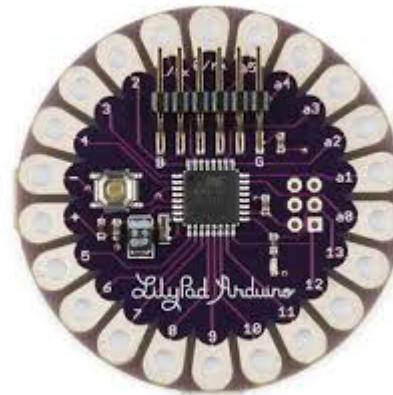
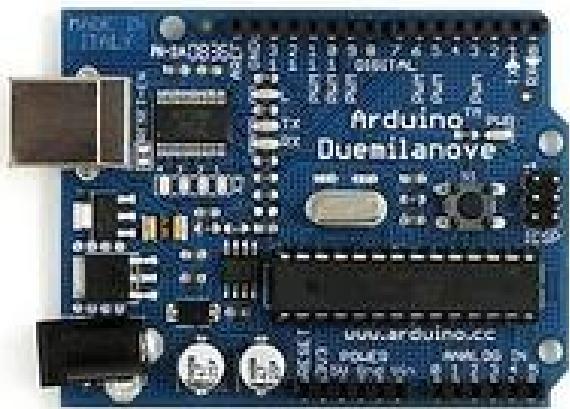
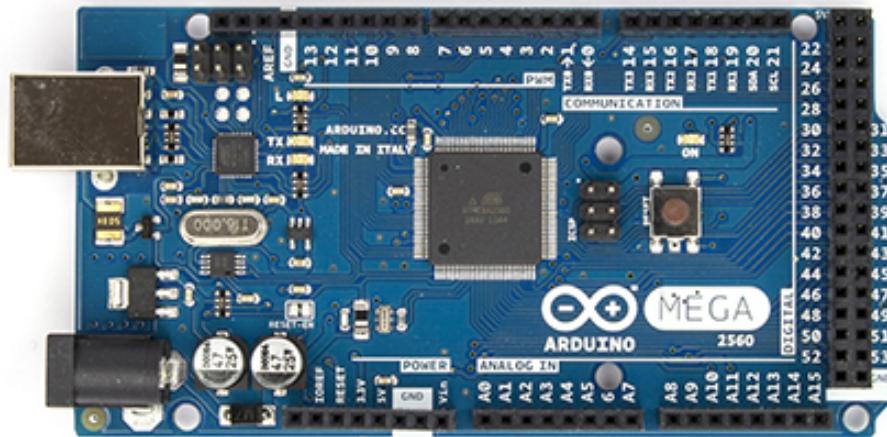


# Arduino



# Introdução a Arduino



- Arduino é um nome engraçado de um projeto famoso mundialmente;
- É uma plataforma gratuita e aberta para criação de placas e invenções eletrônicas;
- Foi projetada para artistas, hobistas, makers / inventores e pessoas que não são da área!
- Open-source hardware e software
- Oferece um IDE e bibliotecas de programação de alto nível;

# Diversas aplicações...



- Robótica;
- Domótica;
- Máquinas CNC;
- Controle de motores em geral;
- Roupas eletrônicas;
- Hacking (do bem!);
- Invenções diversas;

# Histórico do projeto Arduino



- Projeto criado na Itália pelo Mássimo Banzi no Interaction Design Institute Ivrea;
- Nasceu para complementar o aprendizado de programação, computação física e gráfica;
- Nasceu do Processing e Wiring;
- Processing é um ambiente e linguagem de programação para criar imagens, animação e interação;

# Terminologia



- **Microcontrolador:** um microcomputador compacto completo resumido em um circuito. Inclui processador, memórias e I/Os.
- **Firmware:** pedaço de software instalado em um microcontrolador.
- **Bootloader:** pedaço de software instalado na memória de um microcontrolador que permite a carga de firmwares por serial, USB ou outros.

# Arduino: parte técnica



- Plataforma baseada nos **microcontroladores** de 8 bits da Atmel da AVR (ATMegaX68);
- Microcontrolador com **bootloader**
- Transferência de **firmware** via USB
- Programado em C/C++
- Atmega328 é o mais popular microcontrolador:
  - Atmega8
  - Atmega168
  - **Atmega328**
  - Atmega1280
  - **Atmega2560**

# Atmega328



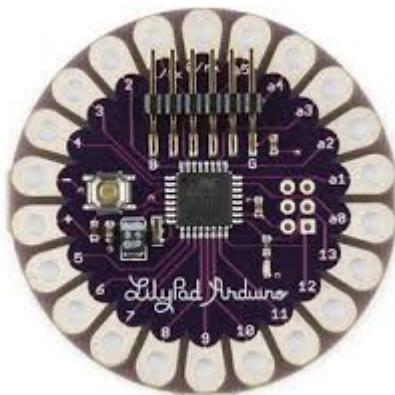
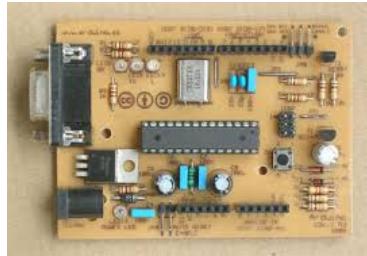
- RISC, 16mhz, 20 MIPS
- 32k memória flash
- 2k RAM Estática
- 1K EEPROM
- 10.000 ciclos na Flash e 100.000 na EEPROM
- 2 interrupções
- 2 contadores / temporizadores de 8bits
- 1 contador / temporizador de 16bits
- 1 temporizador de tempo real com clock a parte
- 20 portas de I/O

# Atmega328

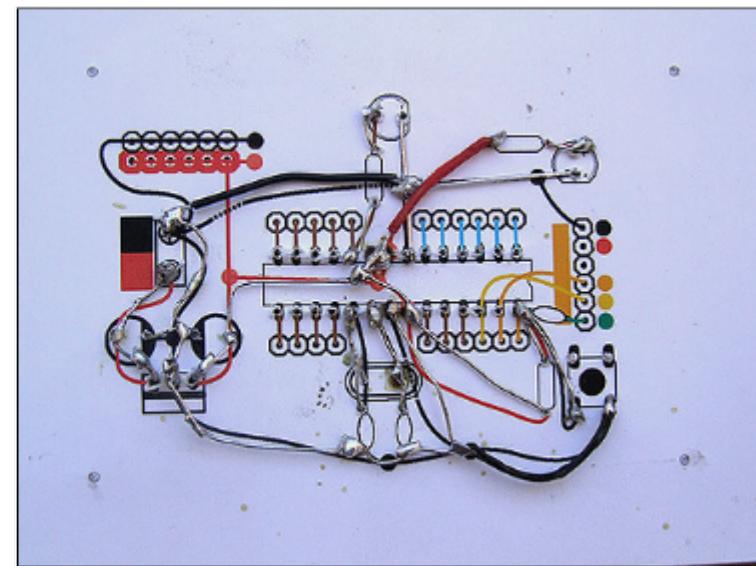
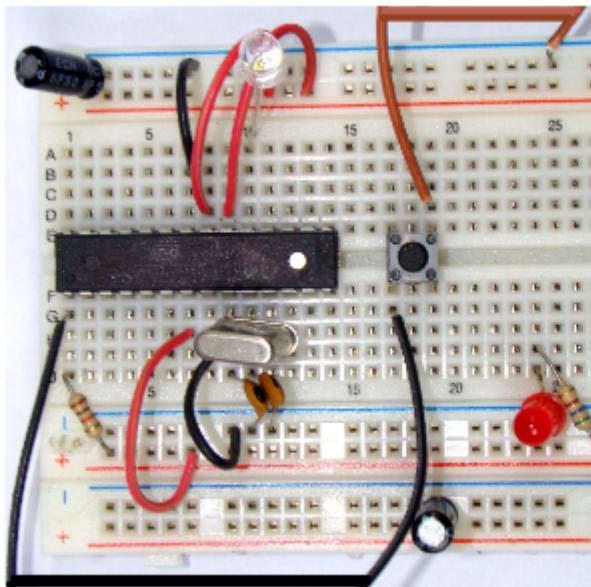


- 14 portas digitais
- 6 canais PWM
- 6 conversores analógico/digital de 10 bits
- 1 serial programável (USART)
- 1 interface SPI (Serial Peripheral Interface)
- 1 interface serial a 2 fios (I2C)
- 1 watch dog timer programável
- 1 comparador analógico no chip
- Interrupção ou wake-up na alteração de estado dos pinos

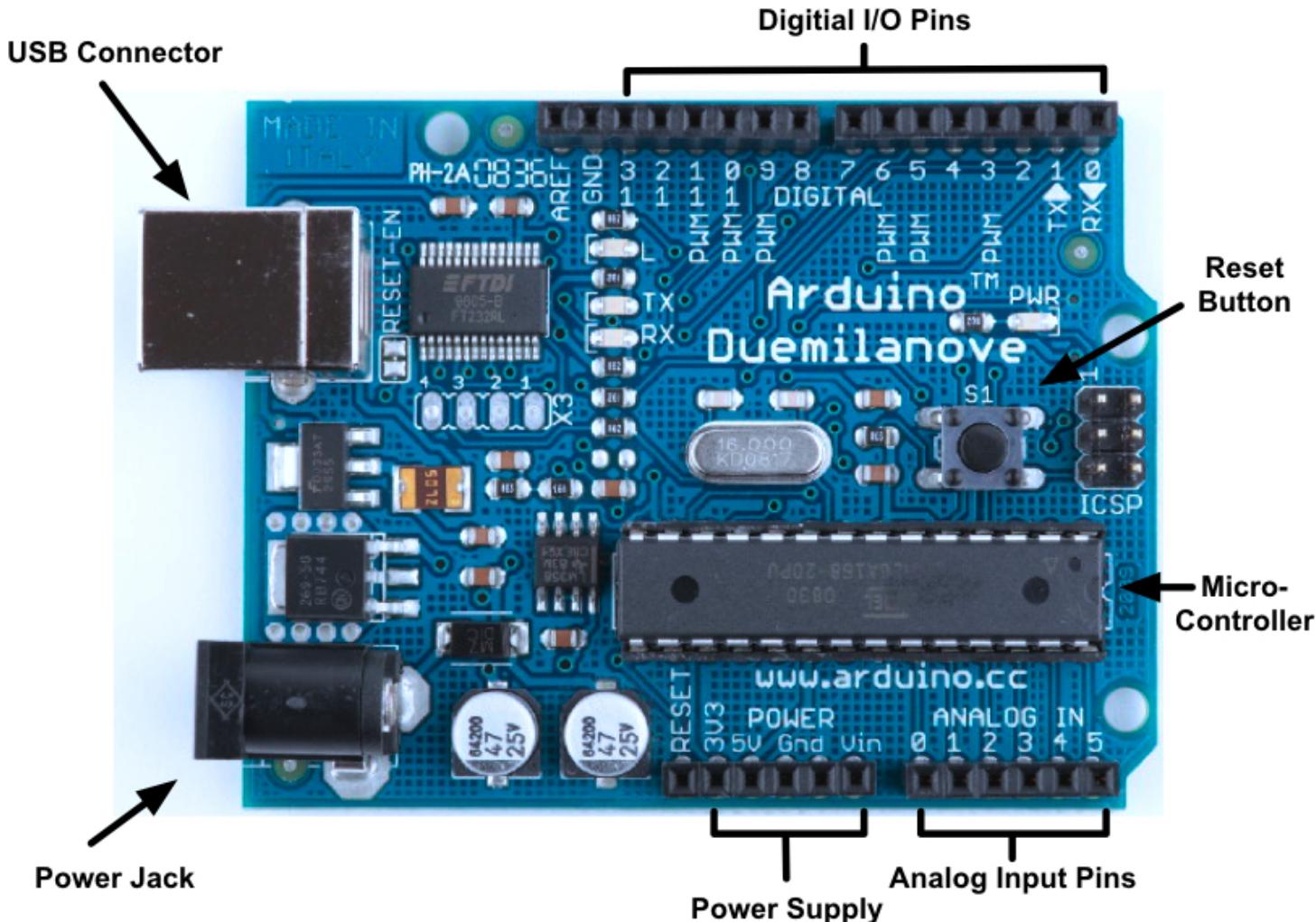
# Diversos tipos de Arduino



# Ou então simplesmente...



# Principais Partes



# Parte Frontal

FT232RL

Conversor USB-Serial

Conector USB

Regular 7805:

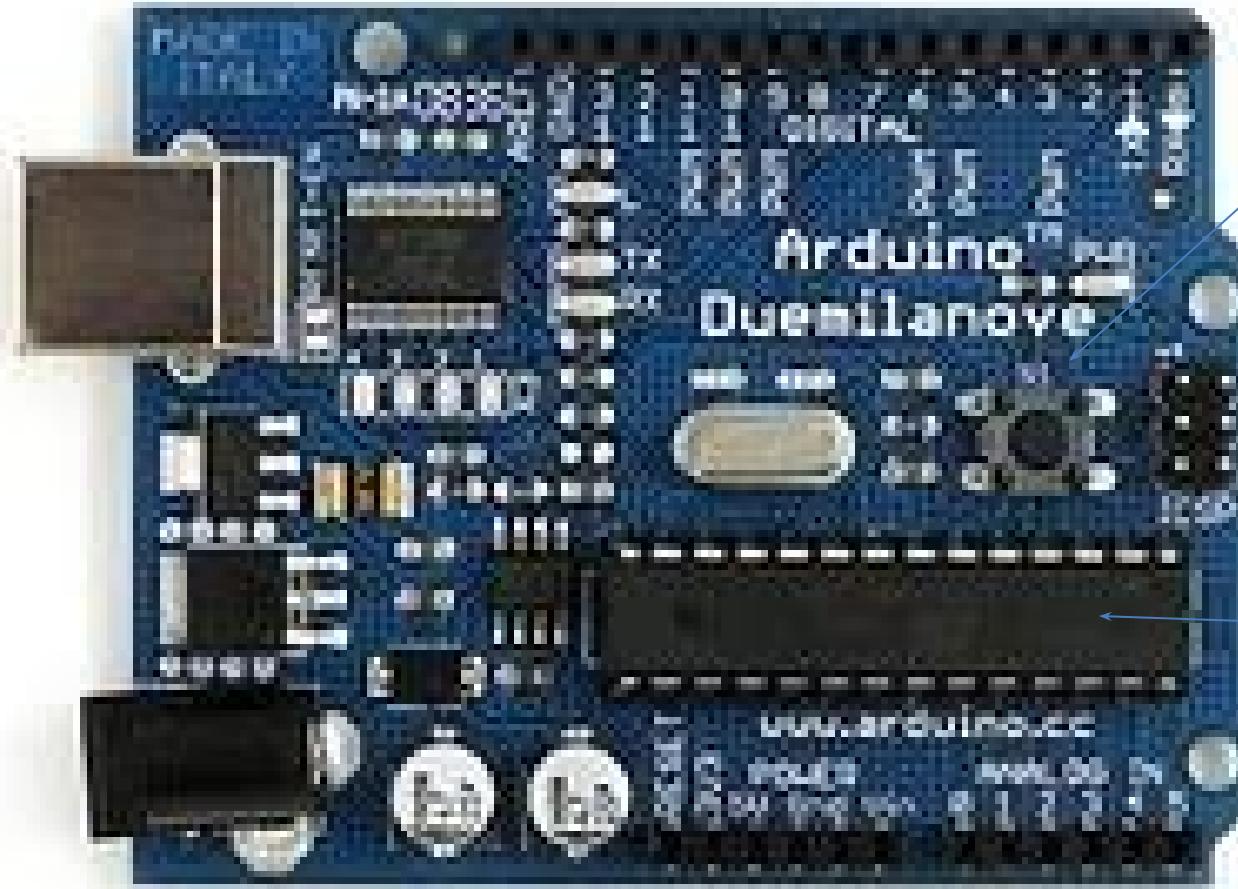
Recebe até 12 volts e  
regula para 5 volts

Alimentação externa:

Até 12 volts



# Núcleo



Botão de reset

**ICSP**

Para gravar bootloader  
ou programas/firmware

AtMega328 /168/8

# Portas Digitais

**AREF**

Referência analógica

Padrão 5 volts

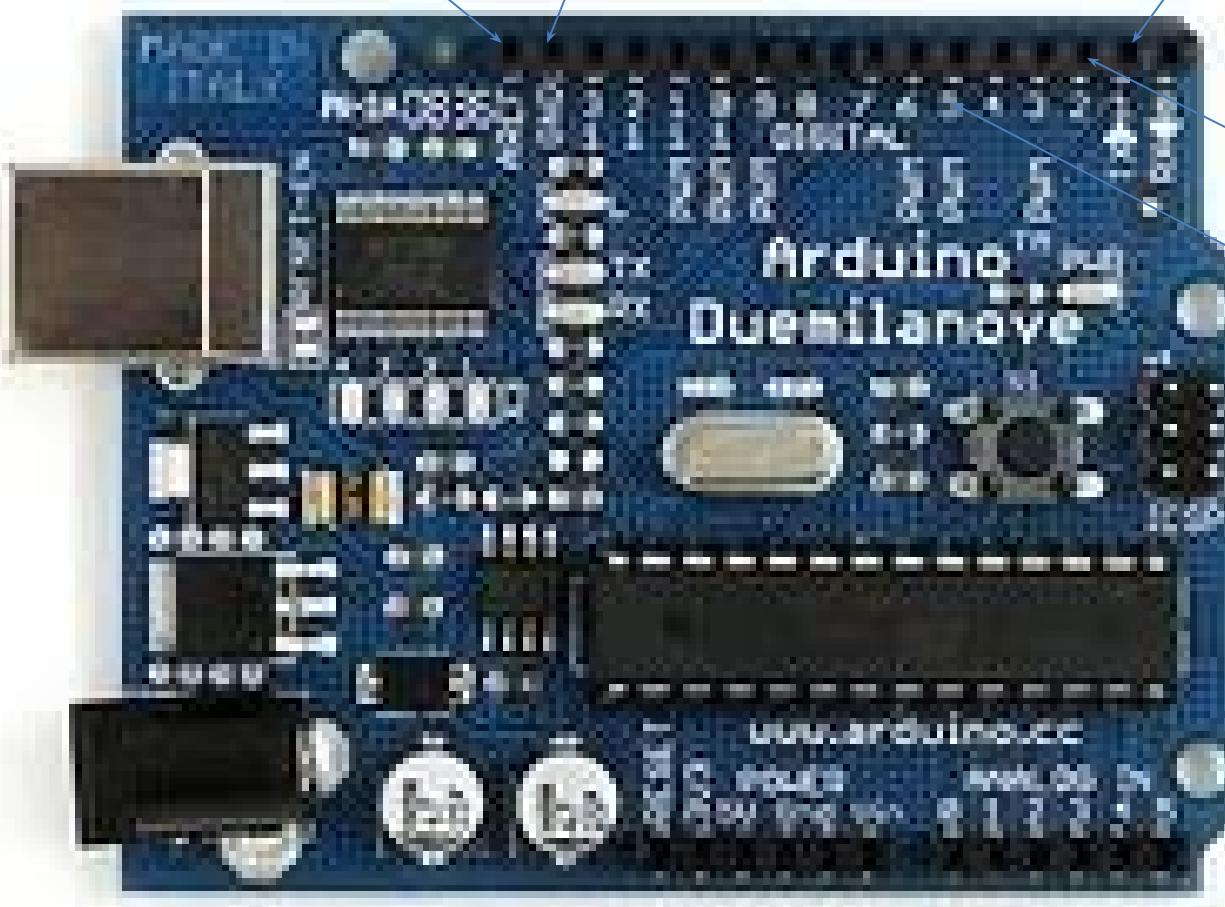
**GND**

**Portas digitais 0 a 13**

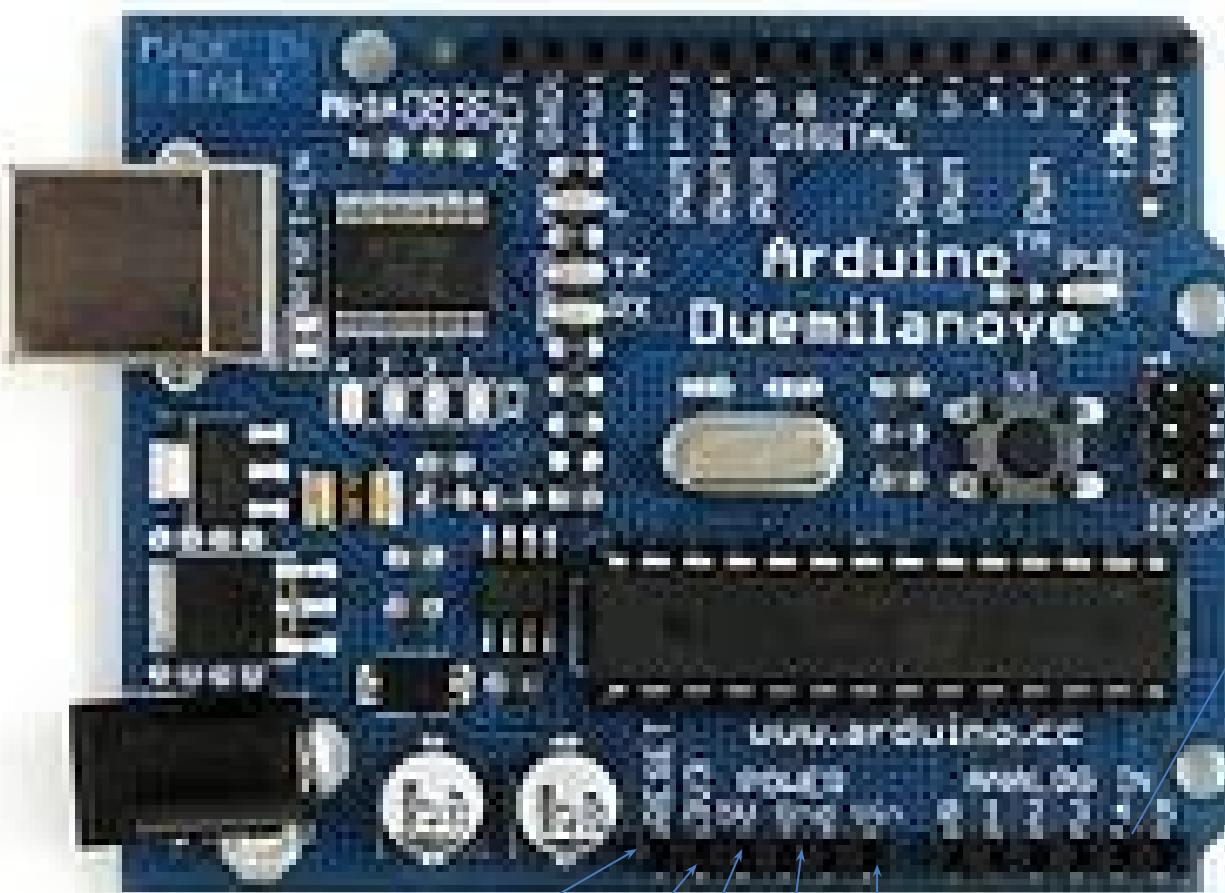
**0 RX 1 TX** = usada durante transferência de sketch e comunicação serial com placa

**2,4,7,8,12,13** = portas digitais convencionais

**3,5,6,9,10,11** = portas PWM



# Portas Analógicas



Reset

3.3 volts

5 volts

VIN

Alimentação de entrada sem regulagem

Portas analógicas de 0 a 5

*Podem funcionar como digitais de 14 a 19*

# Porta Digital Vs. Analógica



- **Digital:** trabalha com 0 e 1 na lógica binária.
- Existem padrões para conversão da tensão para zeros e uns:
  - **TTL:** 0 - 5 volts
  - **CMOS:** 0 - 3.3 volts
  - **Low-voltage CMOS:** 0 - 1.8 volts
- **Digital do Arduino** comum segue **TTL**:
  - 0 a 0,8 volts = 0
  - 2 a 5 volts = 1

# Porta Digital Vs. Analógica

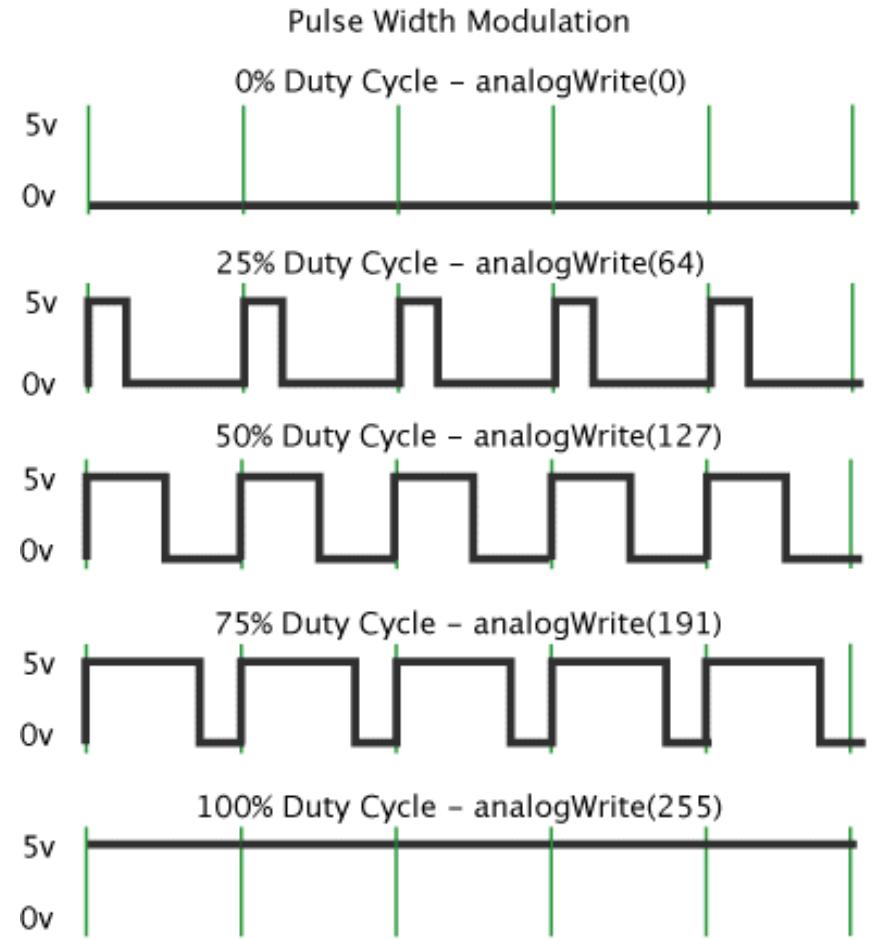


- **Analógica:** valor lido é análogo a tensão.
- **Referência de analogia** é 5 volts
  - 0 volts = 0
  - 2.5 volts= 512
  - 5 volts = 1023
- Atmega contém **Conversor A/D de 10 bits:** 0 a 1023

# Porta PWM



**Uma porta híbrida:**  
digital porém com  
modularização de  
zeros e uns de forma  
que consegue se  
expressar pulsando por  
fatias de tempo;



# Por dentro do MCU



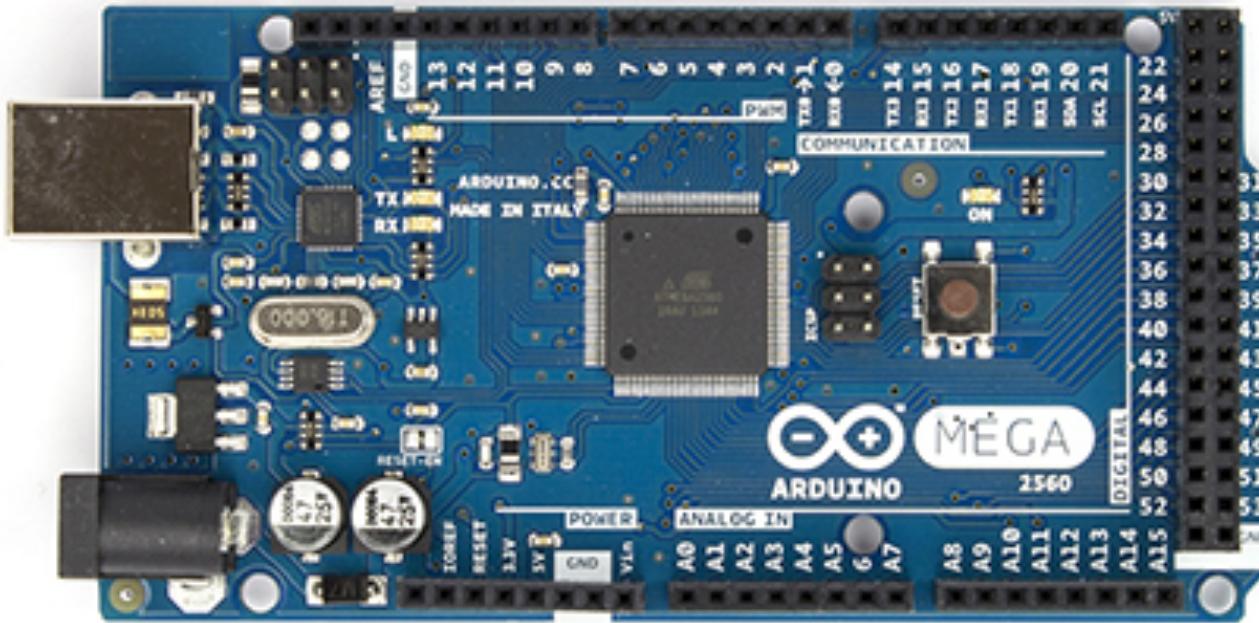
## Atmega168 Pin Mapping

### Arduino function

|                     |                          |    |    |                        |                      |
|---------------------|--------------------------|----|----|------------------------|----------------------|
| reset               | (PCINT14/RESET) PC6      | 1  | 28 | PC5 (ADC5/SCL/PCINT13) | analog input 5       |
| digital pin 0 (RX)  | (PCINT16/RXD) PD0        | 2  | 27 | PC4 (ADC4/SDA/PCINT12) | analog input 4       |
| digital pin 1 (TX)  | (PCINT17/TXD) PD1        | 3  | 26 | PC3 (ADC3/PCINT11)     | analog input 3       |
| digital pin 2       | (PCINT18/INT0) PD2       | 4  | 25 | PC2 (ADC2/PCINT10)     | analog input 2       |
| digital pin 3 (PWM) | (PCINT19/OC2B/INT1) PD3  | 5  | 24 | PC1 (ADC1/PCINT9)      | analog input 1       |
| digital pin 4       | (PCINT20/XCK/T0) PD4     | 6  | 23 | PC0 (ADC0/PCINT8)      | analog input 0       |
| VCC                 | VCC                      | 7  | 22 | GND                    | GND                  |
| GND                 | GND                      | 8  | 21 | AREF                   | analog reference     |
| crystal             | (PCINT6/XTAL1/TOSC1) PB6 | 9  | 20 | AVCC                   | VCC                  |
| crystal             | (PCINT7/XTAL2/TOSC2) PB7 | 10 | 19 | PB5 (SCK/PCINT5)       | digital pin 13       |
| digital pin 5 (PWM) | (PCINT21/OC0B/T1) PD5    | 11 | 18 | PB4 (MISO/PCINT4)      | digital pin 12       |
| digital pin 6 (PWM) | (PCINT22/OC0A/AIN0) PD6  | 12 | 17 | PB3 (MOSI/OC2A/PCINT3) | digital pin 11(PWM)  |
| digital pin 7       | (PCINT23/AIN1) PD7       | 13 | 16 | PB2 (SS/OC1B/PCINT2)   | digital pin 10 (PWM) |
| digital pin 8       | (PCINT0/CLKO/ICP1) PB0   | 14 | 15 | PB1 (OC1A/PCINT1)      | digital pin 9 (PWM)  |

Digital Pins 11,12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

# Arduino Uno Mega 2560

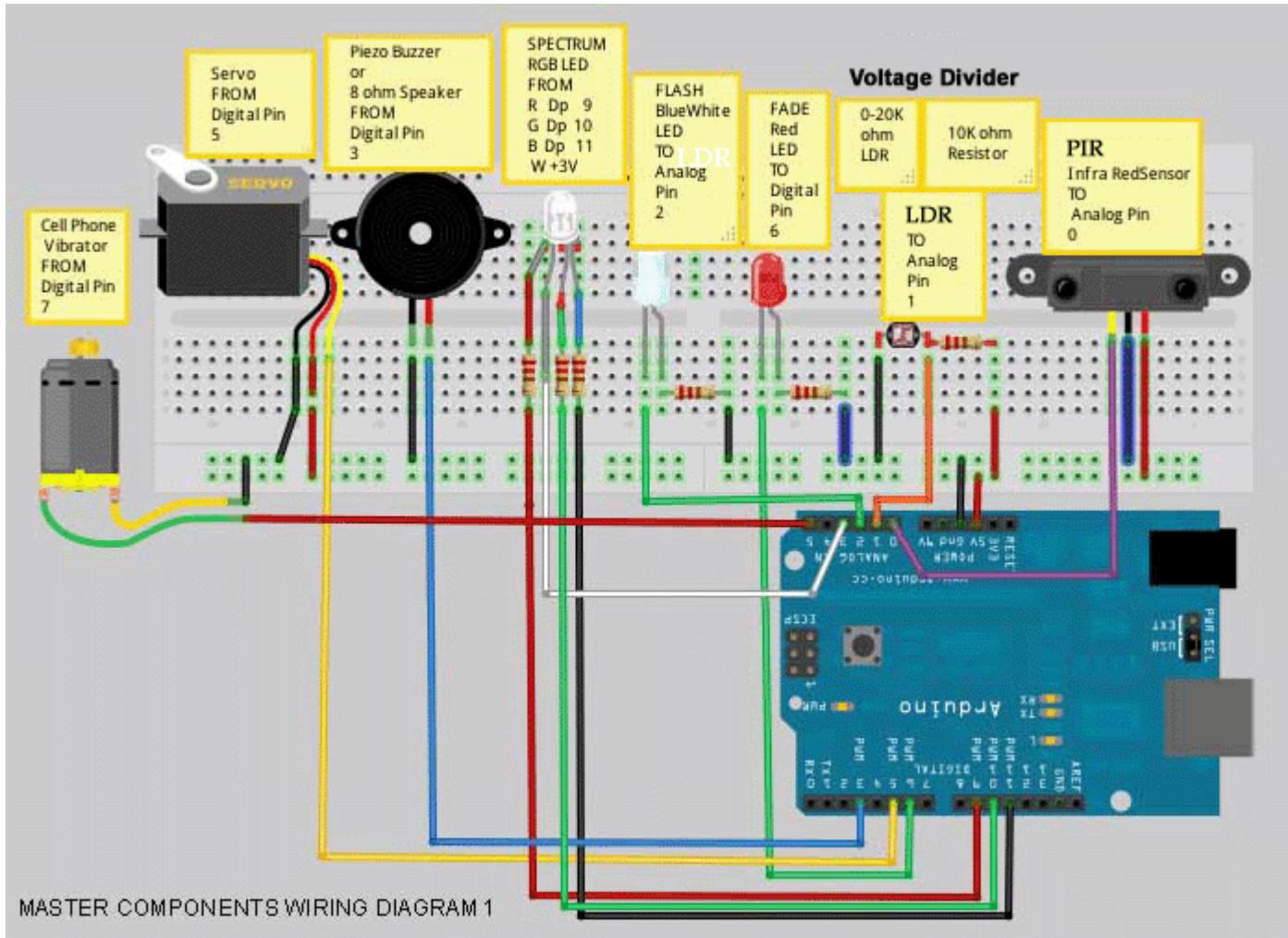


# Arduino Uno Mega 2560



- 54 portas
- 15 canais PWM
- 16 analógicas
- 4 seriais programáveis (USART)
- 6 timers
- 8kb RAM
- 4kb EEPROM
- 256kb memória flash (programas)
- Os mesmos 16mhz...

# Componentes + Portas



# Shields: arquitetura modular



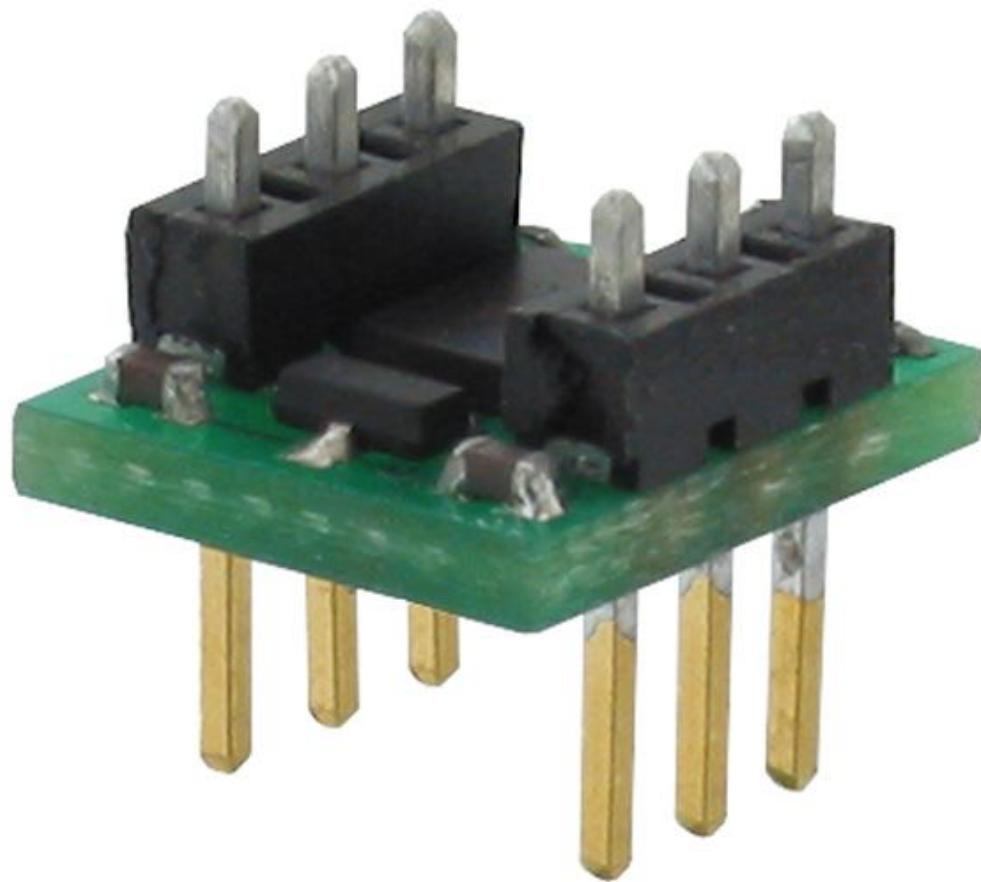
- Arduino estabeleceu um padrão de pinagem que é respeitado por diversas placas shield:



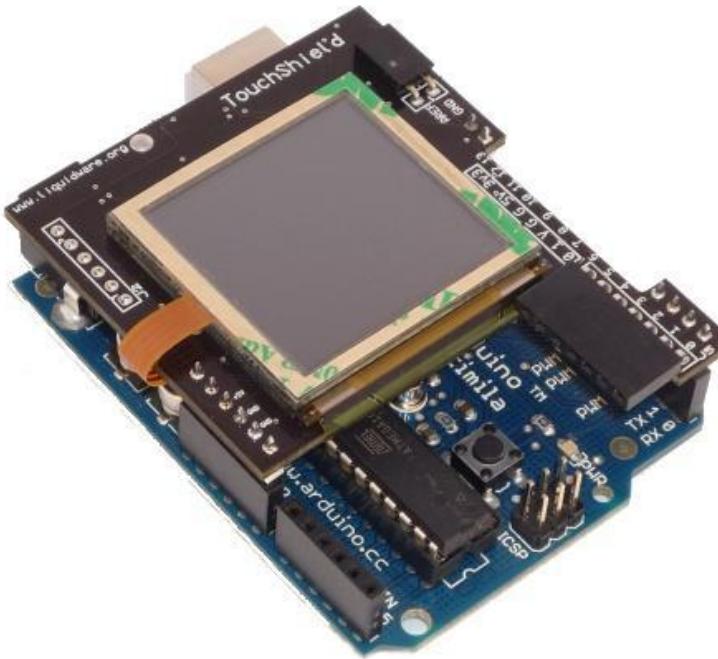
# Ping – Sensor de distância



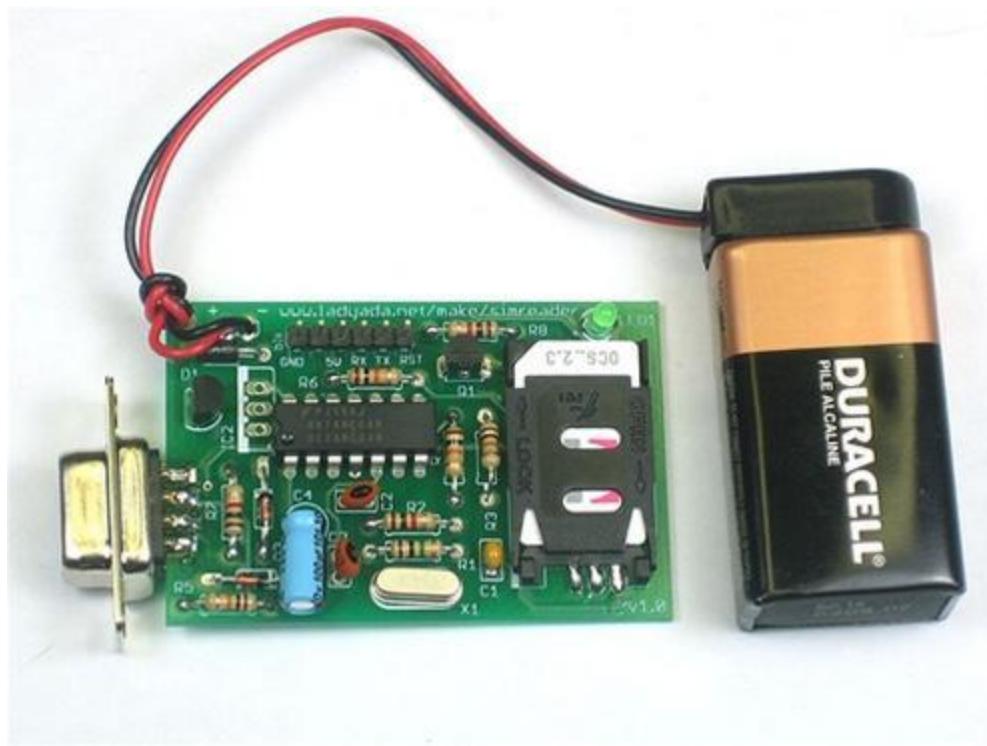
# Bússola



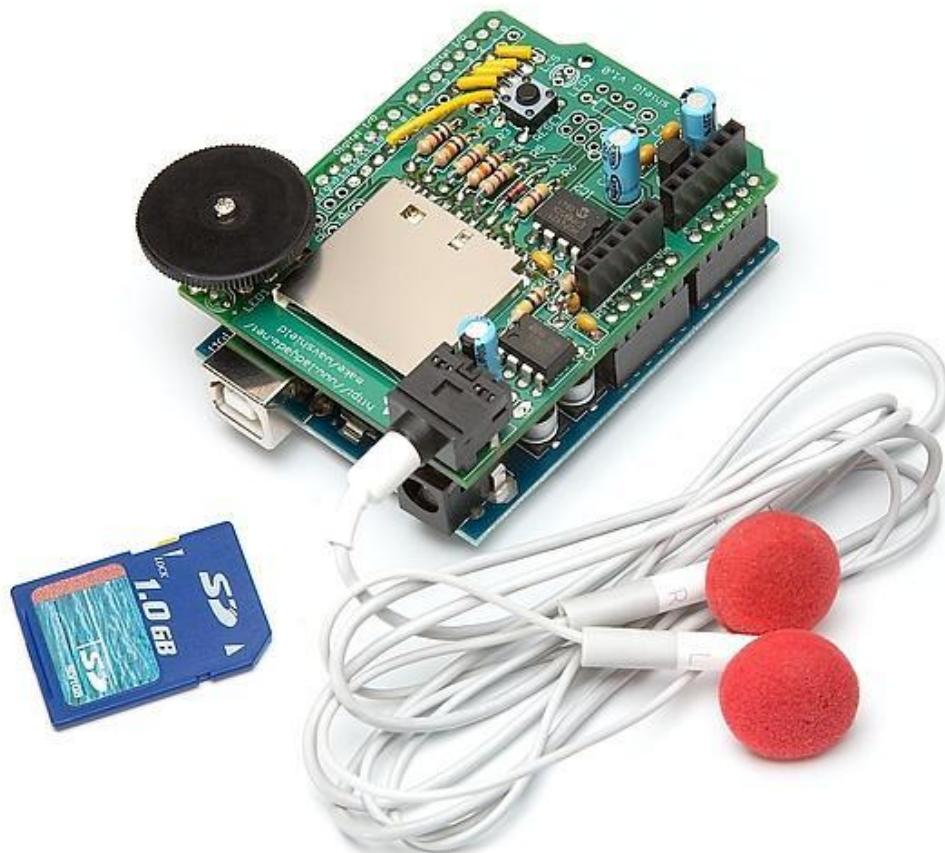
# Shield LCD



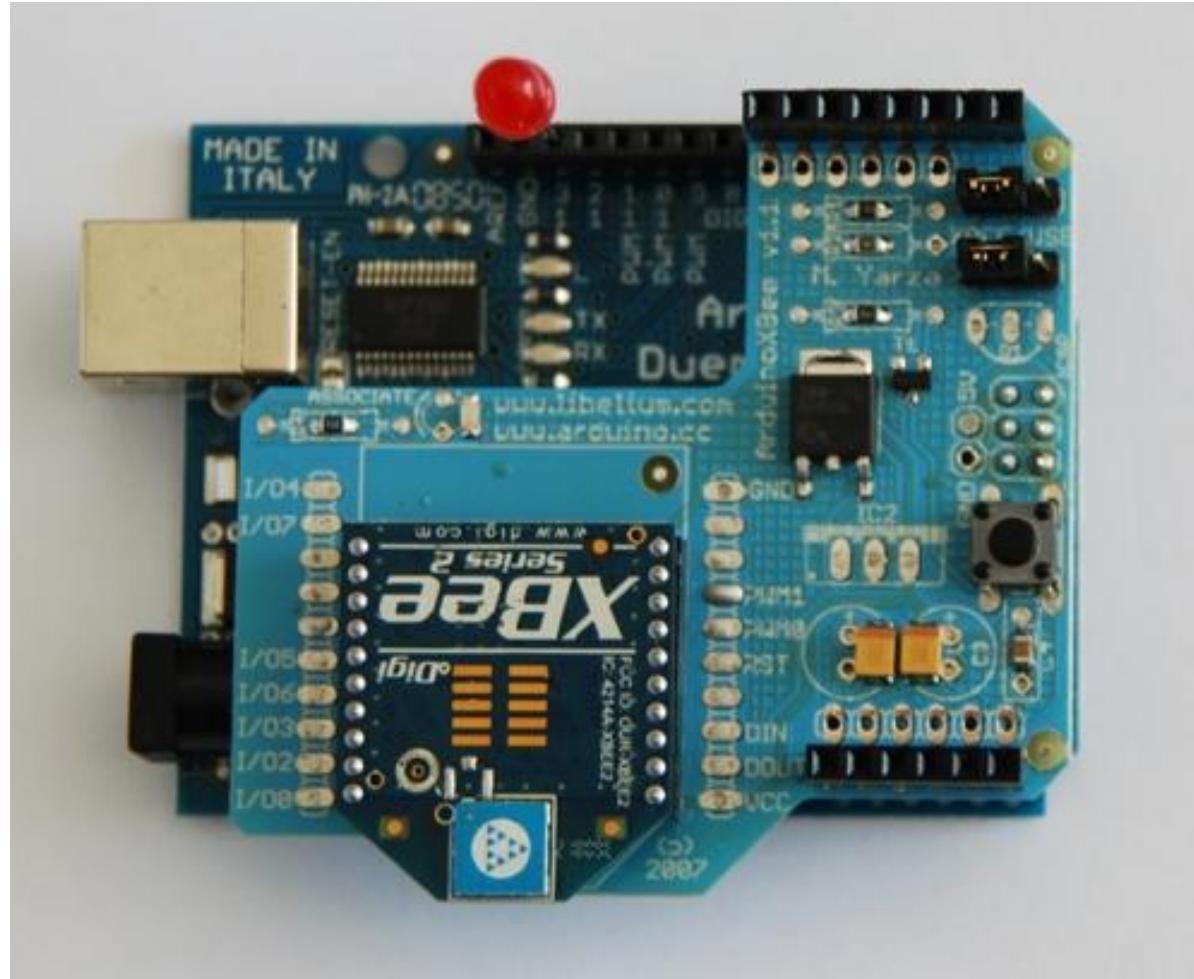
# SimReader



# Mp3 Player Shield



# ZigBee - xBee



# Arduino



Além do padrão  
de placas, oferece  
Gratuitamente  
uma ferramenta  
para  
desenvolvimento:

The screenshot shows the Arduino IDE interface with the title bar "Boat\_Central | Arduino 0021". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar below has icons for play, stop, and other functions. The sketch tab "Boat\_Central" is selected. The code editor contains the following C++ code:

```
#include <LiquidCrystal.h>
#include <MeetAndroid.h>

#define MODE_GPS_LATLONG 0
#define MODE_GPS_SPEED 1
#define MODE_GPS_HEADING 2
#define MODE_GPS_MANE 3
#define MODE_AUTOPILOT_COMPASS 4
#define MODE_AUTOPILOT_ACCELEROMETER 5
#define MODE_DEBUG 6
#define MODE_MANUALPILOT 7
#define MOTOR_A 12
#define MOTOR_B 13
```

# Arduino



- Utilizaremos para escrever todos os nossos programas:

```
void setup() {  
  
    pinMode(13, OUTPUT); //porta 13 em  
    //output  
  
}  
  
void loop() {  
  
    digitalWrite(13, HIGH); //HIGH = 1 =  
    //TRUE  
  
    delay(500);  
  
    digitalWrite(13, LOW); //LOW = 0 =  
    //FALSE  
  
    delay(500);  
  
}
```

The screenshot shows the Arduino IDE interface with the title bar "Boat\_Central | Arduino 0021". The menu bar includes File, Edit, Sketch, Tools, Help, and a tab bar with "Boat\_Central", "EletronLCD", "EletronOS", and "ProgramME". The code editor contains the following C++ code:

```
#include <LiquidCrystal.h>  
#include <MeetAndroid.h>  
  
#define MODE_GPS_LATLONG 0  
#define MODE_GPS_SPEED 1  
#define MODE_GPS_HEADING 2  
#define MODE_GPS_MANE 3  
#define MODE_AUTOPILOT_COMPASS 4  
#define MODE_AUTOPILOT_ACCELEROMETER 5  
#define MODE_DEBUG 6  
#define MODE_MANUALPILOT 7  
#define MOTOR_A 12  
#define MOTOR_B 13
```



# Arduino



```
#include <LiquidCrystal.h>
#include <MeetAndroid.h>

#define MODE_GPS_LATLONG 0
#define MODE_GPS_SPEED 1
#define MODE_GPS_HEADING 2
#define MODE_GPS_MANE 3
#define MODE_AUTOPILOT_COMPASS 4
#define MODE_AUTOPILOT_ACCELEROMETER 5
#define MODE_DEBUG 6
#define MODE_MANUALPILOT 7
#define MOTOR_A 12
#define MOTOR_B 13
```

Botão para compilar

Botão para enviar  
programa para a placa

Área para digitação  
do código

Mensagens de  
sucesso ou erro

# Programando para Arduino



- A IDE foi desenvolvida com Java, portanto precisaremos de um máquina virtual instalada
- Funciona em Windows. Mac OS X e Linux
- (em alguns casos necessita driver)
- Utiliza GCC + GCC Avr para compilação  
*Podemos programar diretamente com GCC!*
- A transferência para a placa é feita via USB pelo IDE;  
*Podemos transferir com gravadores ICSP!*

# A linguagem C



- Linguagem amplamente utilizada para construção de softwares: Linux, Windows, Office e maior parte dos games são feitos em C;
- Programadores que conhecem C tendem a ter maior facilidade de entender outras linguagens;
- A linguagem C pode ficar BASTANTE complicada;
- Faremos um uso básico da linguagem e mesmo assim conseguiremos desenvolver soluções avançadas!

# Programa Arduino Básico



- Temos que obrigatoriamente programar dois métodos:

```
void setup() {  
}
```

```
void loop() {  
}
```

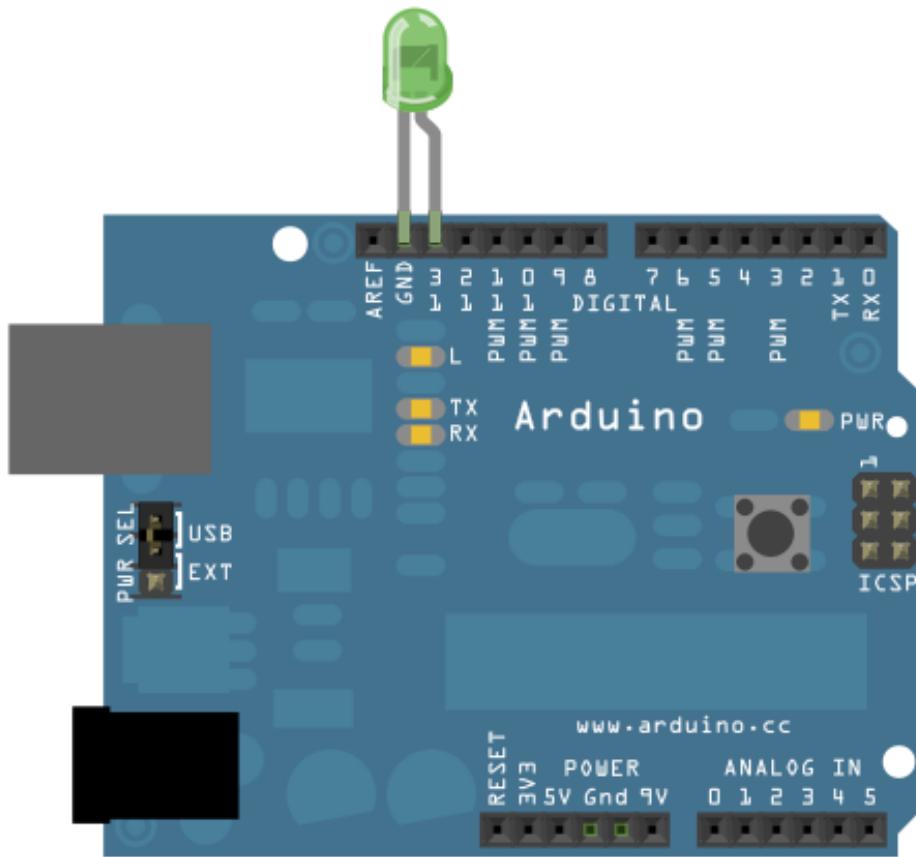
- O setup é executado uma só vez assim que a placa for ligada e o loop terá o código de execução infinita

# Portas digitais e analógicas



- Na prática ligamos componentes em portas digitais e analógicas e através do código Arduino, manipulamos as portas:
  - `pinMode(<porta>, <modo>)`: configura uma porta digital para ser lida ou para enviarmos dados;
  - `digitalWrite(<porta>, 0 ou 1)`: envia 0 ou 1 para porta digital
  - `digitalRead(<porta>)`: retorna um 0 ou 1 lido da porta
  - `analogRead(<porta>)`: retorna de 0 a 1023 com o valor da porta analógica
  - `analogWrite(<porta>, <valor>)`: escreve em uma porta PWM um valor de 0 a 255

# Exemplo “pisca led”



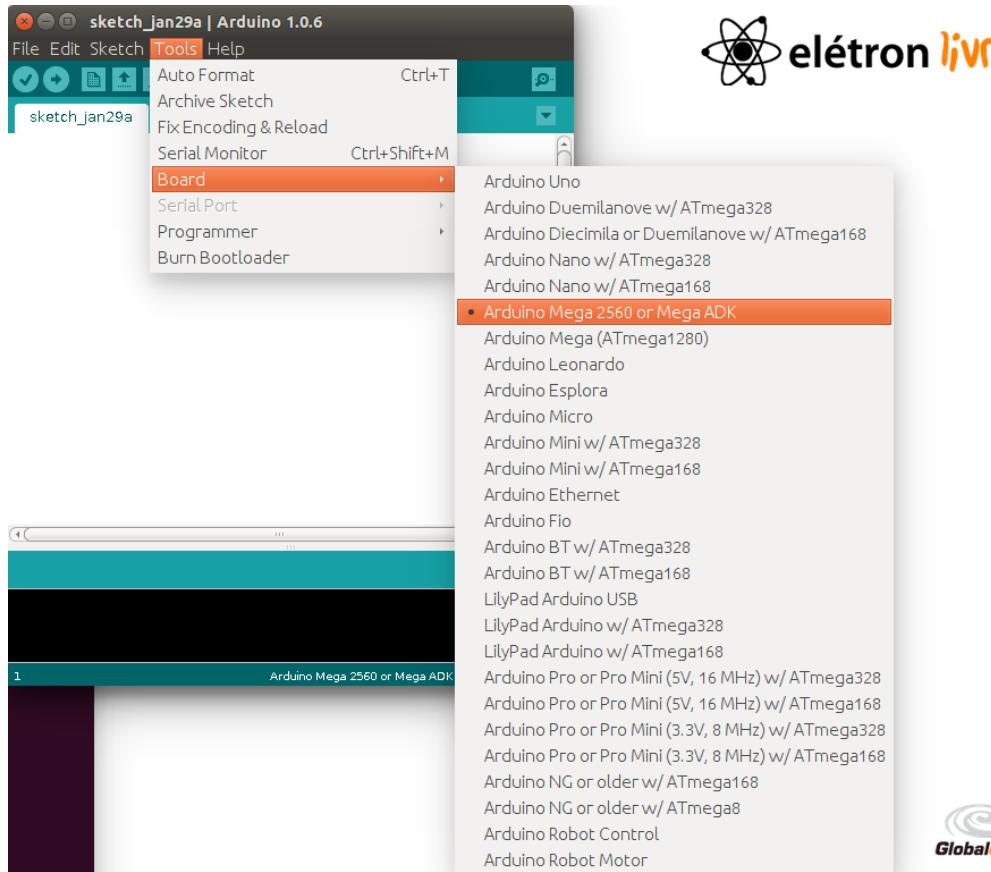
Esta conexão é bem simples somente para efeito de teste para piscar o led.

O correto é ligar um resistor usando uma protoboard.

# Laboratório



- Instale e execute o Arduino IDE (ferramenta)
- Selecione o tipo correto de placa:



# Digite o código:



```
void setup() {  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(500);  
    digitalWrite(13, LOW);  
    delay(500);  
}
```

# Conclusões



- Todo programa Arduino escrito em C é obrigatoriamente composto por no mínimo:

```
void setup() {  
}
```

```
void loop() {  
}
```

- O setup é executado apenas 1 vez quando a placa é ligada;
- O loop executa infinitamente!

# Conclusões



- Para mudar o estado de um LED de desligado para ligado ou vice-versa, precisamos gravar o 0 / 1, HIGH ou LOW na determinada porta:

```
digitalWrite(4, 1);           Número da porta: 4 ou 13  
digitalWrite(13, 0);  
delay(500);                  Ligado ou desligado, 0 ou 1, true ou false  
digitalWrite(4, false);  
digitalWrite(13, true);  
delay(500);
```

# Portas Analógicas



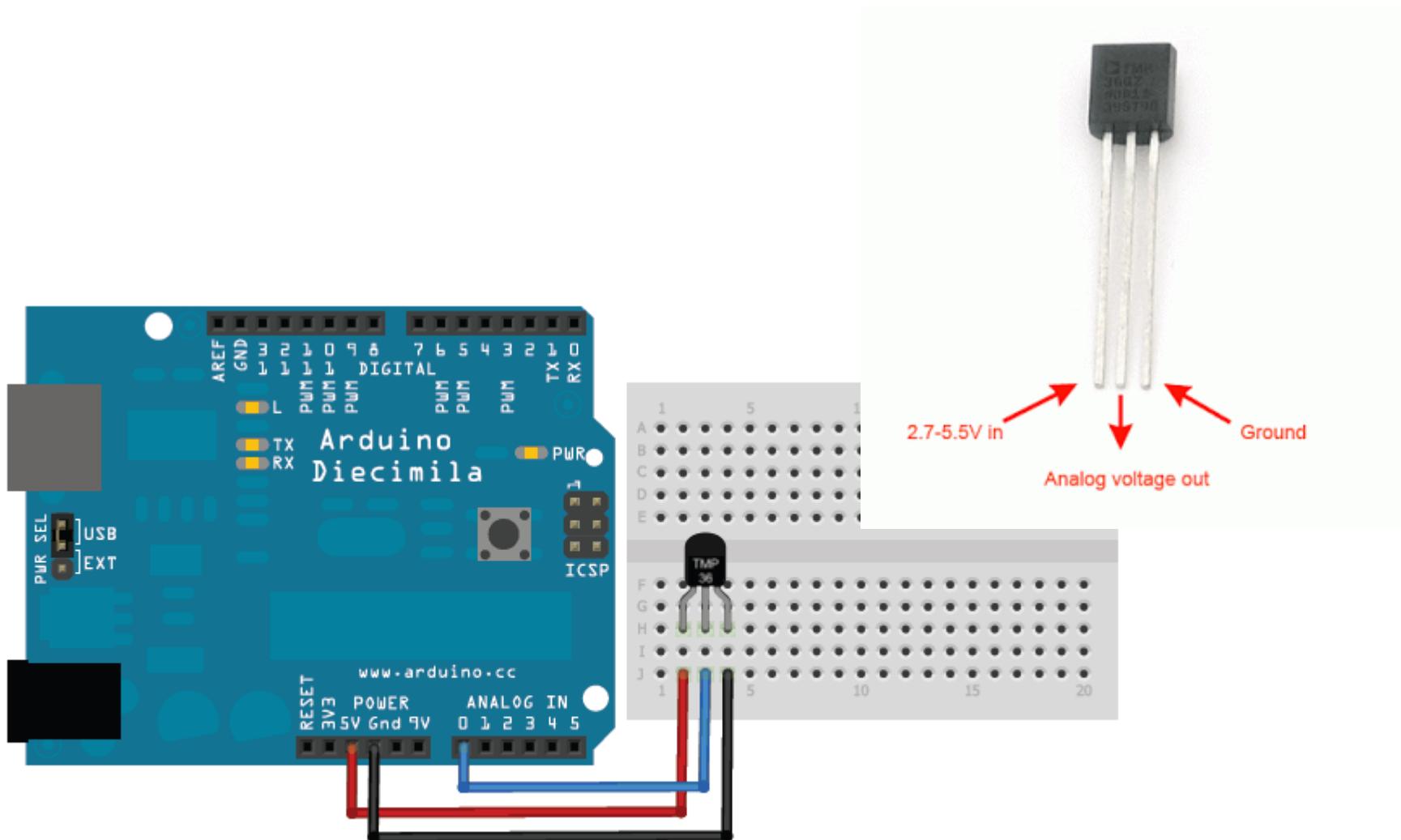
- Aprendemos que nas portas digitais trabalhamos com apenas 0 ou 1, ligada ou desligada.
- Isso é útil para indicar algo que tem apenas dois estados: tomada ligada ou desligada, motor ligado ou desligado, etc.
- Mas quando queremos por exemplo, indicar a temperatura do ambiente?
- Isso é inviável de ser representado por 0 e 1 (no máximo podemos falar quente / frio!)

# Portas Analógicas



- No Arduino e Program-ME contamos com portas analógicas além das portas digitais;
- Uma porta analógica nos fornece informações através de números que viriam entre 0 e 1023;
- Desta forma podemos obter dados com maior precisão através do uso de sensor como luz, temperatura, distância;

# Sensor Temperatura LM35



# Laboratório



- Agora faremos um programa que vai ler os sensores e imprimir os valores lidos na tela do nosso computador;
- Para isso vamos configurar o Arduino para que ele possa enviar dados da placa para o computador:

```
void setup() {  
    //Inicializando conexão com PC via cabo USB  
    Serial.begin(9600);  
}
```

# Laboratório



- Para efetuar a leitura do sensor do analógico em uma porta usamos o comando:

```
analogRead(0); //0 é o número da porta
```

- Para enviar informações da placa para o PC devemos utilizar o comando:

```
Serial.println("informações");
```

- Para enviar o dados do sensor para o PC:

```
Serial.println(analogRead(0));
```

# Laboratório



- O valor lido na porta representará a temperatura análoga a 0 - 1024, para obter o valor em Celsius usamos o seguinte código:

```
const float CELSIUS_BASE = 0.4887585532746823069403714565;

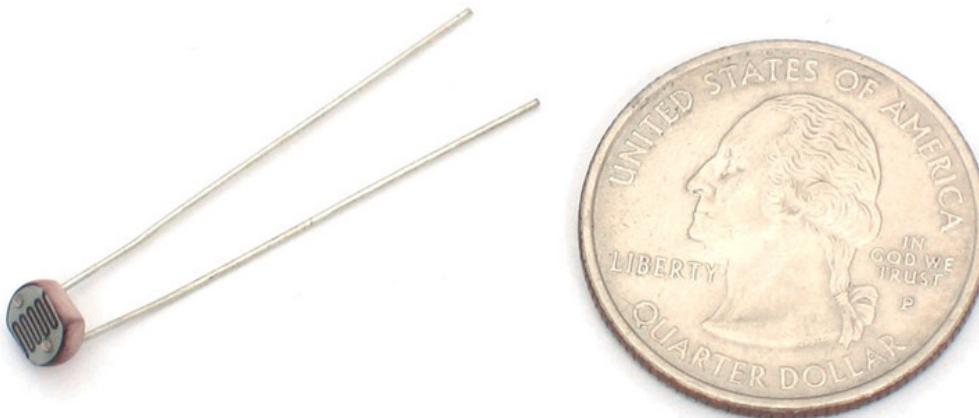
void setup() {
    Serial.begin(9600);
}

void loop() {
    Serial.print("Temperatura: ");
    Serial.println(lerTemperatura());
    delay(1000);
}

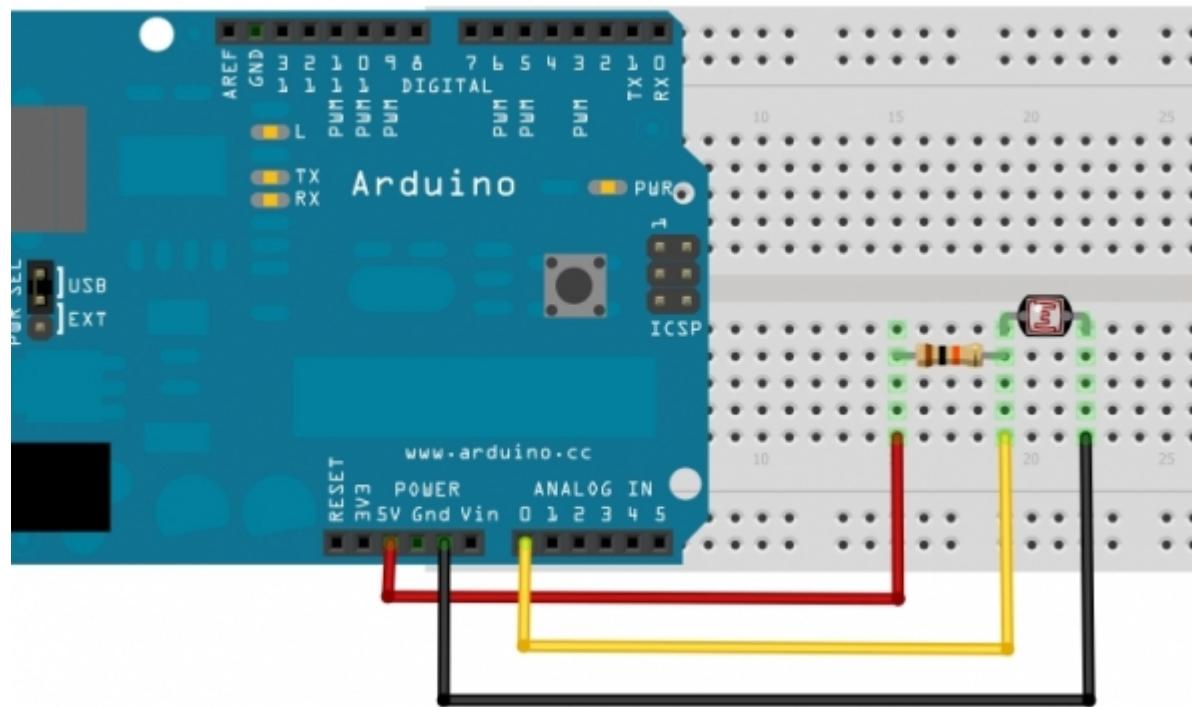
float lerTemperatura(){
    return (analogRead(0) * CELSIUS_BASE);
}
```

# LDR

- Light Dependent Resistor
- Usado como sensor de luz
- Quanto menos luz, maior o valor lido!



# LDR na Protoboard



# Laboratório

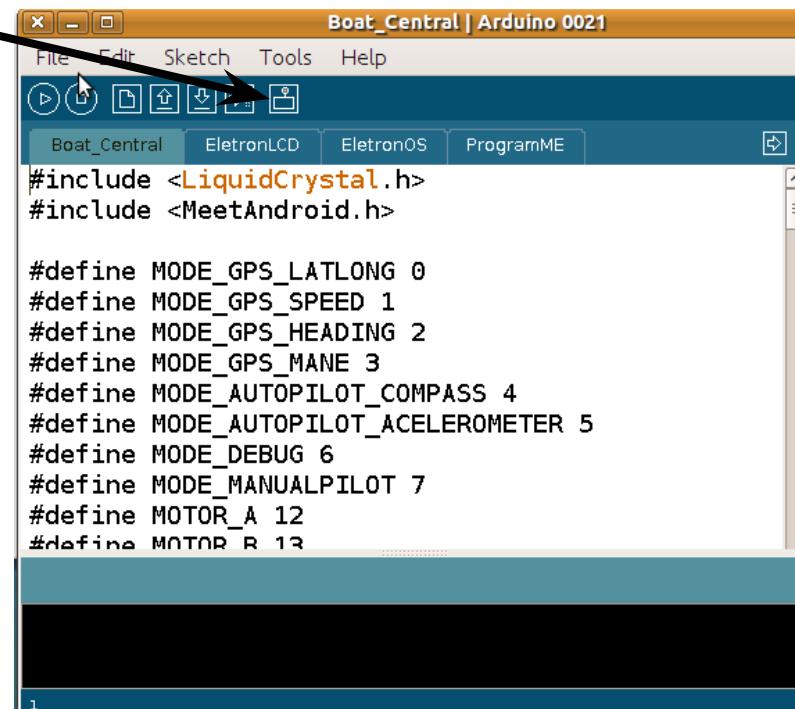


- Agora vamos digitar o código completo e transferir para placa:

```
void setup() {  
    //Inicializando conexão com PC via cabo USB  
    Serial.begin(9600);  
}  
  
void loop() {  
    //envia informações para o PC  
    Serial.println(analogRead(0));  
    delay(500);  
}
```

# Laboratório

- Para visualizar os dados enviados pela placa Arduino para o nosso computador, clique no seguinte ícone:



A screenshot of the Arduino IDE interface. The title bar reads "Boat\_Central | Arduino 0021". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar below has icons for Open, Save, Print, and others. The bottom tab bar shows "Boat\_Central" (selected), EletronLCD, EletronOS, and ProgramME. The main code area contains the following C++ code:

```
#include <LiquidCrystal.h>
#include <MeetAndroid.h>

#define MODE_GPS_LATLONG 0
#define MODE_GPS_SPEED 1
#define MODE_GPS_HEADING 2
#define MODE_GPS_MANE 3
#define MODE_AUTOPILOT_COMPASS 4
#define MODE_AUTOPILOT_ACCELEROMETER 5
#define MODE_DEBUG 6
#define MODE_MANUALPILOT 7
#define MOTOR_A 12
#define MOTOR_B 13
```

# Exemplo “luz ambiente”



```
void setup() {  
    //Inicializando conexão com PC via FT232 - cabo  
    Serial.begin(9600);  
}  
  
void loop() {  
    int luz = analogRead(1);  
    //envia informações para o PC  
    Serial.println(luz);  
    delay(500);  
}
```

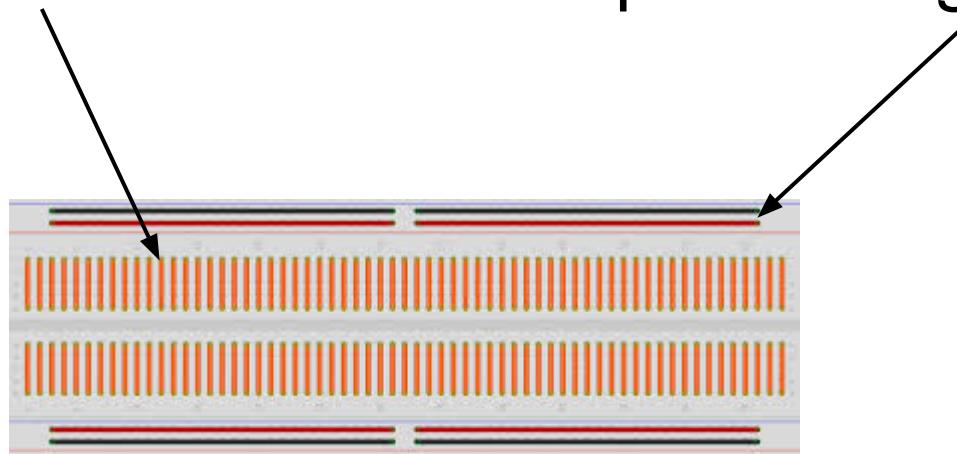
# Exemplo “luz ambiente”



```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.print("Luz: ");  
    if(analogRead(1)>700) Serial.println(" apagada.");  
    else Serial.println(" acesa.");  
    delay(1000);  
}
```

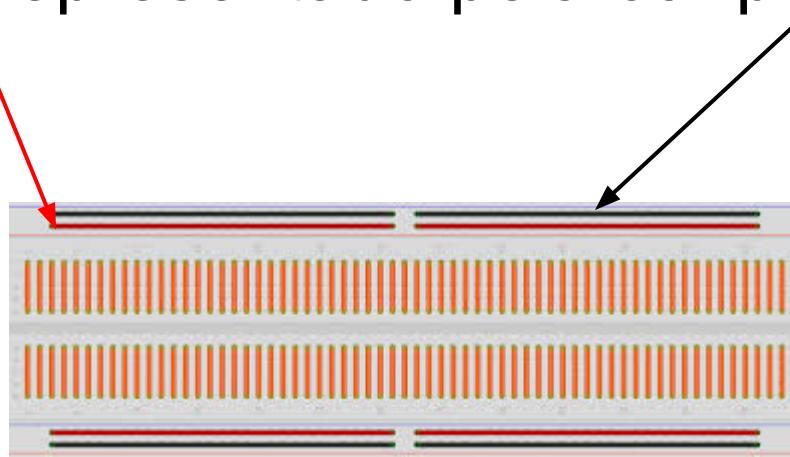
# Conhecendo a protoboard

- Facilita o trabalho de teste e protótipos sem a necessidade de soldar fios;
- Tem seus terminais conectados na vertical para componentes e na horizontal para energia:



# Energizando a protoboard

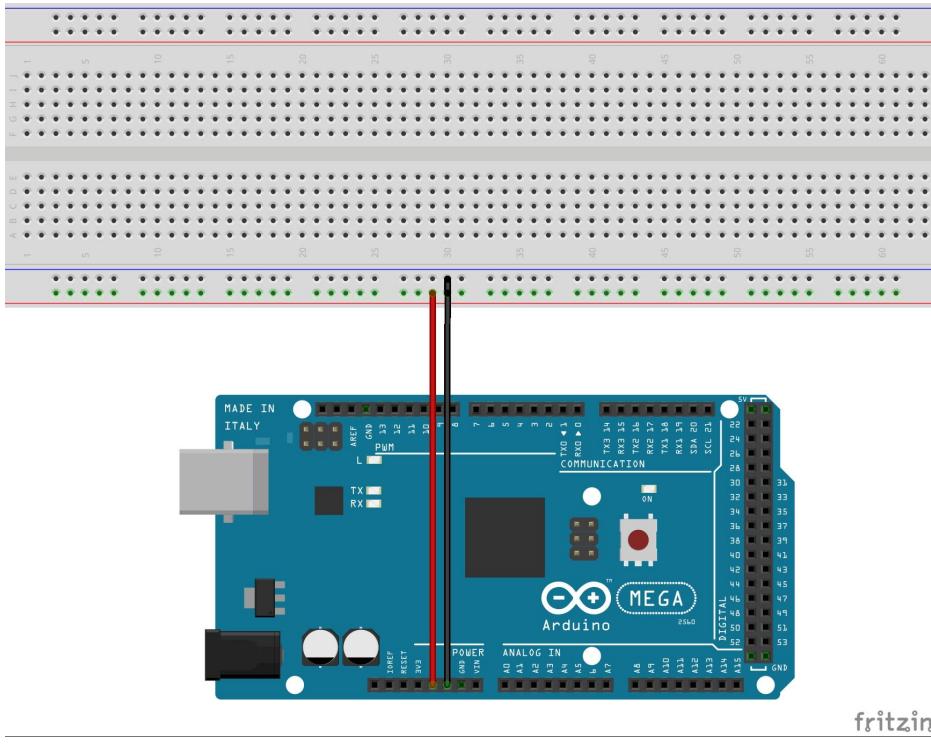
- Quase tudo que ligamos no Arduino terá a necessidade de dois fios para energia:
  - 5volts - representado pela cor vermelha
  - GND - representado pela cor preta



# Energizando a protoboard



- Assim fica a protoboard com os fios de 5volts e ground do Arduino, podemos acessar a energia por toda a horizontal, apenas no lado energizado:

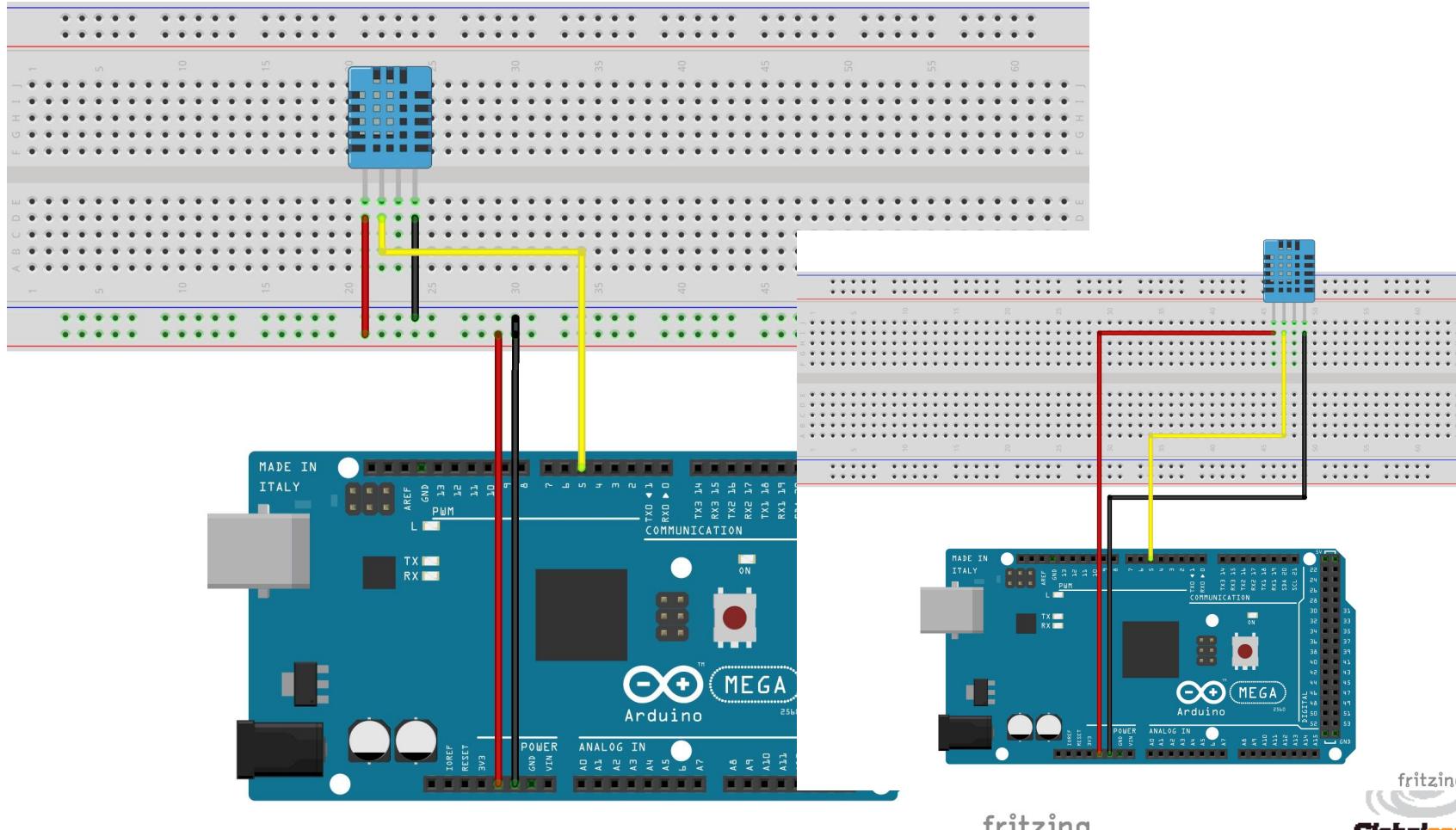


fritzing

# Ligando Componentes



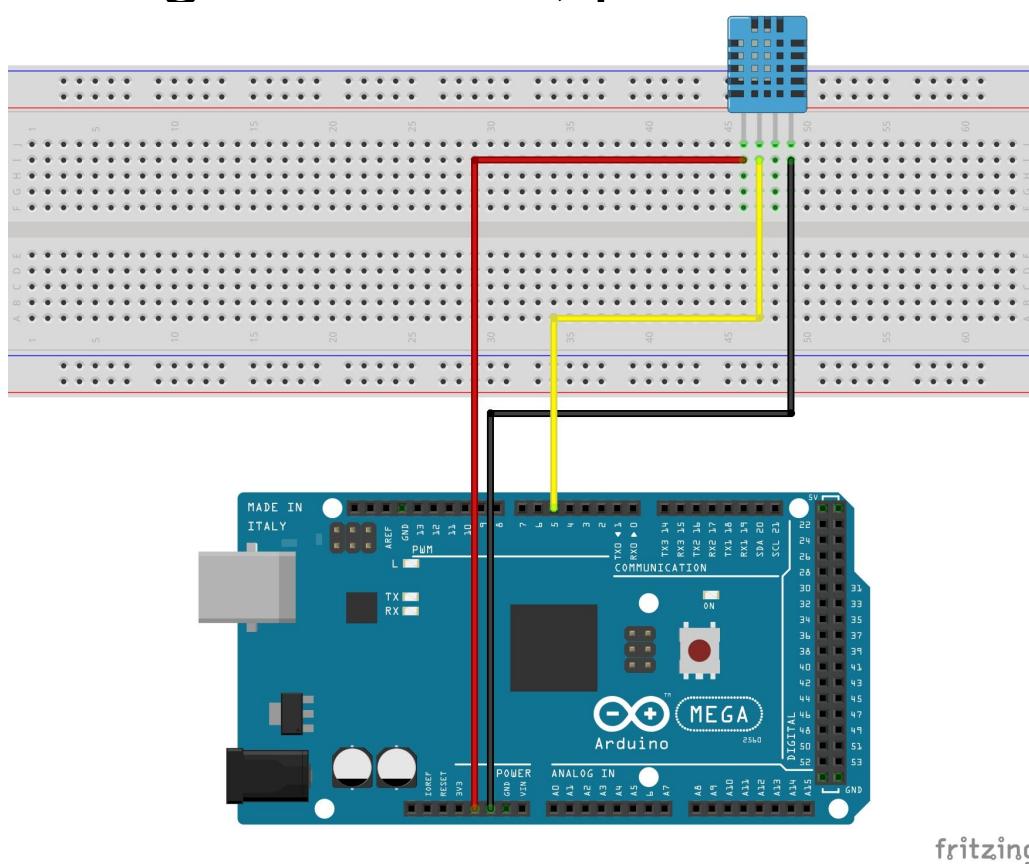
- Agora podemos ligar os componentes desta forma:



# Ligando Componentes



- Podemos usar a protoboard sem as conexões de energia também, porém o Arduino poucos 5volts!



fritzing

# Sensor DHT11

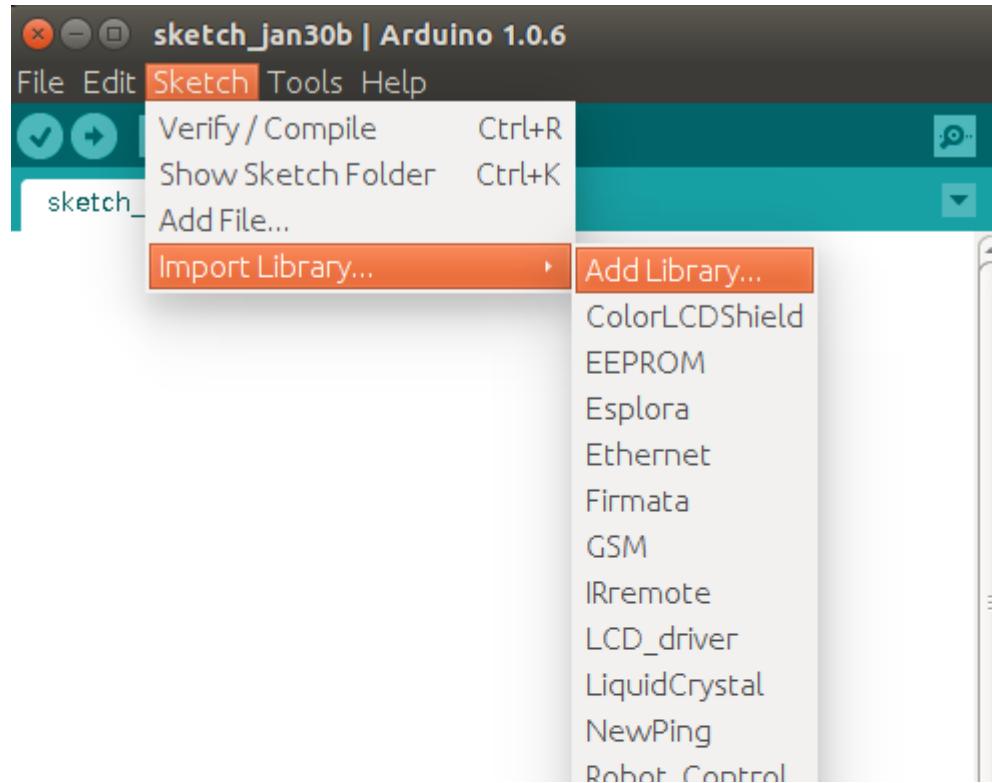


- É um sensor de baixo custo para medir temperatura e umidade;
- Utiliza apenas um pino, porém sua leitura é mais complexa;
- Um colaborador escreveu e disponibilizou um biblioteca Arduino para a família DHT;
- Vamos ver como instalar uma biblioteca externa no Arduino IDE!

# Adicionando bibliotecas...



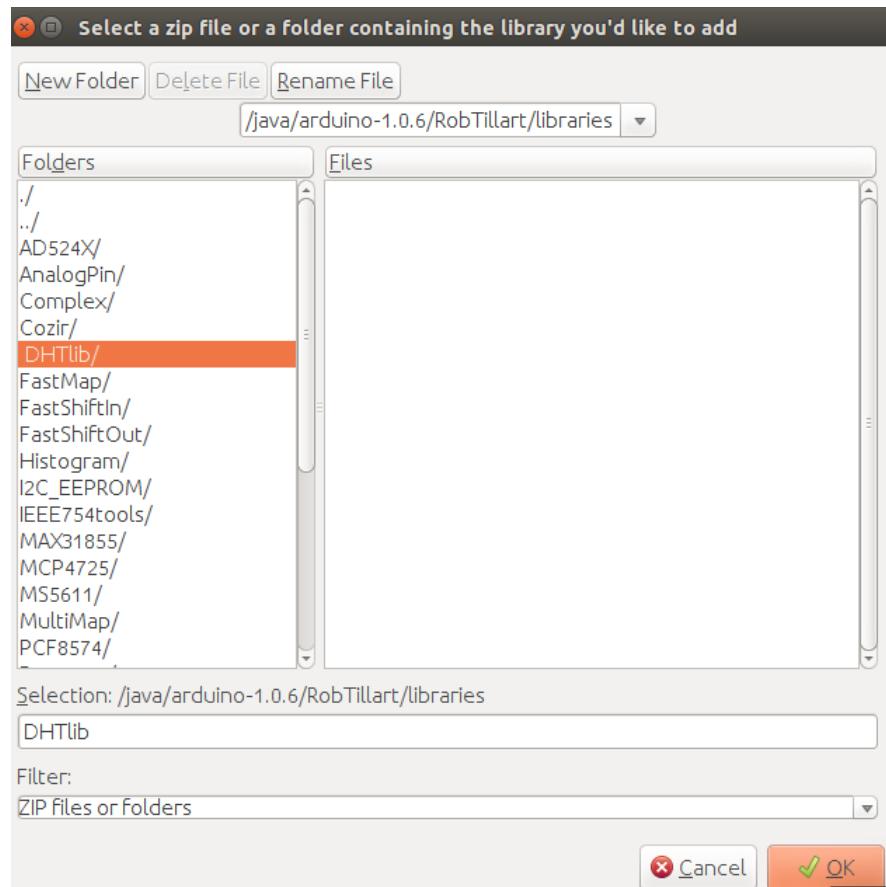
- No menu Sketch -> Import Library -> Add Library



# Adicionando bibliotecas...

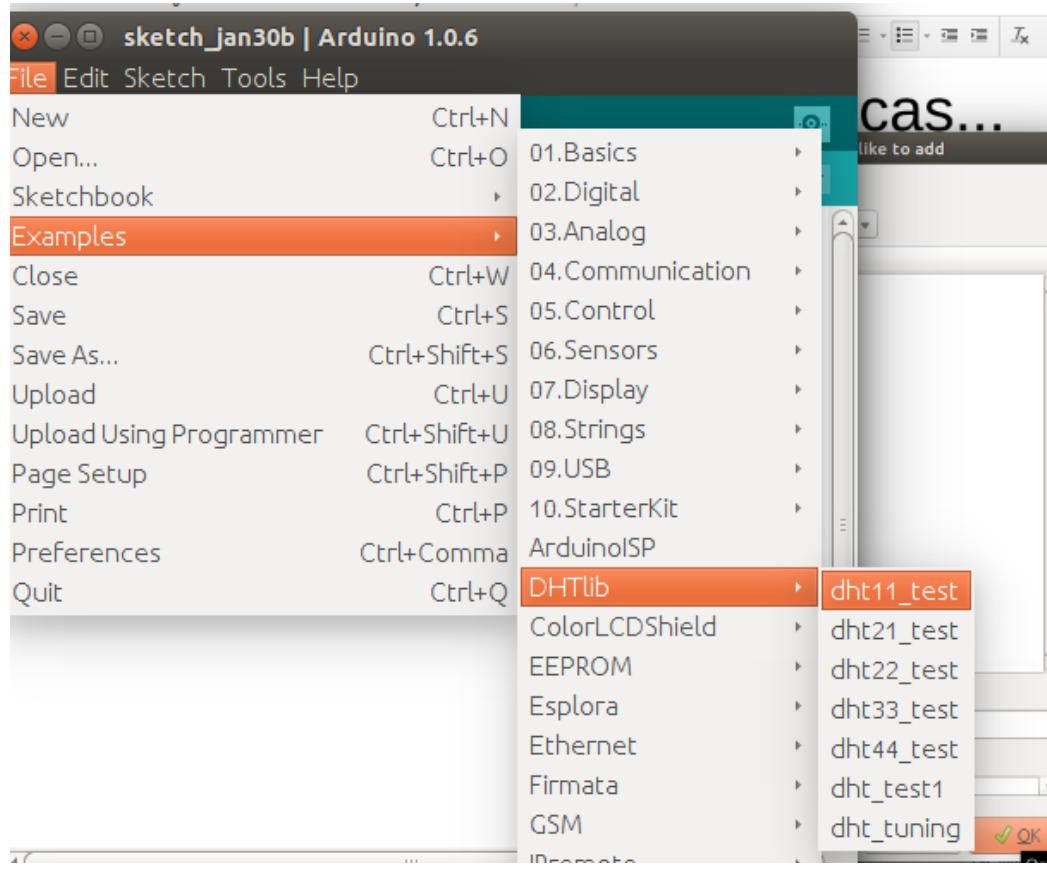


- Escolha o local onde copiou a biblioteca:



# Programa Exemplo

- Em seguida, clique no menu File -> Examples -> DHTLib -> dht11\_test



# Programa Exemplo



Note que há uma definição chamada **DHT11\_PIN** que está configurada para porta digital 5

```
// FILE: dht11_test.ino
// AUTHOR: Rob Tillaart
// VERSION: 0.1.01
// PURPOSE: DHT library test sketch for DHT11 & Arduino
// URL:
//
// Released to the public domain
//
#include <dht.h>

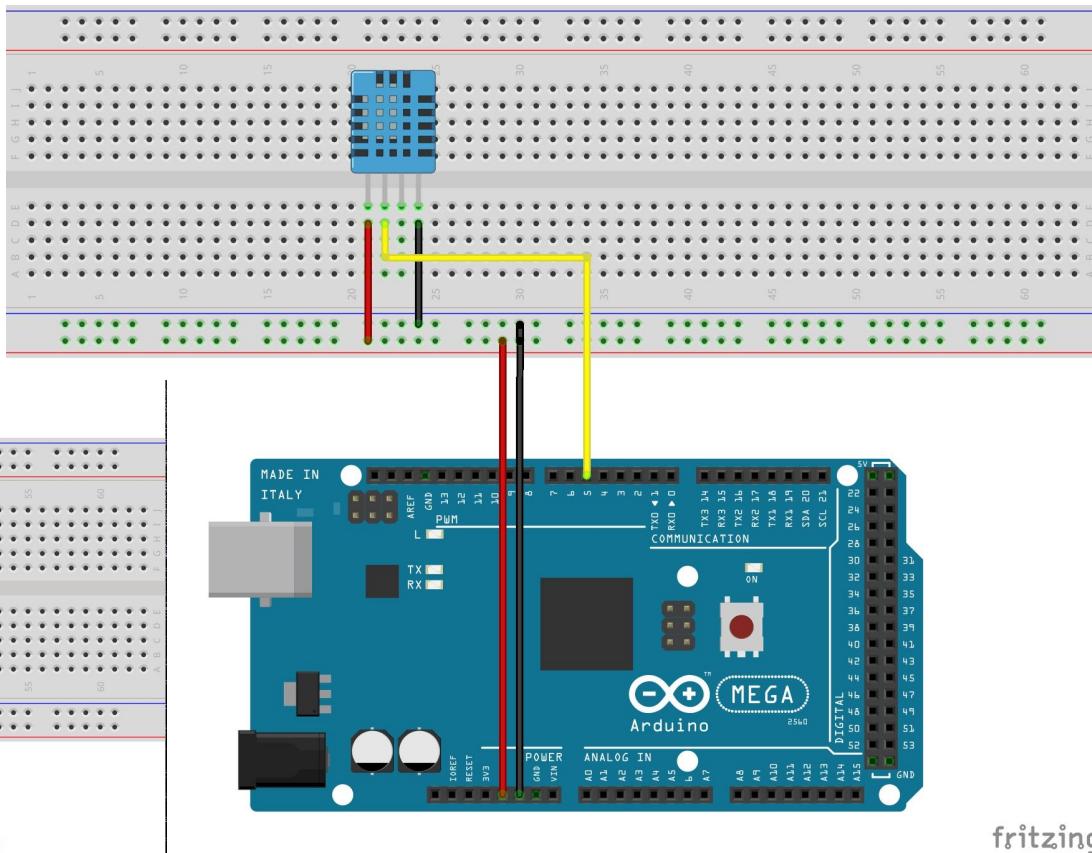
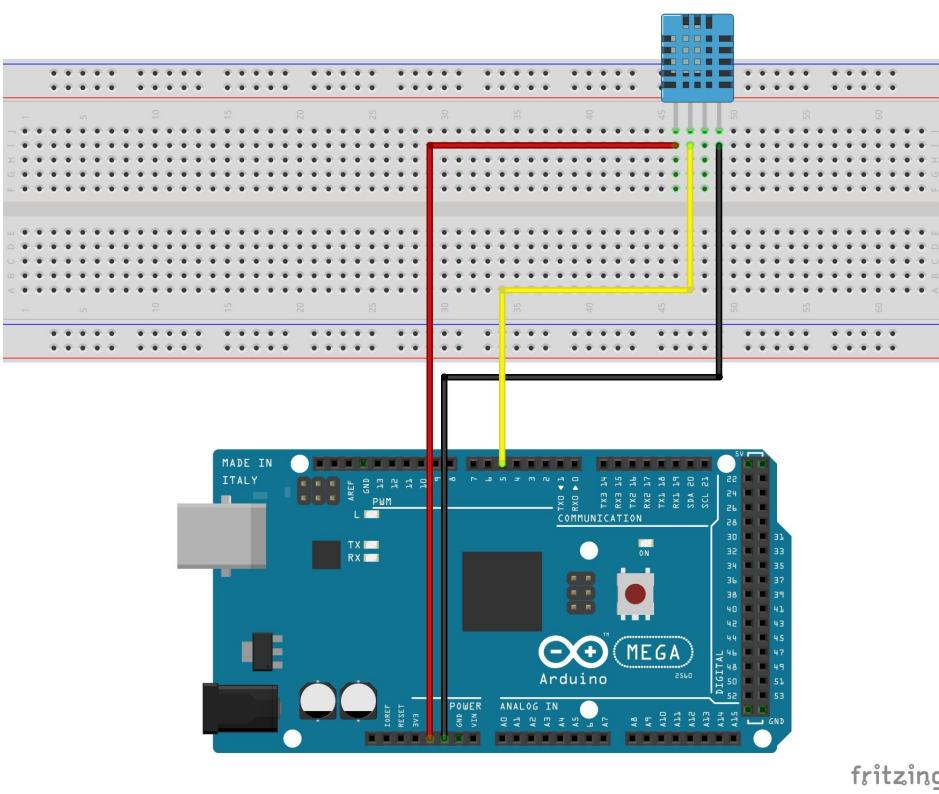
dht DHT;
#define DHT11_PIN 5

void setup()
{
    Serial.begin(115200);
    Serial.println("DHT TEST PROGRAM ");
    Serial.print("LIBRARY VERSION: ");
    Serial.println(DHT_LIB_VERSION);
}

void loop()
{
    float h = DHT.read();
    if (isnan(h)) {
        Serial.println("No reading from DHT sensor!");
        return;
    }
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.println("%");
}
```

Done uploading.  
Binary sketch size: 7,396 bytes (of a 258,048 byte maximum)

# Faça Conexão - CUIDADO!



fritzing

# Upload e Teste



1. Faça Upload
2. Abra o Serial Monitor
3. Configure o baudrate  
e veja o resultado!

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** dht11\_test | Arduino 1.0.6
- Toolbar:** Includes icons for upload, refresh, and other tools.
- Sketch Name:** dht11\_test
- Code Area:** Displays the source code for the DHT11 test sketch. The code includes comments about the file being released to the public domain and defines the DHT11\_PIN as 5. It also includes setup and loop functions for initializing the serial port and printing library information.
- Serial Monitor:** Shows the message "Done uploading." and the binary sketch size information: "Binary sketch size: 7,396 bytes (of a 258,048 byte maximum)".
- Status Bar:** Shows the board as "Arduino Mega 2560 or Mega ADK" and the port as "/dev/ttyACM0".

# Sensor HC-SR04

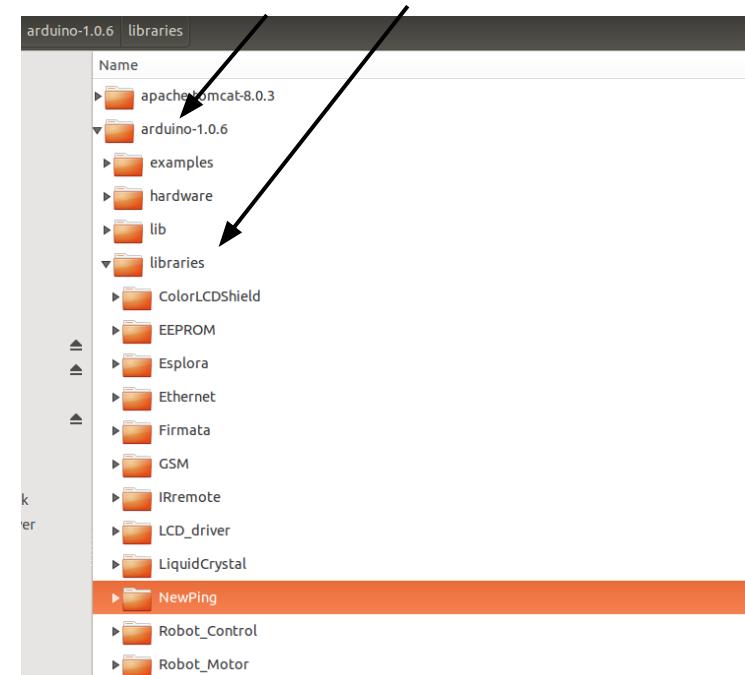


- É um sensor ultrasonico capaz de medir a distância entre ele e um objeto!
- Ele envia um sinal sonoro e mede o tempo de retorno;
- O tempo de retorno será calculado com base na velocidade do som!
- Sensor que requer tempo real;

# Sensor HC-SR04

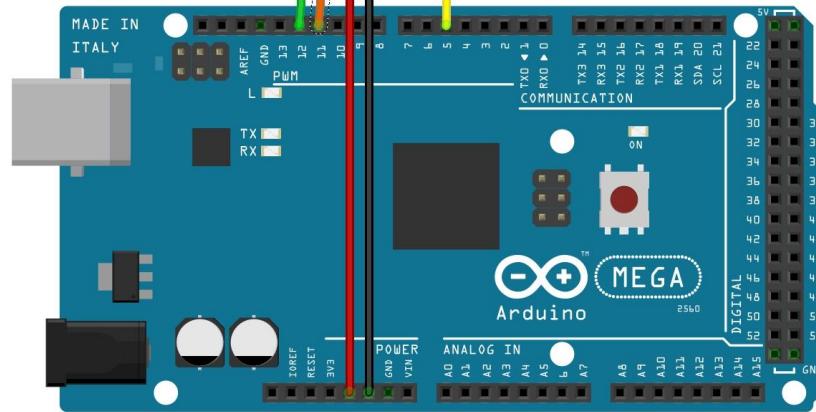
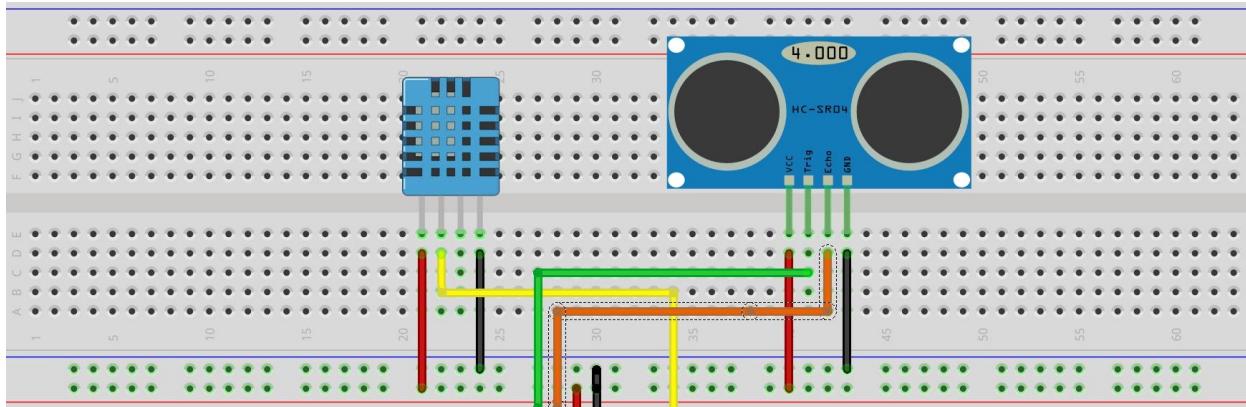


- A comunidade criou uma biblioteca para ele chamada de NewPing;
- Instale a biblioteca conforme apresentado anteriormente ou simplesmente copiando o diretório NewPing para o diretório: arduino/libraries



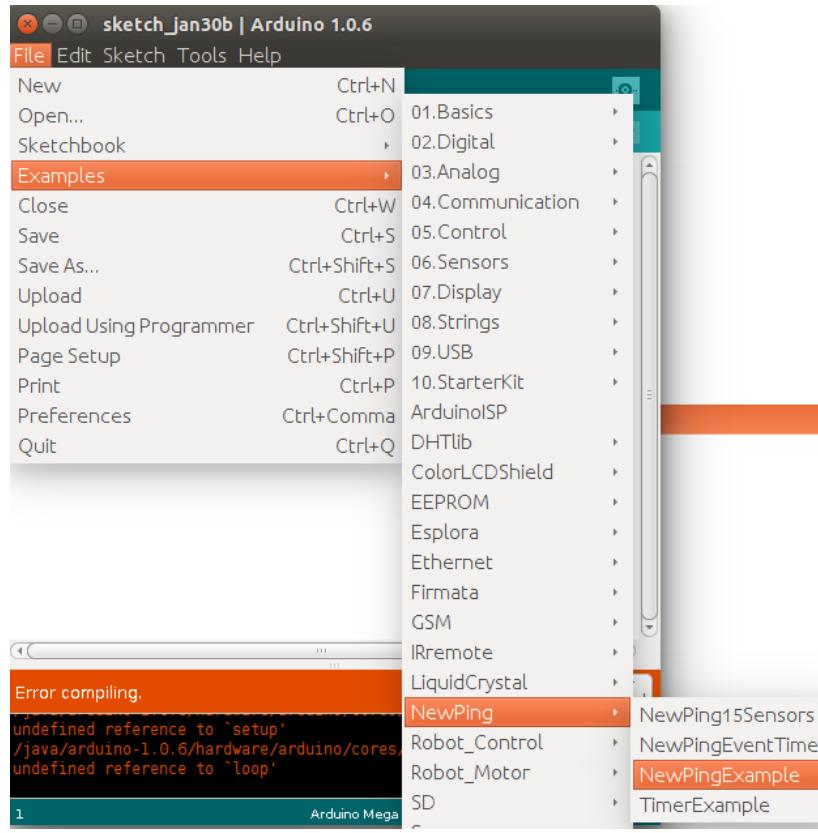
# HC-SR04: fiação

- Trabalha com 4 fios: 5volts / GND / Trigger / Echo



# Sensor HC-SR04

- Ele também contém exemplos, menu:  
File -> Examples - NewPing -> NewPingExample



# Sensor HC-SR04: testes!



- O resultado esperado é:

```
Ping: 8cm
Ping: 9cm
Ping: 0cm
Ping: 19cm
Ping: 17cm
Ping: 6cm
Ping: 8cm
Ping: 11cm
Ping: 0cm
Ping: 0cm
Ping: 0cm
Ping: 4cm
Ping: 9cm
Ping: 13cm
Ping: 10cm
Ping: 16cm
Ping: 11cm
Ping: 10cm
Ping: 18cm
Ping: 10cm
Ping: 9cm
Ping: 17cm
Ping: 10cm
Ping: 12cm
Ping: 10cm
Ping: 10cm
Ping: 12cm
Ping: 12cm
Ping: 11cm
Ping: 13cm
Ping: 11cm
Ping: 11cm
Ping: 12cm
Ping: 12cm
Ping: 13cm
Ping: 11cm
Ping: 11cm
Ping: 12cm
Ping: 11cm
Ping: 12cm
```

Autoscroll      Newline      115200 baud

# Receptor Infravermelho

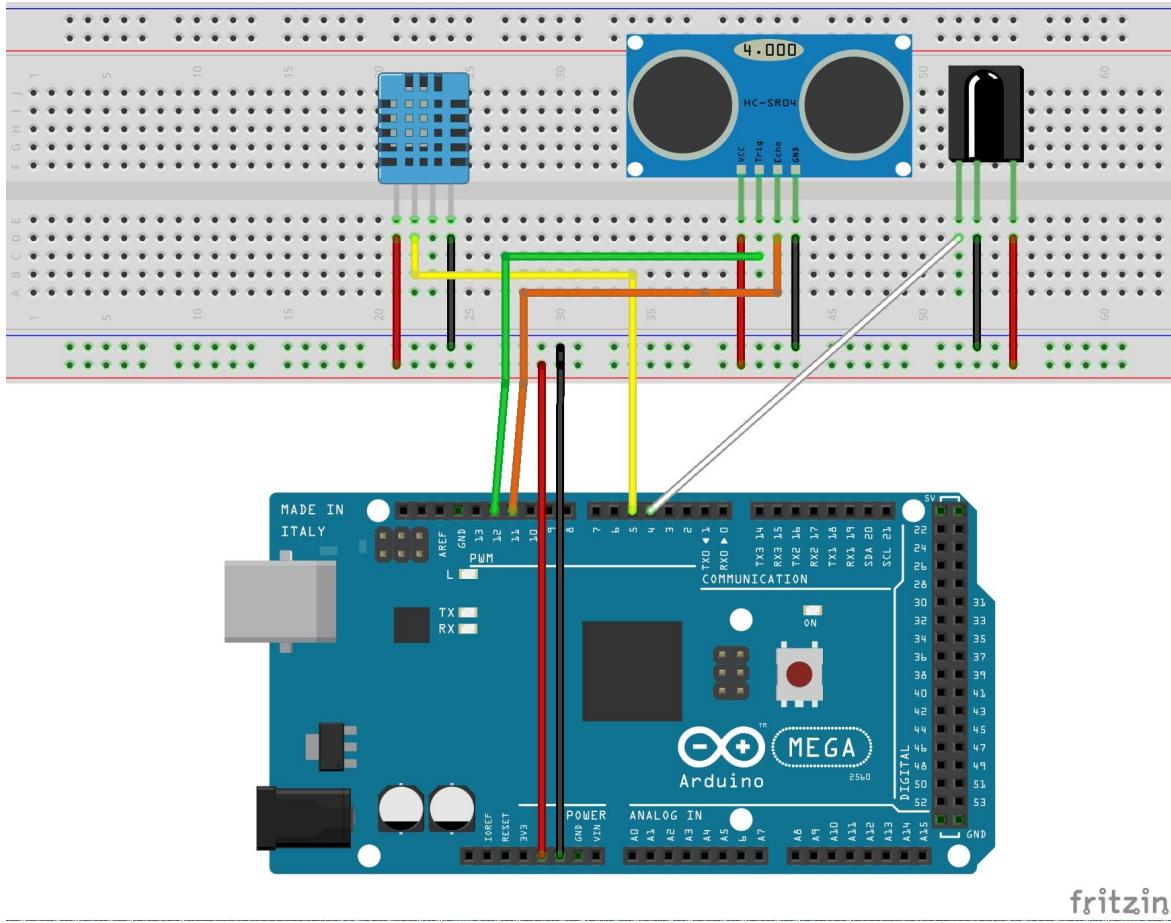


- Infravermelho é algo muito funcional, simples, barato e amplamente suportado no Arduino;
- Existe uma excelente biblioteca para suporte a IR chamada IRRremote que devemos instalar;
- O receptor do IR tem 3 fios: 5v, GND e dados;
- O emissor de IR é um LED convencional;
- Vejamos a seguir a conexão de um receptor IR;

# Receptor Infravermelho



- Estamos ligando os dados no pino 4:



# IR: Programa receptor



The screenshot shows the Arduino IDE interface. The title bar reads "IRrecvDemo | Arduino 1.0.6". The "File" menu is open, displaying various options like "New", "Open...", "Sketchbook", "Examples", "Close", "Save", "Save As...", "Upload", "Upload Using Programmer", "Page Setup", "Print", "Preferences", and "Quit". Below the menu, a code editor window displays the "IRrecvDemo" sketch. The code uses the IRrecv library to receive infrared signals. The "loop" function checks for received data and prints it to the serial monitor. The upload status at the bottom indicates "Done uploading." and a binary sketch size of 11,374 bytes.

```
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
```

Done uploading.  
Binary sketch size: 11,374 bytes (of a 258,000 maximum)

11  
Done uploading.  
Binary sketch size: 11,374 bytes (of a 258,000 maximum)

# IR Código Exemplo



IRRecvDemo | Arduino 1.0.6

File Edit Sketch Tools Help

IRRecvDemo §

```
/*
 * IRremote: IRrecvDemo - demonstrates receiving IR codes with IRrecv
 * An IR detector/demodulator must be connected to the input RECV_PIN.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 */
#include <IRremote.h>

int RECV_PIN = 4; // Change this to your desired pin number

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn(); // Start the receiver
}

void loop() {
    if (irrecv.decode(&results)) {
        Serial.println(results.value, HEX);
        irrecv.resume(); // Receive the next value
    }
    delay(100);
}
```

Done uploading.

Binary sketch size: 11,374 bytes (of a 258,048 byte maximum)

Arduino Mega 2560 or Mega ADK on /dev/ttyACM0

# IR Resultado Captura



```
./dev/ttyACM0
Send
E17AD02F
FFFFFFF
FFFFFFF
E17AD02F
FFFFFFF
FFFFFFF
E17AD02F
FFFFFFF
E17A708F

 Autoscroll
Newline 9600 baud
```

# Relay / Relé



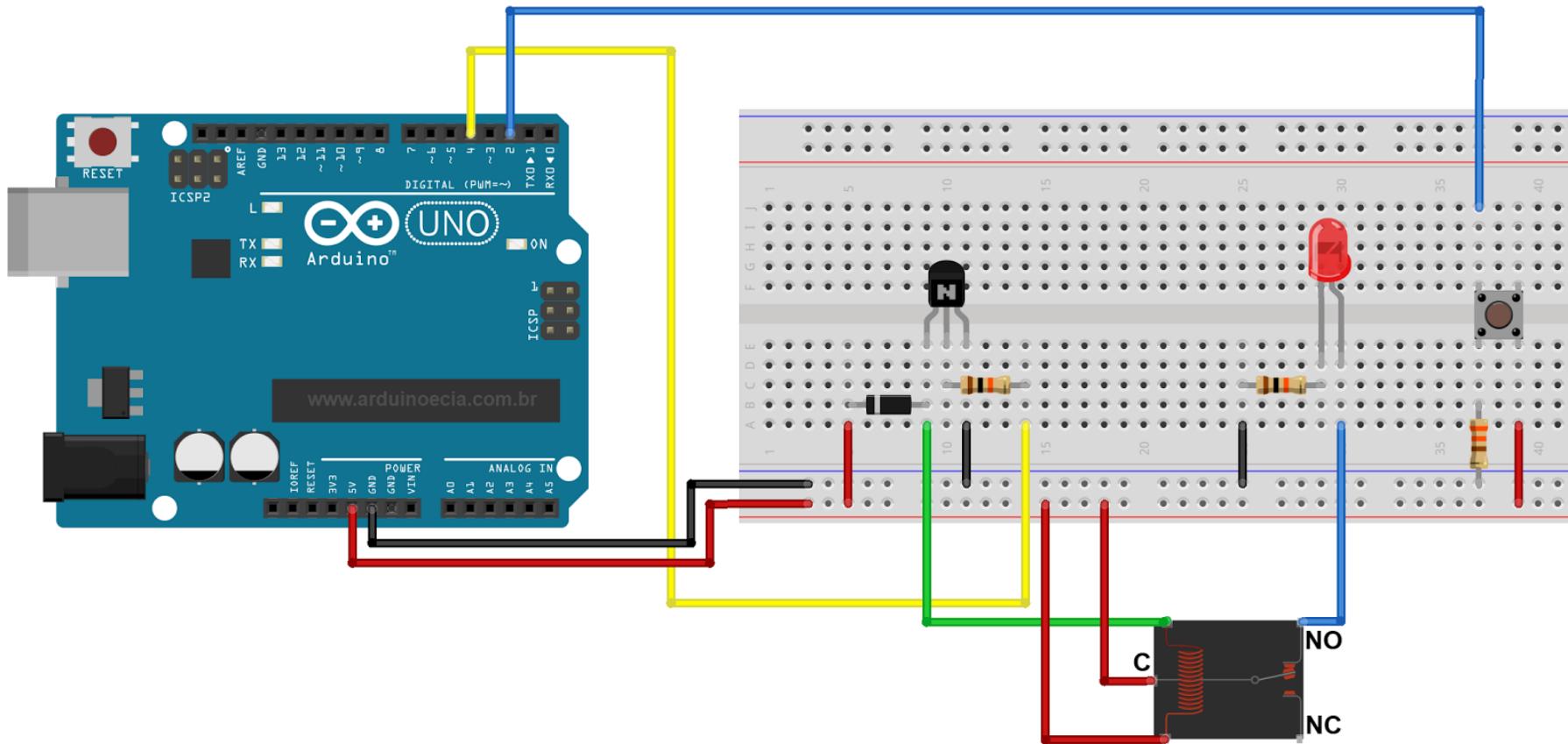
- Relay é um atuador capaz de interromper um corrente elétrica de maior tensão;
- Um microcontrolador consegue ligar e desligar pinos de 5volts apenas!
- Um relé é uma bobina eletromagnética que sendo energizada fecha ou abre um contato;
- Este contato interrompe ou libera a corrente;
- Relés fazer barulho "click";

# Arduino + Relé



- A bobina de acionamento do relé pode ter diferentes tensões: 5volts, 9volts, 12 volts etc..
- Usando um relé 5volts teremos a tensão compatível com o tensão dos pinos Arduino!
- Porém a porta / pino digital do Arduino não corrente suficiente para chavear a bobina do relé;
- Para ampliar a corrente da porta digital, devemos usar um transistor e um diodo de proteção;

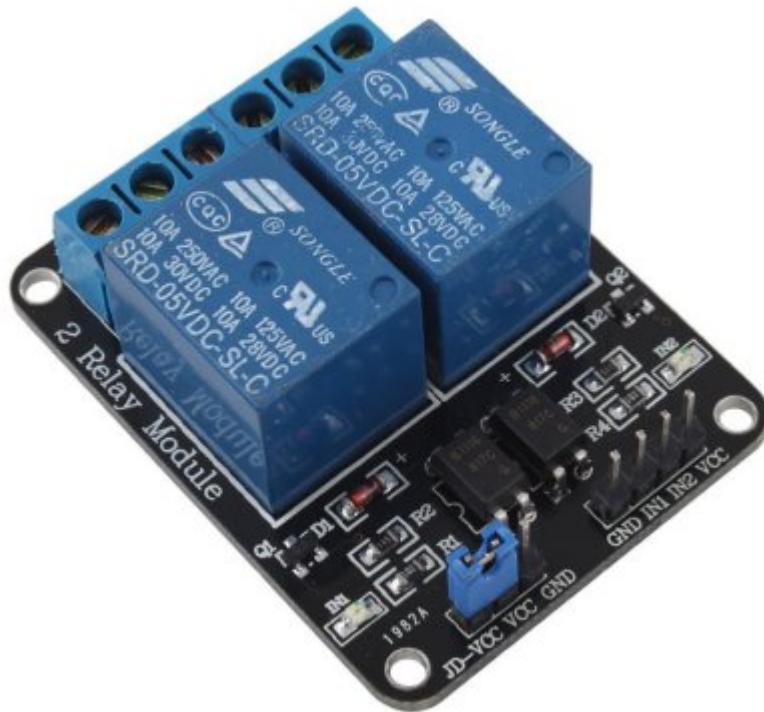
# Arduino + Relé



# Placa Relé 2 Canais

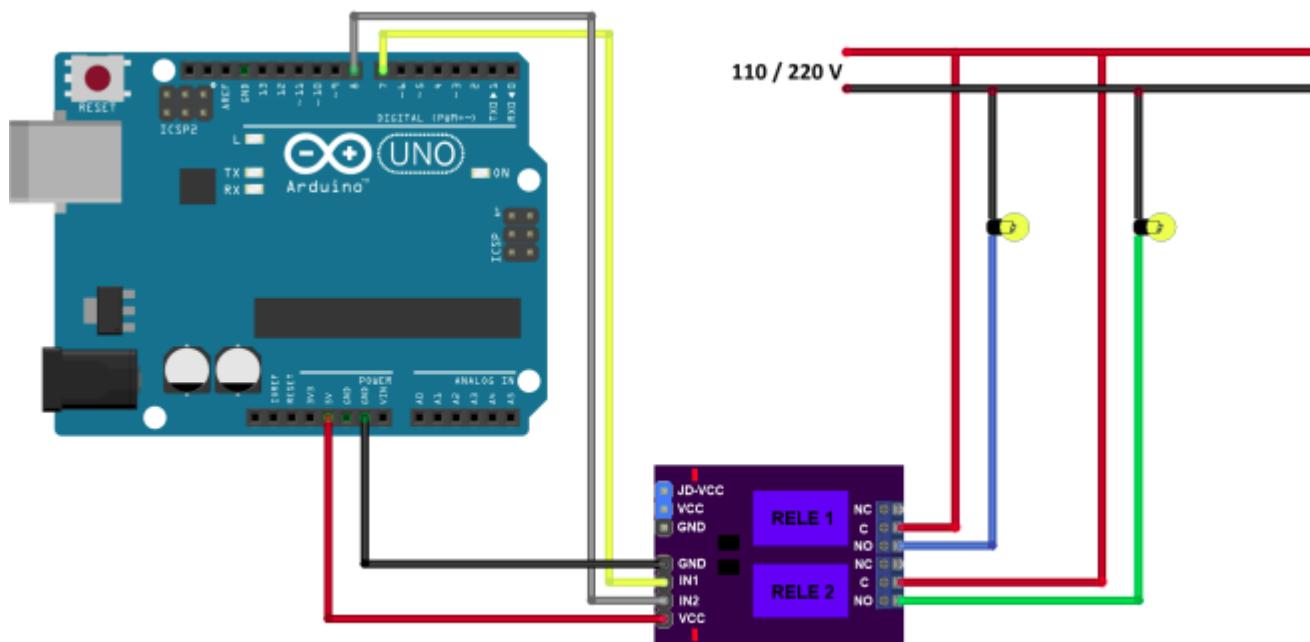


- Para facilitar o uso de relé, vamos utilizar uma placa de relé 2 canais, elá já tem transistor, diodo, resistores e led indicativo!



# Placa Relé - Fiação

- Vamos usar 4 fios em portas digitais convencionais: 5volts - GND - Relé 1 - Relé 2



# Arduino + Relé: código



- Para ligar e desligar um relé, basta colocar a porta utilizada em HIGH / LOW:

```
digitalWrite(7, HIGH);
```

# Motores

- Os tipos mais comuns de motores são:
  - Motor DC / CC / Corrente Contínua
  - Servomotor
  - Motor de passo
  - Motor brushless



# Motores DC / CC

- Os motores de corrente contínua são comuns em carrinhos a pilha, pequenos ventiladores e outros;
- São indicados para girar rápido;
- Pode-se acoplar caixas redutoras para transformar velocidade em força;



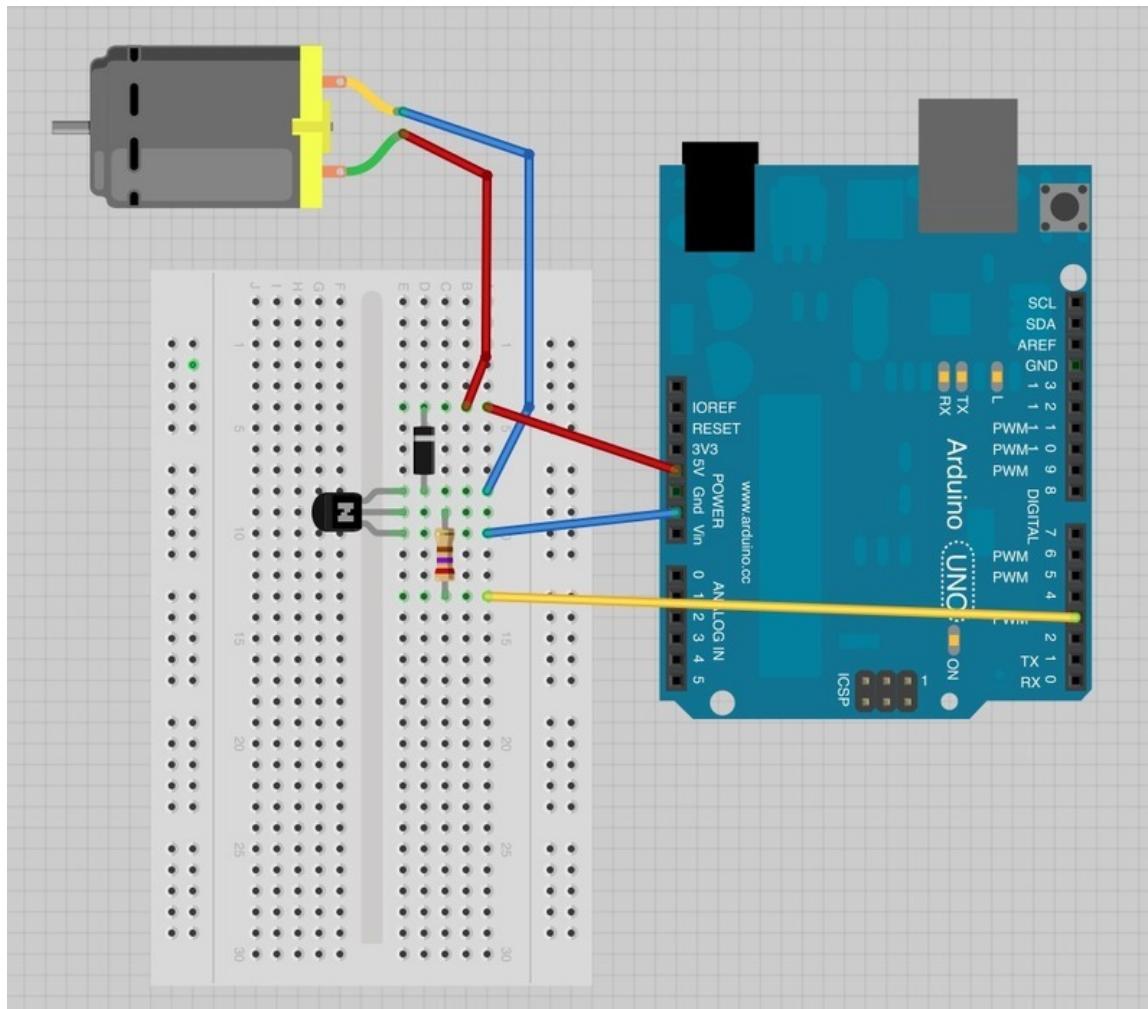
# Mais sobre Motores DC



- Existem motores de diferentes tensões: 3 volts, 6 volts, 9, 12, 24 etc.
- A direção do giro, depende da polaridade;



# Motores DC + Arduino



# Motores DC + Arduino



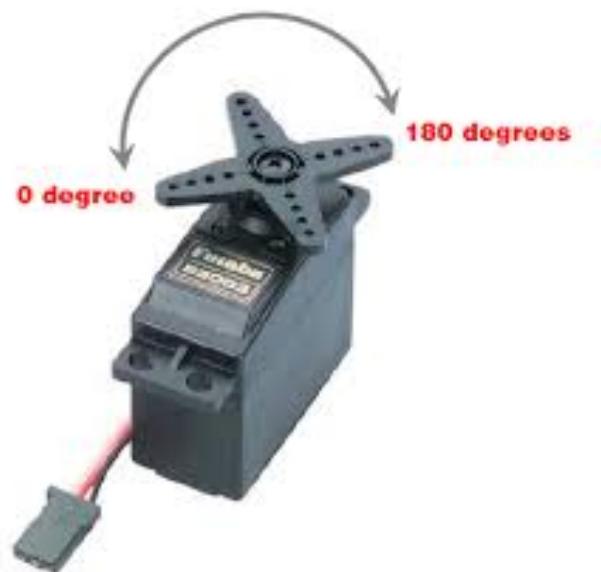
- Para controlar a velocidade do motor usamos a porta PWM, no exemplo 3:

```
analogWrite(3, 255); //velocidade máxima;  
analogWrite(3, 50); //baixa velocidade;
```

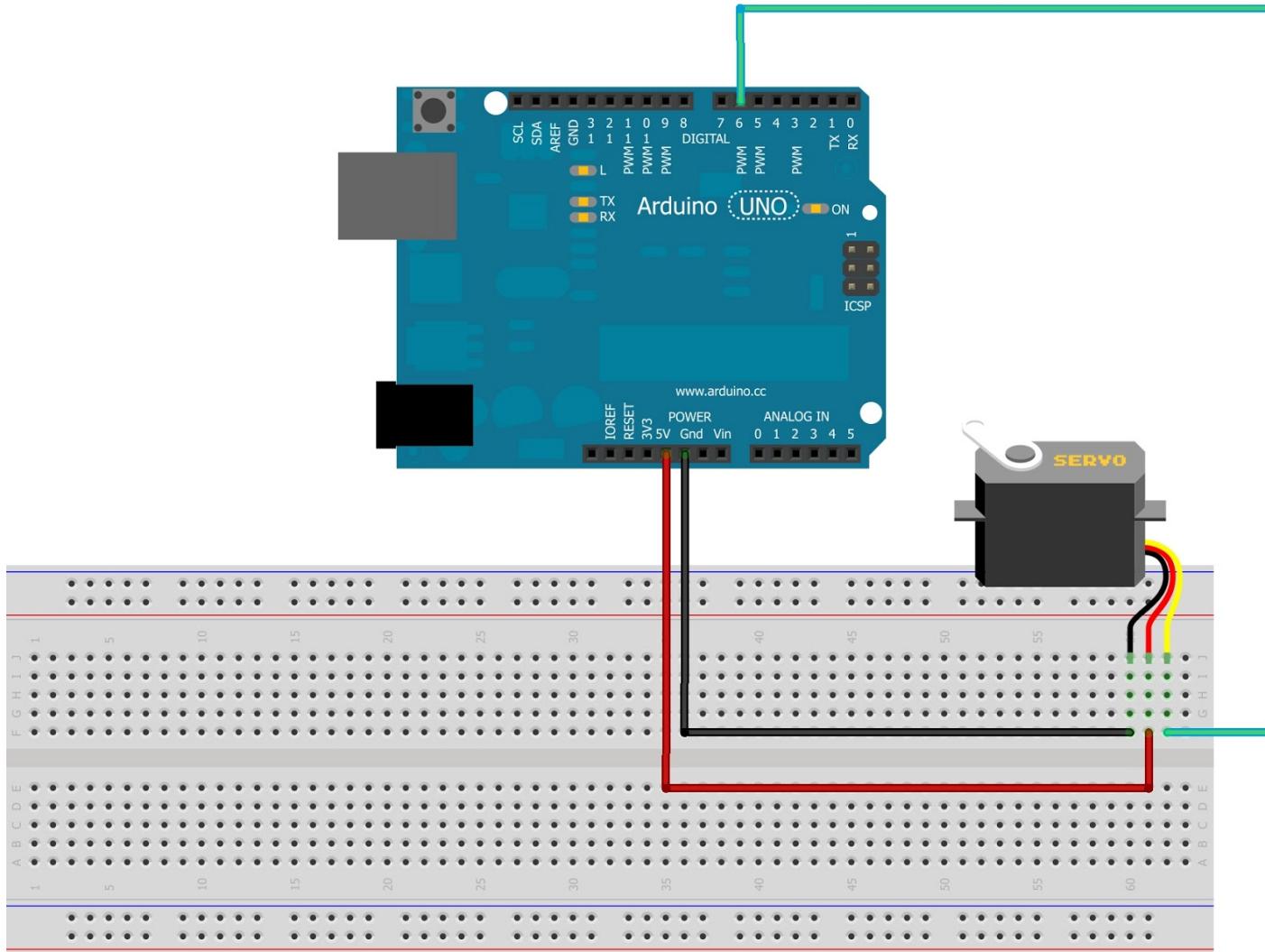
- É possível manipular a direção do giro através de controle de ponte H de transistores ou placa controladora de motor pronta!
- É possível saber se o motor está ou não rodando e quanto através de sistemas de feed-back;

# Servomotor

- É um conjunto de um motor DC + caixa de redução + sistema de feed-back que permite o posicionamento com precisão entre 0 e 180 graus;

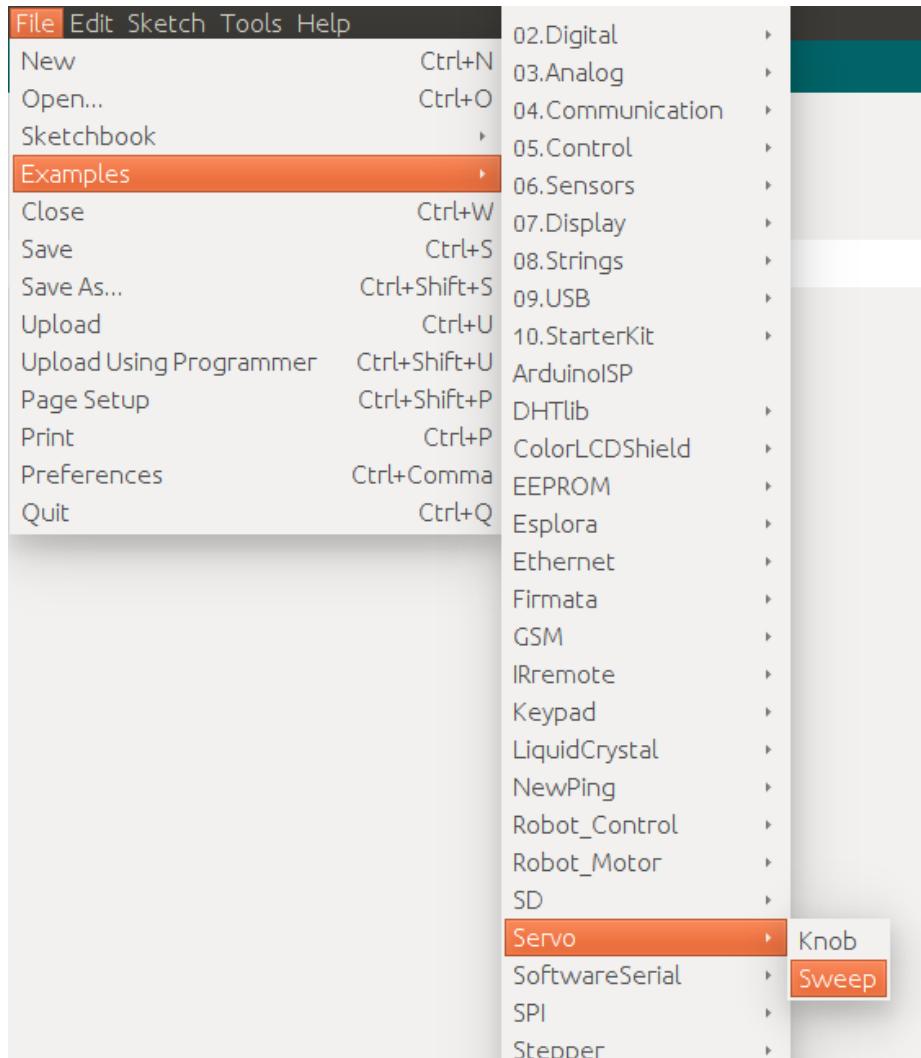


# Servomotor + Arduino

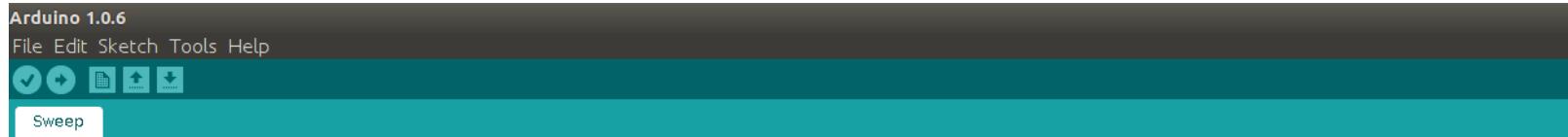


Made with Fritzing.org

# Arduino Servo: exemplo



# Servomotor: código



```
Arduino 1.0.6
File Edit Sketch Tools Help
Sweep

#include <Servo.h>

Servo myservo; // create servo object to control a servo
                // twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup()
{
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

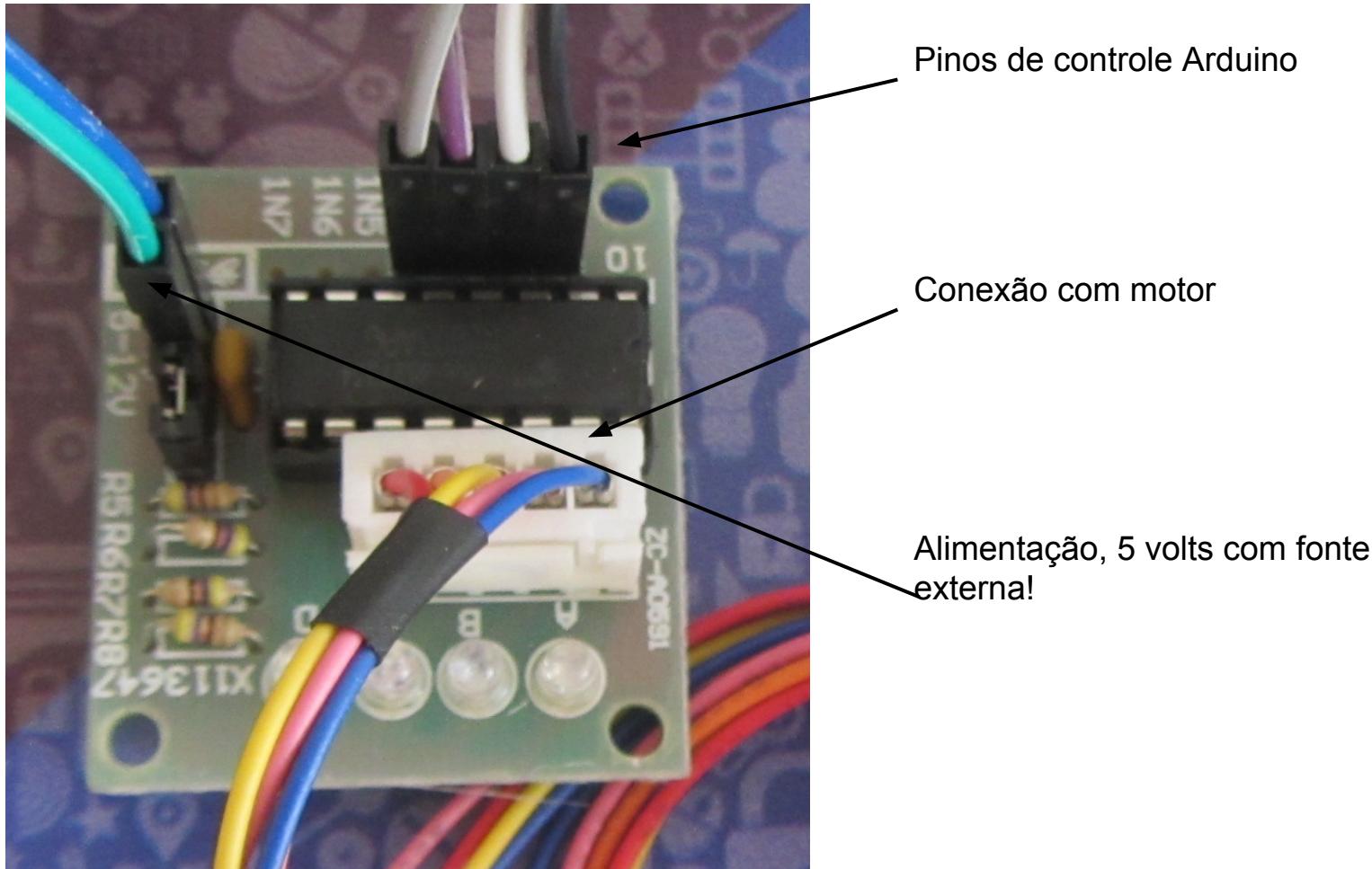
void loop()
{
  for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degrees
  {
    myservo.write(pos);           // tell servo to go to position in variable 'pos'
    delay(15);                  // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=0; pos-=1) // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);           // tell servo to go to position in variable 'pos'
    delay(15);                  // waits 15ms for the servo to reach the position
  }
}
```

# Motores de passo

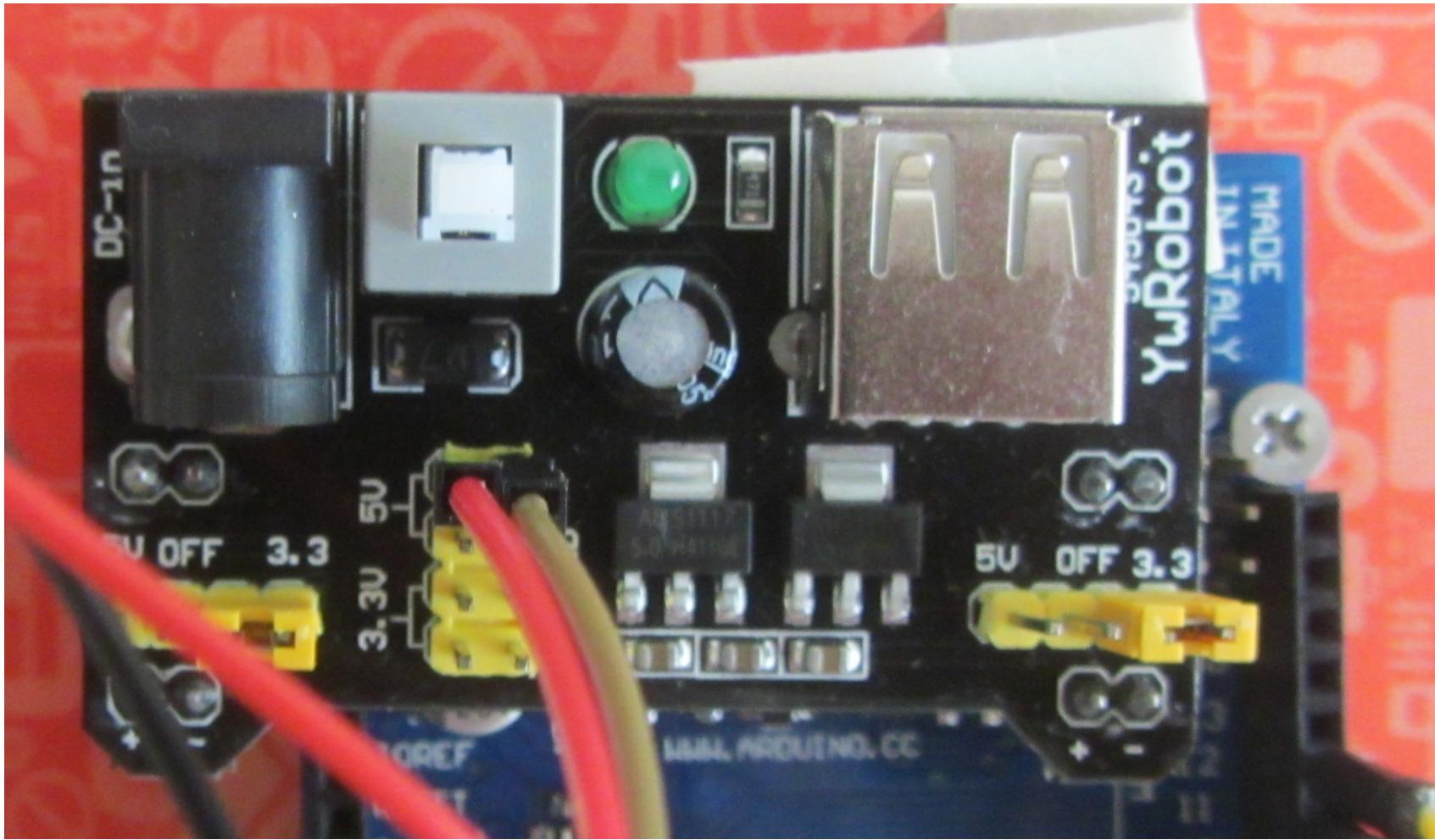


- É um motor que tem força e precisão nos movimentos;
- Muito usado em máquina CNC;
- Mais difícil de controlar;
- Em geral usamos uma placa controladora que é conhecida como Driver de motor de passo;

# Driver motor de passo



# Fonte Externa



# Motor de passo: código



```
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
void setup()
{
    Serial.begin(115200
    Serial.println("DHT
    Serial.print("LIBRAI
    Serial.println(DHT_
    Serial.println();
    Serial.println("Typ
}
void loop()
{
    // READ DATA
    Serial.print("DHT11
    int chk = DHT.read1_----_--_
    switch (chk)
```

The screenshot shows the Arduino IDE interface. The menu bar at the top includes File, Edit, Sketch, Tools, and Help. The Examples menu is currently selected. In the center, there's a code editor with the provided C++ code for a stepper motor driver. A context menu is open over the 'Stepper' entry in the Examples list, with the 'stepper\_oneRevolution' option highlighted. The background of the IDE has a dark theme with colored bars (blue, green, yellow) corresponding to different library categories.

# Motor de passo: código



```
#include <Stepper.h>

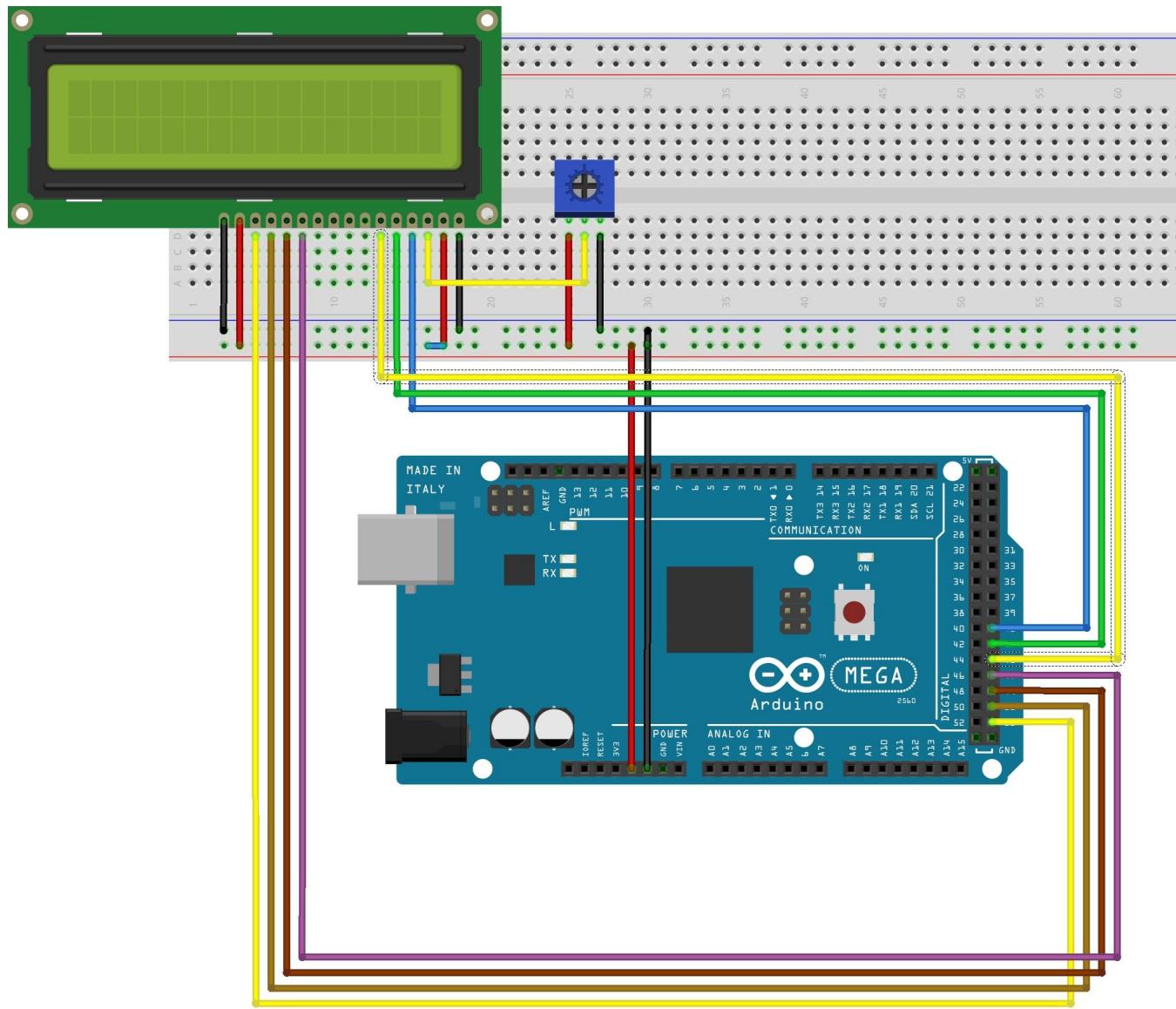
const int stepsPerRevolution = 200; // change this to fit
                                    // for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8,9,10,11);

void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {
  // step one revolution in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);
```

# Display LCD



fritzing



# Display LCD



lcd | Arduino 1.0.6

File Edit Sketch Tools Help

lcd

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/LiquidCrystal>

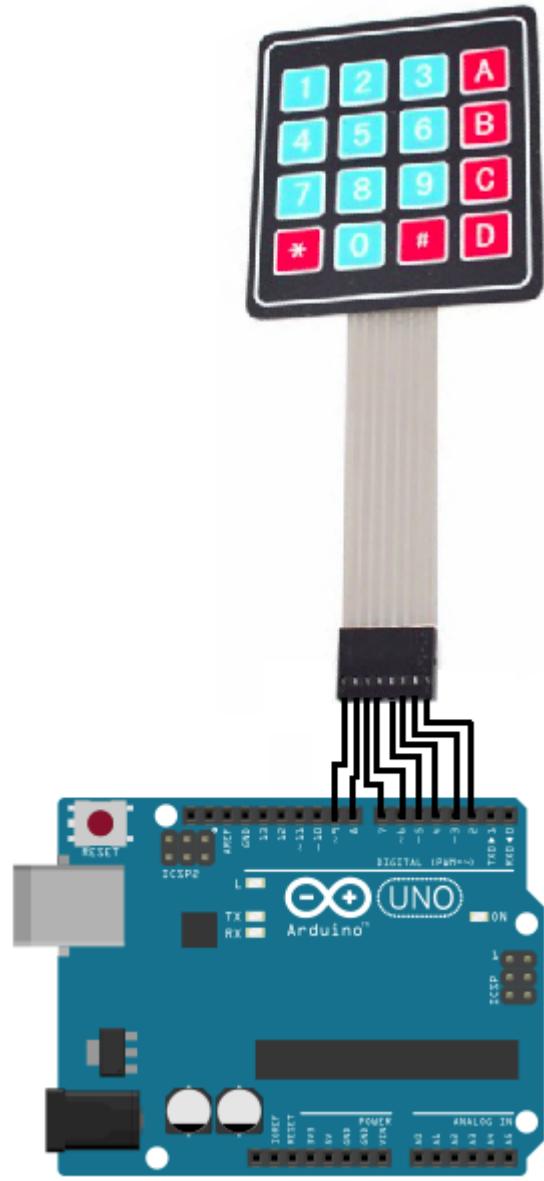
```
/*
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(41, 43, 45, 47, 49, 51, 53);

void setup() {
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.print("hello, world!");
}

void loop() {
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting begins with 0):
    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print(millis()/1000);
}
```

# Teclado



# Teclado: código



```
teclado | Arduino 1.0.6
File Edit Sketch Tools Help
teclado S
*/
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};
byte rowPins[ROWS] = {22, 24, 26, 28}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {30, 32, 34, 36}; //connect to the column pinouts of the keypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
}

void loop(){
  char customKey = customKeypad.getKey();

  if (customKey){
    Serial.println(customKey);
  }
}
```