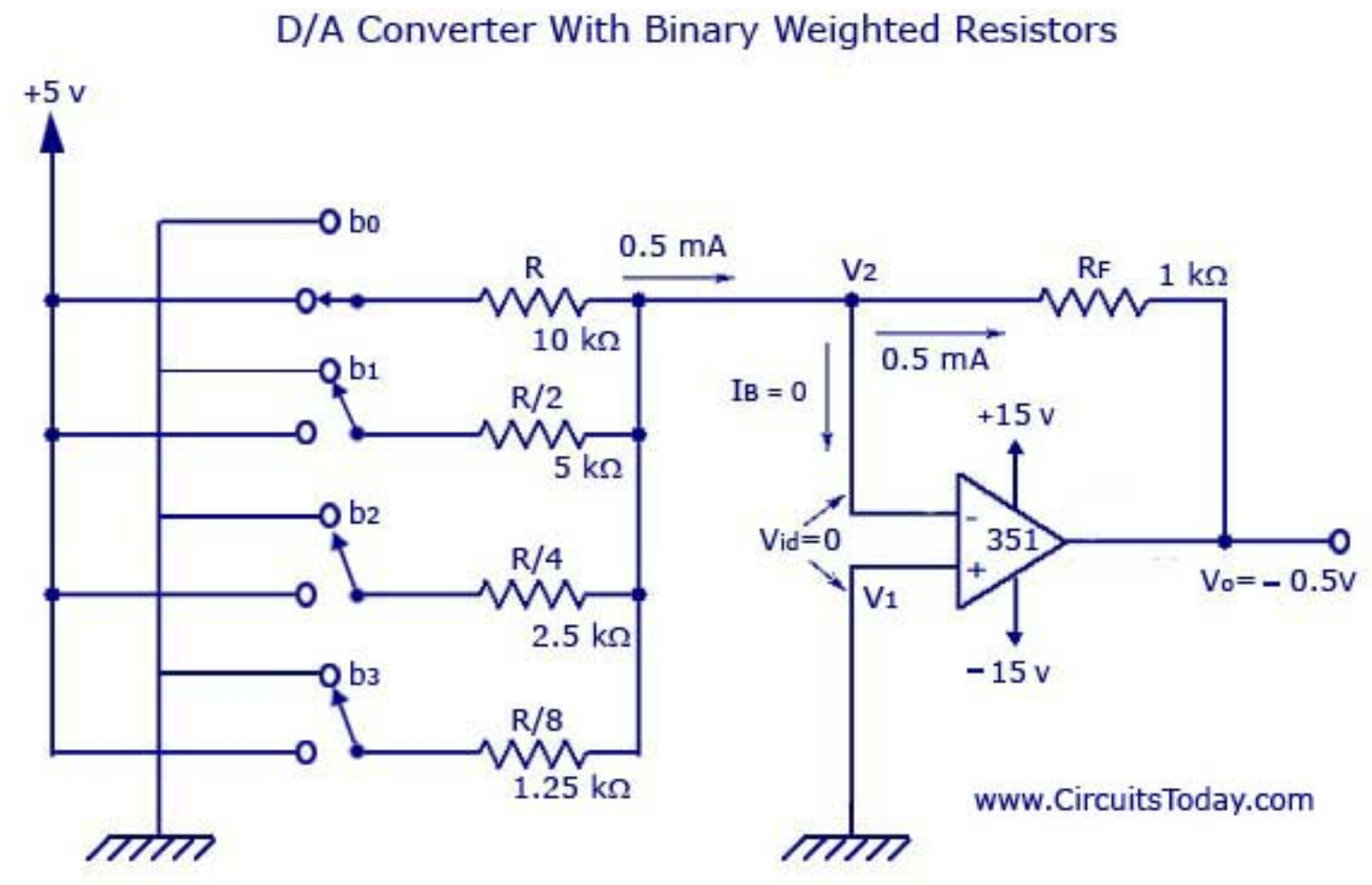
Introdução

- Um **sensor** é um dispositivo que mede uma quantidade física. Um **atuador** é um dispositivo que altera uma quantidade física.
- Sensores e atuadores podem ser digitais (dois estados) ou de faixa linear.
- Os sistemas que utilizam faixa linear necessitam de conversores A/D e/ou D/A.
- A precisão depende do número de bits dos conversores A/D e D/A.
- Exemplos: sensor de efeito hall, de temperatura (vários tipos), acelerômetros, etc.
- Atuadores podem ser válvulas, solenoides, aquecedores, etc

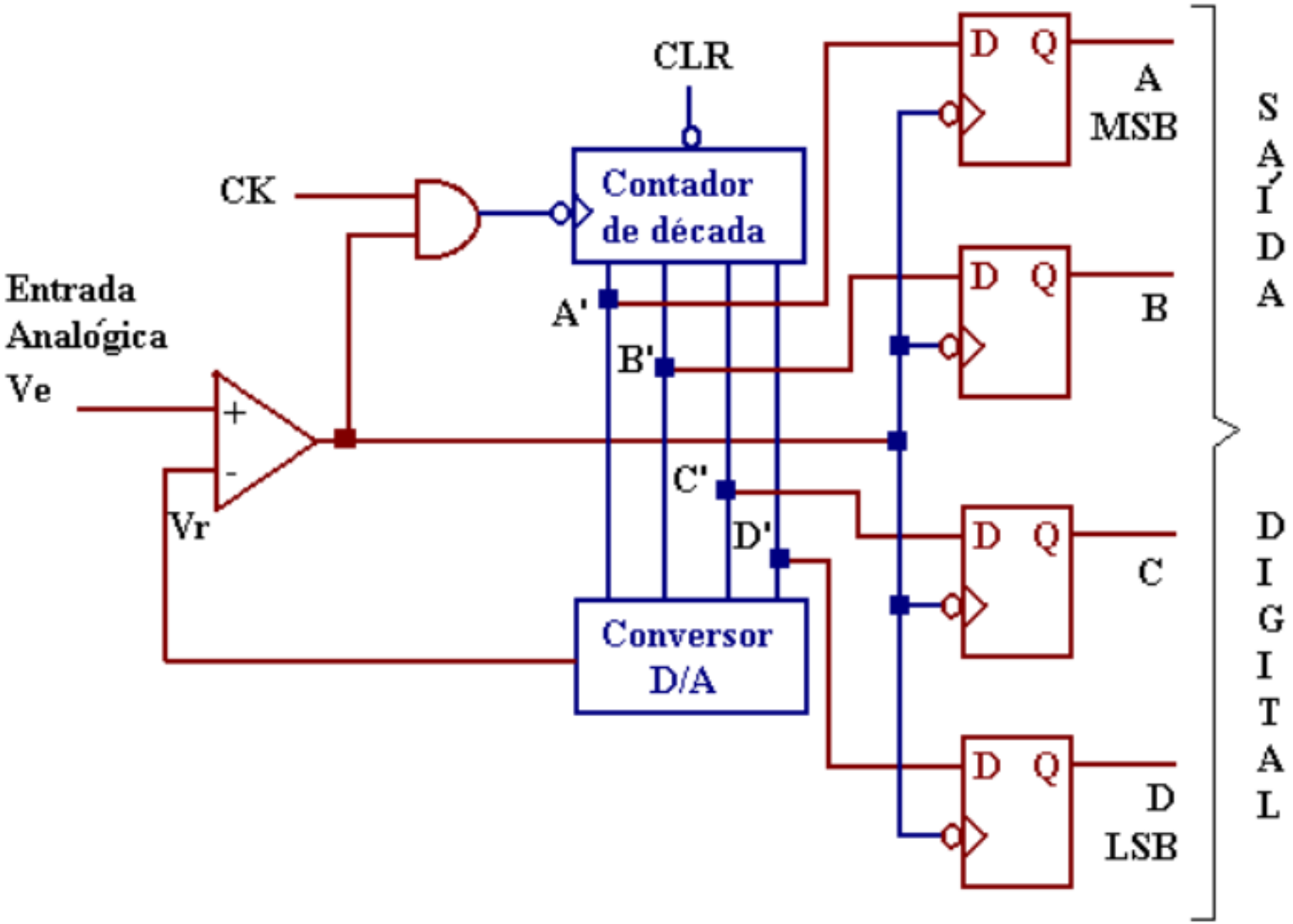
Conversor A/D



Conversor A/D



Conversor D/A



Propriedades dos Sensores e Atuadores

- Linearidade
- Range, range dinâmico e precisão
- Quantização
- Ruido
- Amostragem
- Distorção Harmônica

Linearidade

- Sensores lineares e de função afim
- É chamada função afim toda função polinomial do primeiro grau. A constante a é o coeficiente angular do gráfico de f e b é o coeficiente linear, ou o ponto de intersecção com o eixo y
- Suponha que uma quantidade física $x(t)$ no instante t é reportada pelo sensor tendo o valor $f(x(t))$
- Se $x(t)$ for do tipo $x(t) = ax(t) + b$ o sensor é dito **afim**. Se $b = 0$ o sensor é **linear**
- Geralmente o tipo linear é atribuído aos dois casos.
- O termo b é denominado bias.

Range

- Sensores e atuadores assumem valores limitados
- Os valores podem ficar **saturados**
- Na verdade a função afim de um sensor é definida como:

$$f(x(t)) = \begin{cases} ax(t) + b & \text{if } L \leq x(t) \leq H \\ aH + b & \text{if } x(t) > H \\ aL + b & \text{if } x(t) < L, \end{cases}$$

A limitação para atuadores deve ser forçada no software

Exemplo: Controle PID de motor BLDC

```
if(u0>12000) u0=12000;
if(u0<0) u0=0;
pwm = map(u0, 0, 12000, 255, 0);
```

Range Dinâmico e Precisão

- A precisão **p** de um sensor é a **menor diferença absoluta** entre dois valores de uma grandeza física cujas leituras do sensor são distinguíveis.
- A faixa dinâmica D de um sensor digital é a razão:

$$D = \frac{H - L}{p}$$

- O alcance dinâmico é geralmente medido em decibéis:

$$D_{dB} = 20log_{10}\left(\frac{H - L}{p}\right)$$

Decibéis

- O termo "decibel" é literalmente um décimo de um bel, nome dado em homenagem a Alexander Graham Bell.
- Desenvolvida por engenheiros de telefone da Bell Telephone Labs para designar a relação entre a potência de dois sinais
- Um bel é definido como sendo um fator de 10 entre duas potências.
- Assim, um secador de cabelo de 1000 watts dissipa 1 bel, ou 10 dB, mais energia do que uma lâmpada de 100 watts.

$$decibeis = 10.log\left(\frac{p_1}{p_2}\right)$$

Quantização

- Um sensor digital representa uma quantidade física usando um número de n bits (conversor A/D), onde n é um inteiro pequeno.
- Existem apenas 2^n números distintos, de modo que tal sensor pode produzir apenas 2^n medições distintas.

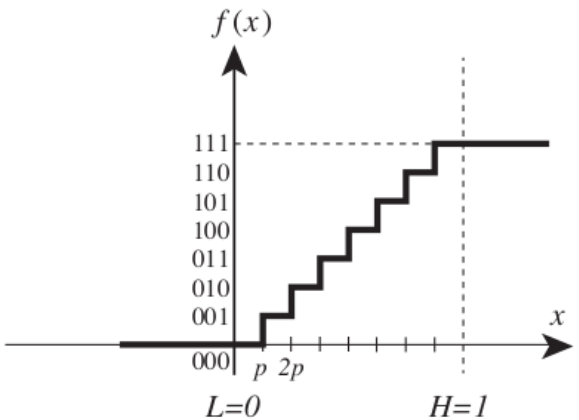


Figure 7.1: Sensor distortion function for a 3-bit digital sensor capable of measuring a range of zero to one volt, where the precision $p = 1/8$.

Ruído

- Por definição, o ruído é a parte indesejável de um sinal.
- Queremos medir $x(t)$ no tempo t, mas na verdade medimos $x'(t)$. Então o ruído é a diferença:

$$n(t) = x'(t) - x(t)$$

- Uma forma comum de se caracterizar o ruido é através do RMS:

$$\lim_{T \to \infty} \sqrt{\frac{1}{2T} \int_{-T}^{T} (n(\tau))^2 d\tau}$$

Ruído

- A Razão sinal/ruído (SNR) é medida em decibéis:

$$SNR_{dB} = 20log_{10}\left(\frac{X}{N}\right)$$

- onde X é o valor RMS do sinal x.

Amostragem

- Um sensor digital irá amostrar a quantidade física em pontos específicos no tempo para criar um sinal discreto.
- Em amostragem uniforme, o tempo fixo T entre as amostras é o chamado intervalo de amostragem.
- O sinal resultante pode ser modelado como uma função $s : \mathbb{Z} \to \mathbb{R}$ definida como segue:

$$\forall n \in \mathbb{Z}, \quad s(n) = f(x(nT))$$

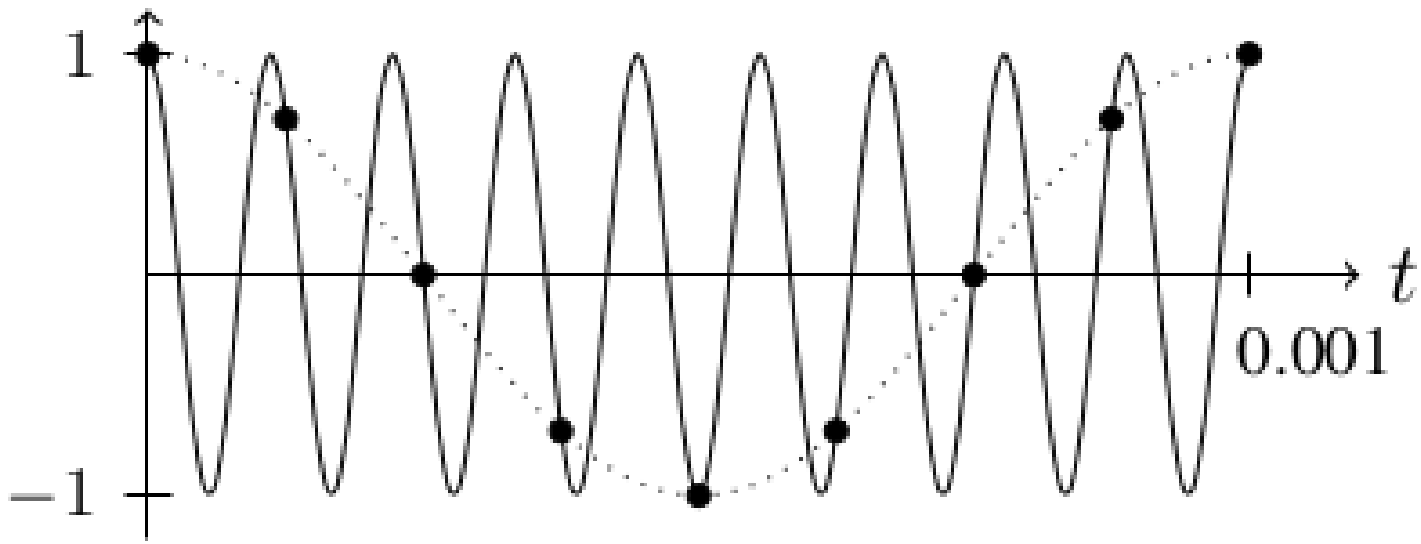
- onde $\mathbb{Z}$ é o conjunto de inteiros. Ou seja, a grandeza física $x(t)$ é observada apenas nos momentos $t = nT$.

Amostragem

- A taxa de amostragem é $\frac{1}{T}$, cuja unidade é amostras por segundo, muitas vezes dadas como Hertz (Hz)
- Na prática, quanto menor o intervalo de amostragem T, mais custoso se torna fornecer mais bits em um ADC.
- Com o mesmo custo, os ADCs mais rápidos normalmente produzem menos bits e, portanto, têm um erro de quantização mais alto ou um intervalo menor.

Amostragem

- Uma preocupação importante na amostragem de sinais é que existem muitas funções distintas x que, quando amostradas, produzirão os mesmos sinais s. Esse fenômeno é conhecido como **aliasing**.



Amostragem

- Uma regra prática útil para a amostragem uniforme é dada pelo teorema de **Nyquist-Shannon**.
- O estudo do assunto requer o uso das transformadas de Fourier. .
- Informalmente, este teorema afirma que um conjunto de amostras na taxa de amostragem $R = \frac{1}{T}$ define exclusivamente um sinal de tempo contínuo que é uma soma de componentes senoidais com frequências menores que $\frac{R}{2}$

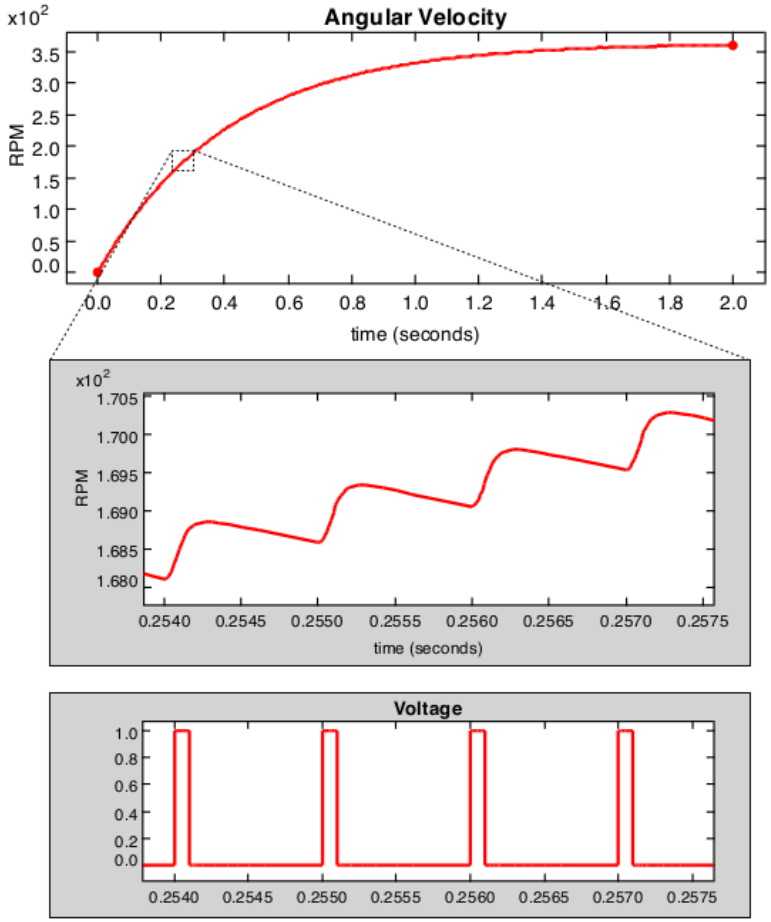
Sensores Comuns

- Tilt e aceleração
- Posição e velocidade
- Rotação
- Sensor sonoro
- Campo magnético
- Outros

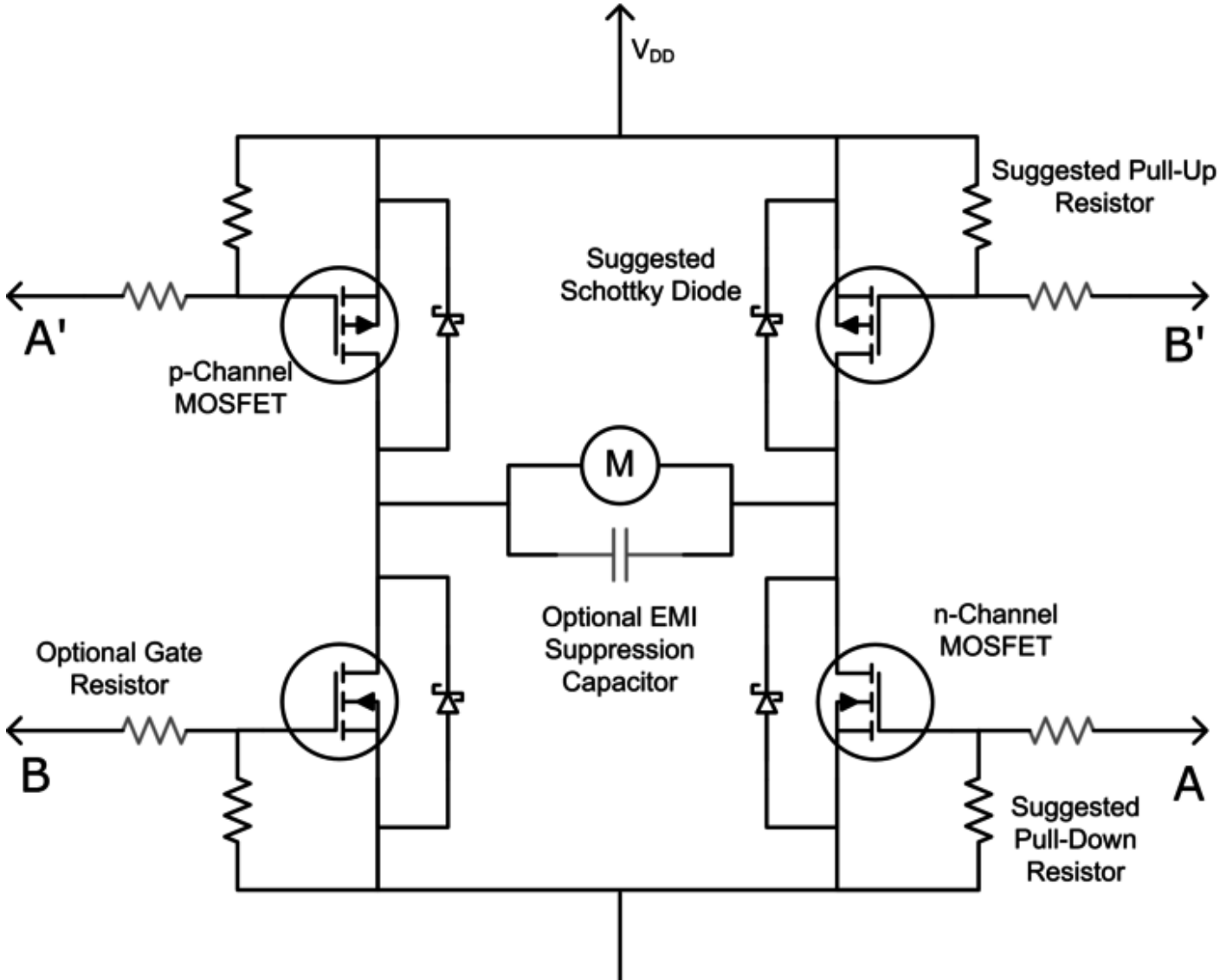
Atuadores

- LEDs
- Controle de motores
  - Motores de CC
  - Motores de passo
  - Servo motores

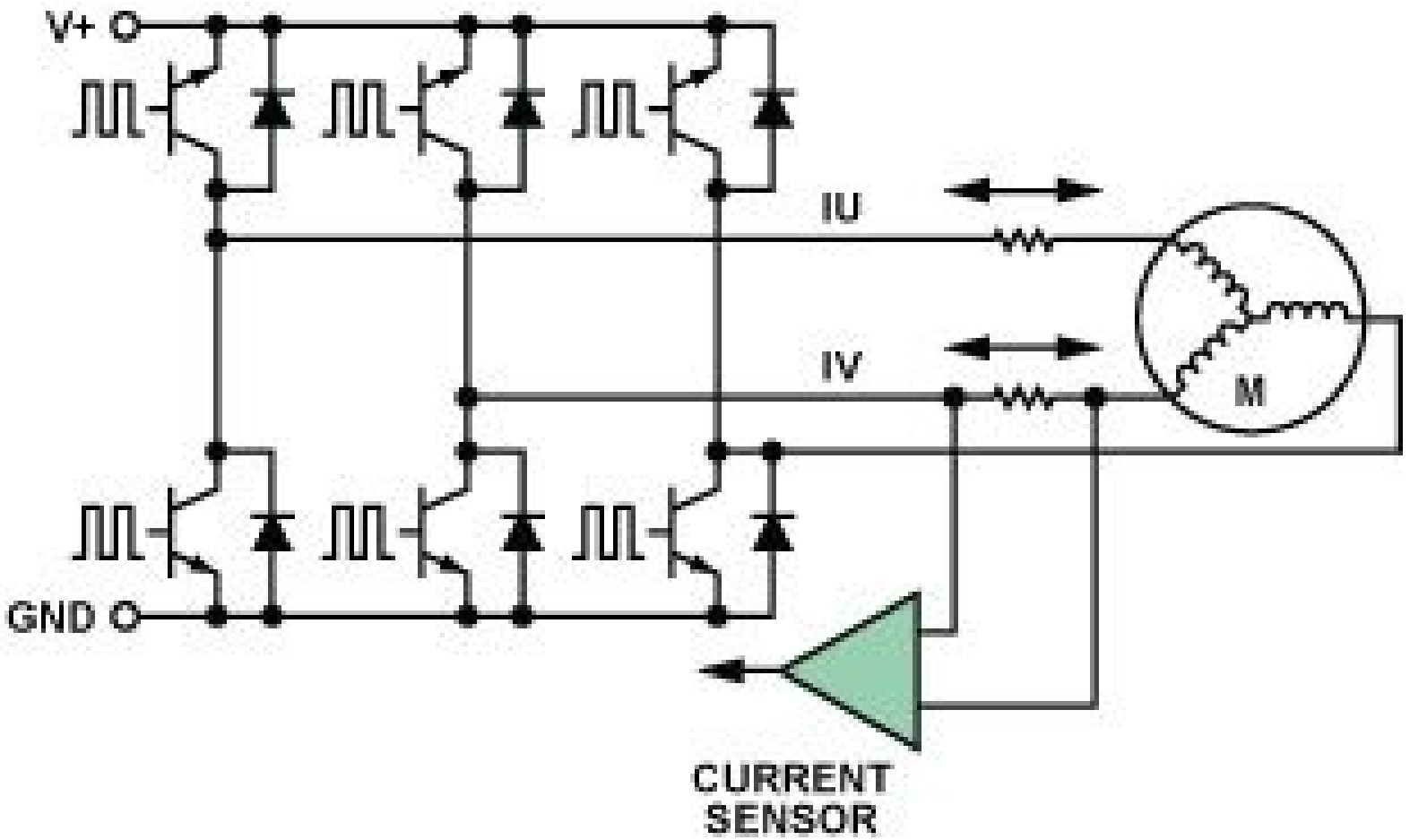
Controle PWM de um motor DC



Circuito de controle: Pontes



Circuito de controle: Pontes



Processadores para Sistemas Embarcados

Introdução

- O conjunto de instruções x86 domina hoje a computação de propósito geral (general-purpose computing)
- Não existe essa dominância na computação embarcada (Talvez o INTEL 8051)
- Diferença entre instruction set architecture (ISA) e a a realização de um processador (chip)
- Os processadores para sistemas embarcadostrazem muitas vantagens: gasta menos energia, utiliza baterias menores, incluem dispositivos internos

Microcontroladores

- Um microcontrolador (μC) é um pequeno computador em um único circuito integrado que consiste em um núcleo CPU relativamente simples, combinado com dispositivos periféricos, como memórias, dispositivos de E/S e temporizadores.
- Exemplos: ATMEL ATMEGA328 (Arduino Uno), ESP8266 (NodeMCU - arquitetura RISC de 32 bits), PIC16F877 (Microship)

Exemplo: ATMEL ATMEGA328

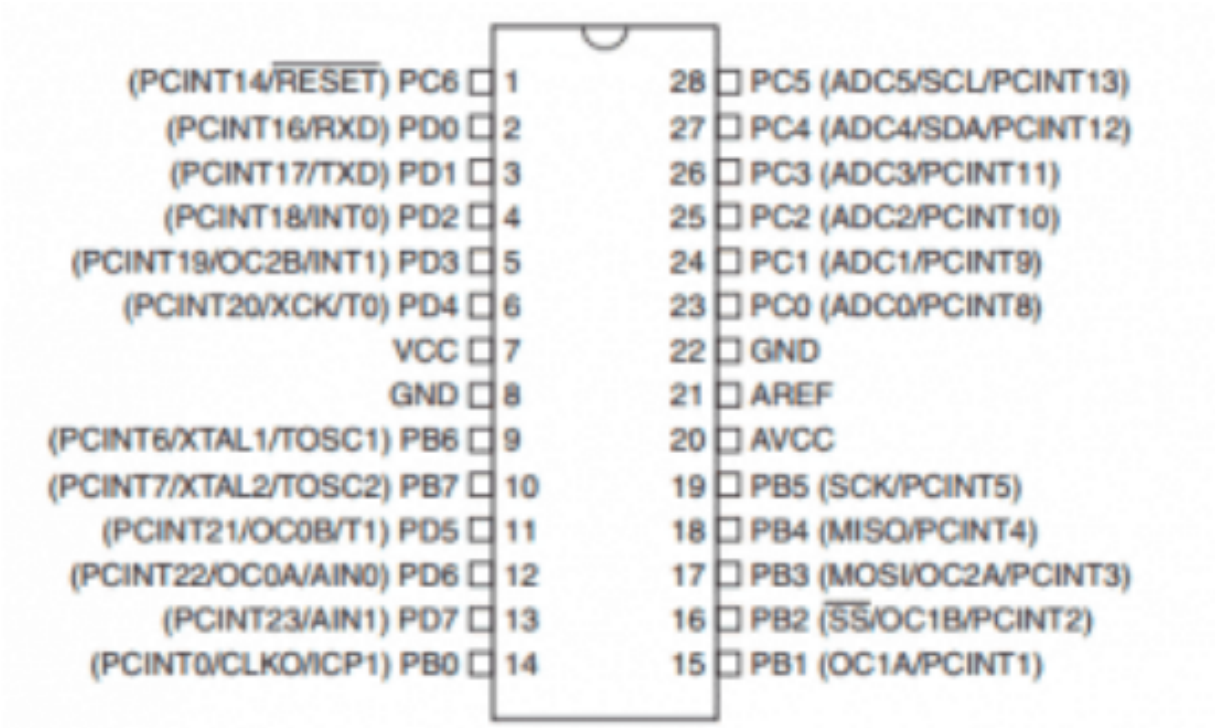
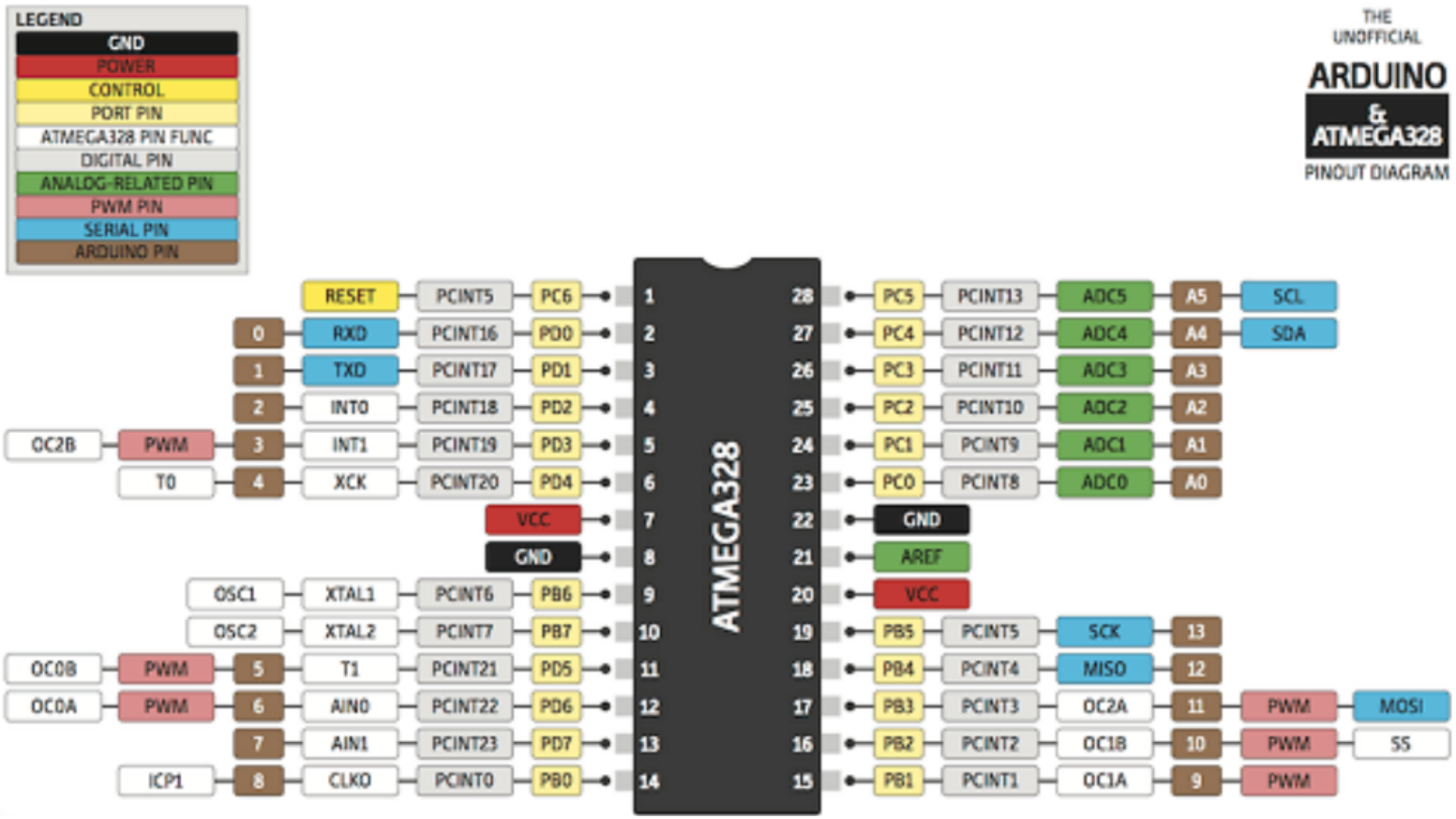


Figura 8 - Pinagem ATmega328 usado no Arduino UNO

fonte: https://www.embarcados.com.br/

Exemplo: ATMEL ATMEGA328

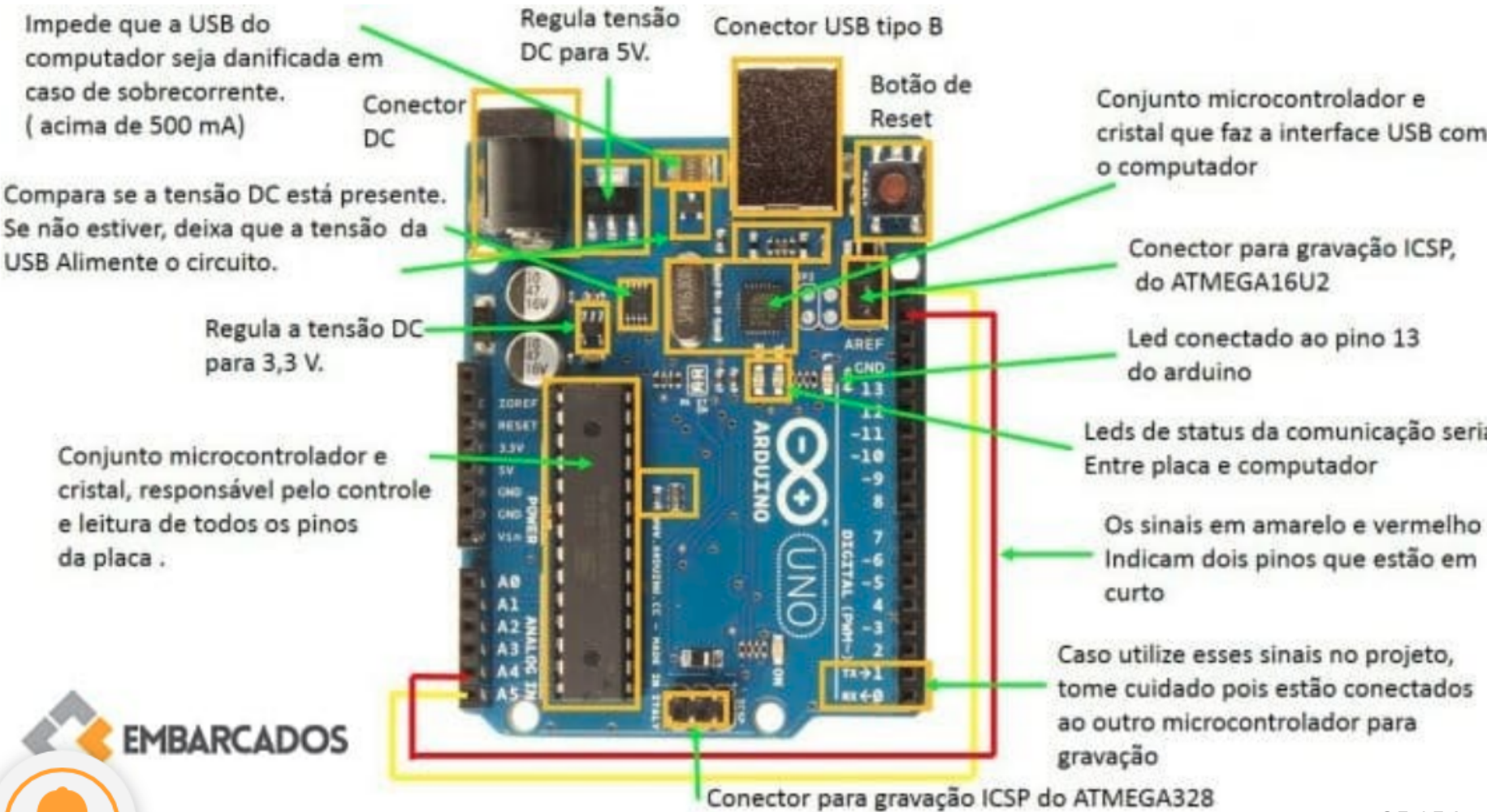


fonte: https://www.embarcados.com.br/

Exemplo: ATMEL ATMEGA328

fonte: https://www.embarcados.com.br/

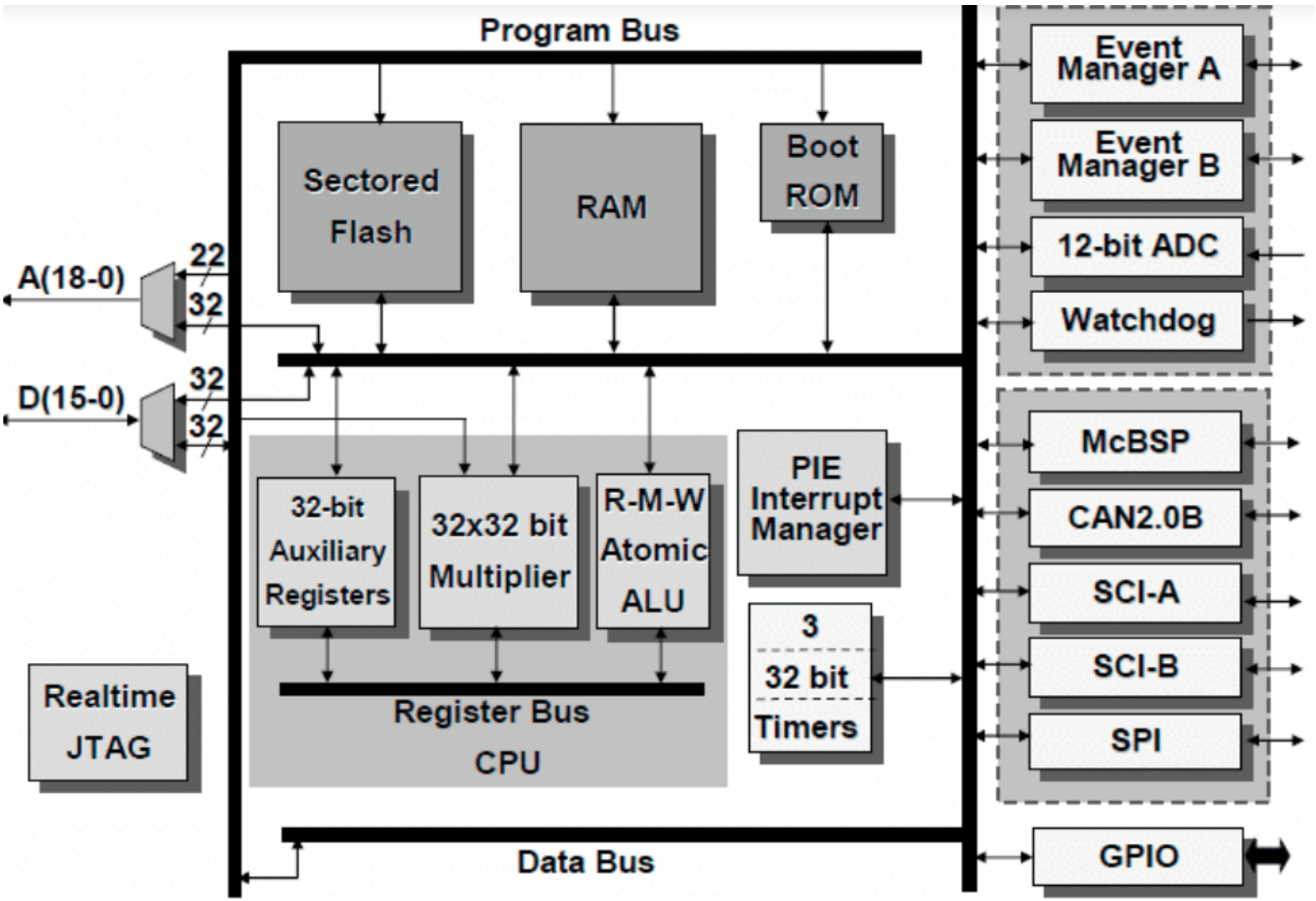
Exemplo: ATMEL ATMEGA328



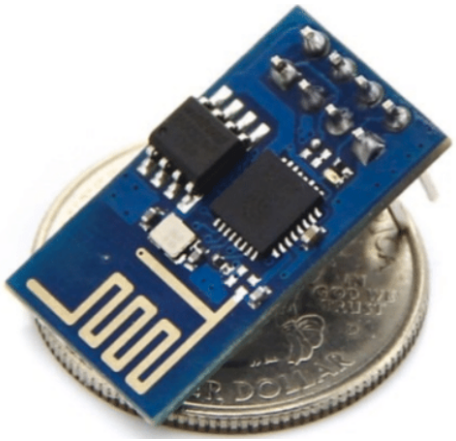
Processadores Digitais de Sinais (DSP)

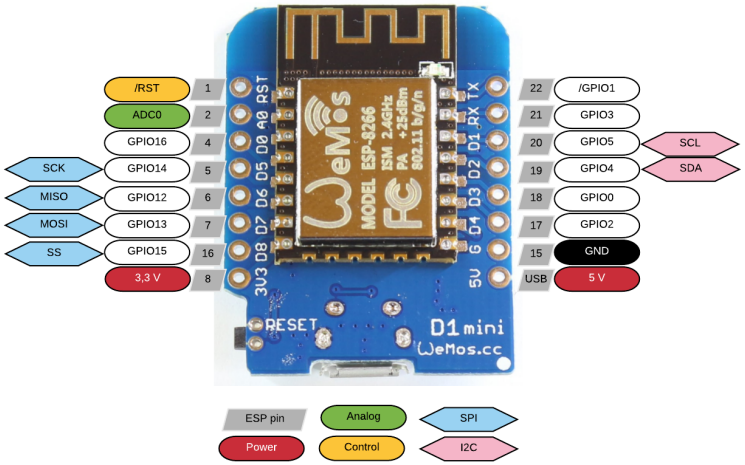
- O TMS320F2812 é um dispositivo denominado controlador digital de sinais (DSC)
- O DSP apresenta unidades de hardware projetadas para realizar operações matemáticas com alto desempenho.
- O princípio de operação do DSP é fazer todo processamento necessário antes de obter uma nova informação.

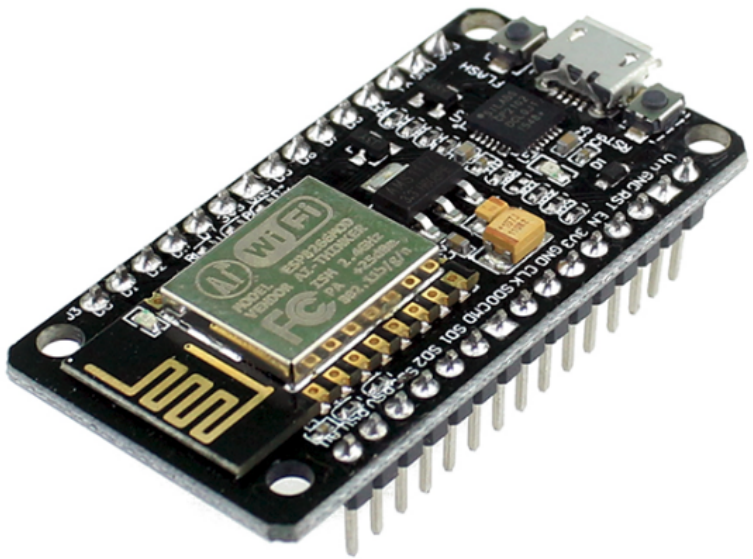
TMS320F2812



ESP8266 e NodeMCU





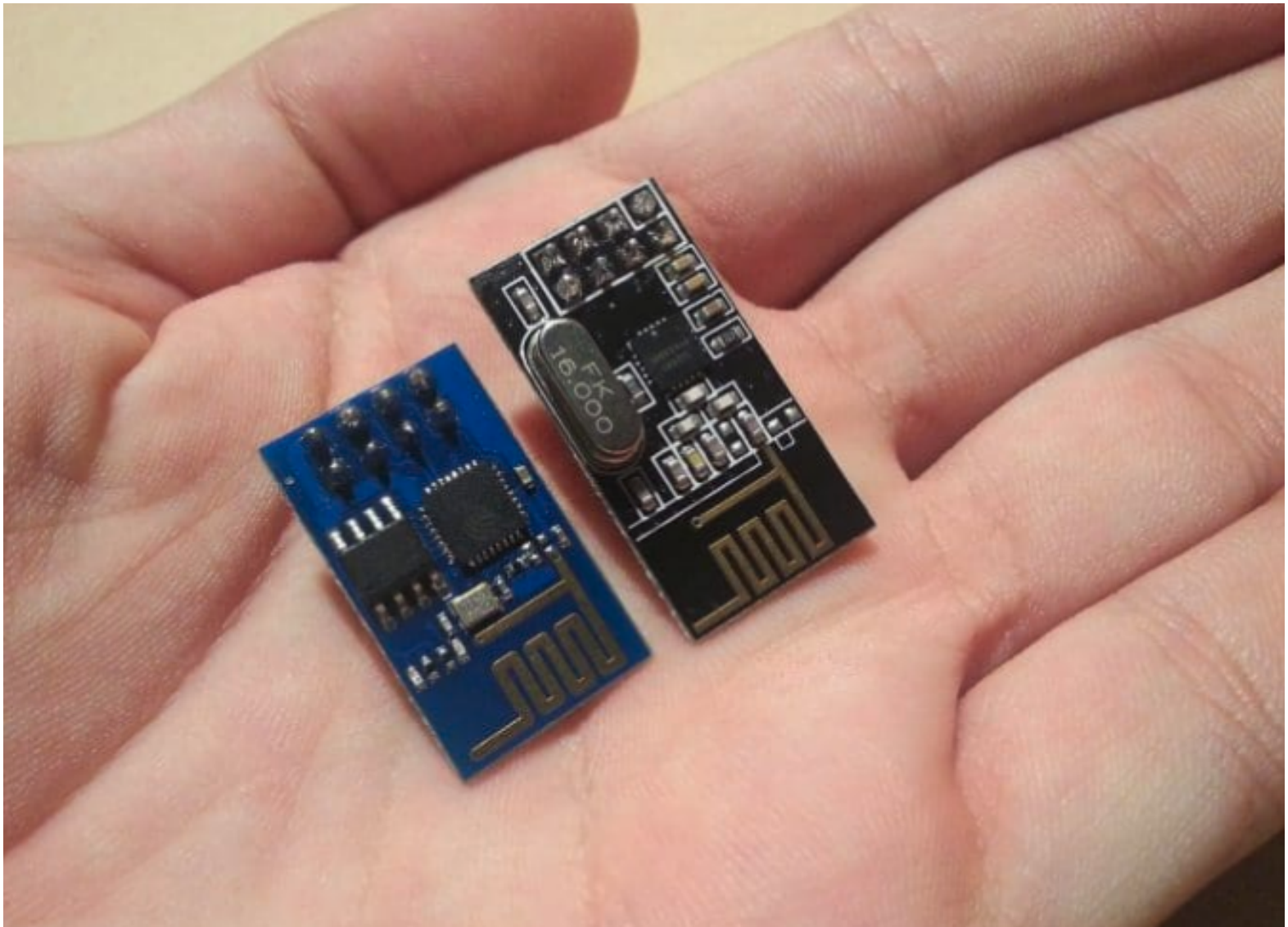


Características do ESP8266

- É um System-On-Chip com Wi-Fi embutido;
- Tem conectores GPIO, barramentos I2C, SPI, UART, entrada ADC, saída PWM sensor interno de temperatura;
- CPU que opera em 80MHz, com possibilidade de operar em 160MHz;
- Arquitetura RISC de 32 bits;
- 32KBytes de RAM para instruções;
- 96KBytes de RAM para dados;
- 64KBytes de ROM para boot;
- Possui uma memória Flash SPI Winbond W25Q40BVNIG de 512KBytes;
- O núcleo é baseado no IP Diamand Standard LX3 da Tensilica;
- Fabricado pela Espressif;
- Existem módulos de diferentes tamanhos e fabricantes.

Família ESP8266

Comparação de modelos ESP01



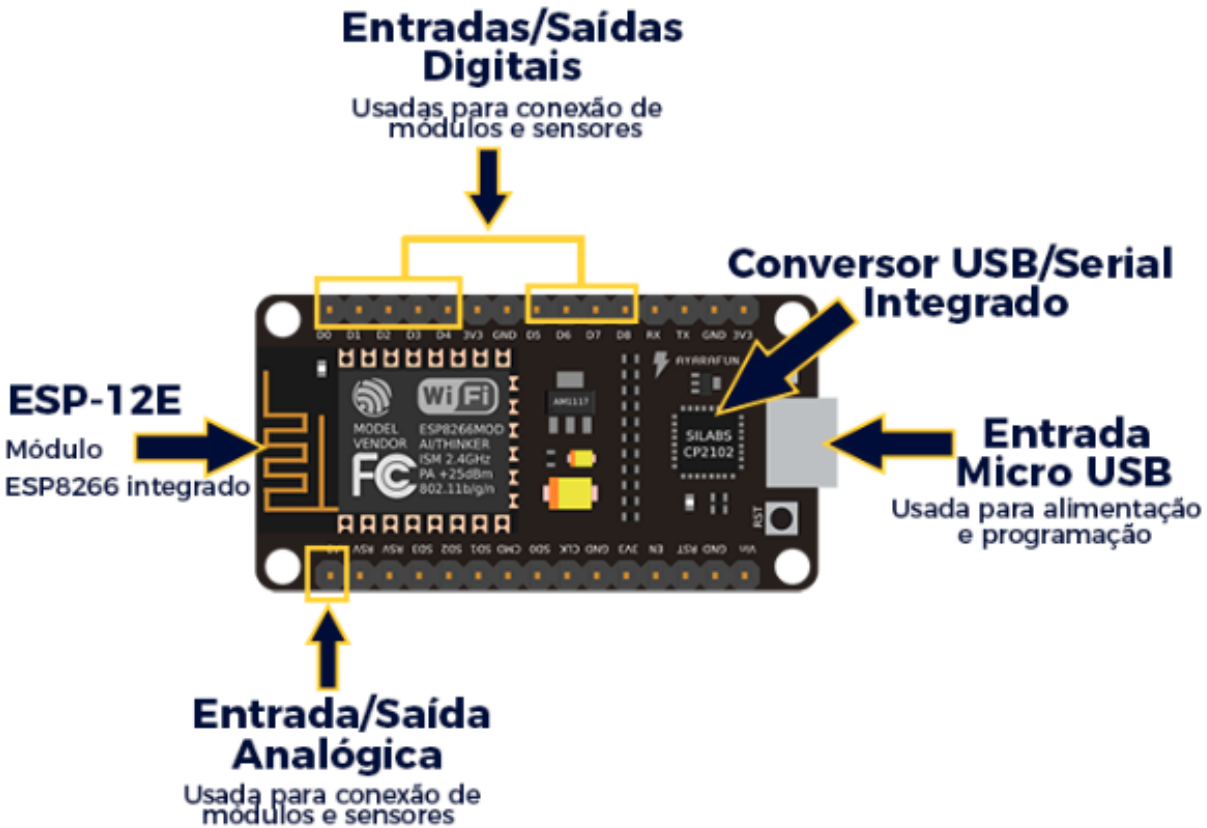
Consumo do ESP8266

Modo	Típico	Unidades
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm	215	mA
Transmit 802.11b, CCK 11Mbps, POUT=+18.5dBm	197	mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm	145	mA
Transmit 802.11n, MCS7, POUT=+14dBm	135	mA
Receive 802.11b, packet length=1024 byte, -80dBm	60	mA
Receive 802.11g, packet length=1024 byte, -70dBm	60	mA
Receive 802.11n, packet length=1024 byte, -65dBm	62	mA
Standby	0.9	mA
Deep sleep	10	uA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.86	mA
Total shutdown	0.5	uA

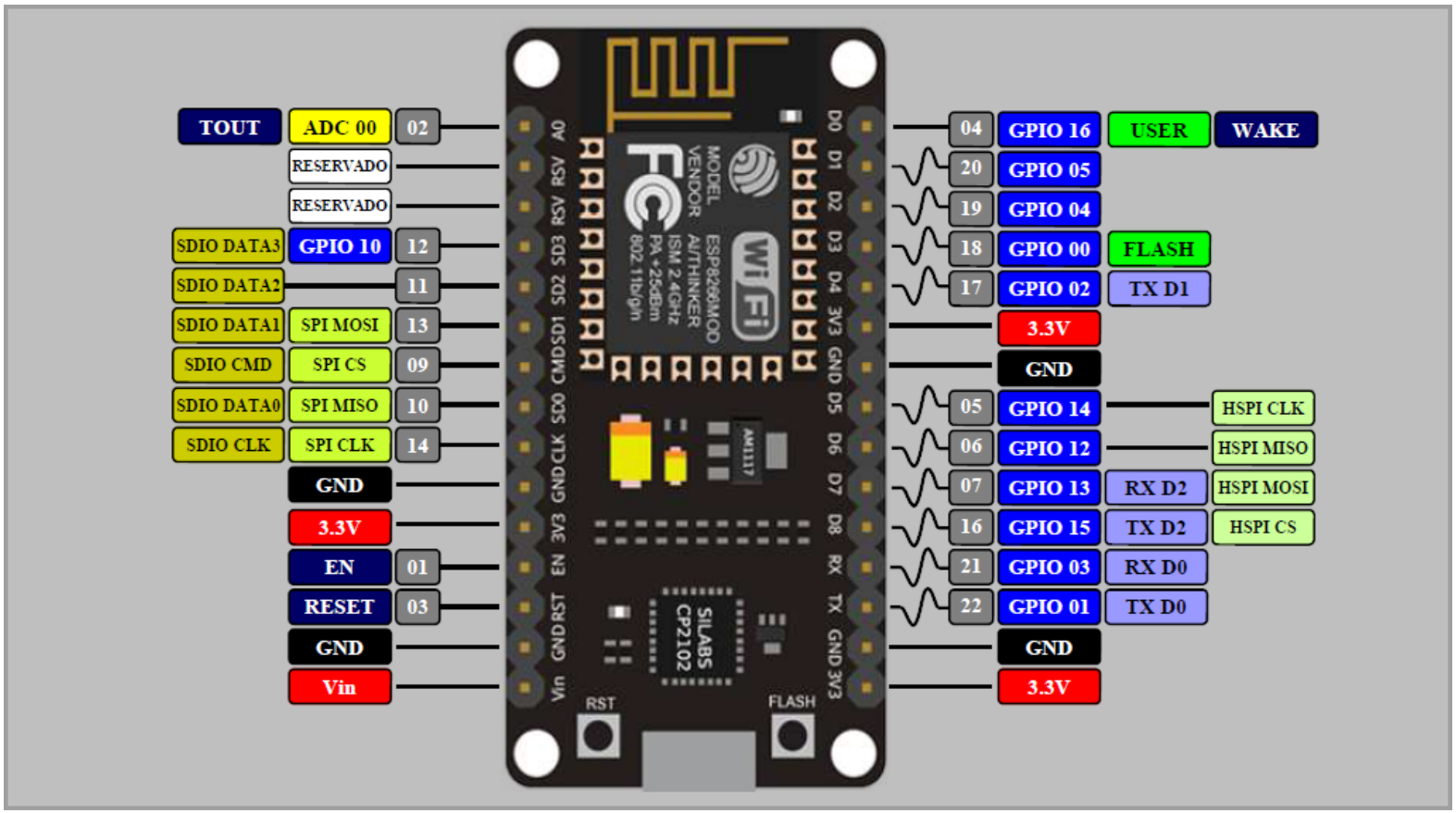
Aplicações

- Tomadas inteligentes;
- Automação residencial;
- Monitoramento remoto;
- Segurança doméstica, comercial e industrial;
- Rede de sensores;
- Controle industrial sem-Fio;
- Monitores de bebês e crianças;
- Eletrônica vestível;
- Dispositivos para localização via Wi-Fi;
- Tags de identificação para segurança;
- Câmeras IP;
- Robótica;
- E muito mais.

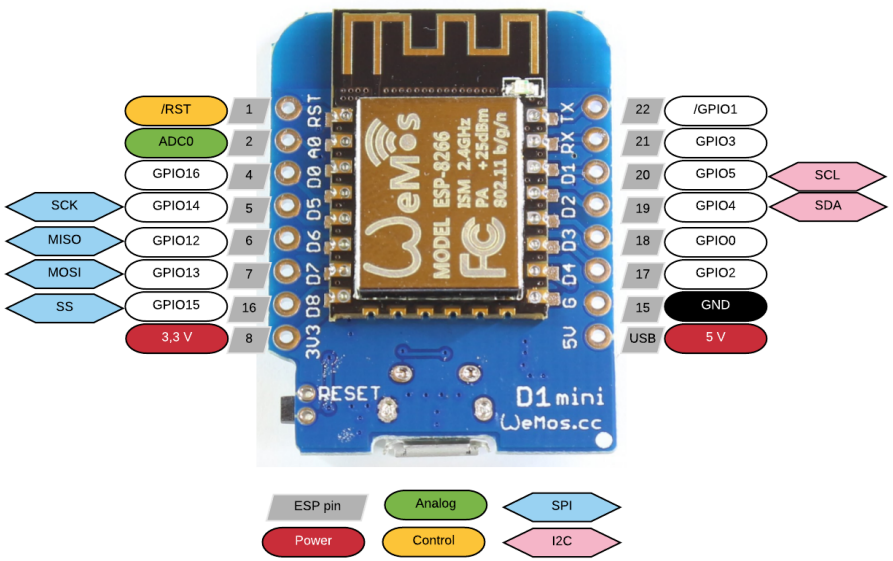
NodeMCU


Entradas/Saídas Digitais
Usadas para conexão de módulos e sensores
Conversor USB/Serial Integrado
ESP-12E
Módulo ESP8266 integrado
Entrada Micro USB
Usada para alimentação e programação
Entrada/Saída Analógica
Usada para conexão de módulos e sensores

Pinos do NodeMCU Development Board v2



Pinos do NodeMCU Wemos D1 Mini



Programando o NodeMCU em Lua

Primeiro passo: Configurar um firmware no site: https://nodemcu-build.com/

Select modules to include

☑ ADC	☐ end user setup	☐ perf	☐ Switec
☐ ADS1115	☑ file	☑ PWM	☐ TCS34725
☐ ADXL345	☐ gdbstub	☐ RC (no docs)	☐ TM1829
☐ AM2320	☑ GPIO	☐ rfswitch	☑ timer
☐ APA102	☐ HDC1080	☐ rotary	☐ TSL2561
☑ bit	☐ HMC5883L	☐ RTC fifo	☐ U8G
☐ Bloom filter	☑ HTTP	☐ RTC mem	☑ UART
☐ BME280	☐ HX711	☑ RTC time	☐ UCG
☐ BME680	☑ I²C	☐ Si7021	☐ websocket
☐ BMP085	☐ L3G4200D	☐ Sigma-delta	☑ WiFi
☐ CoAP	☐ MCP4725	☐ SJSON	☐ WiFi monitor
☐ color utils	☐ mDNS	☐ SNTP	☐ WPS
☐ Cron	☑ MQTT	☐ Somfy	☐ WS2801
☐ crypto	☑ net	☑ SPI	☑ WS2812
☐ DHT	☑ node	☐ SQLite 3	☐ WS2812 effects
☑ DS18B20	☐ 1-Wire	☐ struct	☐ XPT2046
☐ encoder	☐ PCM		

Programando o NodeMCU em Lua

Vamos testar um programa que faz uso dos timers em Lua

```
led_amarelo = 1
led_verde = 2
led_vermelho = 3

state = {0,0,0}

function blink(pin)
        if state[pin]==0 then
                state[pin]=1
                gpio.write(pin,gpio.LOW)
        else
                state[pin]=0
                gpio.write(pin,gpio.HIGH)
        end
end

tmr.alarm(0,500, tmr.ALARM_AUTO, function() blink(led_amarelo) end )
tmr.alarm(1,700, tmr.ALARM_AUTO, function() blink(led_verde) end )
tmr.alarm(2,1100, tmr.ALARM_AUTO, function() blink(led_vermelho) end )
```

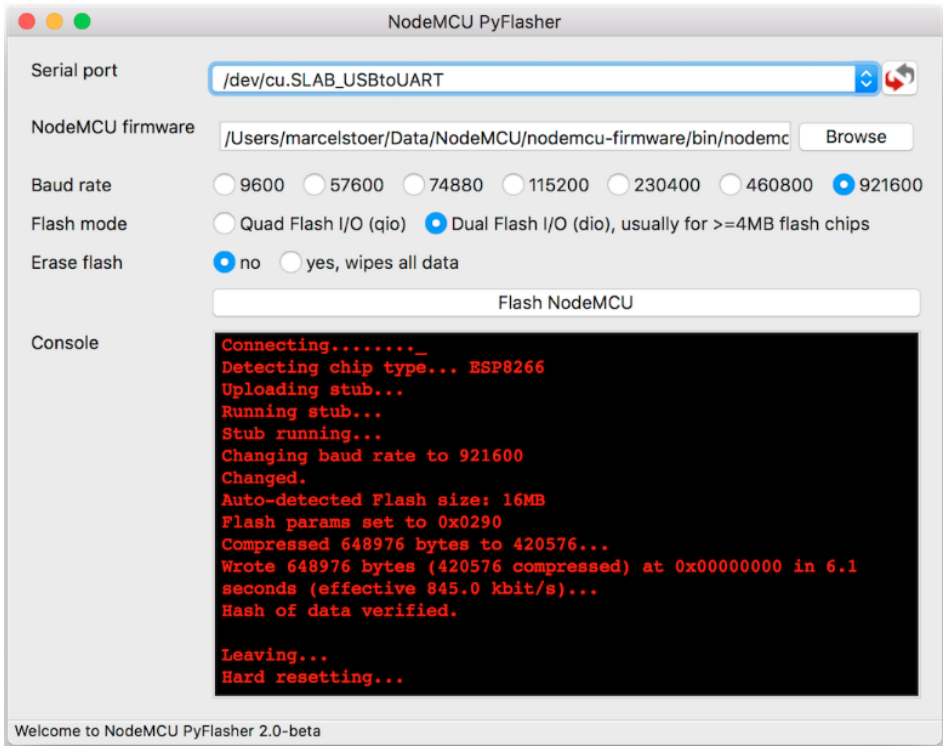
Programando o NodeMCU em Lua

Segundo passo: Escolher as ferramentas

- ESPlorer
- NodeMCU PyFlasher
- esptool.py
- minicom
- gtkterm
- luatool.py

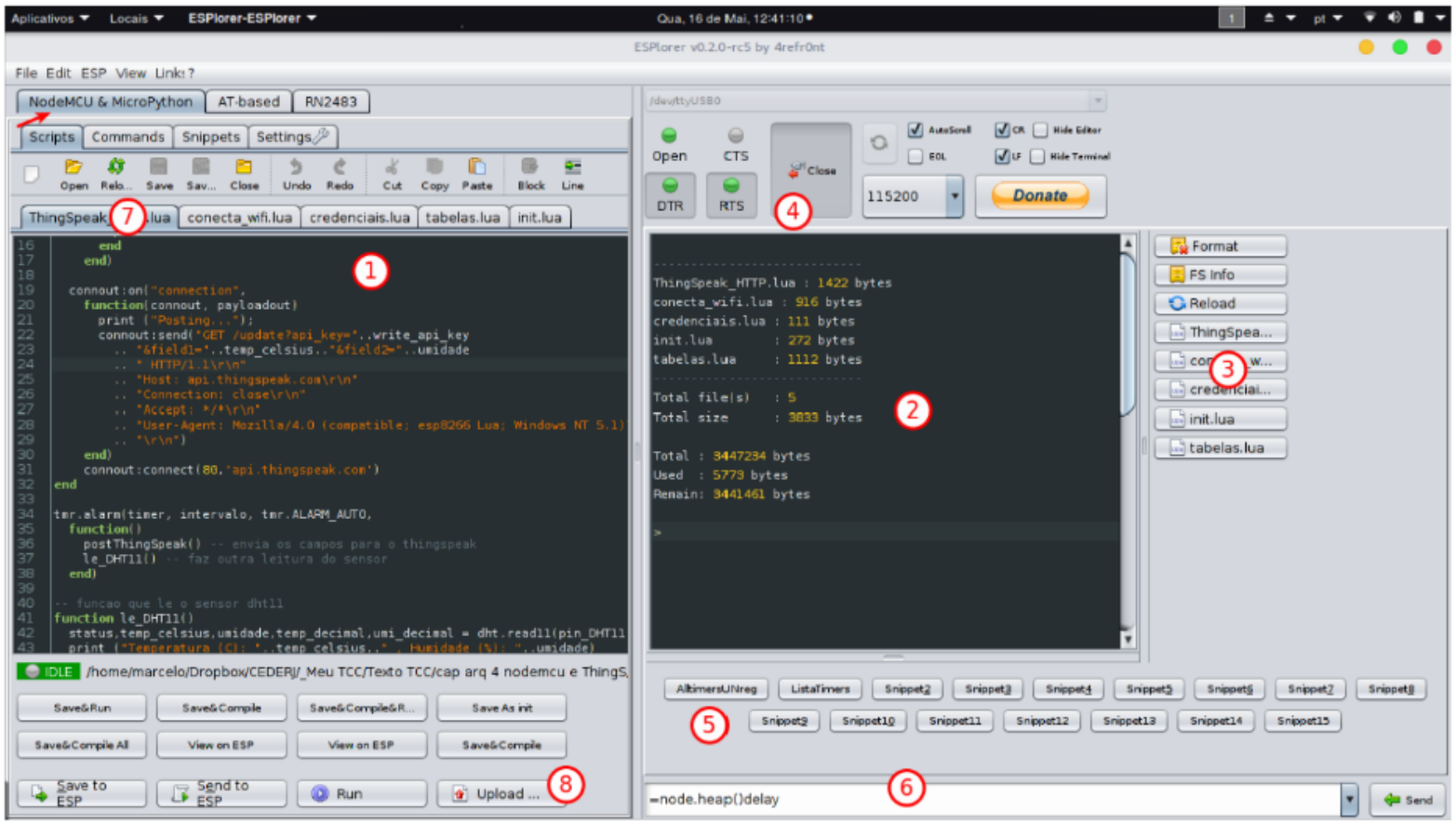
Programando o NodeMCU em Lua

NodeMCU PyFlasher: Self-contained NodeMCU flasher with GUI based on esptool.py and wxPython.



Programando o NodeMCU em Lua

ESPlorer



Programando o NodeMCU em Lua

1. Área de digitação do código fonte
2. Monitor serial
3. Gerenciador de arquivos que permite executar, compilar, renomear e remover um arquivo
4. Controle serial
5. Área de Snippets
6. Caixa de entrada de comandos no modo interativo.
7. Menu arquivo
8. Upload do código para a memória flash do NodeMcu.

Programando o NodeMCU em Lua

Passo a passo no Linux:

1. Instalar a ferramenta esptool:
   pip install esptool
2. Gravar o firmware:
   esptool.py - -port (port) write_flash -fm=dio -fs=32m 0x00000 nodemcu-master...-float.bin
3. Digitar o script Lua em um editor como o GVim
4. Instalar a ferramenta de upload:
   git clone https://github.com/4refr0nt/luatool.git
5. Carregar o programa:
   python2 luatool.py - -port (port) –src init.lua –dest init.lua - -verbose
6. resetar o NodeMCU

Links

1. https://www.embarcados.com.br
2. https://hackaday.io/project/159471/instructions
3. http://mundoprojetado.com.br/introducao-ao-nodemcu-aula-1/