Introdução a Arduino e Raspberry Pi

Prof. Marcel Silva
DCC/IM/UFRRJ

Curso de férias - julho de 2016

Roteiro de hoje

- Programação da semana
- Introdução
  - Sistemas embarcados
  - Motivação
- Arduino
  - Características de hardware e software
  - ‘Hello LED’
- Raspberry Pi
  - Caraterísticas de hw e sw
  - GPIO
  - Node-red

Programação da semana

[18/07] Conceitos básicos sobre Arduino e Raspberry Pi

[19/07] Hands on: LEDs, Matriz de LEDs

[20/07] Hands on: Sensores e outros Inputs

[21/07] Hands on: Display Nokia5110

[22/07] Hands on: Motores elétricos

Antes de começar...

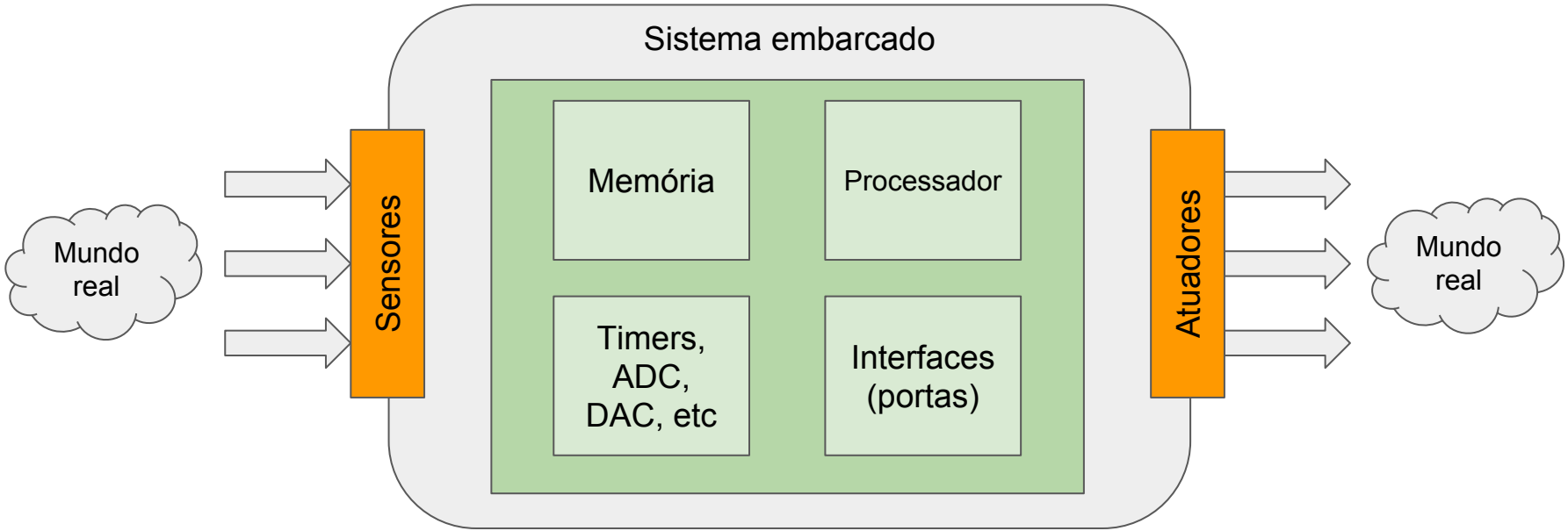
https://vimeo.com/174411588

Sistema embarcado ou embutido

- Microprocessador encapsulado em um chip e que é dedicado a um propósito específico → geralmente controlar um dispositivo ou sistema

- Diferente de computadores de propósito geral (PCs)
  - Projeto específico → reduz tamanho, custo e consumo

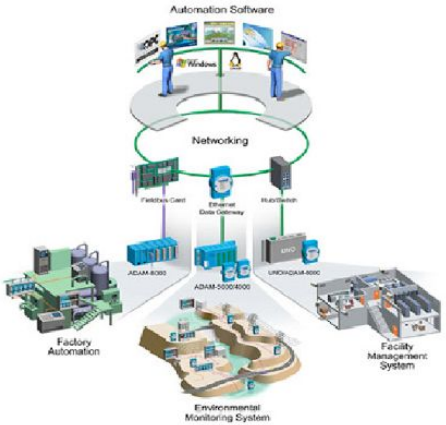
Sistemas embarcados - diagrama geral

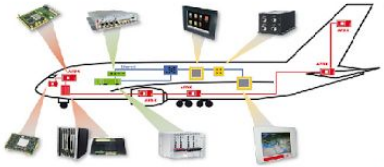


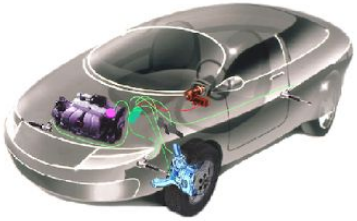
Sistemas embarcados - exemplos

- MP3 player
- relógio digital
- calculadora
- injeção eletrônica
- navegador com GPS
- ‘maquininha’ de cartão
- robô
- semáforo
- roteador wireless
- DVD ou BD player
- controles aviônicos
- mísseis teleguiados
- console de vídeo game
- lavadora de roupas
- marca-passo
- quadricóptero (VANT)
- forno de micro-ondas
- tablet
- medidor de pressão arterial
- televisão
- e vários outros...

Sistemas embarcados - exemplos







Futuro? - Internet das Coisas

- *Internet of Things* (IoT) → TUDO possuirá um sistema embarcado, com possibilidade de comunicação
- Possibilitará aplicações inovadoras
- Exemplos
  - Telemedicina
  - Controle de estoque
  - Ambientes inteligentes (*smart cities*)

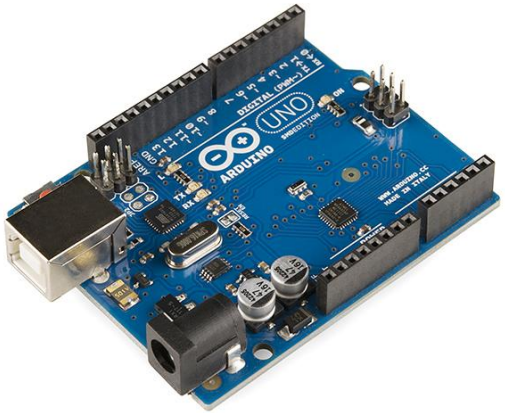
- Na atualidade: Fase de transição → muitas coisas já possuem sistemas embarcados, mas nem todas com capacidade de comunicação

O que isso tem a ver com Arduino e Raspberry Pi?

- Tudo!
- Ambas as plataformas facilitam o desenvolvimento de protótipos de sistemas embarcados (as vezes sendo usados até como solução ‘de fato’)
- Arduino e Raspberry são basicamente pequenos computadores programáveis
- Programáveis em diferentes níveis
  - Arduino → baseado em um microcontrolador programával simples
  - Raspberry Pi → mais parecido com PC convencional, compactado em um único chip e com arquitetura mais simples (ARM)

ARDUINO™

Arduino

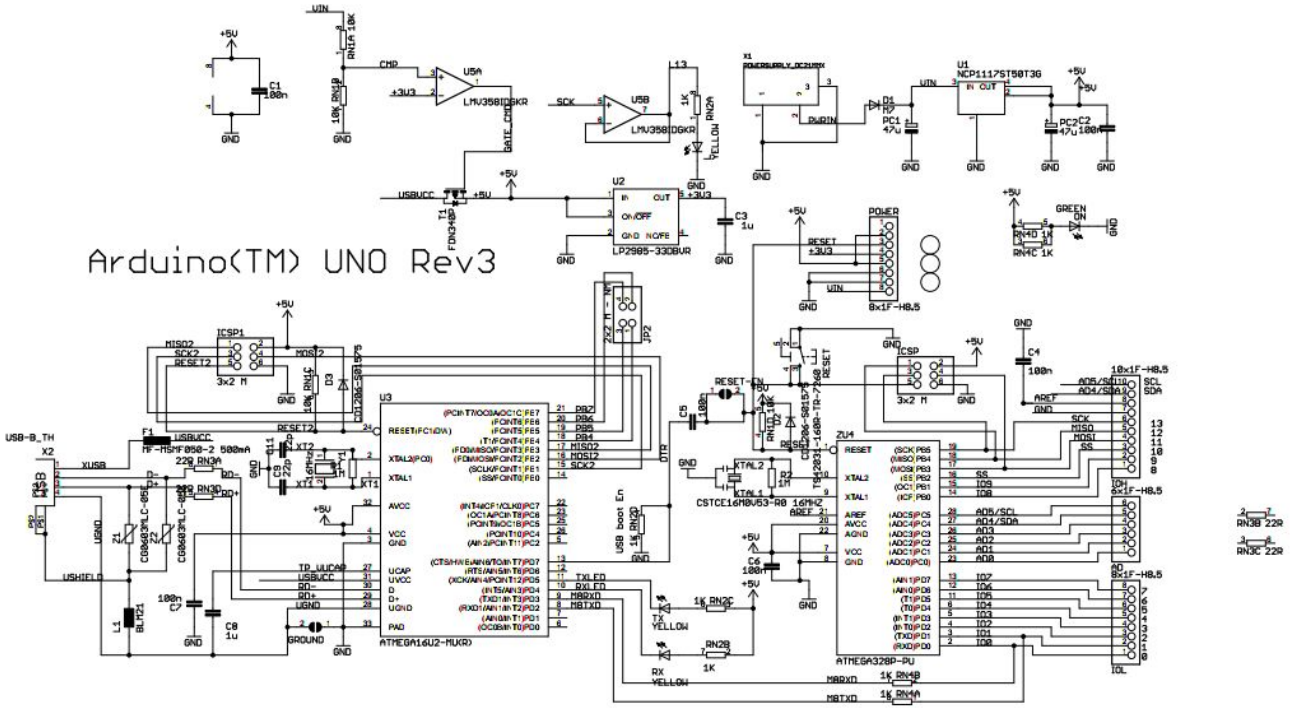


Arduino - O que é?

- Plataforma de ‘hardware livre’ para a prototipagem de sistemas digitais
  - Permite criar sistemas de baixo custo e com baixo tempo de desenvolvimento
- Programável em linguagem de alto nível
  - Muito parecida com C/C++
  - Software é desenvolvido em um host e posteriormente carregado na memória da placa
- Utiliza como base um microcontrolador em conjunto com interfaces de entrada/saída de sinais
  - Placa única
  - Também possui interface serial ou USB para comunicação com host
  - Em conjunto com outros componentes eletrônicos permite criar sistemas digitais poderosos

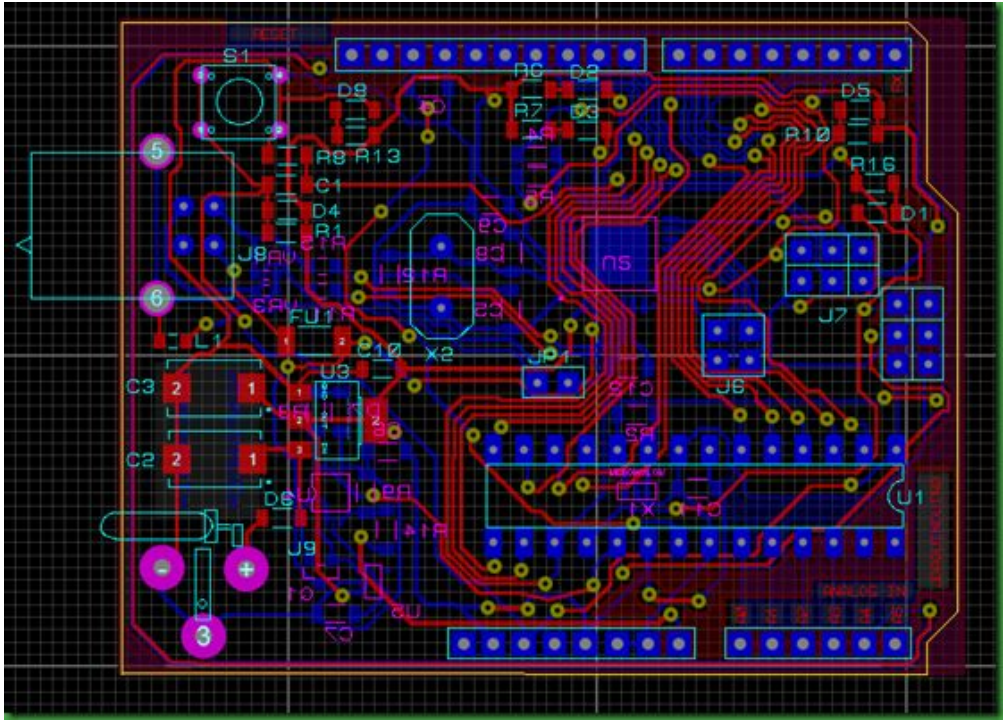
Arduino - Hardware

- **Hardware open-source** → qualquer um pode construir uma placa arduino!



Arduino - Hardware

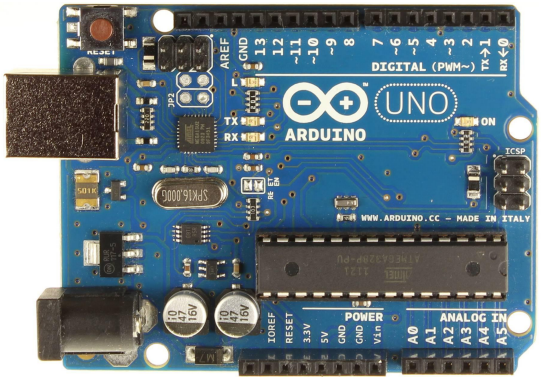
- **Hardware open-source** → qualquer um pode construir uma placa arduino

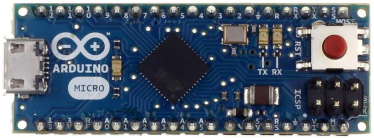


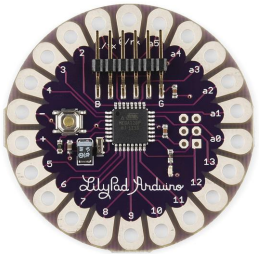
Arduino - Hardware

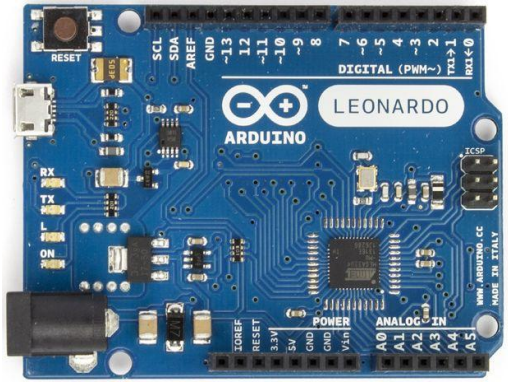
- **Hardware open-source** → qualquer um pode construir uma placa arduino
- Mas marca Arduino é registrada! → briga judicial pelos direitos de uso da marca
  - Entre startup que desenvolvia os projetos e IDE (Arduino LLC) e startup que fabricava as placas (Smart Projects)
  - Mais sobre o caso na versão do Massimo (LLC): http://makezine.com/2015/03/19/massimo-banzi-fighting-for-arduino/

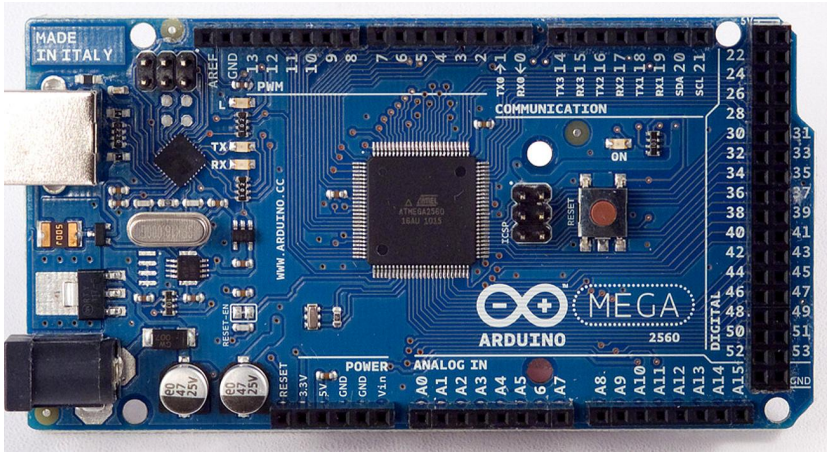
Arduino - Hardware - versões mais antigas



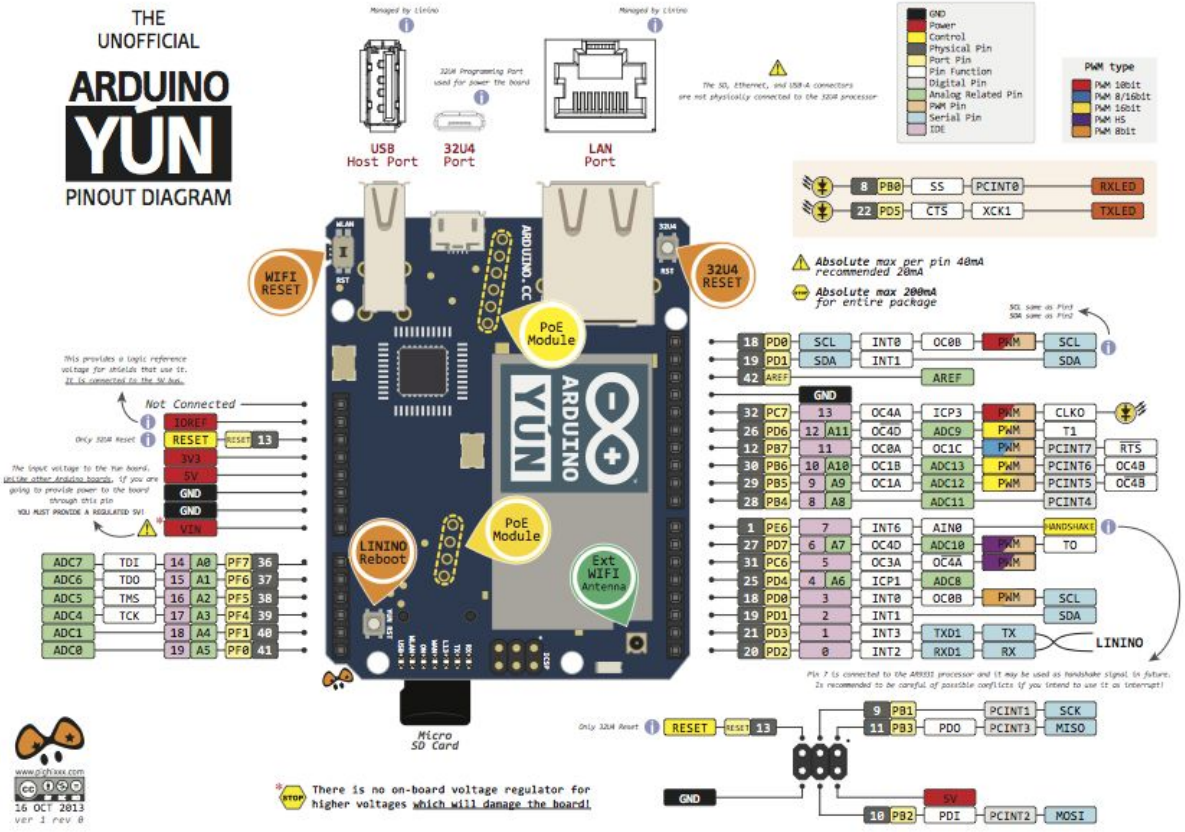








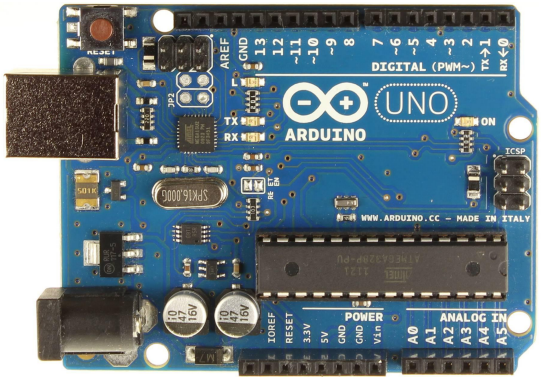
Arduino - Hardware - muitas novas versões...



Arduino - Hardware - versões alternativas

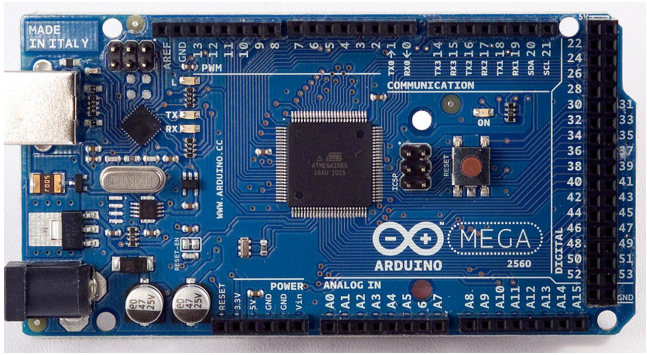
- Clones de arduino
  - Freeduino - http://www.freeduino.org/
  - Seeduino - http://www.seeedstudio.com/wiki/Seeeduino_v2.21
  - Brasuino - http://brasuino.holoscopio.com/
- Similares
  - ChipKIT - http://chipkit.net/
  - Olimexino - https://www.olimex.com/Products/Duino/STM32/OLIMEXINO-STM32/
  - Texas Instrument LaunchPad - http://www.ti.com/tool/msp-exp430g2
- Compatíveis
  - Intel Galileo - https://software.intel.com/pt-br/iot/hardware/galileo

Arduino - Hardware - Detalhes do UNO



- Microcontrolador: ATmega328
- Tensão de operação: 5V
- Tensão de entrada (recomendada): 7-12V
- Pinos digitais de E/S: 14 (6 podem ter sinal PWM)
- Pinos com entrada analógica: 6
- Corrente máxima por pino de E/S: 40 mA
- Memória Flash (de programa): 32 kB, com 0,5 kB usados pelo bootloader
- Memória SRAM: 2 kB - EEPROM: 1 kB
- Frequência de clock: 16 MHz

Arduino - Hardware - Detalhes do Mega2560



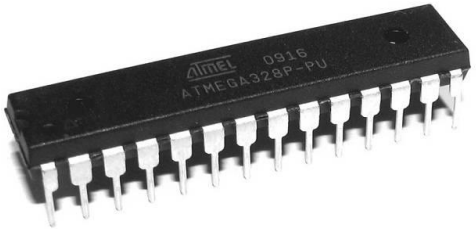
- Microcontrolador: ATmega2560
- Tensão de operação: 5V
- Tensão de entrada (recomendada): 7-12V
- Pinos digitais de E/S: 54 (15 podem ter sinal PWM);
- Pinos com entrada analógica: 16
- Corrente máxima por pino de E/S: 40 mA
- Memória Flash (de programa): 256 kB, com 8 kB usados pelo bootloader
- Memória SRAM: 8 kB - EEPROM: 4 kB
- Frequência de clock: 16 MHz

Arduino - Hardware - Microcontrolador

- Chip - “coração” do sistema
- CPU de pequeno porte, capaz de executar um conjunto de instruções
  - Ou seja, possui um microprocessador!
- Instruções simples e rápidas
- Possui memória(s)
- Possui periféricos
- Pode se comunicar com outros periféricos

Arduino - Hardware - Microcontrolador

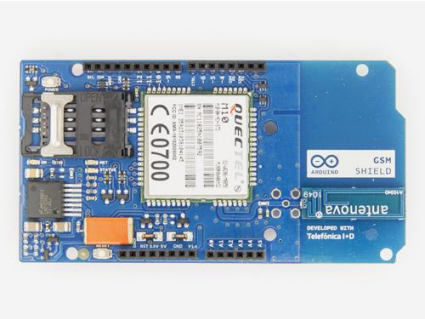
- Arduino - Família MegaAVR (ATMEL)
  - http://www.atmel.com/pt/br/products/microcontrollers/avr/megaAVR.aspx
  - Existem outras famílias produzidas por outros fabricantes (Intel, Motorola, Texas Instruments, Microchip, etc)
- Exemplos de MegaAVR usados em arduinos:
  - **ATmega168**: Diecimila, Duemilanove, Nano, LilyPad
  - **ATmega328P**: Duemilanove, Nano, Fio, LilyPad, Uno
  - **ATmega1280**: Mega
  - **ATmega2560**: Mega2560
  - **ATmega32u4**: Leonardo, Esplora, LilyPad USB, Yún, Robot
  - **AT91SAM3X8E**: Due

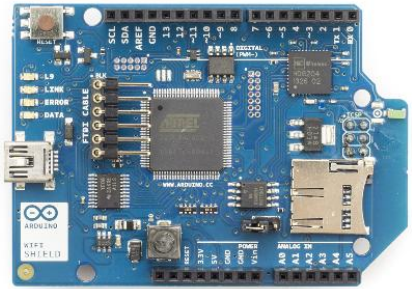


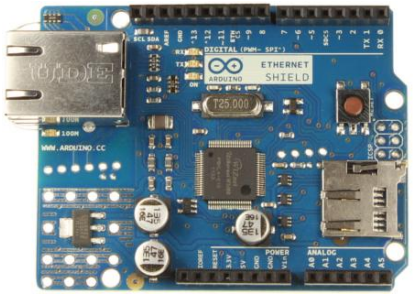
Ardunio - Hardware - shields

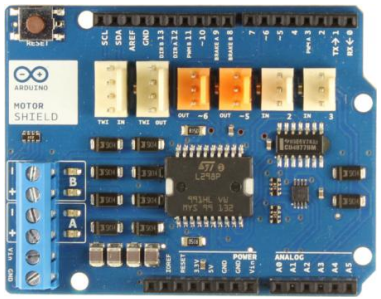
Site com uma lista de shields:
http://shieldlist.org

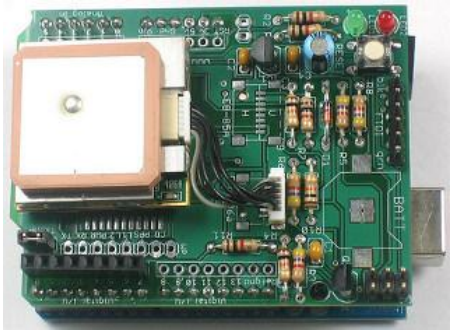
- Shields → placas de extensão para o arduino
  - Permitem adicionar funcionalidades nas placas convencionais
- Exemplos:
  - WiFi shield
  - Ethernet shield
  - GSM shield
  - Motor shield
  - GPS shield





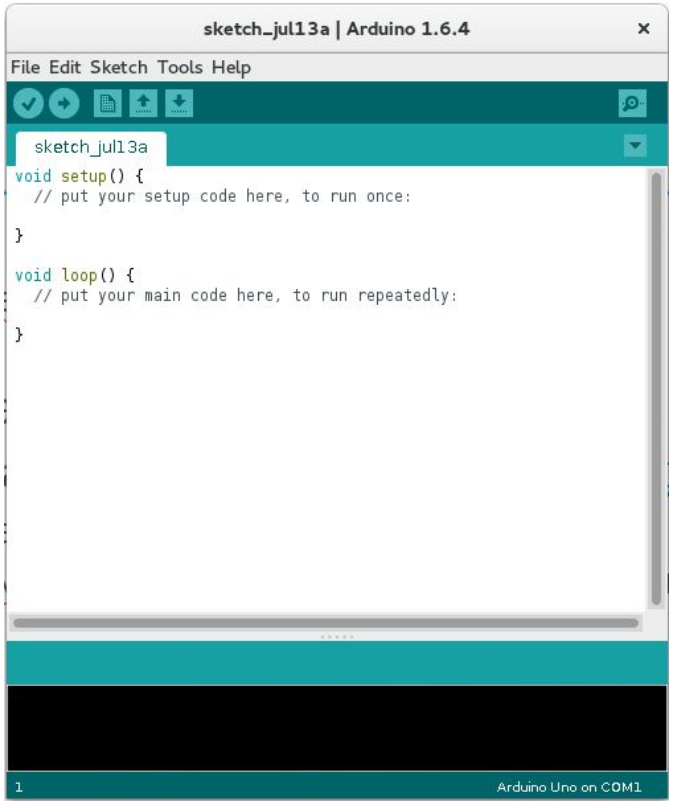






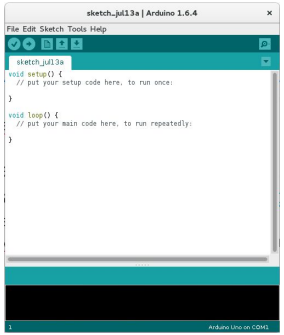
Arduino - Software

- Microcontrolador executa instruções armazenadas em sua memória
  - Linguagem AVR (um tipo de assemble)
  - Difícil programar!
- Programação facilitada através de IDE open source disponibilizada pelos desenvolvedores do arduino → plataforma Wiring (http://wiring.org.co/)
  - Linguagem de mais alto nível → semelhante a C/C++
  - IDE faz a conversão do código para AVR e compila no formato aceito pelo microcontrolador



Arduino - software - etapas de desenvolvimento

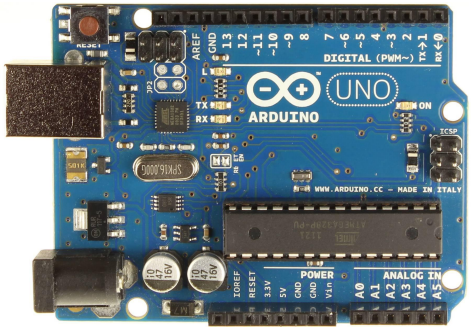
IDE



compilador

arquivo.hex

programador



Arduino - software - “hello LED”

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - software - “hello LED”

‘Setup’ - executado apenas uma vez quando a placa é iniciada

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - software - “hello LED”

‘Setup’ - executado apenas uma vez quando a placa é iniciada

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}
```

‘Loop’ - executado infinitas vezes após a execução da parte de inicialização

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - software - “hello LED”

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - software - “hello LED”

pinMode() → Pino 13 configurado como saída (OUTPUT)

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - software - “hello LED”

pinMode() → Pino 13 configurado como saída (OUTPUT)

digitalWrite() → escreve nível lógico (HIGH ou LOW) no pino especificado

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - software - “hello LED”

pinMode() → Pino 13 configurado como saída (OUTPUT)

digitalWrite() → escreve nível lógico (HIGH ou LOW) no pino especificado

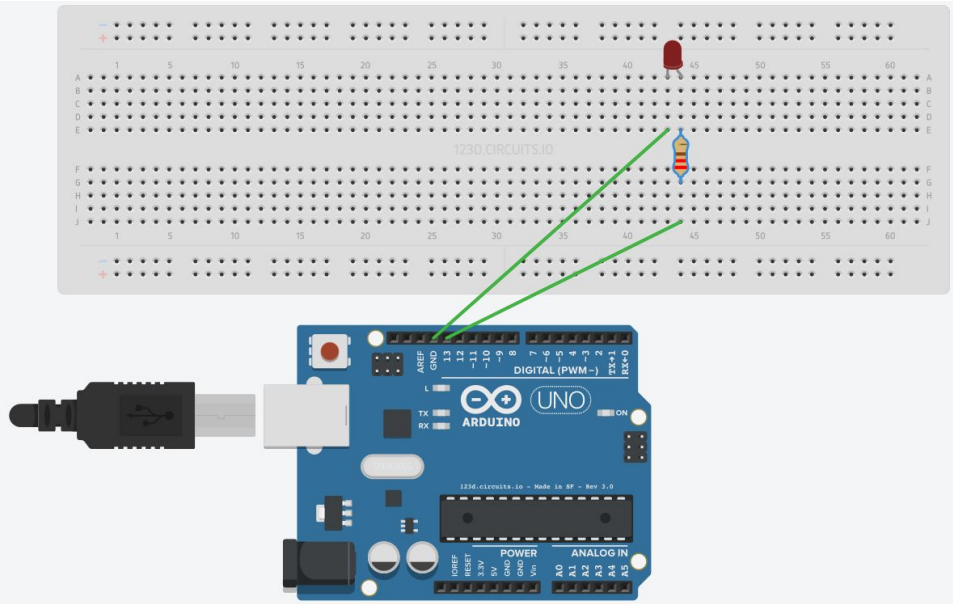
delay() → espera por um inteiro em milisegundos

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino - “hello LED” - simulação

- Exemplo usando plataforma online de simulação - circuits.io
  - Protoboard, LED, Resistor e Arduino UNO



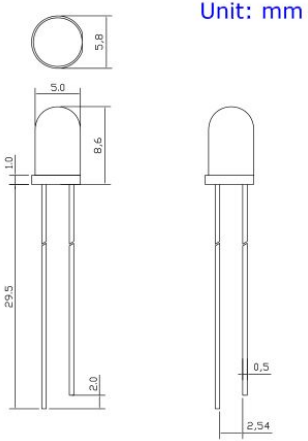
https://circuits.io/circuits/2436819-hello-led/embed

Arduino - “hello LED” - detalhes

- O que acontece na prática?
  - digitalWrite(13,HIGH) → faz o nível de tensão no pino 13 igual à 5V
  - induz passagem de corrente no circuito
  - diagrama no quadro
- Para que serve o resistor?
  - evita uma corrente muito alta no circuito → poderia ‘queimar’ o LED
- V=R.I (tensão igual a resistência multiplicada pela corrente)
  - com V=5V e R=220 Ohms, qual o valor da corrente?
  - aproximadamente 22,7 mA
  - quanto maior a corrente, mais intenso o brilho do LED
  - qual o limite?

Arduino - “hello LED” - detalhes

- Todo componente eletrônico possui uma documentação que o descreve → **dataSheet**
  - Informa em detalhes o comportamento do componente em cada situação
  - Limites máximos toleráveis para o bom funcionamento
- Ex.: datasheet LED 5mm (https://www.sparkfun.com/datasheets/Components/LED/COM-09590-YSL-R531R3D-D2.pdf)

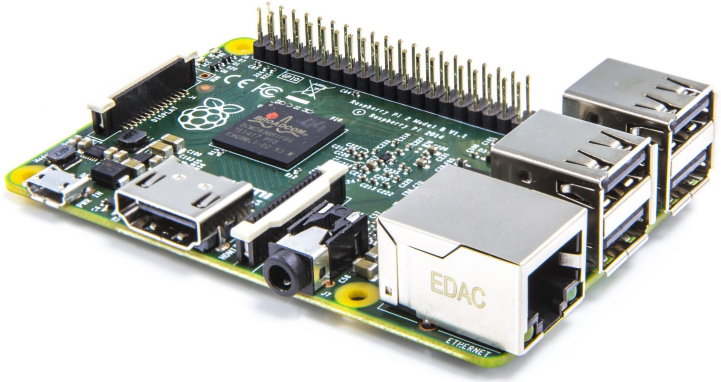


ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage (V_R=5V)	I_R	10	uA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	℃
Storage Temperature	T_{STG}	-40 ~ 100	℃
Lead Soldering Temperature	T_{SOL}	Max. 260℃ for 3 Sec. Max. (3mm from the base of the expoxy bulb)	

Arduino - “hello LED” - detalhes

- Como usar um componente que eu não conheço?
  - Primeiro passo → encontrar o dataSheet!
  - De posse dos detalhes do seu funcionamento, podemos construir circuitos utilizando o componente
  - Faremos isso bastante nas próximas aulas!

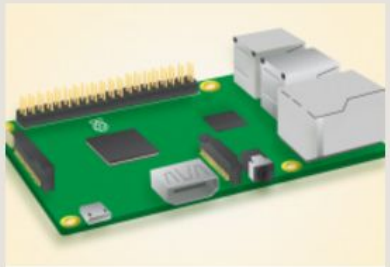
Raspberry Pi



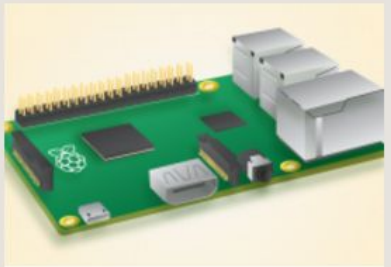
Raspberry Pi - www.raspberrypi.org

- Produzido originalmente para ser um computador de baixo custo
  - Computador completo por apenas 35 dolares (no Brasil, 250-300 reais)
  - Totalmente open source (hardware e software (Linux))
  - Com propósito de facilitar o ensino de informática (programação) para crianças
- Diferente do arduino, é um computador mais completo em um único chip
  - GPU (com capacidade de reproduzir vídeos em 1080p, saída HDMI e interface dedicada para conexão de display LCD)
  - Interface de áudio (com saída P2)
  - Interface de rede (Ethernet 10/100 Mbps e também WiFi+Blutooth na versão 3)
  - Interface dedicada para câmera
  - Portas USB (recebe qualquer tipo de dispositivo USB)
  - Comunicação serial (SPI) e paralela (DPI)
  - Interface GPIO (pinos semelhantes aos do arduino que podem ser usados para controlar outros tipos de hardware!)

Raspberry Pi - versões



RASPBERRY PI 3 MODEL B

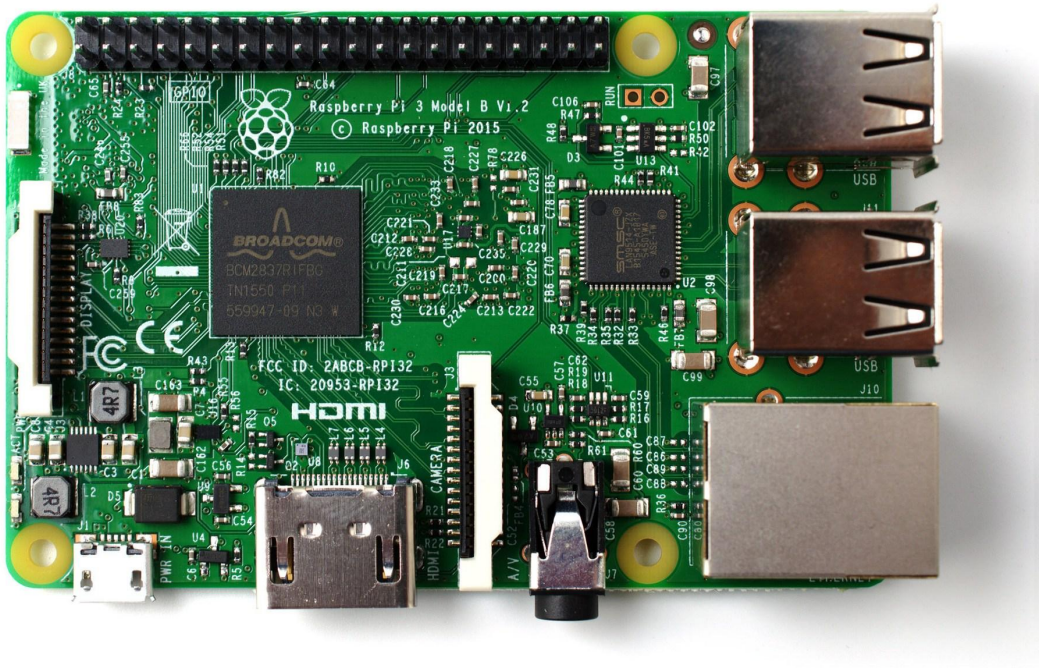


RASPBERRY PI 2 MODEL B

RASPBERRY PI 1 MODEL B+

RASPBERRY PI 1 MODEL A+

Raspberry Pi - versão 3 modelo B



Raspberry Pi - comparação entre versões

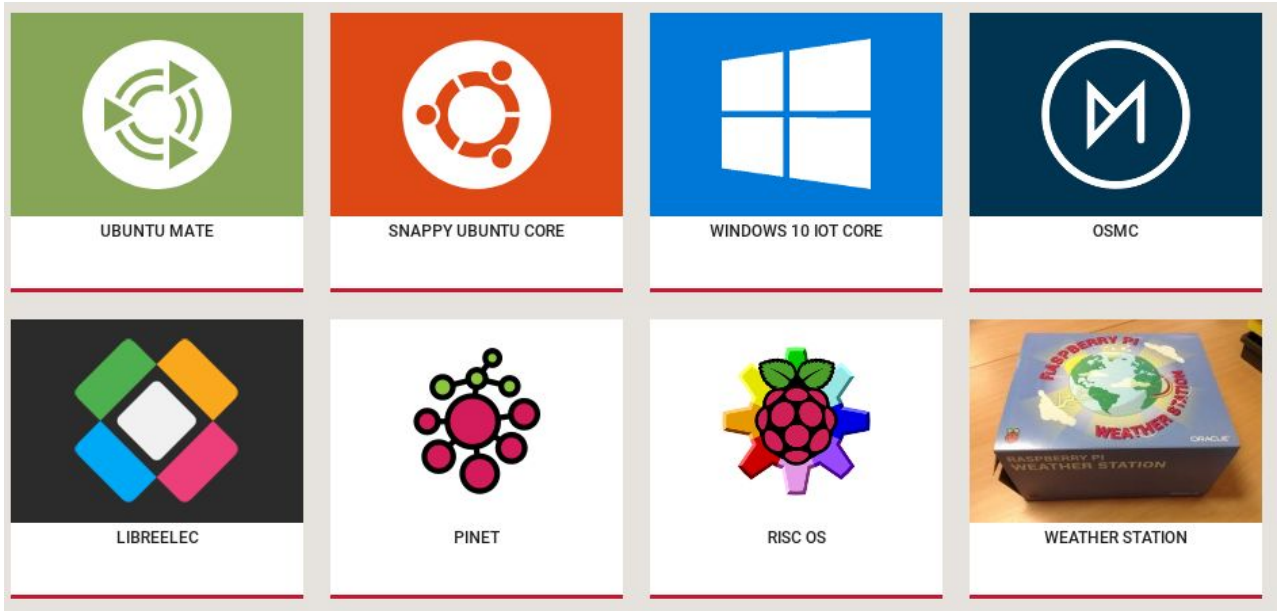
Type	Model A		Model B			
Generation	1	1 +	1	1 +	2	3
Release date	February 2013	November 2014[41]	April–June 2012	July 2014[42]	February 2015[17]	February 2016[18]
Target price	25 US$	20 US$[45]	35 US$[46]	25 US$[47]	35 US$	35 US$
Architecture	ARMv6 (32-bit)				ARMv7 (32-bit)	ARMv8 (64/32-bit)
SoC	Broadcom BCM2835[15]				Broadcom BCM2836	Broadcom BCM2837
CPU	700 MHz single-core ARM1176JZF-S[15]				900 MHz 32-bit quad-core ARM Cortex-A7	1.2 GHz 64-bit quad-core ARM Cortex-A53
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz)[48][49] OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS) MPEG-2 and VC-1 (with license),[50] 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder[15] (BCM2837: 1080p60)					
Memory (SDRAM)	256 MB (shared with GPU)	512 MB (shared with GPU) as of 4 May 2016. Older boards had 256 MB (shared with GPU)[51]			1 GB (shared with GPU)	
USB 2.0 ports[32]	1 (direct from BCM2835 chip)		2 (via the on-board 3-port USB hub)[52]		4 (via the on-board 5-port USB hub)[42][31]	
Video input	15-pin MIPI camera interface (CSI) connector, used with the Raspberry Pi camera or Raspberry Pi NoIR camera[53]					

Raspberry Pi - comparação entre versões

Type	Model A		Model B			
Generation	1	1 +	1	1 +	2	3
Video outputs	HDMI (rev 1.3) composite video (RCA jack)	HDMI (rev 1.3), composite video (3.5 mm TRRS jack)	HDMI (rev 1.3), composite video (RCA jack)	HDMI (rev 1.3), composite video (3.5 mm TRRS jack)		
Audio inputs	As of revision 2 boards via I²S[61]					
Audio outputs	Analog via 3.5 mm phone jack; digital via HDMI and, as of revision 2 boards, I²S					
On-board storage[32]	SD / MMC / SDIO card slot (3.3 V with card power only)	MicroSDHC slot[42]	SD / MMC / SDIO card slot	MicroSDHC slot		
On-board network[32]	None[62]		10/100 Mbit/s Ethernet (8P8C) USB adapter on the USB hub[52]			10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1
Low-level peripherals	8× GPIO[63] plus the following, which can also be used as GPIO: UART, I²C bus, SPI bus with two chip selects, I²S audio[64] +3.3 V, +5 V, ground[48][65]	17× GPIO plus the same specific functions, and HAT ID bus	8× GPIO plus the following, which can also be used as GPIO: UART, I²C bus, SPI bus with two chip selects, I²S audio +3.3 V, +5 V, ground. An additional 4× GPIO are available on the P5 pad if the user is willing to make solder connections	17× GPIO plus the same specific functions, and HAT ID bus		
Power ratings	300 mA (1.5 W)[67]	200 mA (1 W)[68]	700 mA (3.5 W)	600 mA (3.0 W)[42]	800 mA[69] (4.0 W)[70]	

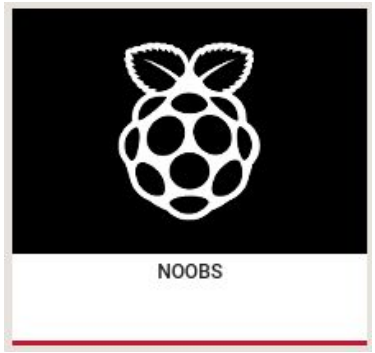
Raspberry Pi - software

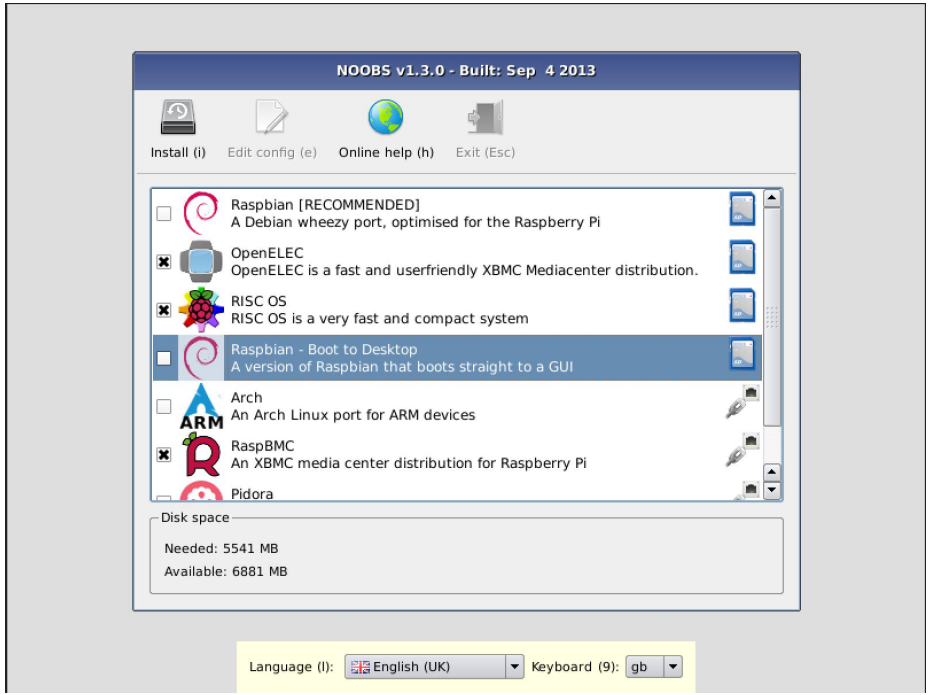
- Executa sistema linux própio para arquitetura ARM
  - Instalado no cartão de memória
- Diferentes distribuições:



Raspberry Pi - software

- Noobs → Ferramenta para facilitar a instalação do SO
  - Já possui uma instalação básica do Raspbian
  - Outras distros podem ser baixadas pela Internet
- Interface gráfica permite escolher o sistema a ser instalado
  - Permite multiboot
  - Interface acessível sempre que desejar

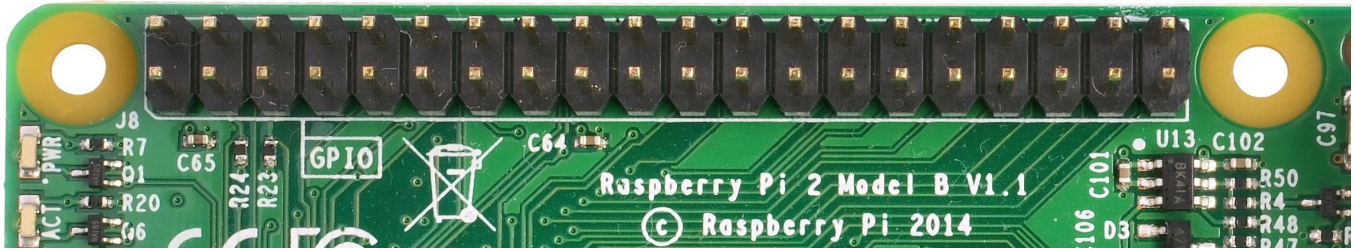


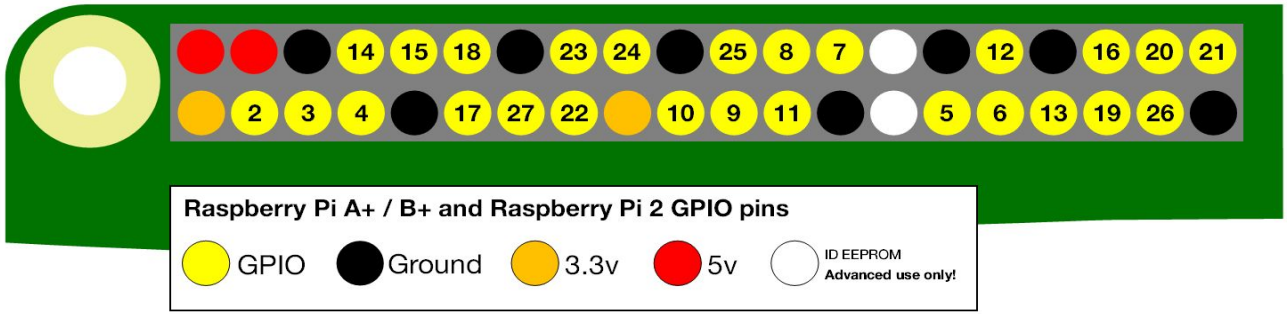


Raspberry Pi - software - como programar?

- Importante: Arquitetura é ARM!!!
  - Programas devem ser compilados adequadamente (Cross-compiling)
  - Maioria das distros Linux possuem ferramentas adequadas
- Alternativa é compilar diretamente no raspberry
  - Mais lento do que num computador convencional
- Linguagens interpretadas não tem este problema
  - Raspberry tem suporte para uma vasta gama de linguagens

Raspberry Pi - GPIO *General Purpose Input/Output*





Raspberry Pi - GPIO

- Várias bibliotecas disponíveis
- WiringPi fornece o comando gpio
- Exemplo: blink.sh

```
#! /bin/bash

gpio mode 0 out

while true; do
    gpio write 0 1
    sleep 1
    gpio write 0 0
    sleep 1
done
```

Raspberry Pi - GPIO

- Mesmo exemplo de antes, mas em linguagem C (wiringPi.h)
- Exemplo: blink.c

```
#include <wiringPi.h>
int main (void)
{
  wiringPiSetup () ;
  pinMode (0, OUTPUT) ;
  for (;;)
  {
      digitalWrite (0, HIGH) ; delay (500) ;
      digitalWrite (0,  LOW) ; delay (500) ;
  }
  return 0 ;
}
```

Raspberry Pi - Node-red - http://nodered.org/

- Ferramenta para programar tarefas no raspberryPi
  - Na verdade, roda em qualquer máquina
- Permite criar fluxos de tarefas a serem realizadas
- Exemplo: controle do LED

- O que não fazer com GPIO?
  - Controle de equipamentos que demandam potência elevada (motores, etc)
  - Pode causar danos irreversíveis a placa (brick!)
  - Arduino e semelhantes são mais indicados