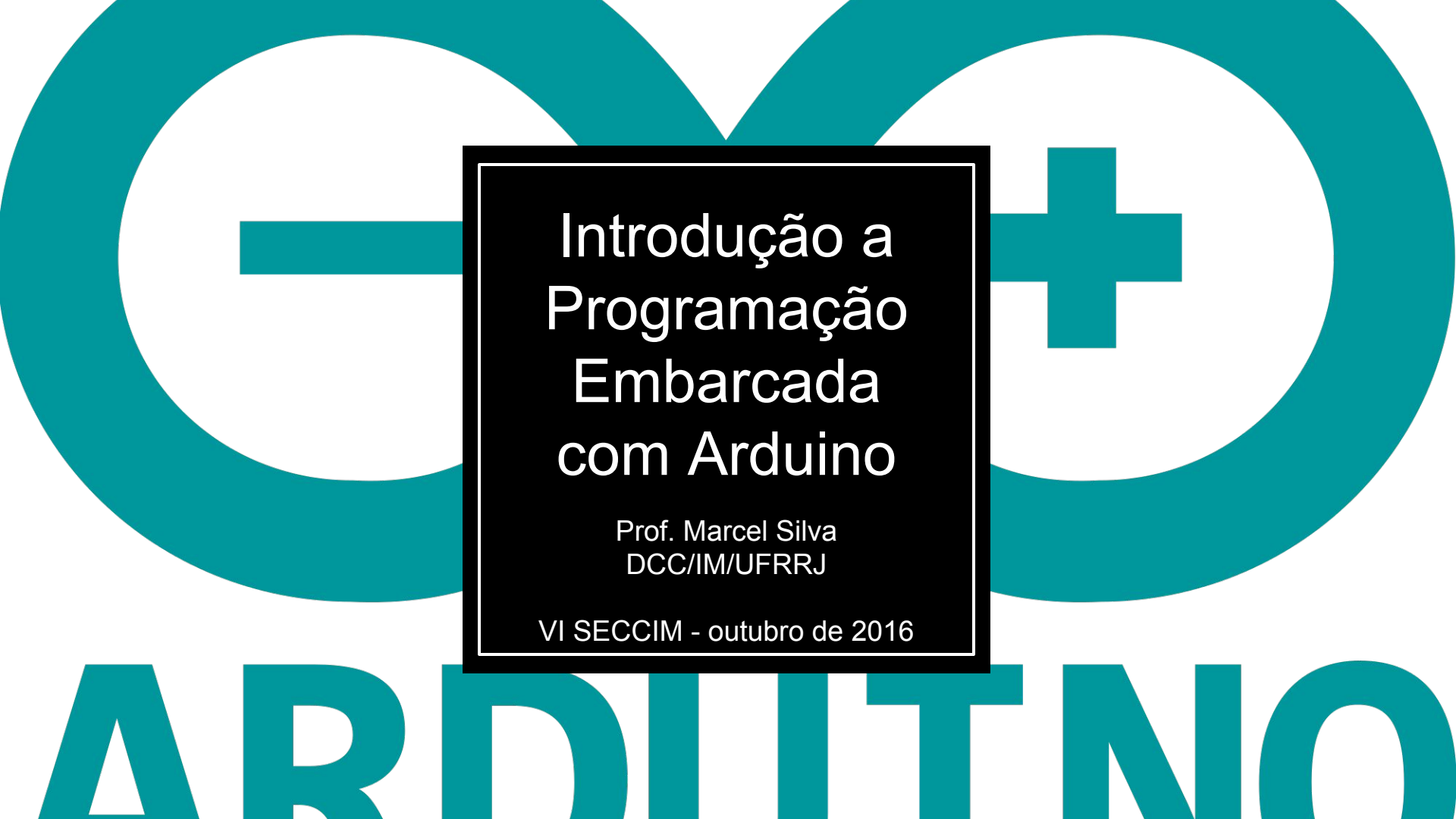


Introdução a Programação Embarcada com Arduino

Prof. Marcel Silva
DCC/IM/UFRRJ

VI SECCIM - outubro de 2016

Roteiro

- Introdução
  - Sistemas embarcados
  - Motivação
- Arduino
  - Características de hardware
  - Características de software
- Exemplos
  - Hello LED
  - LED pulse
  - Sensor ultrassônico
  - Motor elétrico
- Prática final - “sensor de estacionamento”

Antes de começar, um vídeo...

https://www.dcc.ufrrj.br/~marcel/arduino-seccim/batalha.mp4

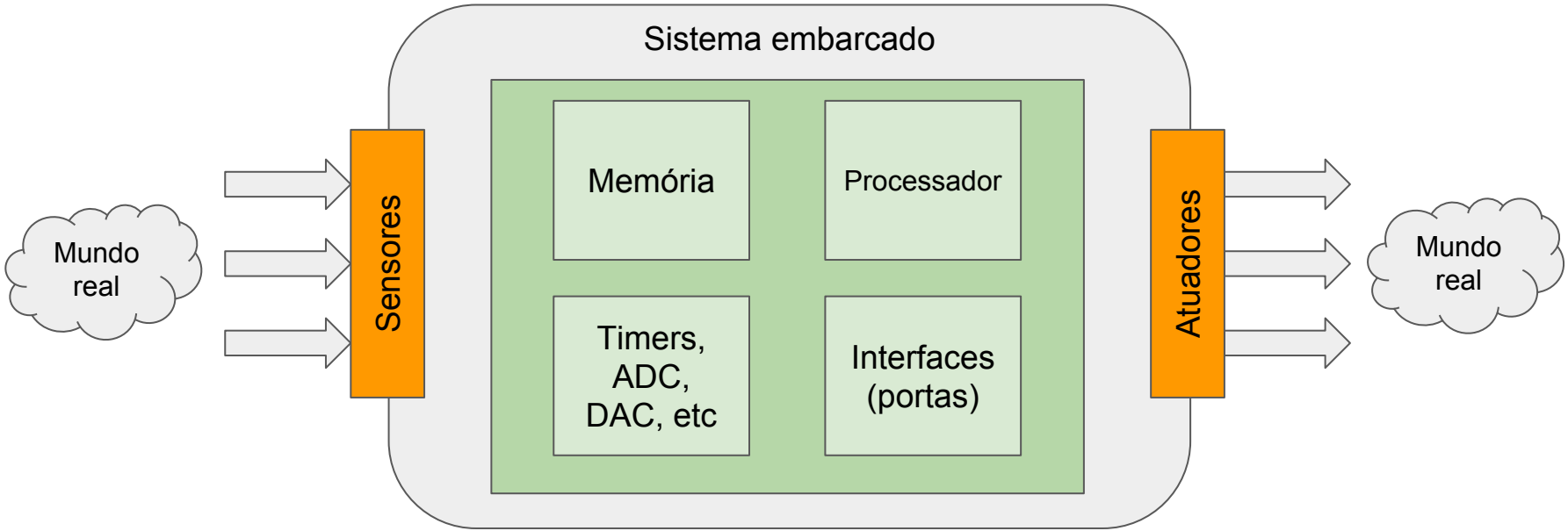
Sistema embarcado ou embutido

- Não é um computador de propósito geral (ex.: PCs)

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

- Projeto específico → reduz tamanho, custo e consumo

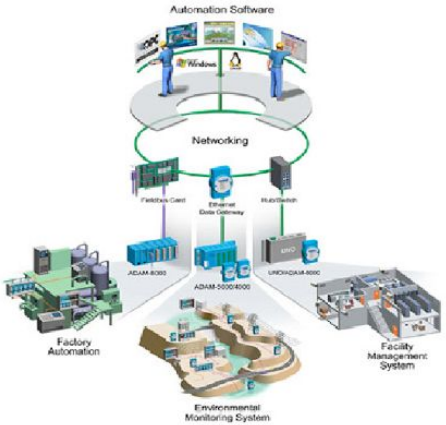
Sistemas embarcados - diagrama geral

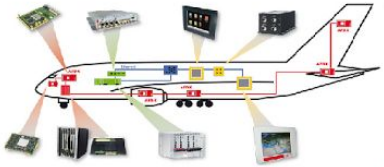


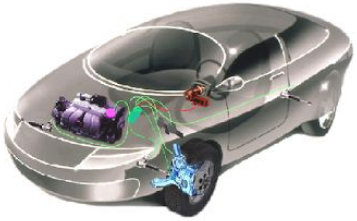
Sistemas embarcados - inúmeros exemplos

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

Sistemas embarcados - inúmeros exemplos







Futuro? - Internet das Coisas

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

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

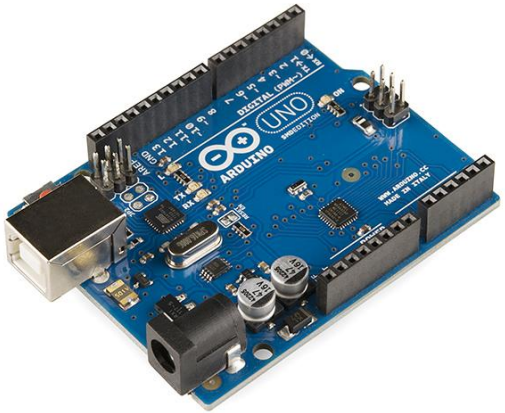
O que isso tem a ver com Arduino?

- Tudo!

- É uma plataforma para o desenvolvimento de sistemas embarcados, que facilitam a criação de protótipos e, as vezes, é usado até como solução ‘de fato’

ARDUINO™

Arduino

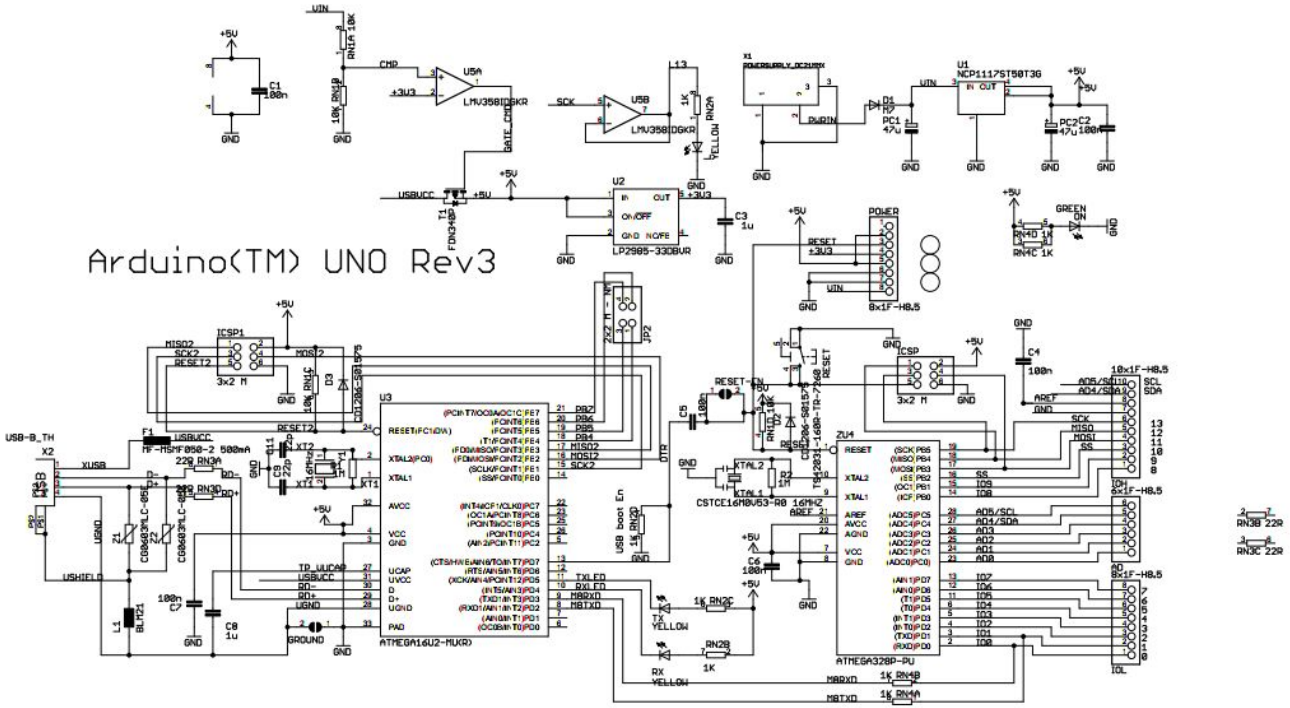


Arduino - O que é?

- 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

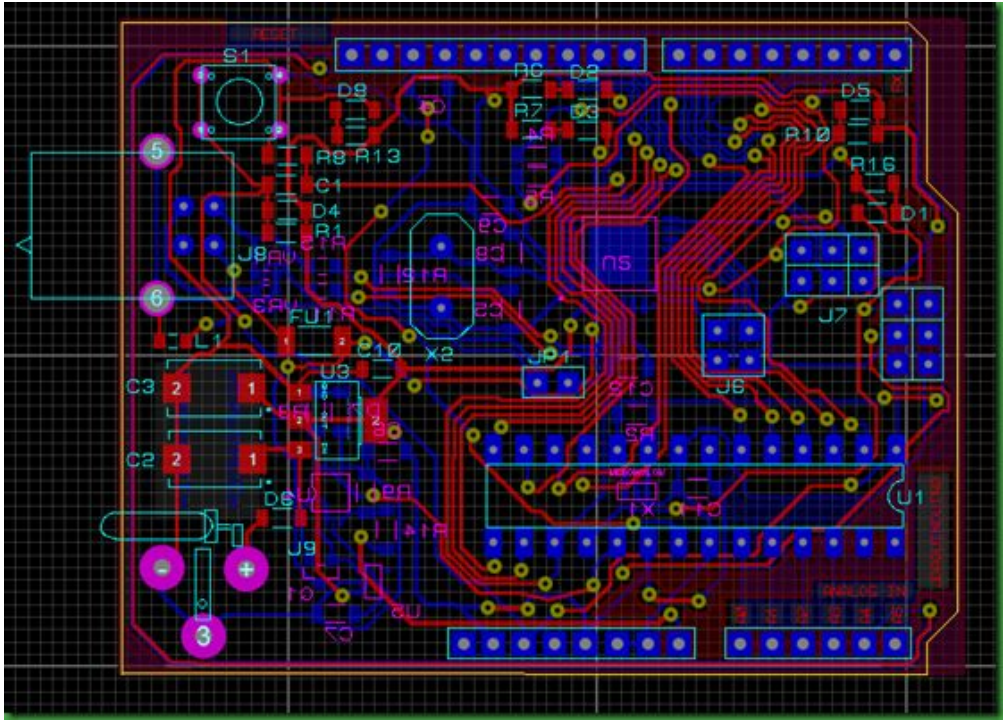
Arduino - Hardware

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



Arduino - Hardware

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

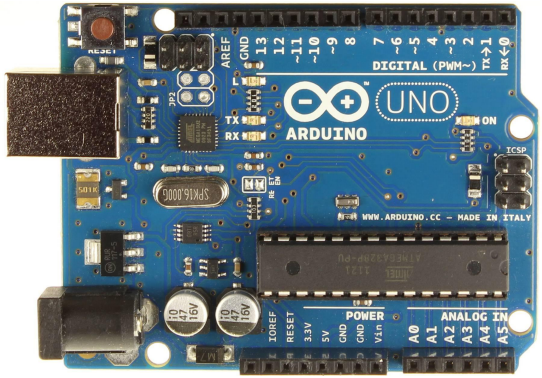


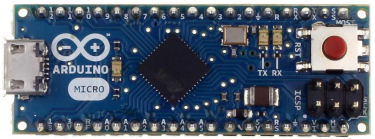
Arduino - Hardware

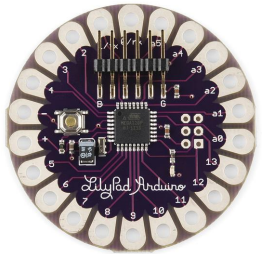
- Apesar do projeto ser aberto, a marca Arduino é registrada
- Briga judicial pelos direitos de uso da marca
  - Entre startup que projetava placas e a IDE (Arduino LLC) e startup que fabricava e vendia as placas (Smart Projects)
  - Mais sobre o caso na versão do Massimo (LLC):

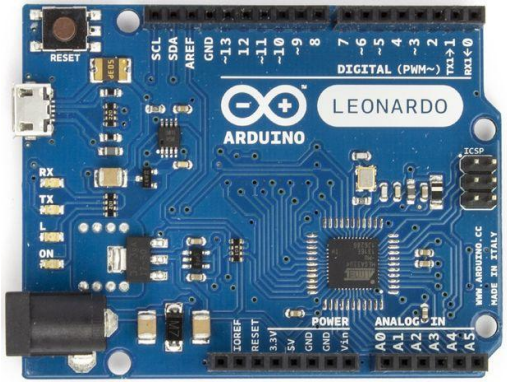
http://makezine.com/2015/03/19/massimo-banzi-fighting-for-arduino/

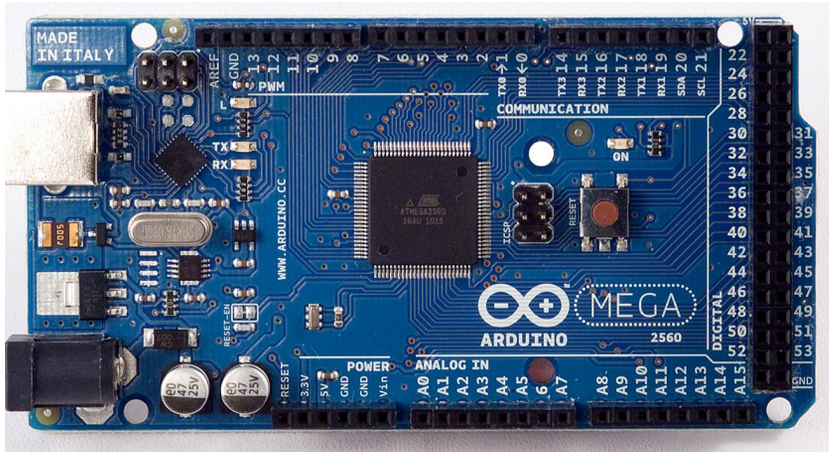
Arduino - Hardware - versões mais antigas



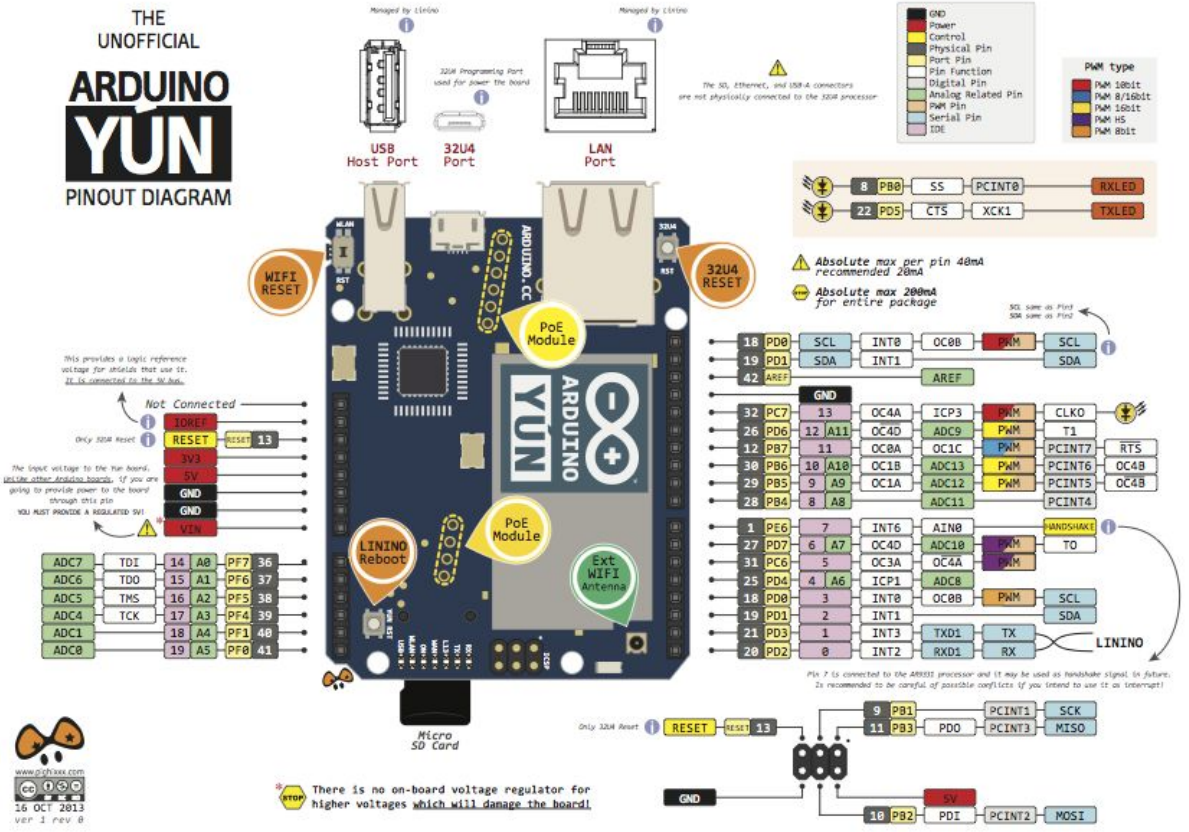








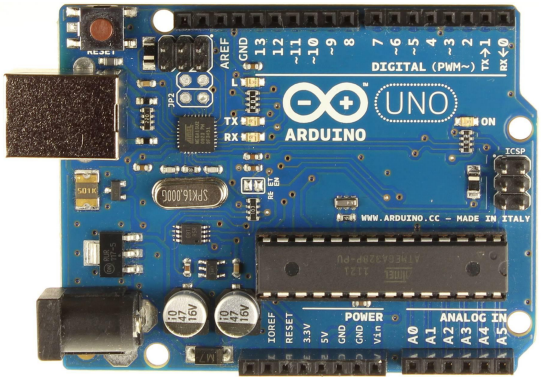
Arduino - Hardware - muitas novas versões...



Arduino - Hardware - versões alternativas

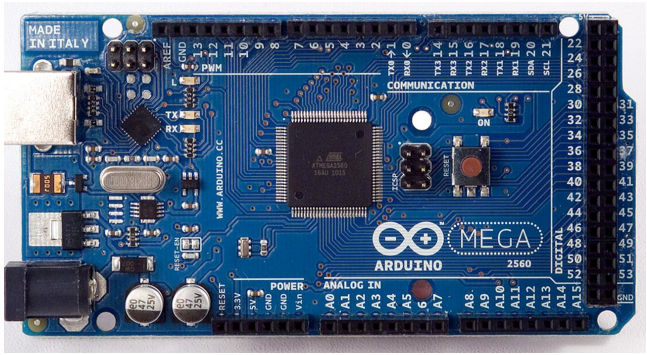
- Clones de arduino
  - Freeduino - http://www.freeduino.org/
  - Seeduino - http://www.seeedstudio.com/wiki/Seeeduino_v2.21
  - Brasuino - http://brasuino.holoscopio.com/
- Similares
  - ChipKIT - http://chipkit.net/
  - Olimexino - https://www.olimex.com/Products/Duino/STM32/OLIMEXINO-STM32/
  - Texas 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

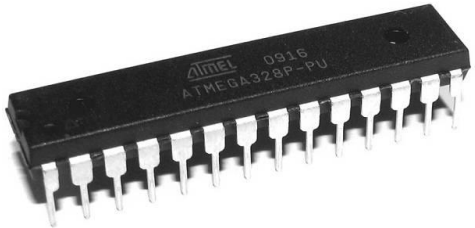
Arduino - Hardware - Detalhes do Mega2560



- 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
- Microprocessador de pequeno porte, capaz de executar um pequeno conjunto de instruções
  - Instruções simples e rápidas
  - Possui memória(s)
  - Possui periféricos
  - Pode se comunicar com outros periféricos

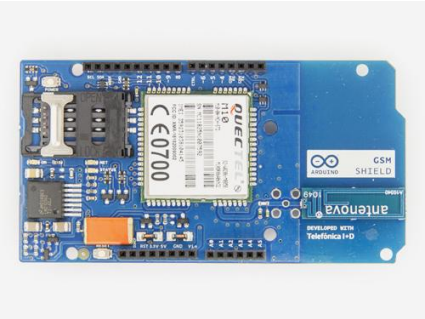


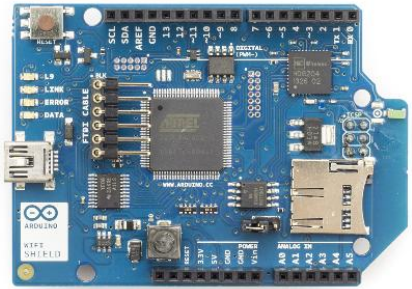
- Arduino - Família MegaAVR (ATMEL)
  - http://www.atmel.com/pt/br/products/microcontrollers/avr/megaAVR.aspx
  - Existem outras famílias produzidas por outros fabricantes (Intel, Motorola, Texas Instruments, Microchip, etc)

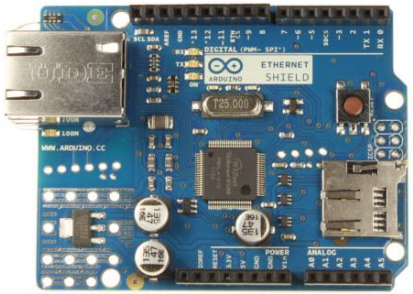
Arduino - Hardware - Shields

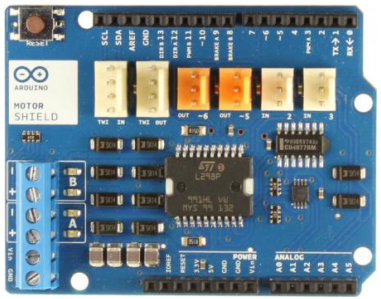
Site com uma lista de shields:
http://shieldlist.org

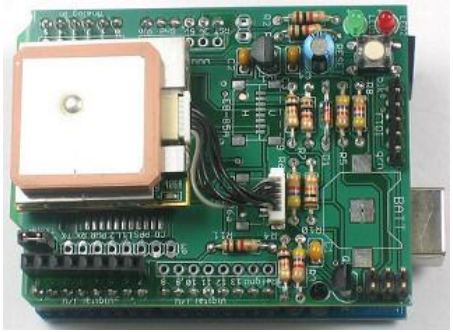
- Shields → placas de extensão para o arduino
  - Permitem adicionar funcionalidades nas placas convencionais
- Exemplos:
  - WiFi shield
  - Ethernet shield
  - GSM shield
  - Motor shield
  - GPS shield





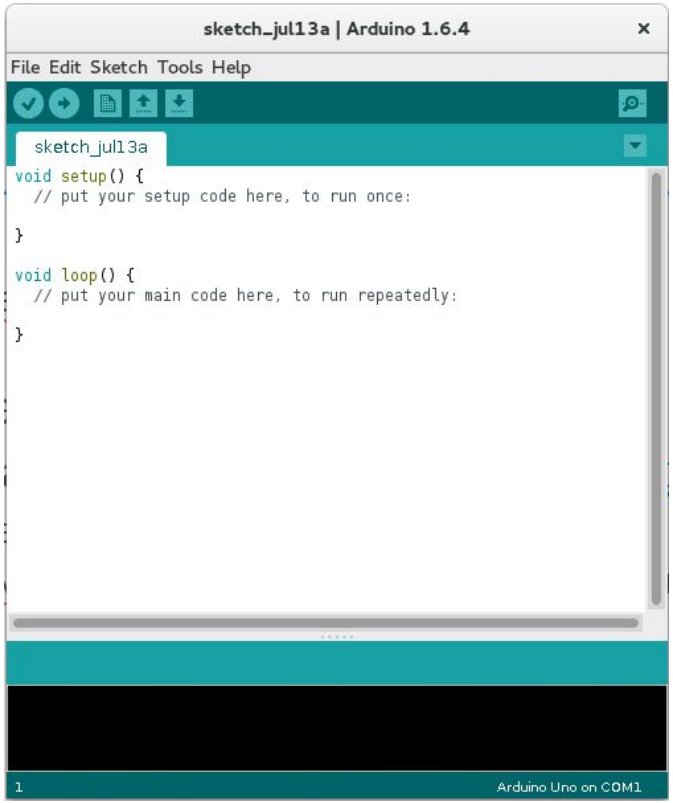






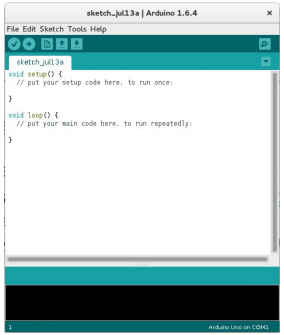
Arduino - Software

- Microcontrolador executa instruções armazenadas em sua memória
  - Linguagem AVR (um tipo de assemble)
  - Difícil programar na linguagem nativa!
- Programação facilitada através de IDE open source disponibilizada pelos desenvolvedores do arduino → baseado no framework Wiring (http://wiring.org.co/)
  - Programação em linguagem de 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

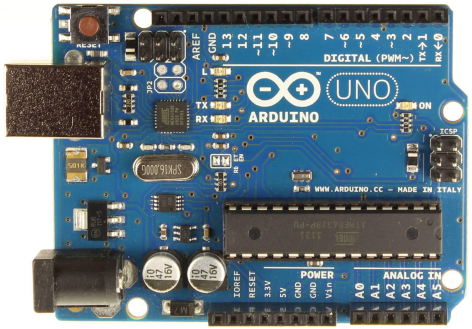
IDE



compilador

arquivo.hex

programador



Arduino - Software - IDE

- Já disponível como pacote em algumas distribuições, mas pode ser executada sem a necessidade de instalação
  - Download: https://www.dcc.ufrrj.br/~marcel/arduino-seccim/arduino-1.6.12-linux64.tar.xz
- Vejamos as funcionalidades na prática...

Arduino - Software - Exemplo1 - “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 - Exemplo1 - “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 - Exemplo1 - “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 - Exemplo1 - “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 - Exemplo1 - “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 - Exemplo1 - “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 - Exemplo1 - “hello LED”

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

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

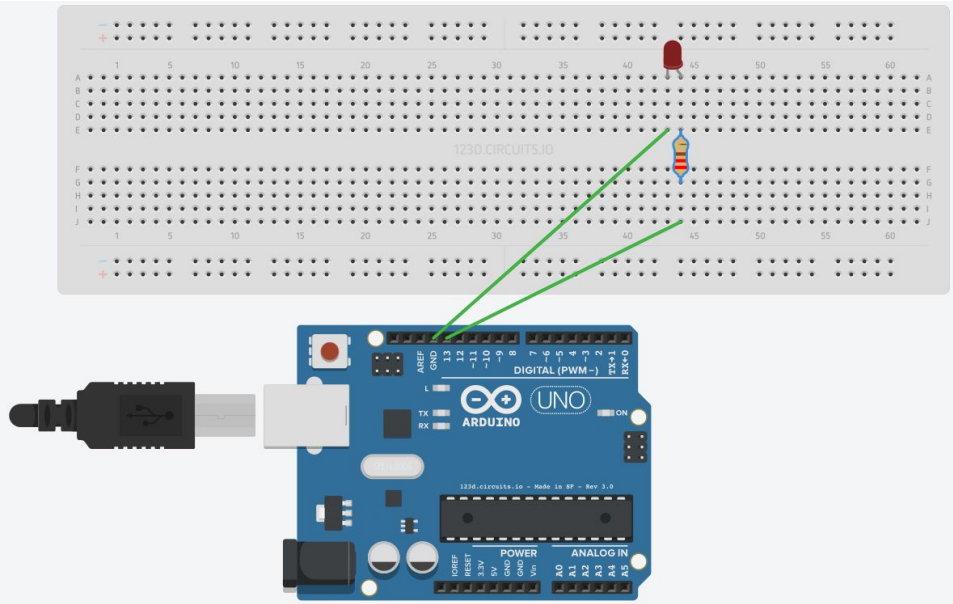
delay() → espera por um inteiro em milisegundos

```
// 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 - Exemplo1 - “hello LED” - simulação

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



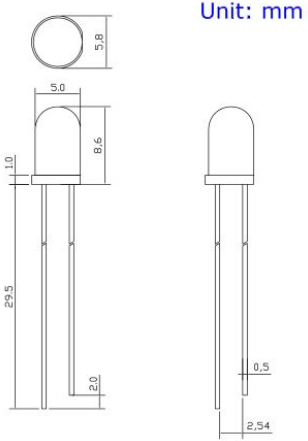
https://circuits.io/circuits/2436819-hello-led/embed

Arduino - Exemplo1 - “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
  - se corrente muito alta, o LED pode queimar
  - mas qual o limite?

Arduino - Exemplo1 - “hello LED” - detalhes

- Todo componente eletrônico possui uma documentação que o descreve → **data sheet** (folha de dados)
  - 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)

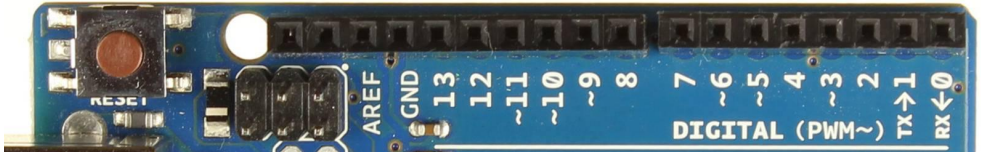


ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage (V_R=5V)	I_R	10	uA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	℃
Storage Temperature	T_{STG}	-40 ~ 100	℃
Lead Soldering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

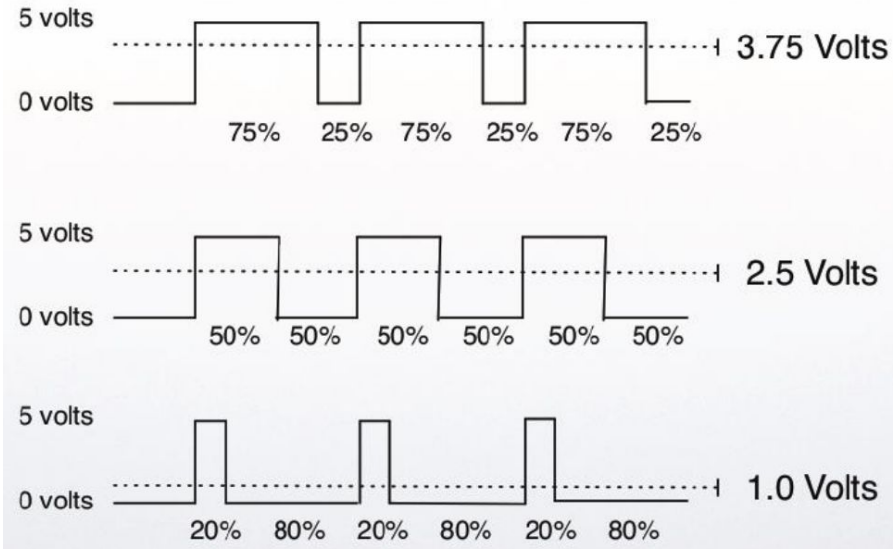
Arduino - Exemplo1 - “hello LED” - detalhes

- 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
    - Muito importante para evitar danos aos equipamentos

Exemplo2 - LED pulse



- Como piscar o LED do forma ‘suave’?
- Usar escrita analógica, pinos PWM (~)
- Pulse Width Modulation
  - Emulam uma tensão analógica
  - Através do ‘duty cycle’ - tensão média
- analogWrite(pino,valor)
  - valor entre 0 e 255 define a razão entre os tempos em 0V e 5V
- Tarefa:
  - Modificar o programa de piscar o LED para pulsar

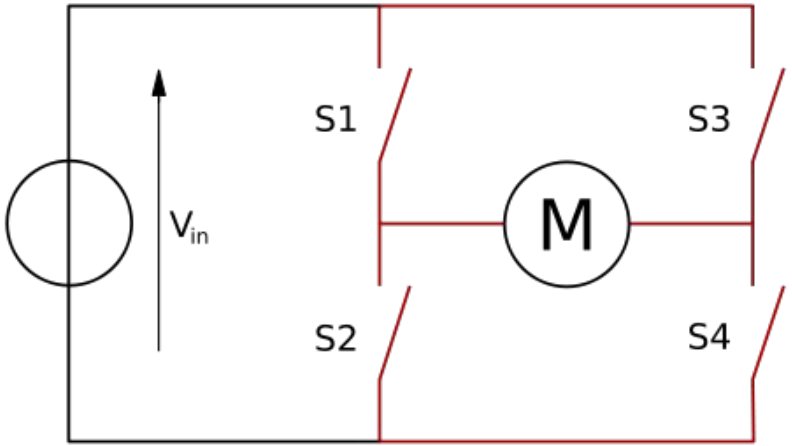


Exemplo3 - Sensor ultrassônico

- HC-SR04
- Emite um sinal ultrassônico e capta o retorno
- Intervalo entre os sinais (emitido e recebido) determina a distância para o obstáculo
- Exemplo: Medidor de distância
  - http://blog.filipeflop.com/sensores/sensor-ultrassonico-hc-sr04-ao-arduino.html

Exemplo4 - Motor elétrico DC

- Motor elétrico ativado com corrente contínua
- Funciona através do efeito eletromagnético em bobinas
- Para controlar o sentido do giro do motor, deve-se usar um circuito “ponte h”
  - Escolha das chaves determina sentido do motor



Exemplo4 - Motor elétrico DC

- Shield de motor possui dois CIs que implementam duas pont h cada
  - Adafruit Motor shield v1
  - Controla até 4 motores DC por vez
  - Biblioteca associada disponível no repositório
- Exemplo:
  - http://blog.filipeflop.com/motores-e-servos/controle-motor-dc-arduino-motor-shield.html
- Tarefa:
  - Movimentar o carrinho

Tarefa final - “sensor de estacionamento”

- Integrar LED + sensor ultrassônico + motores
- Carrinho deve estacionar ao encontrar um obstáculo
  - Piscar o LED conforme distância
  - Reduzir velocidade dos motores quando perto do obstáculo