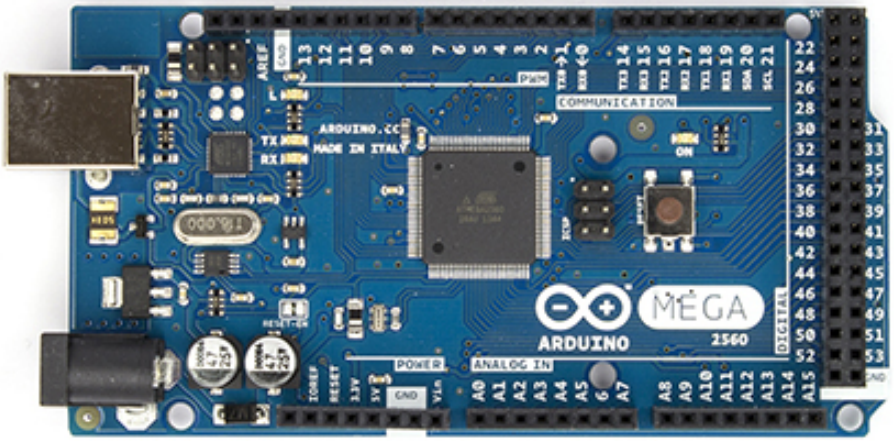
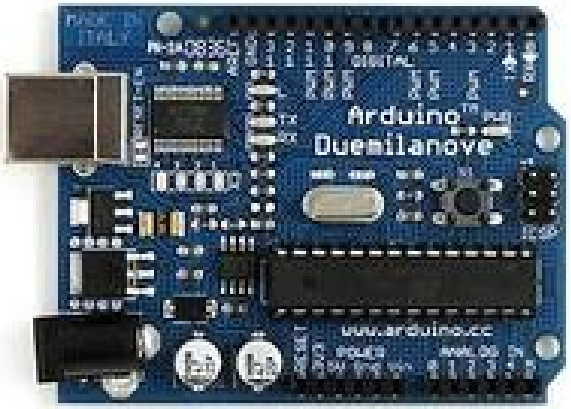
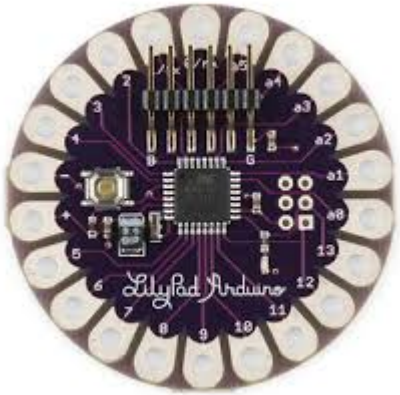
Arduino

elétron livre







Globalcode

Introdução a Arduino

elétron livre

- 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...

elétron livre

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

Histórico do projeto Arduino

elétron livre

- → 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

elétron livre

- **Microcontrolador**: um microcomputador compacto completo resumido em um circuito. Inclui processador, memórias e I/Os.

- **Firmware:** pedaço de software instalado em um microcontrolador.

- **Bootloadder:** pedaço de software instalado na memória de um microcontrolador que permite a carga de firmwares por serial, USB ou outros.

Globalcode

Arduino: parte técnica

elétron livre

- 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**

Globalcode

Atmega328

elétron livre

- 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

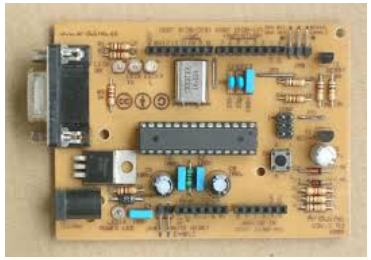
elétron livre

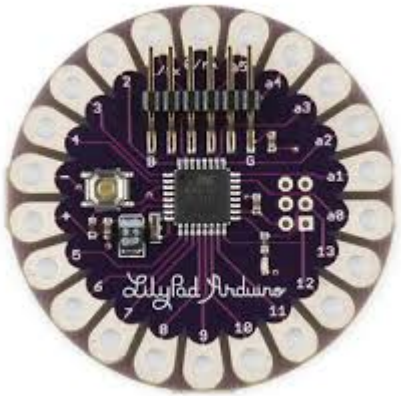
- 14 portas digitais
- 6 canais PWM
- 6 conversores analógico/digital de 10 bits
- 1 serial programável (USART)
- 1 interface SPI (Serial Peripheral Interface)
- 1 interface serial a 2 fios (I2C)
- 1 watch dog timer programável
- 1 comparador analógico no chip
- Interrupção ou wake-up na alteração de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

Globalcode

Diversos tipos de Arduino

elétron livre

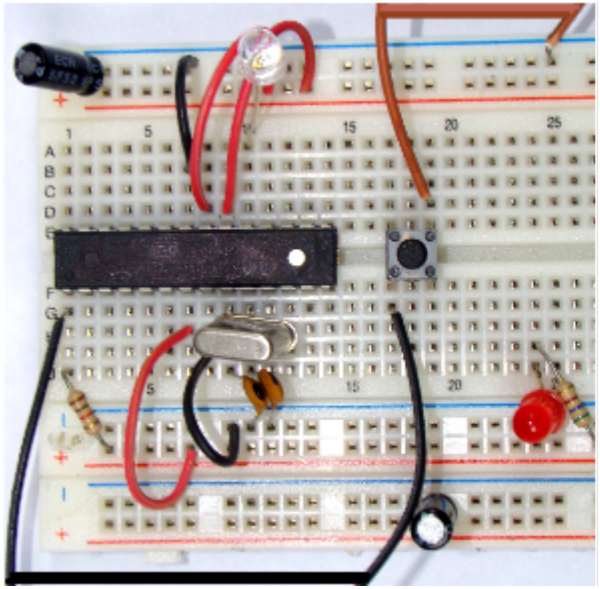


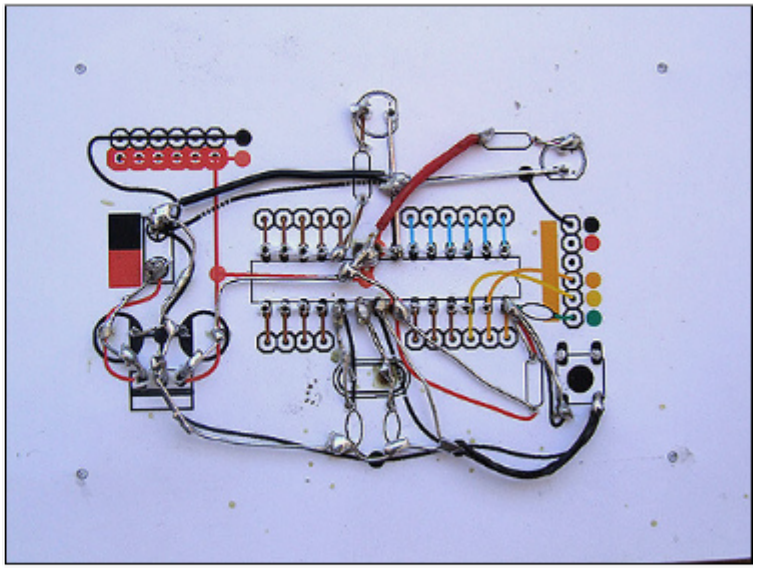


Globalcode

Ou então simplesmente...

elétron livre

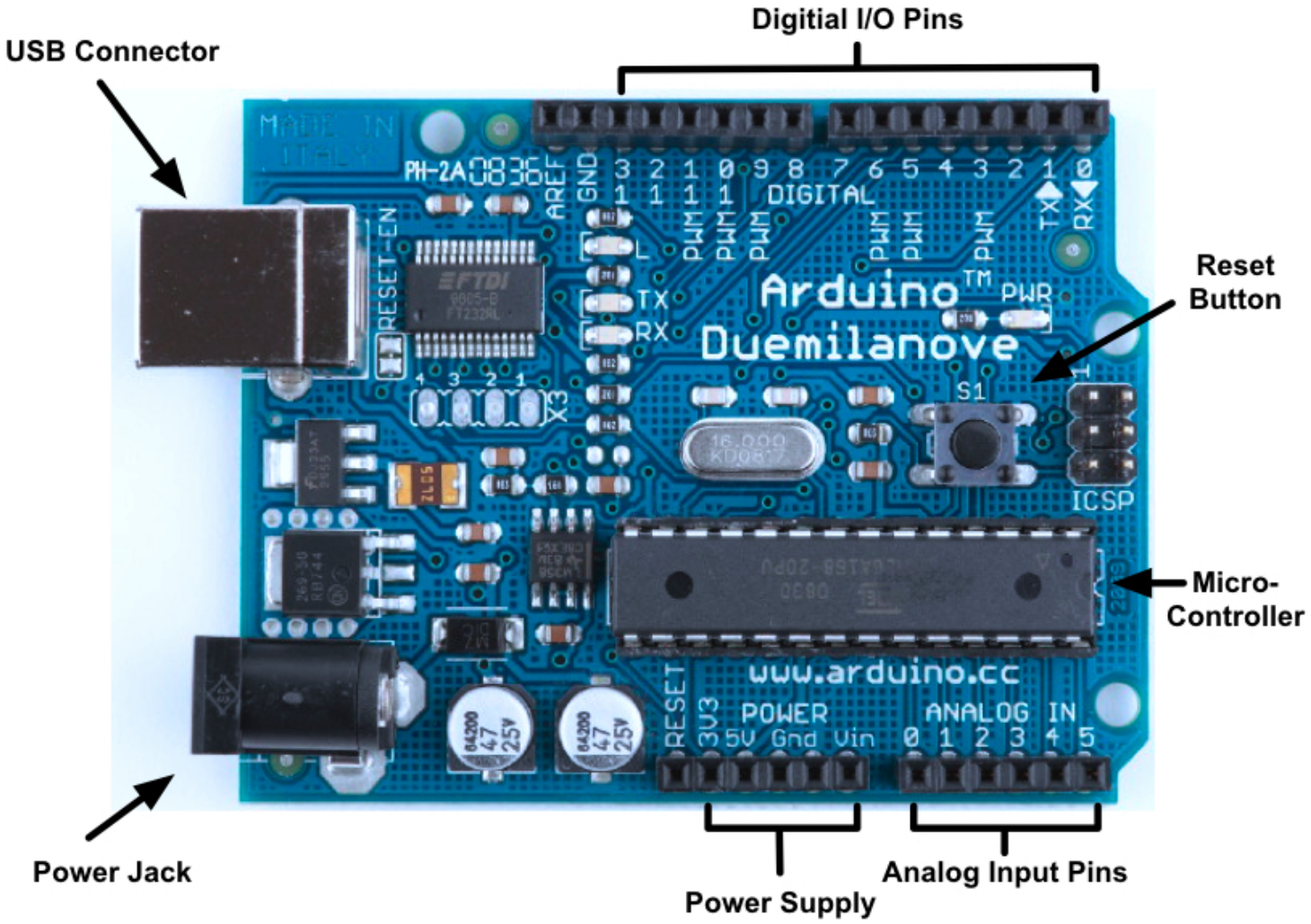




Globalcode

Principais Partes

elétron livre



Globalcode

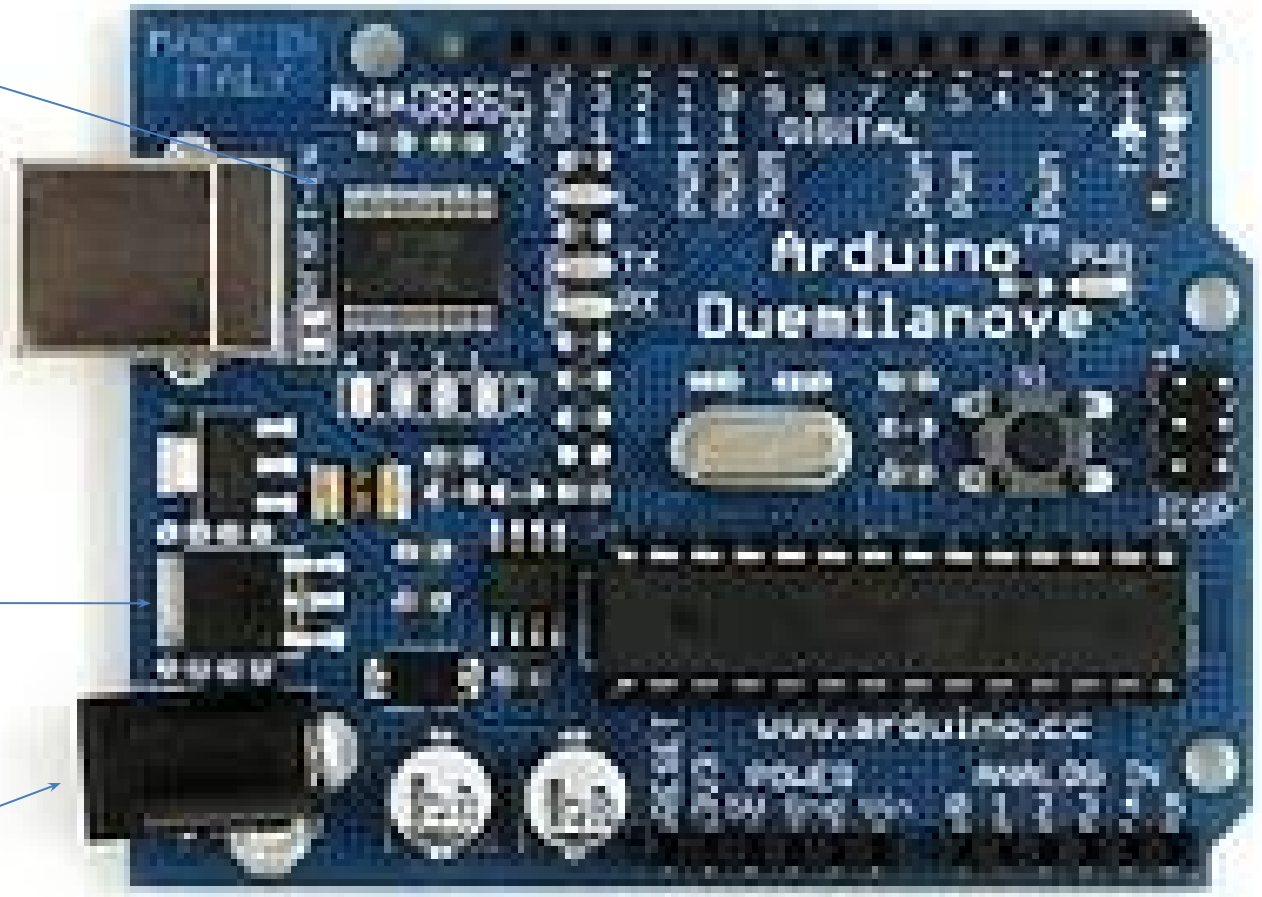
Parte Frontal

FT232RL
Conversor USB-Serial

Conector USB

Regular 7805:
Recebe até 12 volts e
regula para 5 volts

Alimentação externa:
Até 12 volts



Núcleo

elétron livre

Botão de reset

ICSP
Para gravar bootloader
ou programas/firmware

AtMega328 /168/8

Globalcode

Portas Digitais

elétron livre

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

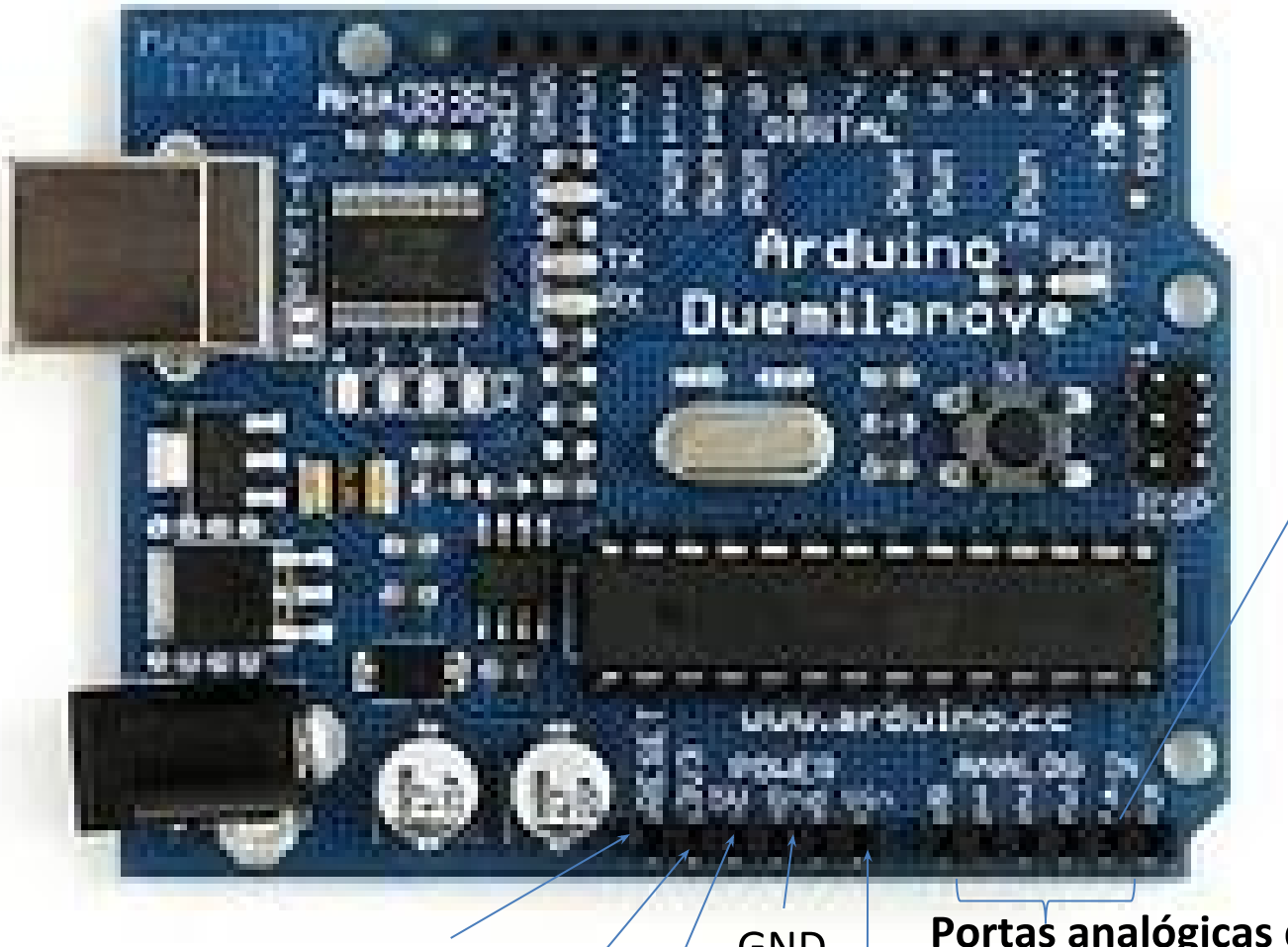
2,4,7,8,12,13 = portas digitais convêncionais

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

Globalcode

Portas Analógicas

elétron livre



Portas analógicas 4 e 5
São as portas utilizadas para conexões via I2C / TWI.

Reset

3.3 volts

5 volts

GND

VIN
Alimentação de entrada sem regulagem

Portas analógicas de 0 a 5
Podem funcionar como digitais de 14 a 19

Globalcode

Porta Digital Vs. Analógica

elétron livre

- **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

Globalcode

Porta Digital Vs. Analógica

elétron livre

- **Analógica**: valor lido é análogo a tensão.
- **Referência** de analogia é 5 volts
  - 0 volts = 0
  - 2.5 volts= 512
  - 5 volts = 1023

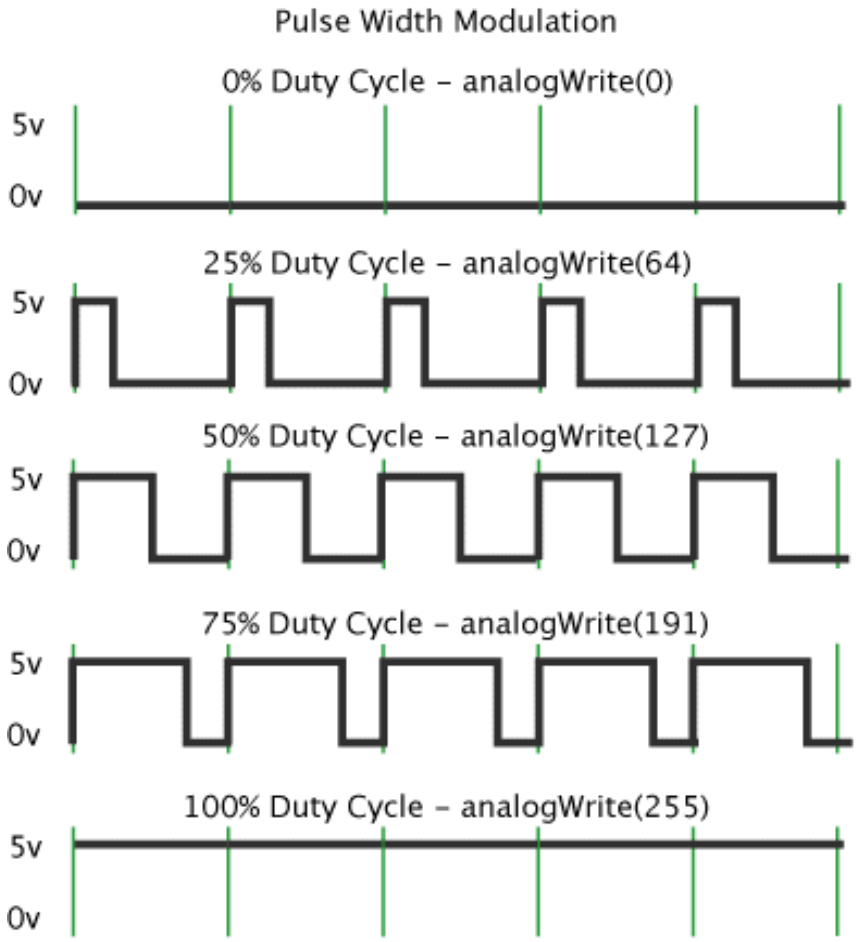
- Atmega contém **Conversor A/D** de **10 bits**: 0 a 1023

Globalcode

Porta PWM

elétron livre

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

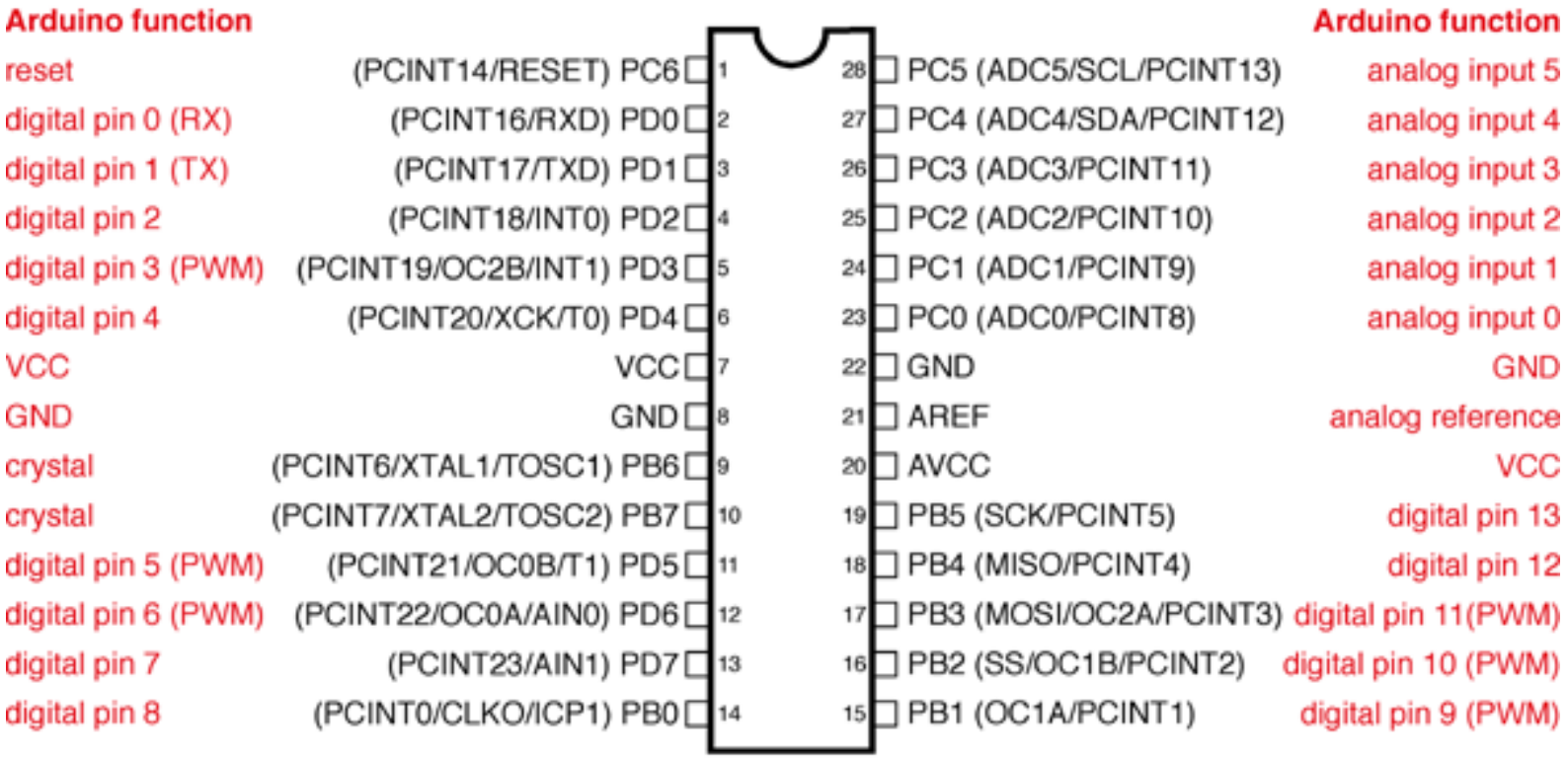


Globalcode

Por dentro do MCU

elétron livre

Atmega168 Pin Mapping

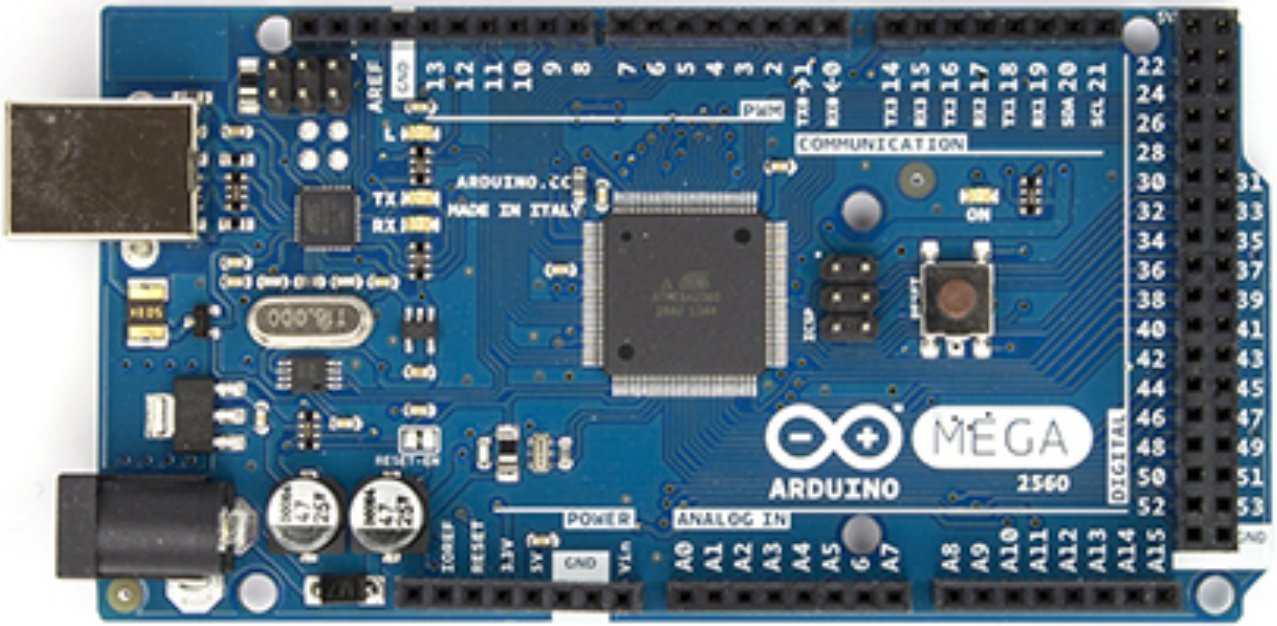


Digital Pins 11,12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Globalcode

Arduino Uno Mega 2560

elétron livre



Globalcode

Arduino Uno Mega 2560

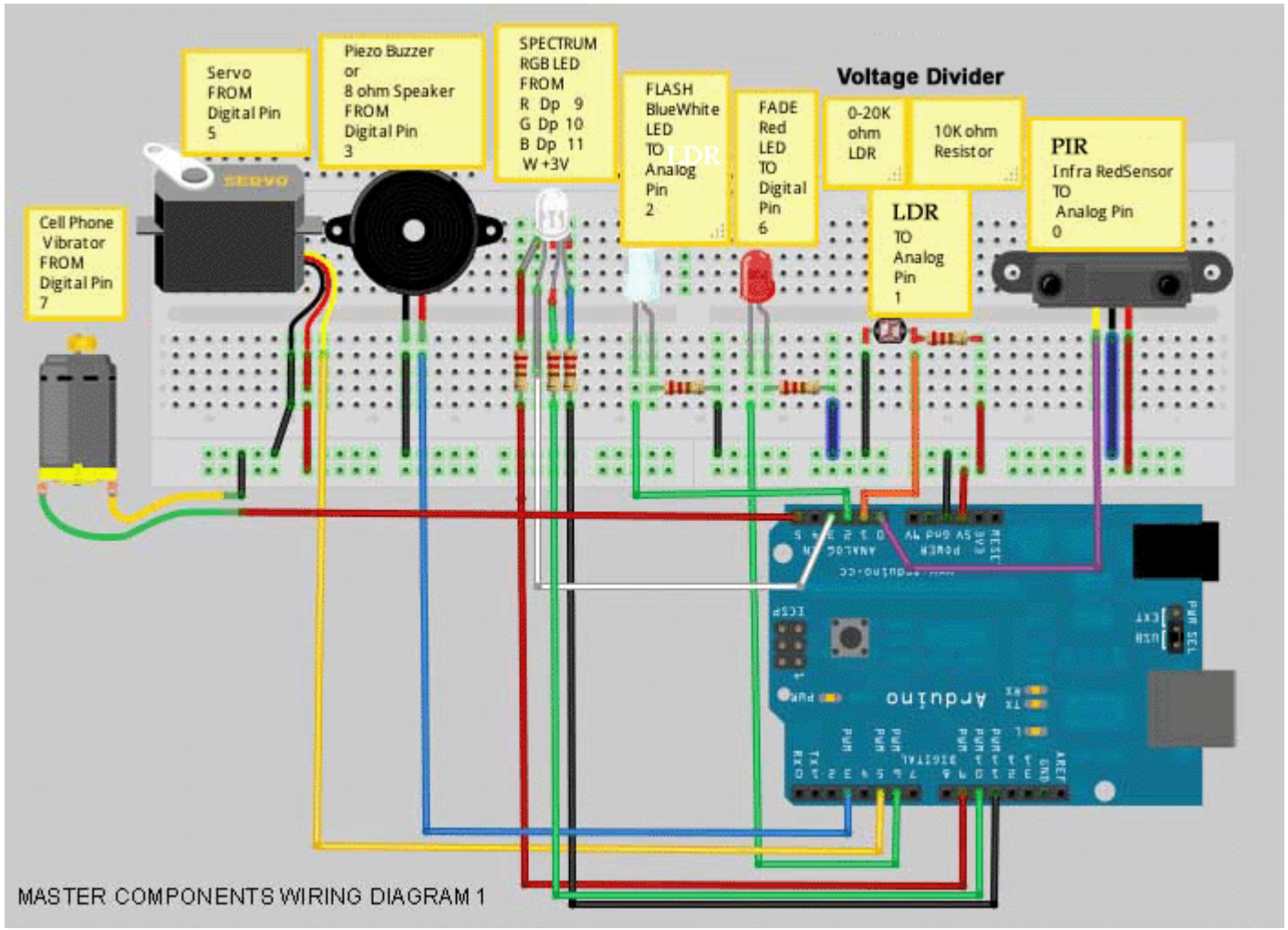
elétron livre

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

Globalcode

Componentes + Portas

elétron livre



Globalcode

Shields: arquitetura modular

elétron livre

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

Globalcode

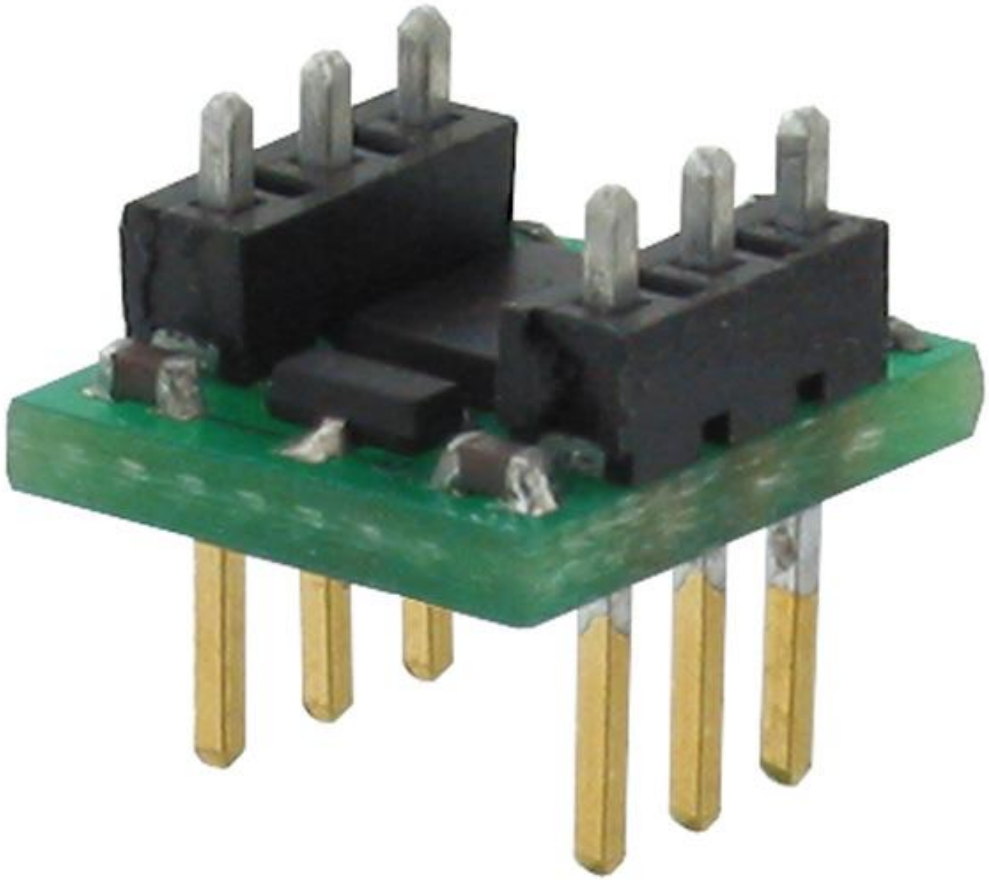
Ping – Sensor de distância

elétron livre

Globalcode

Bússola

elétron livre



Globalcode

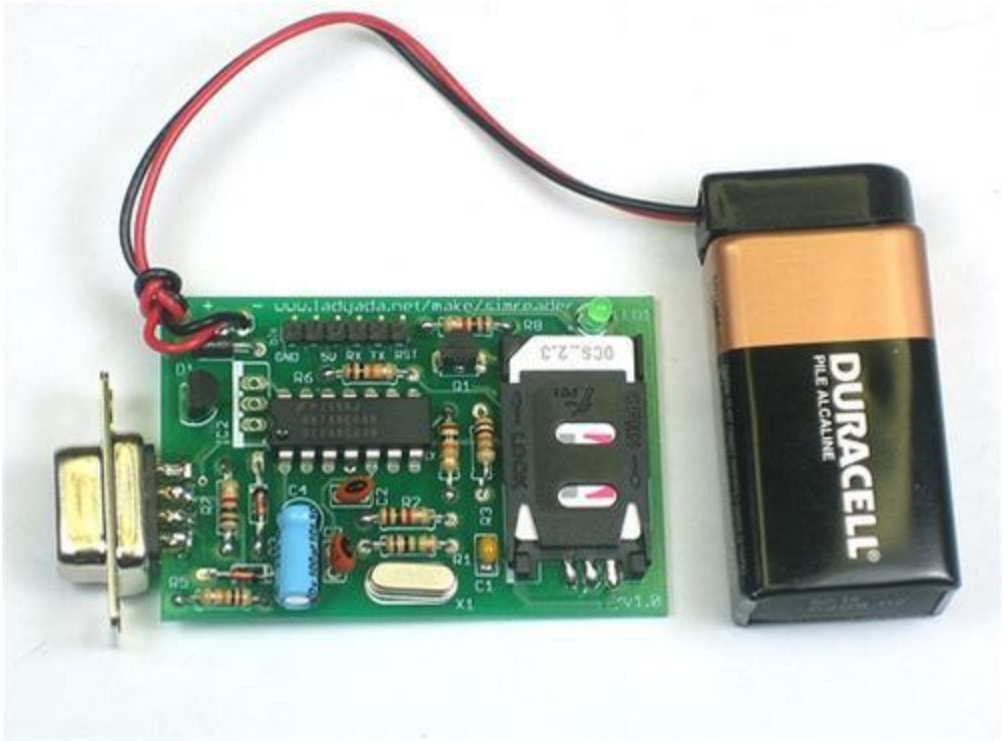
Shield LCD

elétron livre

Globalcode

SimReader

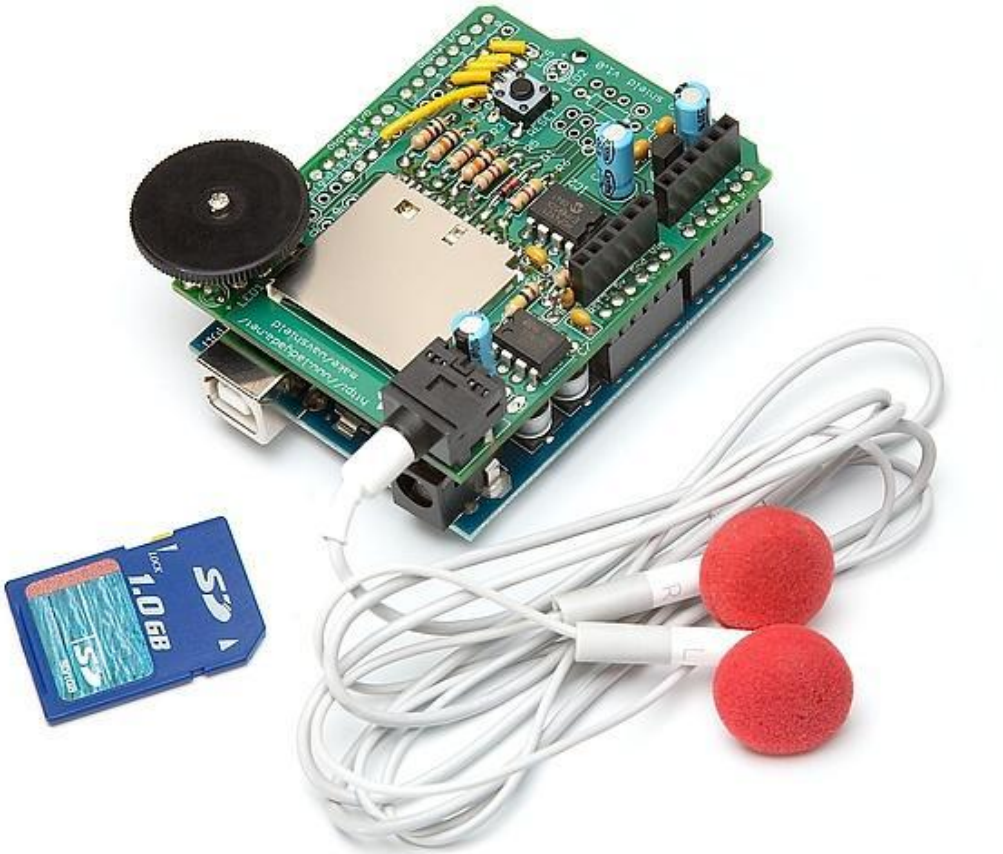
elétron livre



Globalcode

Mp3 Player Shield

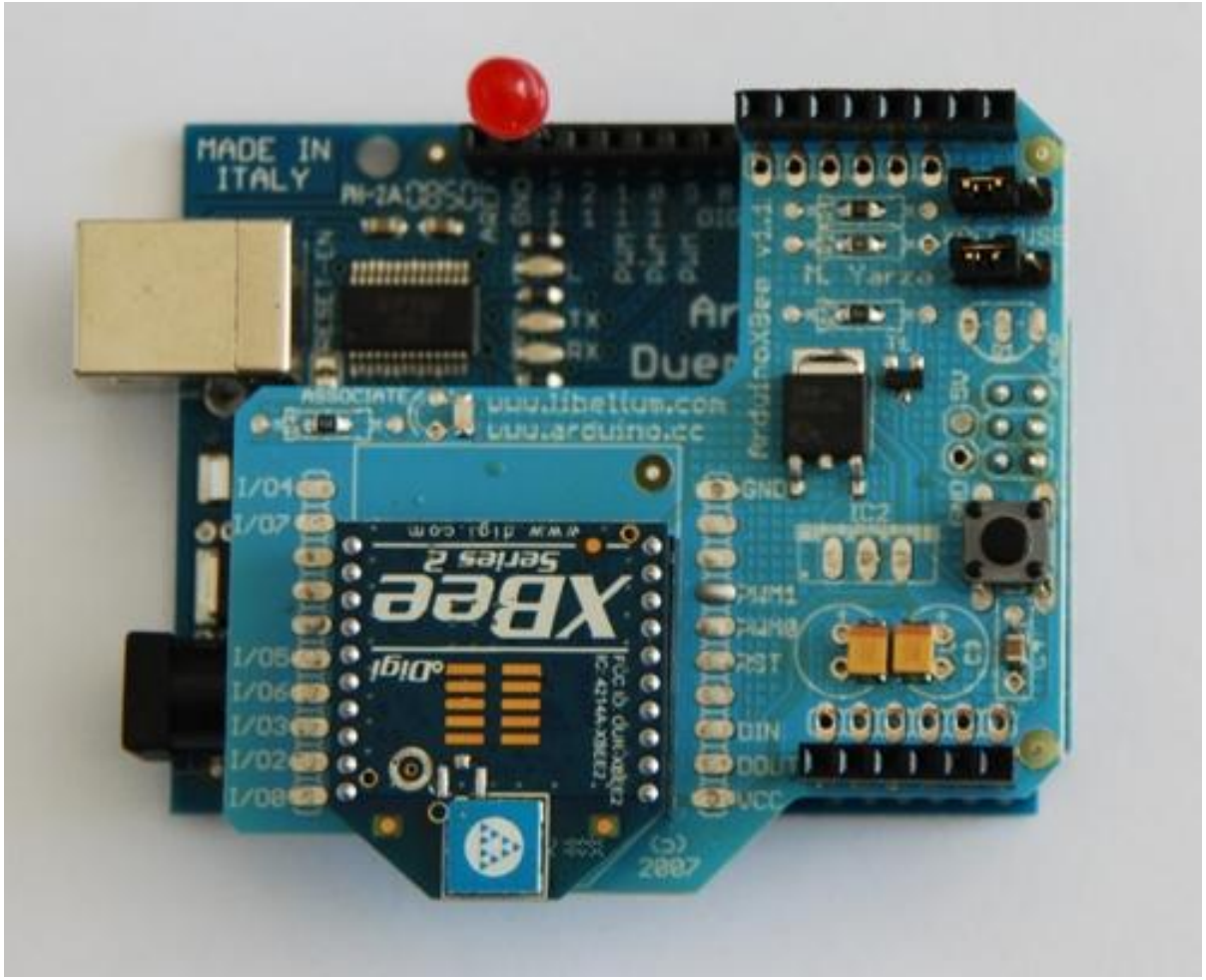
elétron livre



Globalcode

ZigBee - xBee

elétron livre

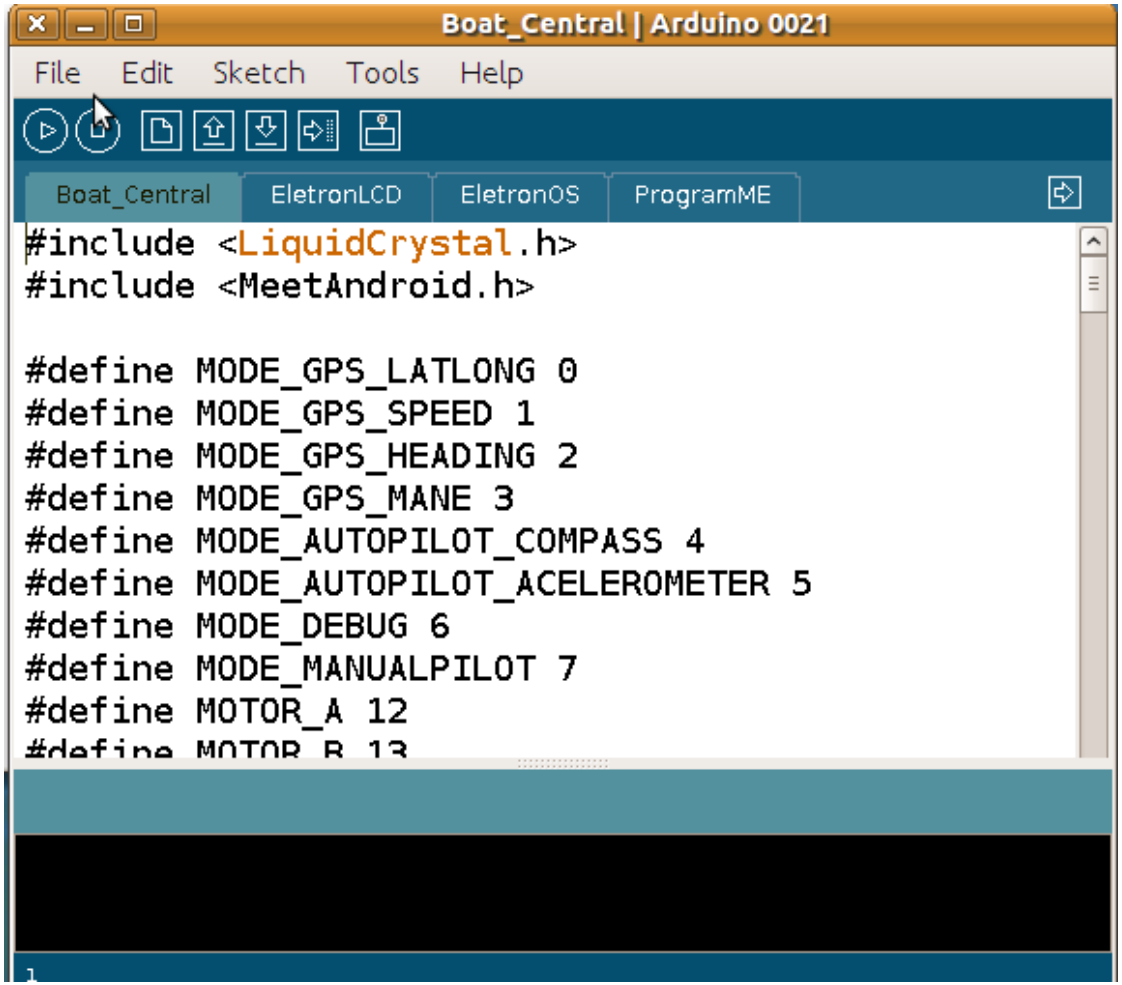


Globalcode

Arduino

elétron livre

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

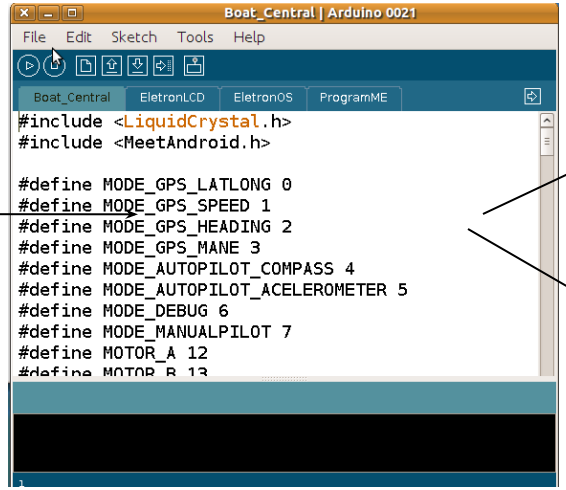


Globalcode

Arduino

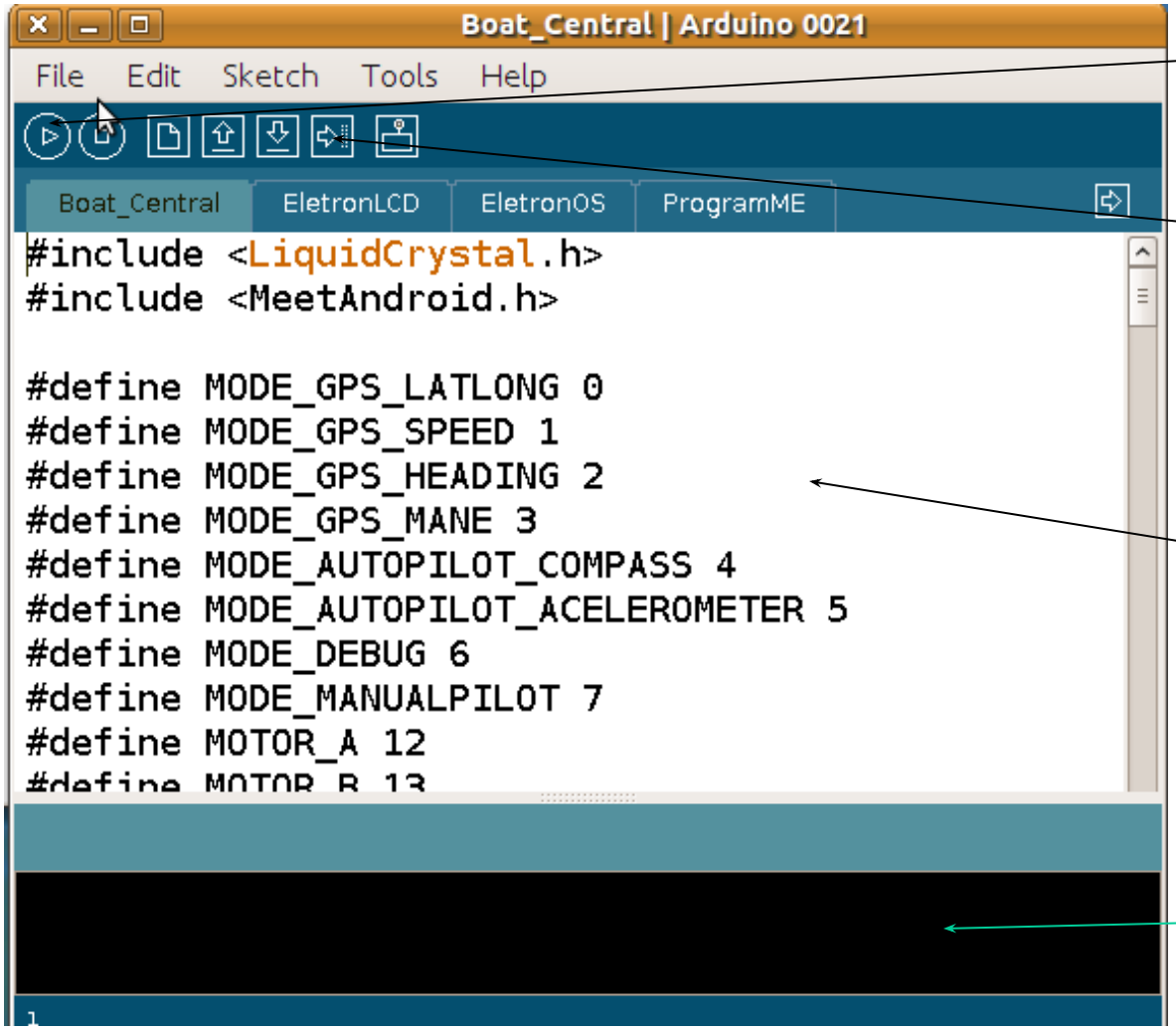
- Utilizaremos para escrever todos os nossos programas:

```
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);

}
```



Arduino

elétron livre



Botão para compilar

Botão para enviar programa para a placa

Área para digitação do código

Mensagens de sucesso ou erro

Globalcode

Programando para Arduino

elétron livre

- 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!

Globalcode

A linguagem C

elétron livre

- 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

elétron livre

- Temos que obrigatoriamente programar dois métodos:

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

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

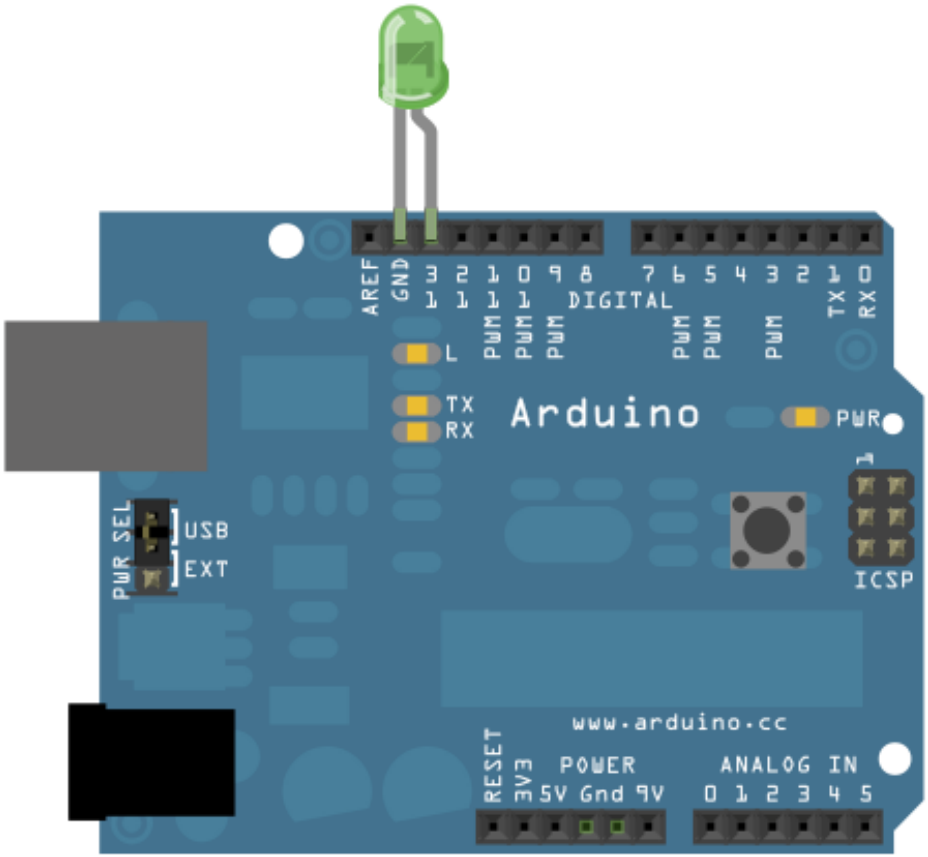
Globalcode

Portas digitais e analógicas

elétron livre

- 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”

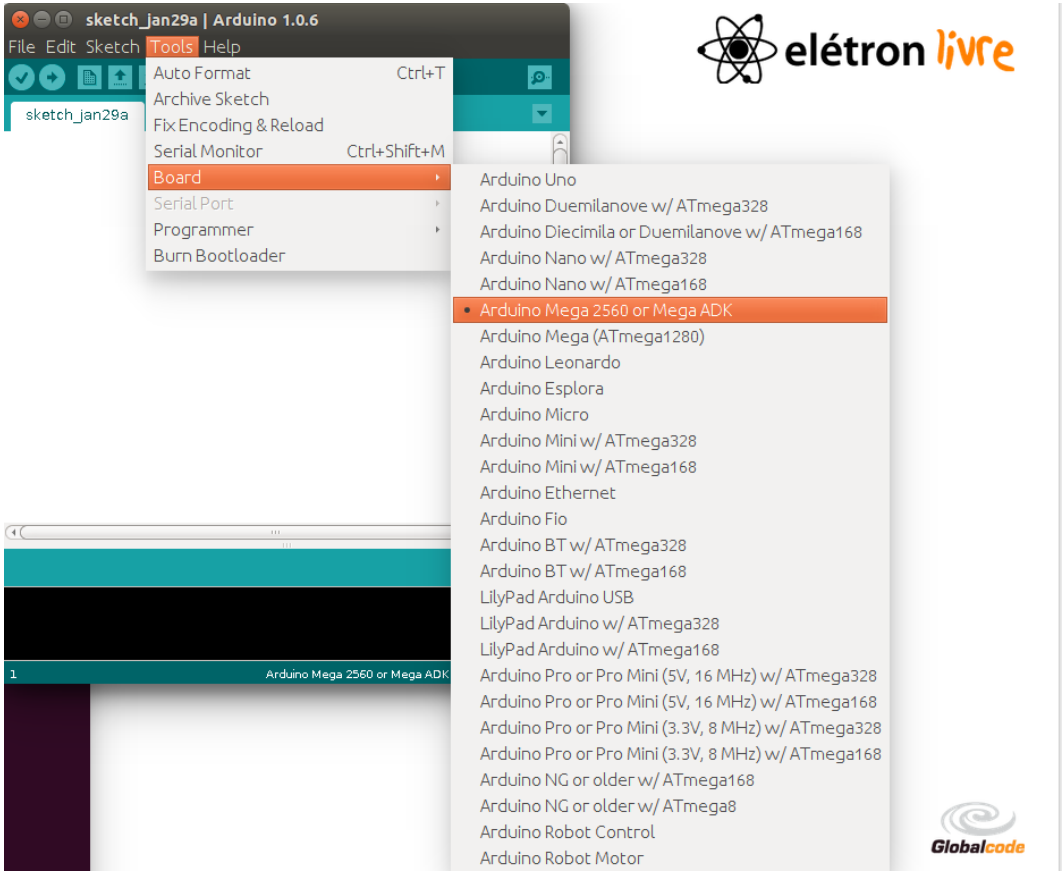


Esta conexão é bem simples somente para efeito de teste para piscar o led.

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:

elétron livre

```
void setup() {
  pinMode(13, OUTPUT);
}

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

Globalcode

Conclusões

elétron livre

- 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!

Globalcode

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);
digitalWrite(13, 0);
delay(500);
digitalWrite(4, false);
digitalWrite(13, true);
delay(500);
```

Número da porta: 4 ou 13

Ligado ou desligado, 0 ou 1, true ou false

Portas Analógicas

elétron livre

- 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

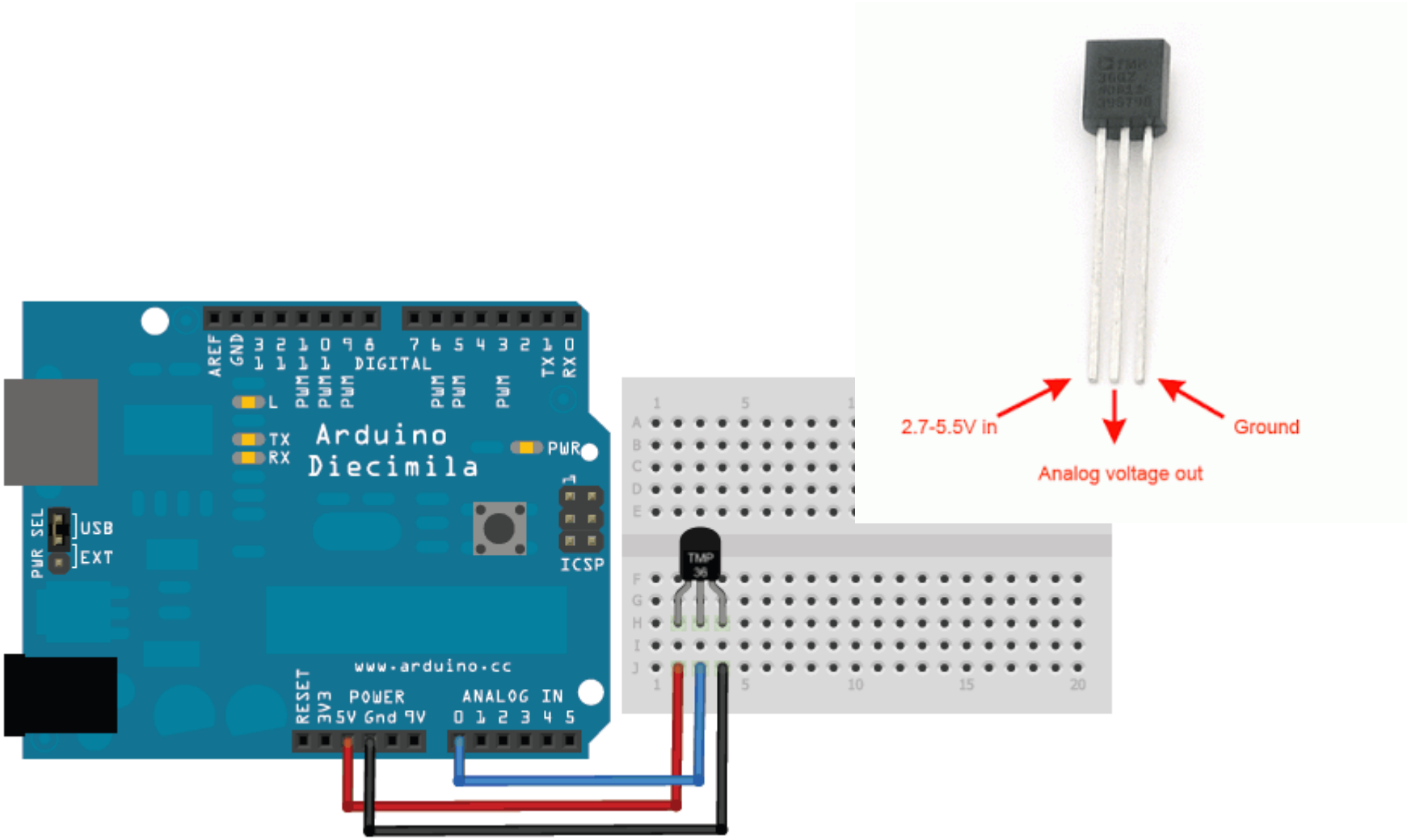
elétron livre

- 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;

Globalcode

Sensor Temperatura LM35

elétron livre



Globalcode

Laboratório

elétron livre

- 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);
}
```

Globalcode

Laboratório

elétron livre

- 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));
```

Globalcode

Laboratório

elétron livre

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

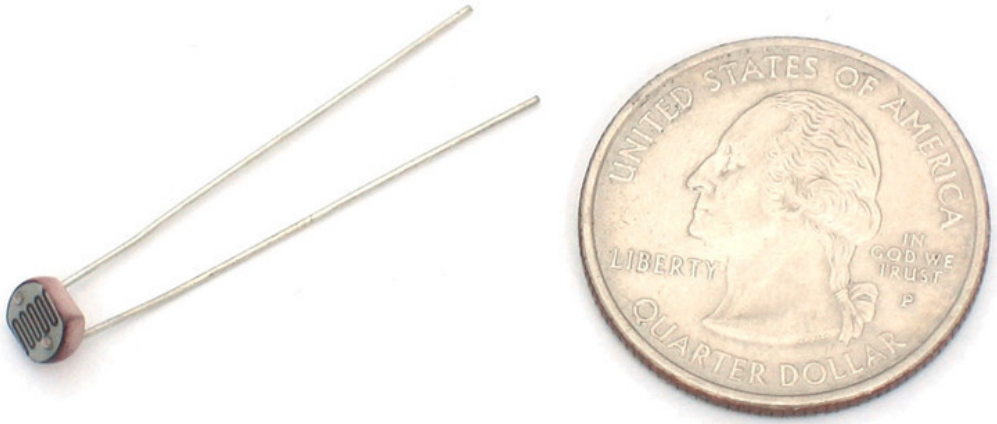
```
const float CELSIUS_BASE = 0.48875855327468230694037145 65;

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

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

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

Globalcode

LDR

elétron livre

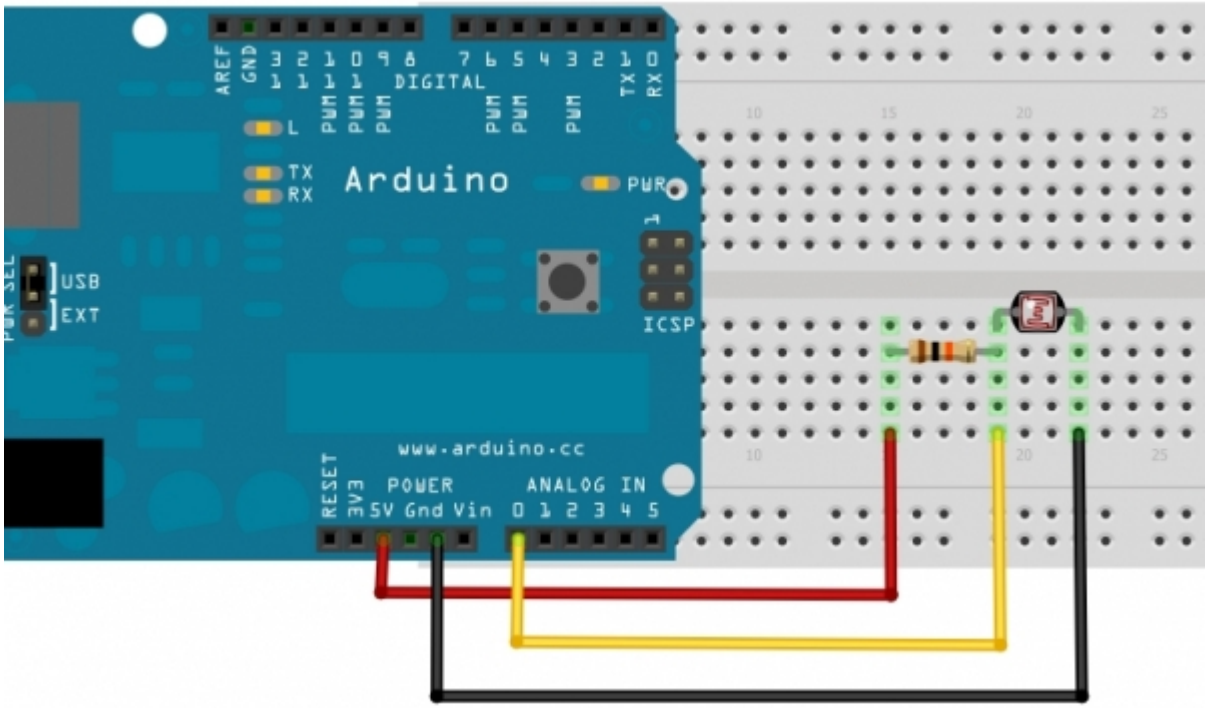
- Light Dependent Resistor
- Usado como sensor de luz
- Quanto menos luz, maior o valor lido!



Globalcode

LDR na Protoboard

elétron livre



Globalcode

Laboratório

elétron livre

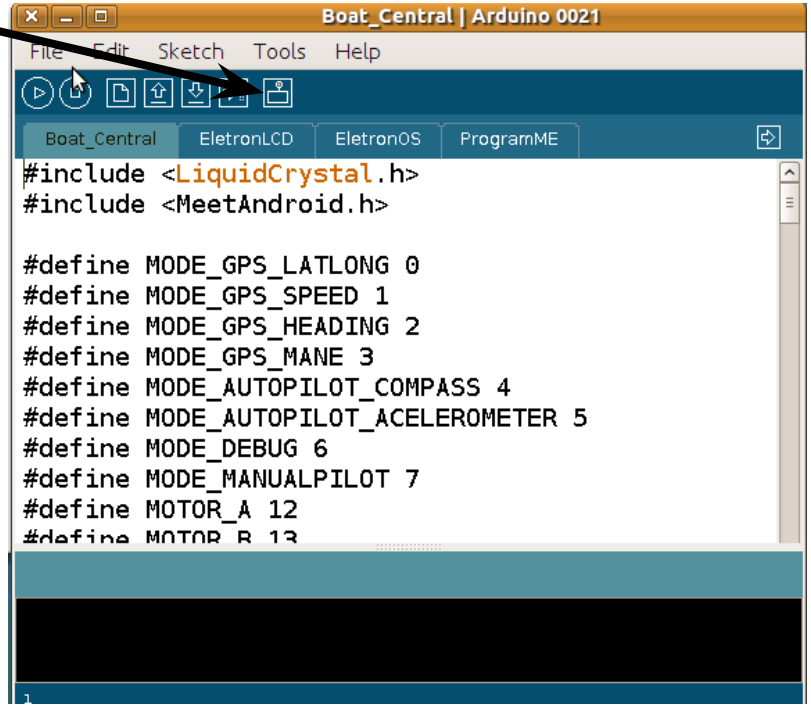
- Agora vamos digitar o código completo e transferir para placa:

```
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);
}
```

Globalcode

Laboratório

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



Exemplo “luz ambiente”

elétron livre

```
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);
}
```

Globalcode

Exemplo “luz ambiente”

elétron livre

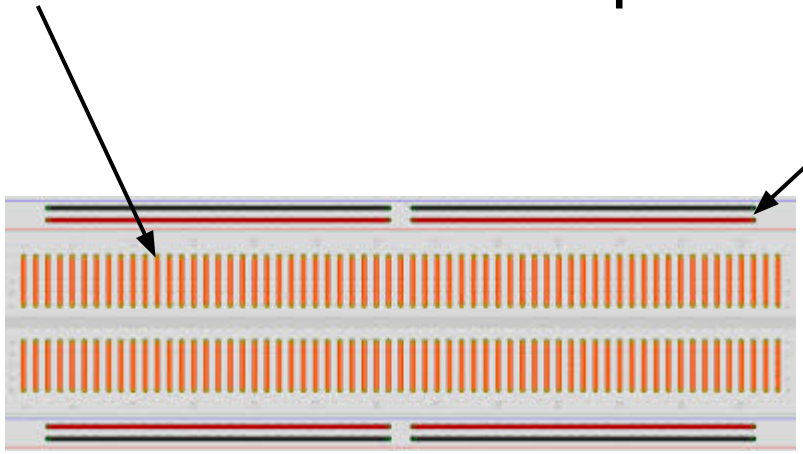
```
void setup() {
  Serial.begin(9600);
}

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

Globalcode

Conhecendo a protoboard

elétron livre

- 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:

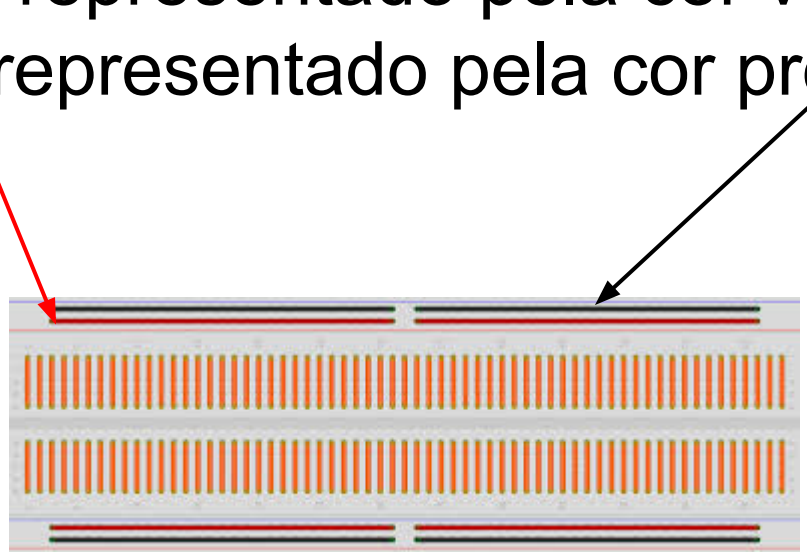


Globalcode

Energizando a protoboard

elétron livre

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

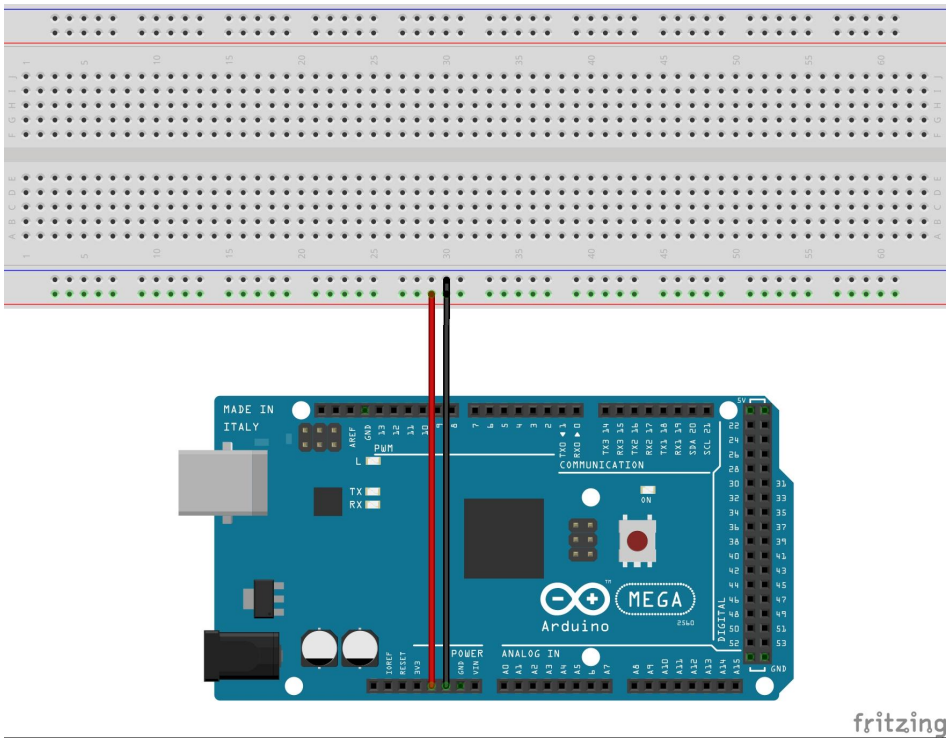


Globalcode

Energizando a protoboard

elétron livre

- 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:

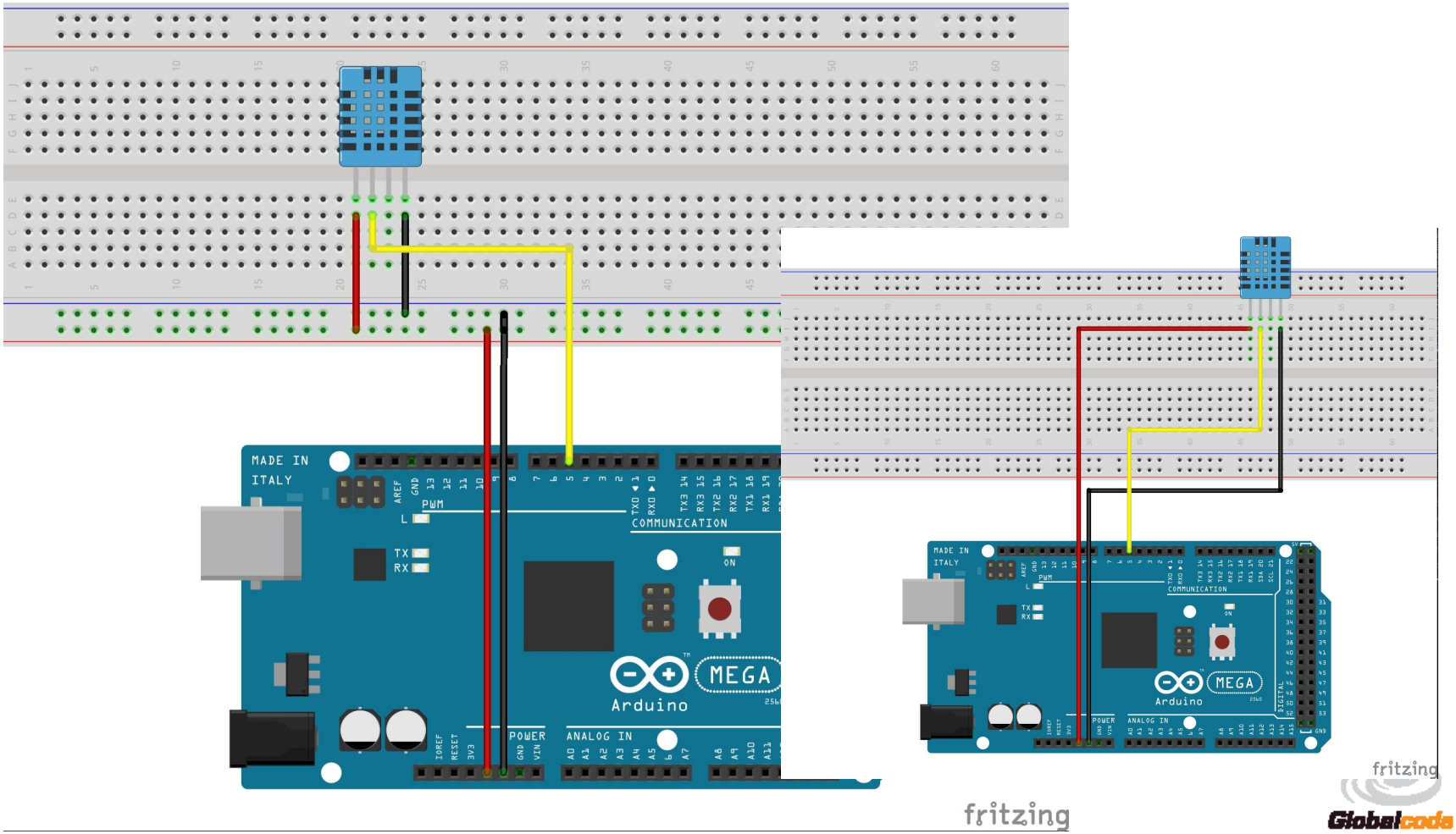


Globalcode

Ligando Componentes

elétron livre

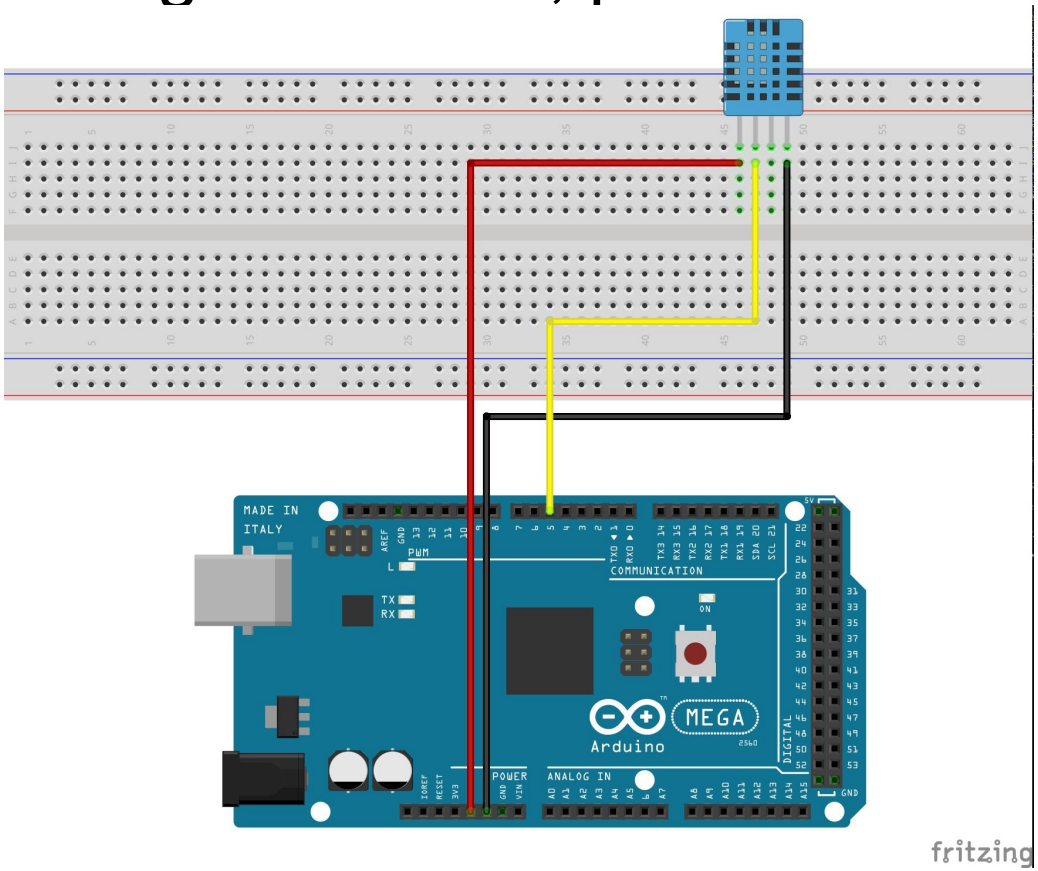
- Agora podemos ligar os componentes desta forma:



Globalcode

Ligando Componentes

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



Sensor DHT11

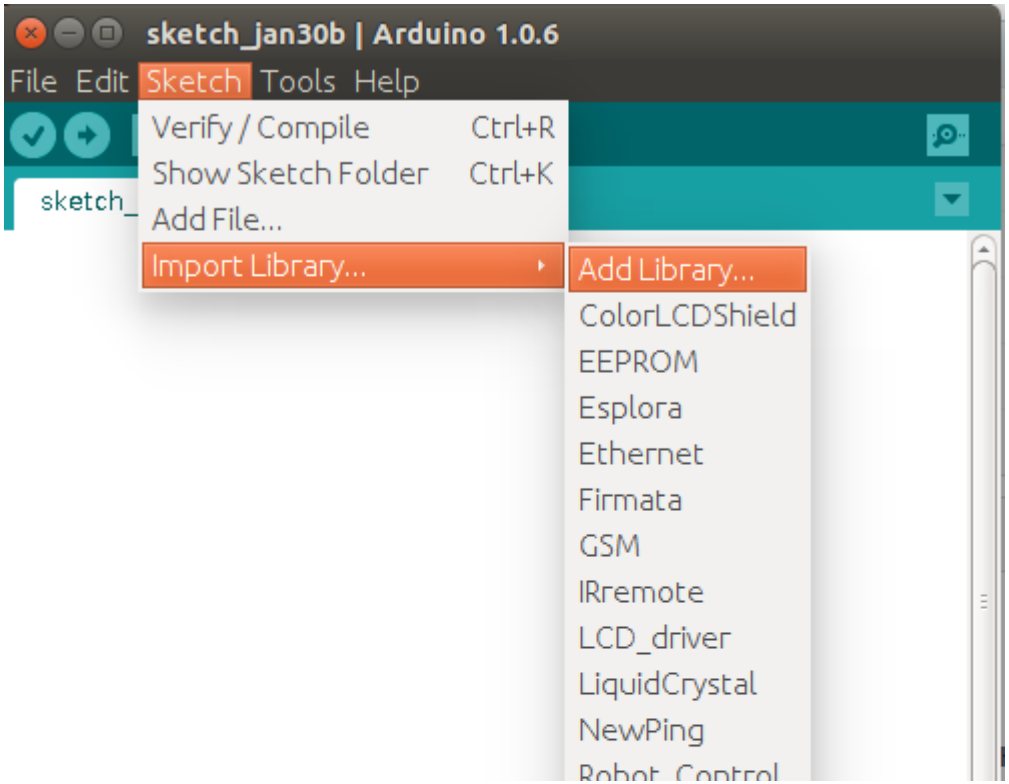
elétron livre

- 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!

Globalcode

Adicionando bibliotecas...

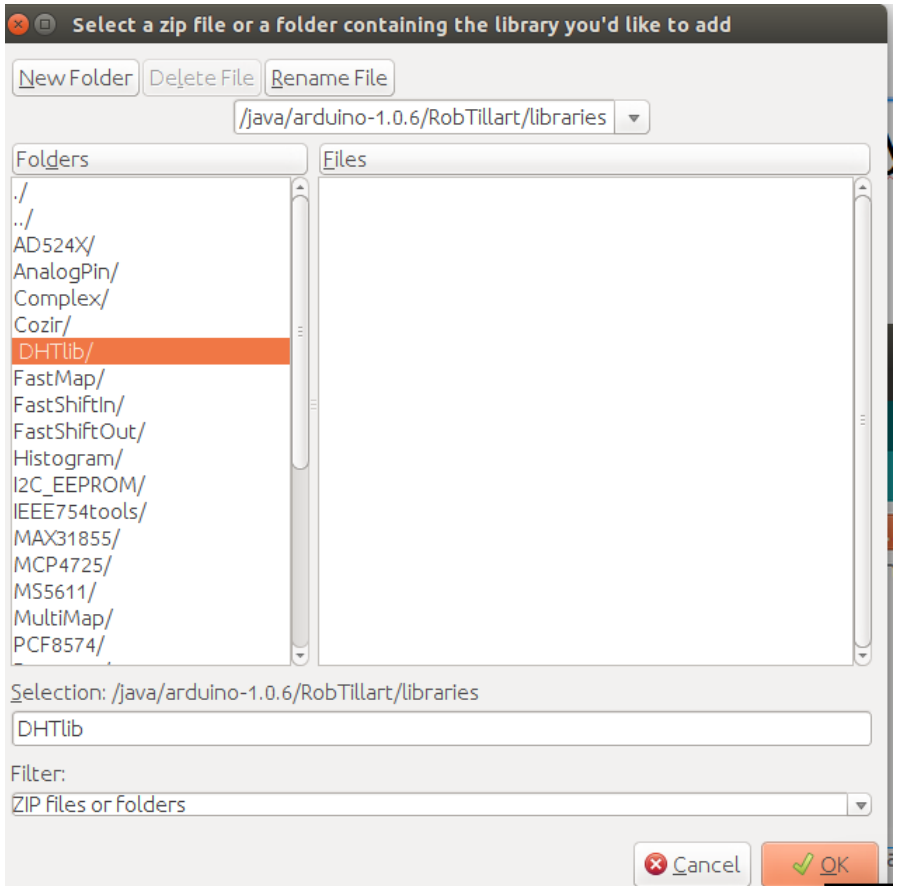
- No menu Scketch -> Import Library -> Add Library



Adicionando bibliotecas...

elétron livre

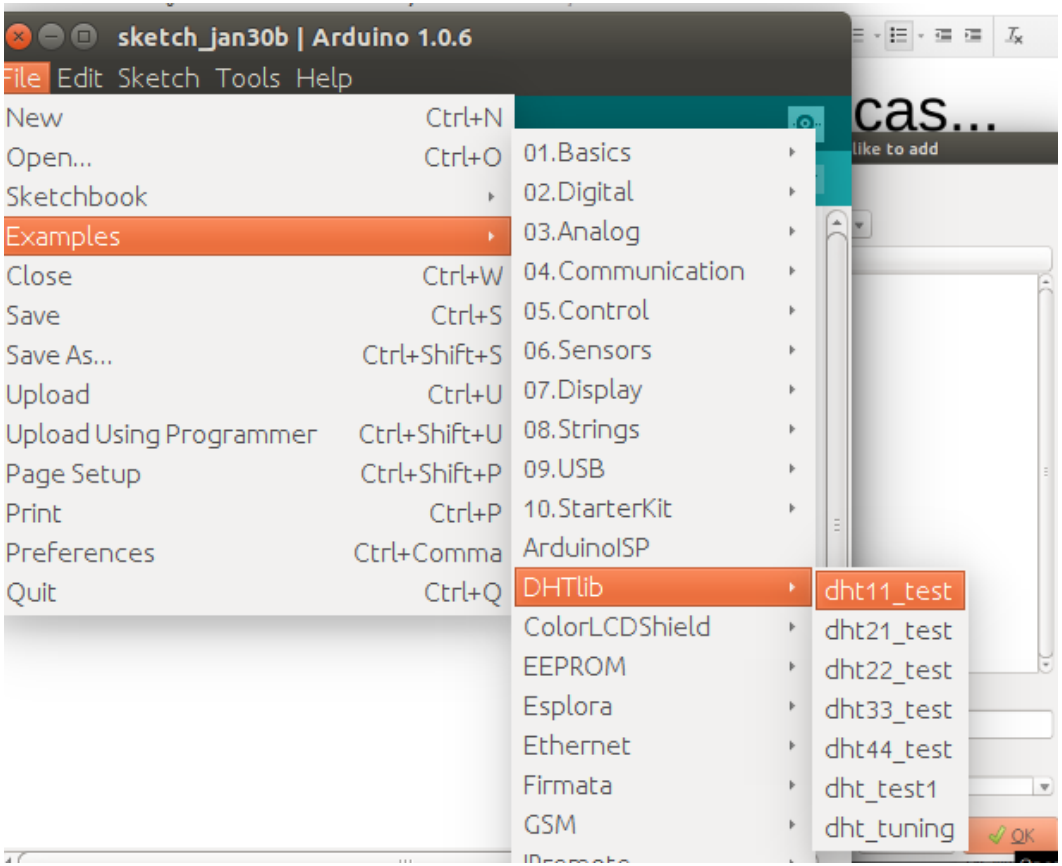
- Escolha o local onde copiou a biblioteca:



Globalcode

Programa Exemplo

- Em seguida, clique no menu File -> Examples -> DHTLib - > dht11_test

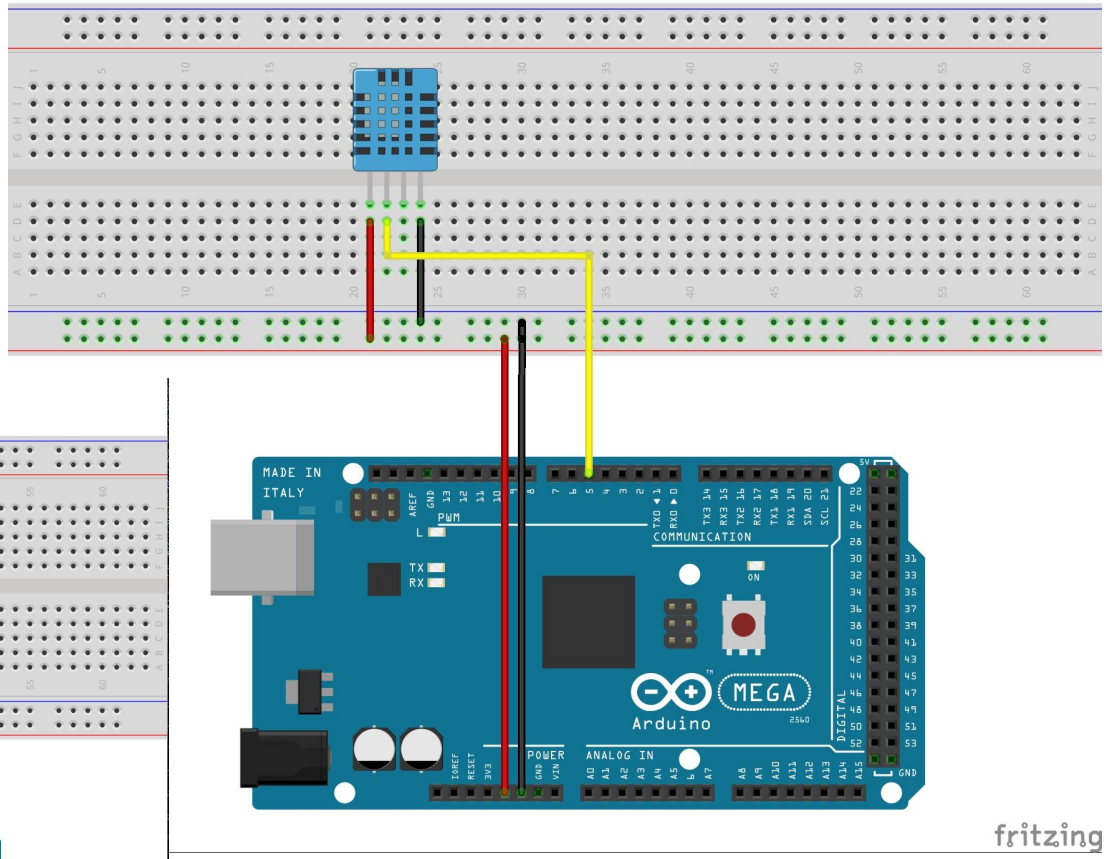


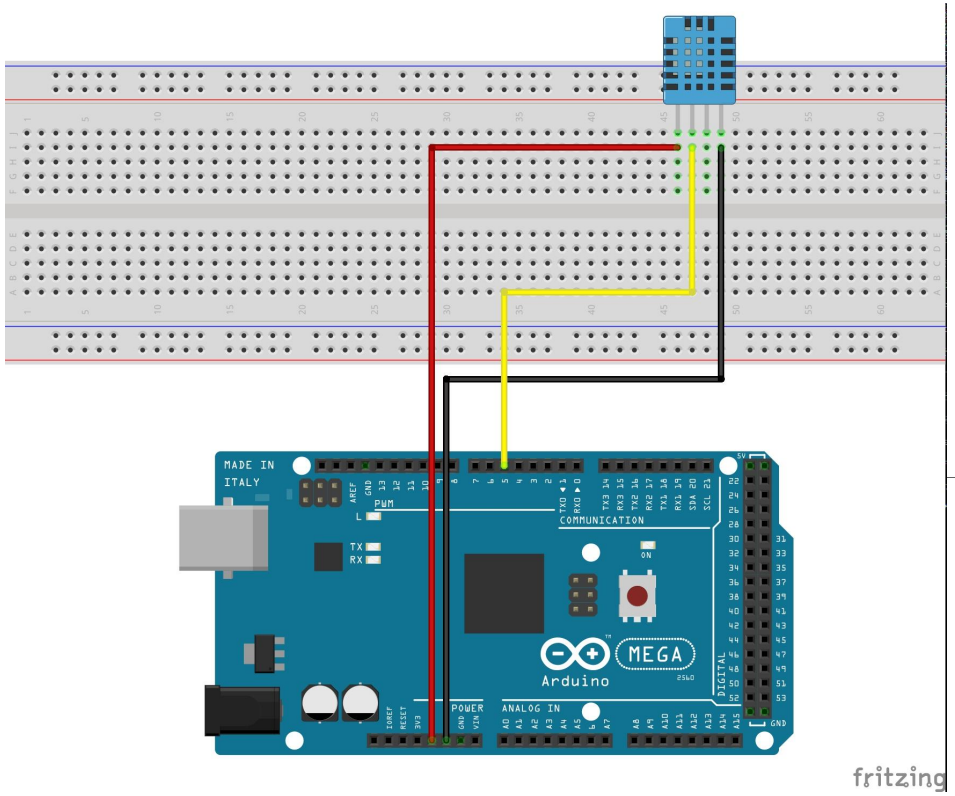
Programa Exemplo

Note que há uma definição chamada DHT11_PIN que está configurada para porta digital 5

Faça Conexão - CUIDADO!

elétron livre





Globalcode

Upload e Teste

elétron livre

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

Globalcode

Sensor HC-SR04

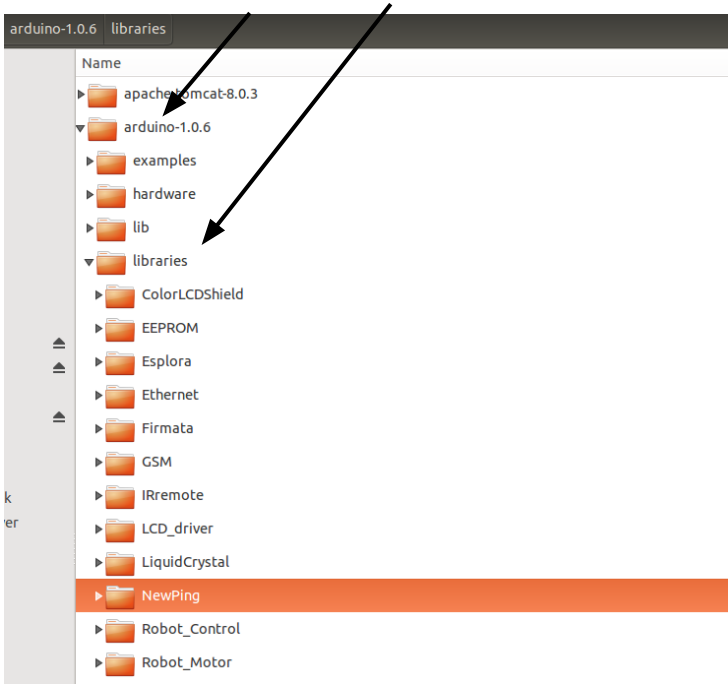
elétron livre

- 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 cálculado com base na velocidade do som!
- Sensor que requer tempo real;

Globalcode

Sensor HC-SR04

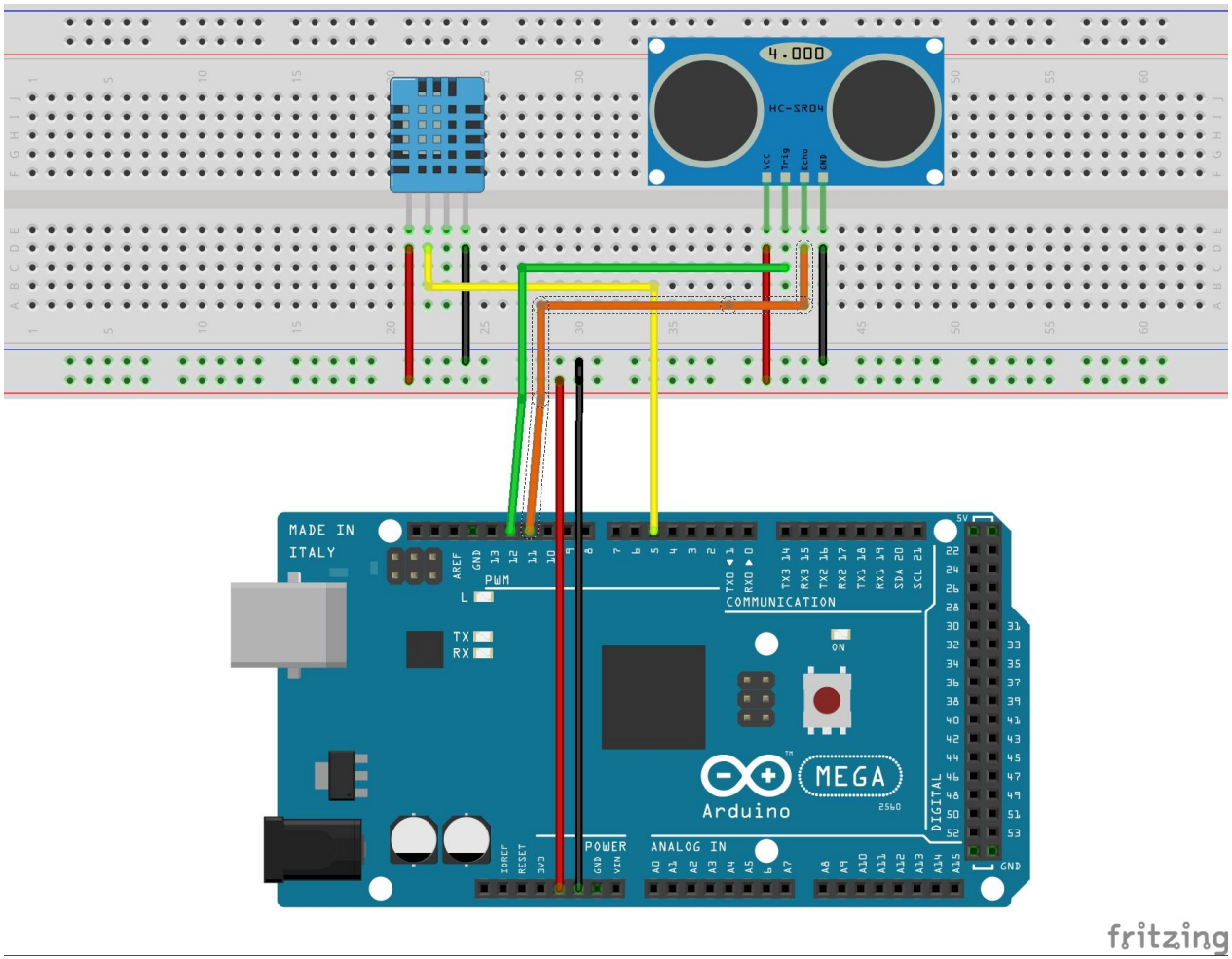
- A comunidade criou uma biblioteca para ele chamada de NewPing;
- Instale a biblioteca conforme apresentado anteriormente ou simplesmente copiando o diretório NewPing para o diretório: arduino/libraries



HC-SR04: fiação

elétron livre

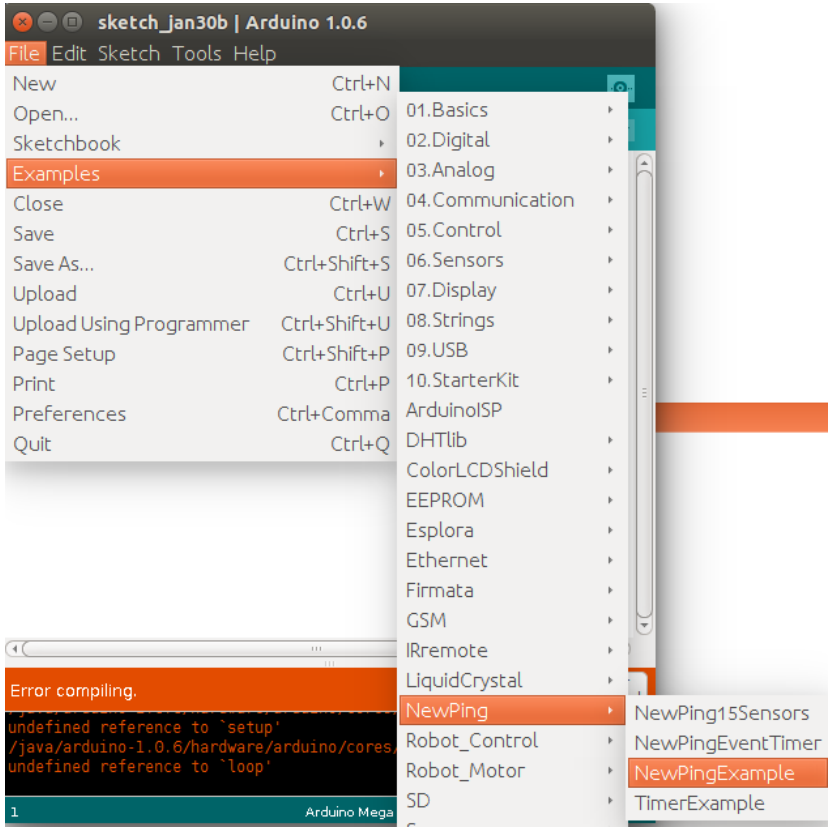
- Trabalha com 4 fios: 5volts / GND / Trigger / Echo



Globalcode

Sensor HC-SR04

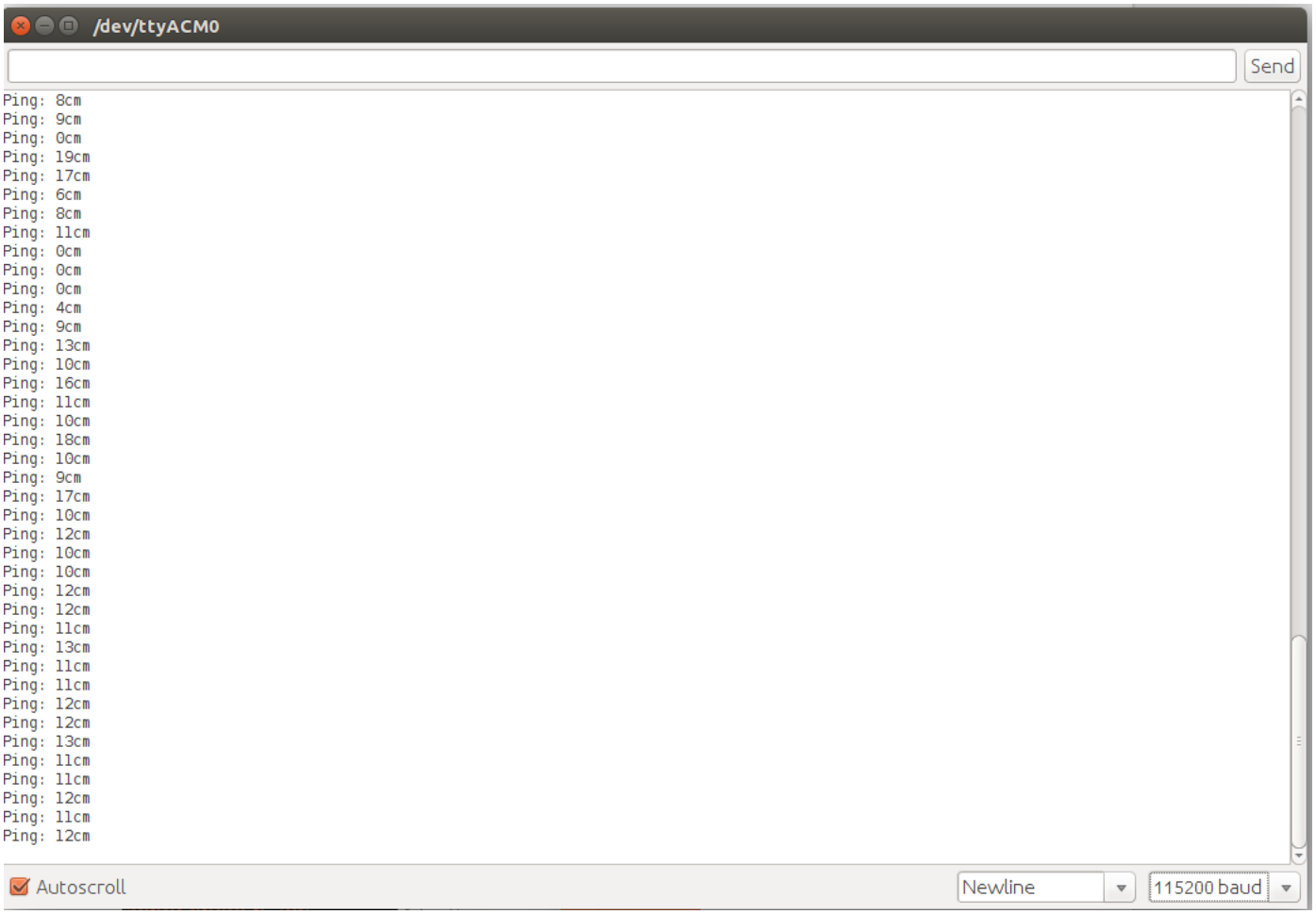
- Ele também contém exemplos, menu:
  File -> Examples - NewPing -> NewPingExample



Sensor HC-SR04: testes!

elétron livre

- O resultado esperado é:



Globalcode

Receptor Infravermelho

elétron livre

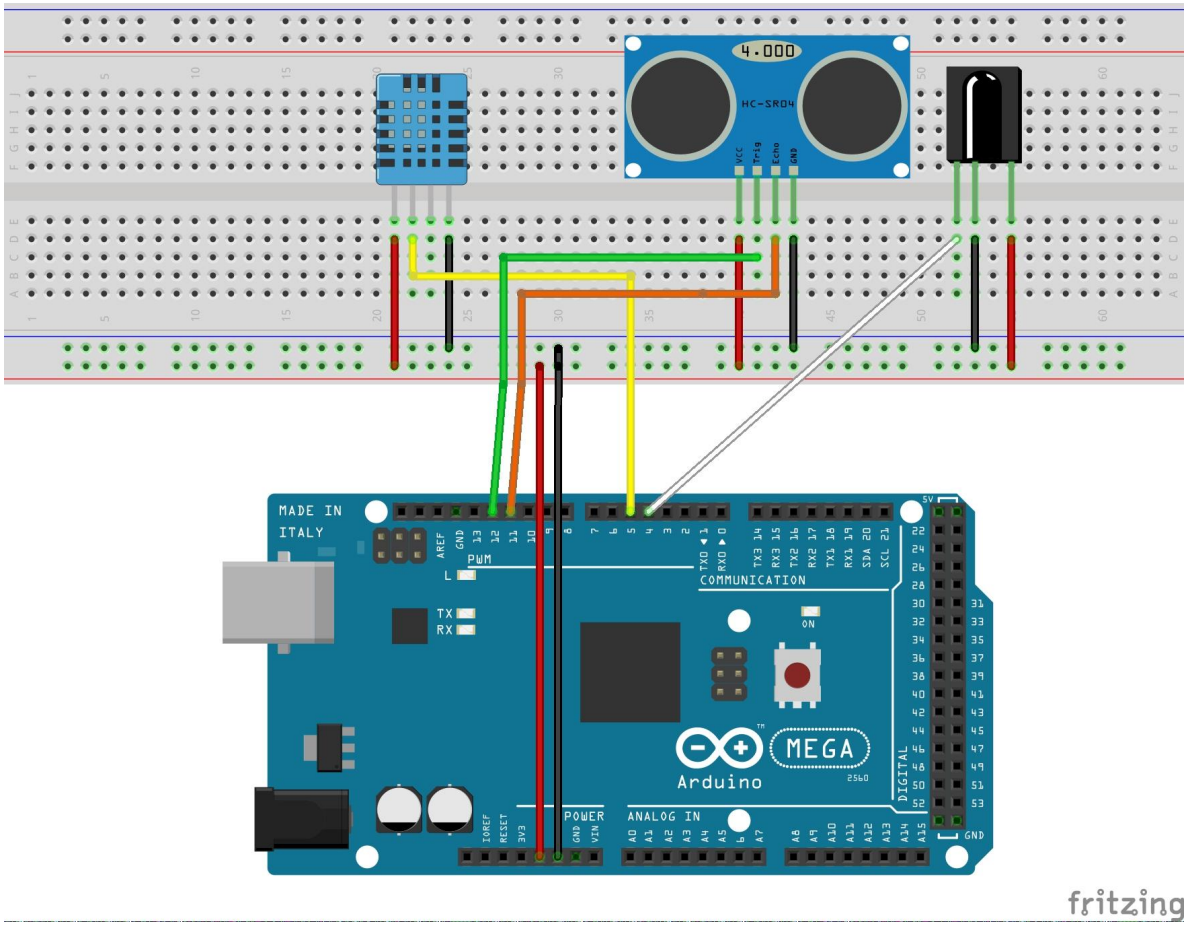
- Infravermelho é algo muito funcional, simples, barato e amplamente suportado no Arduino;
- Existe uma excelente biblioteca para suporte a IR chamada IRRemote que devemos instalar;
- O receptor do IR tem 3 fios: 5v, GND e dados;
- O emissor de IR é um LED convencional;
- Vejamos a seguir a conexão de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

Globalcode

Receptor Infravermelho

elétron livre

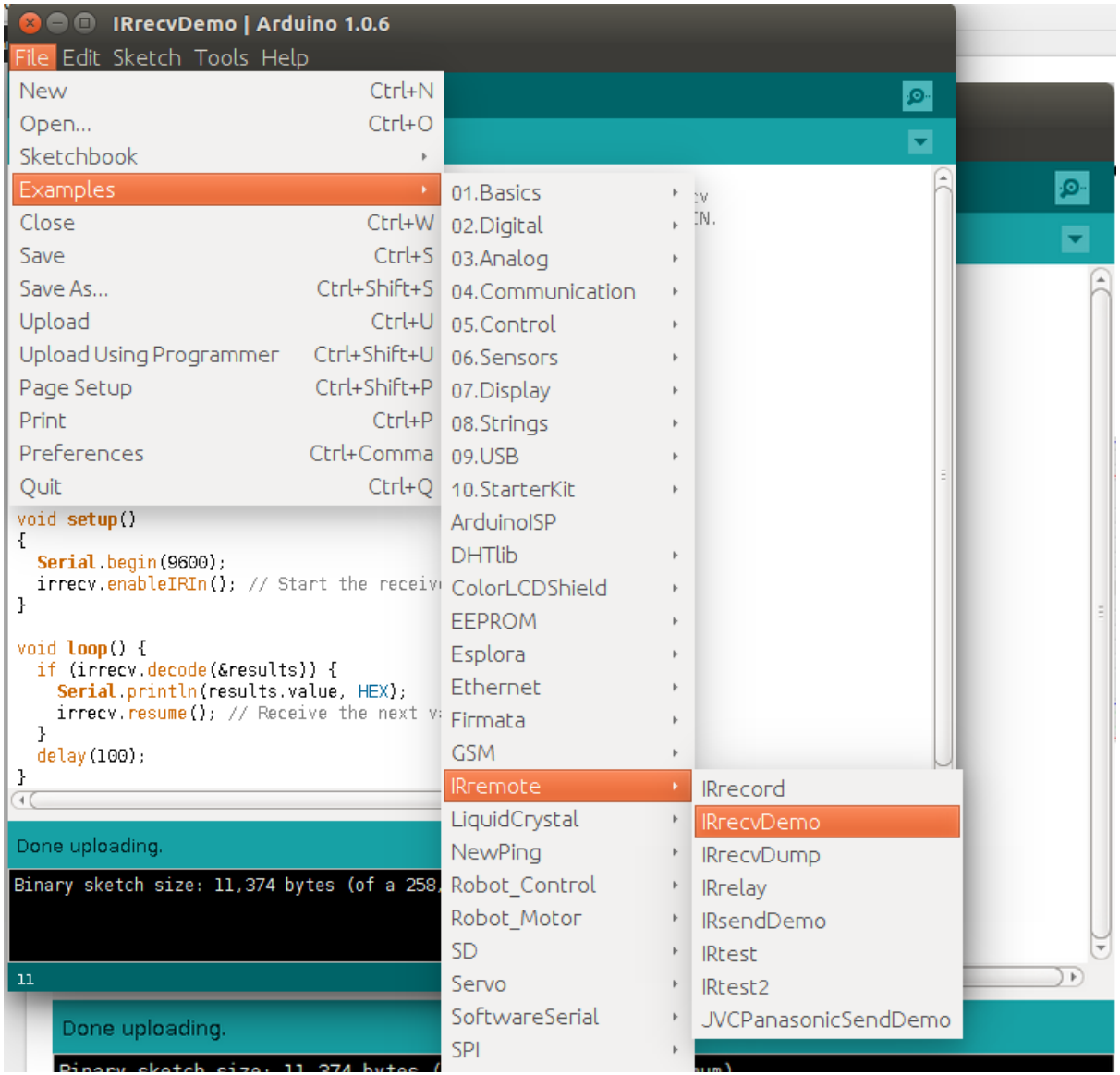
- Estamos ligando os dados no pino 4:



Globalcode

IR: Programa receptor

elétron livre



Globalcode

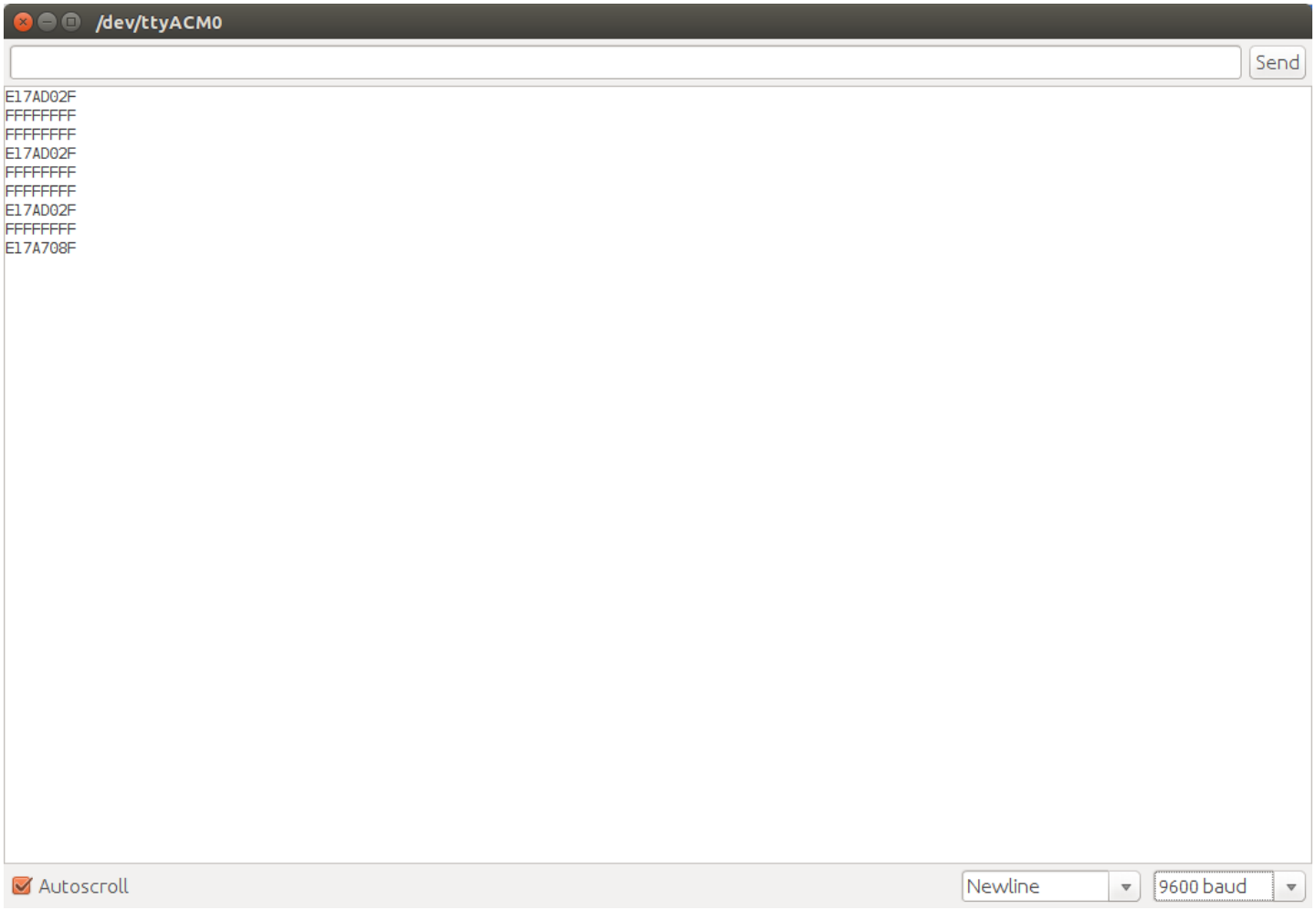
IR Código Exemplo

elétron livre

Alterar para a porta escolhida!

Globalcode

IR Resultado Captura



Relay / Relé

elétron livre

- 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";

Globalcode

Arduino + Relé

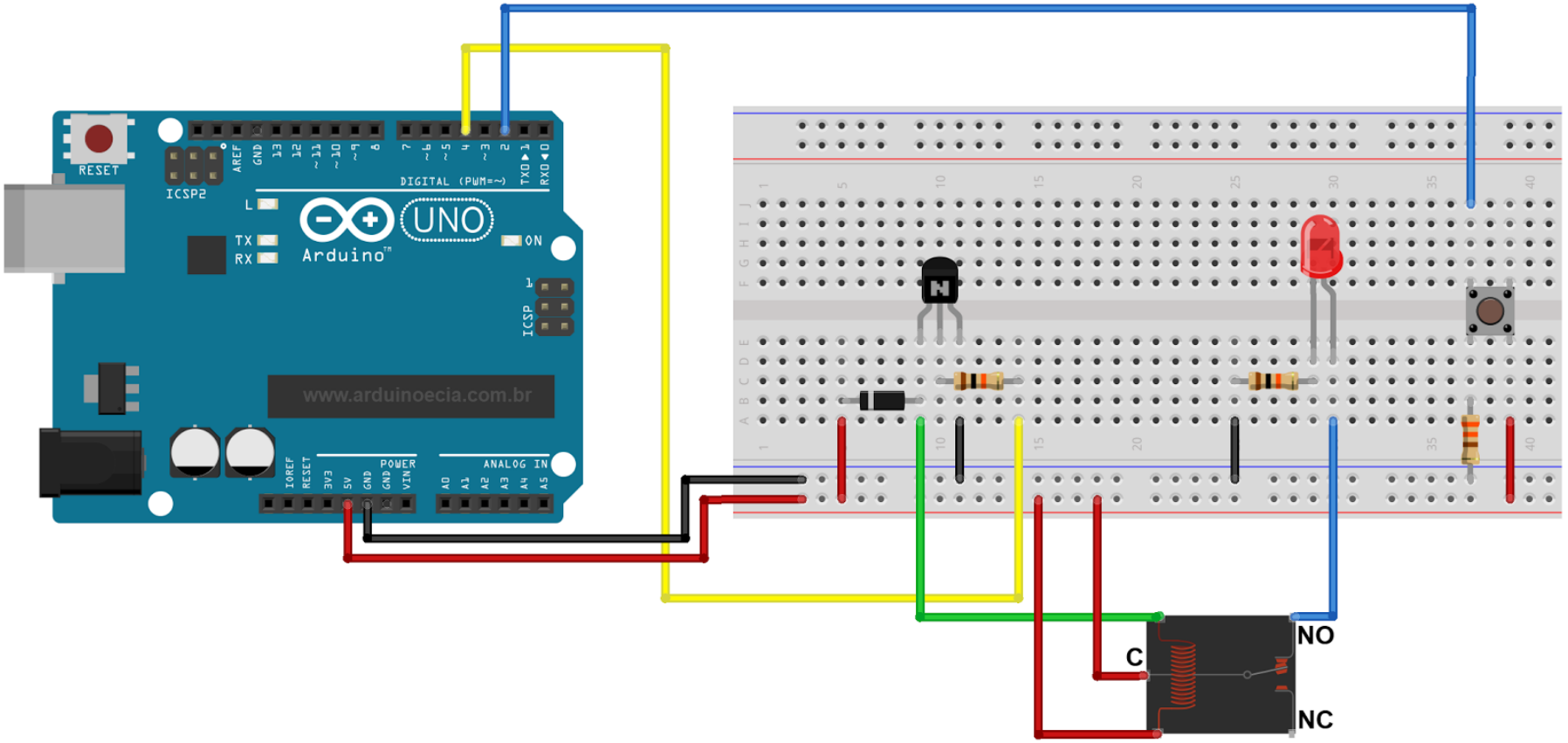
elétron livre

- A bobina de acionamento do relé pode ter diferentes tensões: 5voolts, 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;

Globalcode

Arduino + Relé

elétron livre

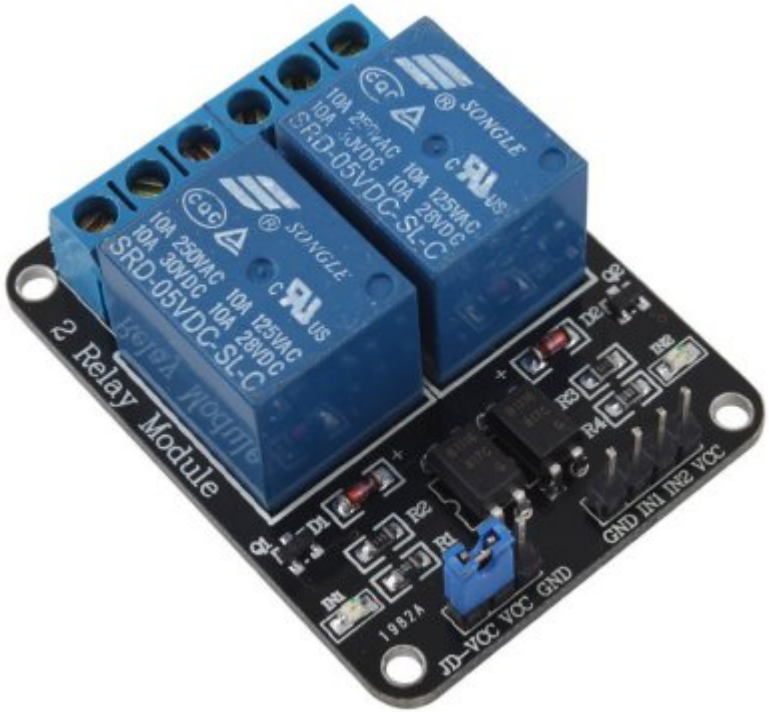


Globalcode

Placa Relé 2 Canais

elétron livre

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

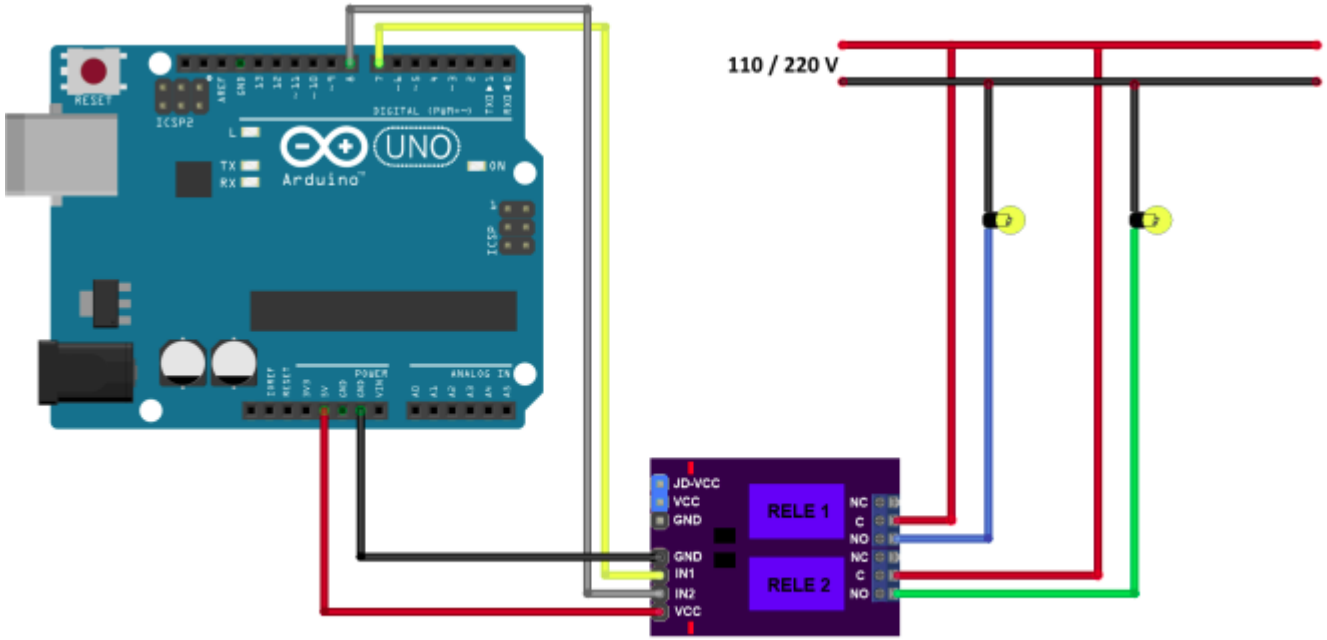


Globalcode

Placa Relé - Fiação

elétron livre

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



Globalcode

Arduino + Relé: código

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

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

Motores

elétron livre

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

Globalcode

Motores DC / CC

elétron livre

- 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 transforma velocidade em força;

Globalcode

Mais sobre Motores DC

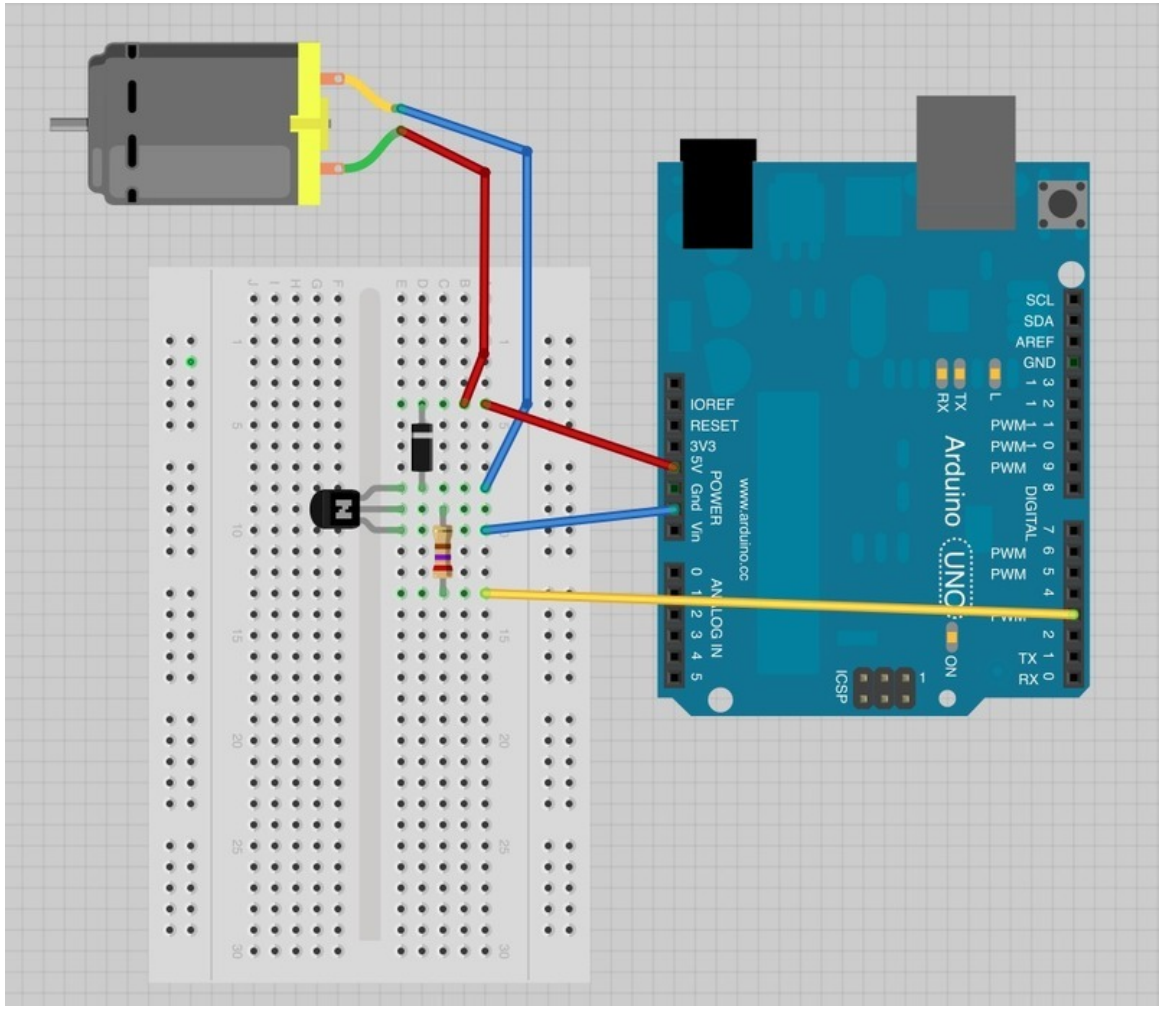
elétron livre

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

Globalcode

Motores DC + Arduino

elétron livre



Globalcode

Motores DC + Arduino

elétron livre

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

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

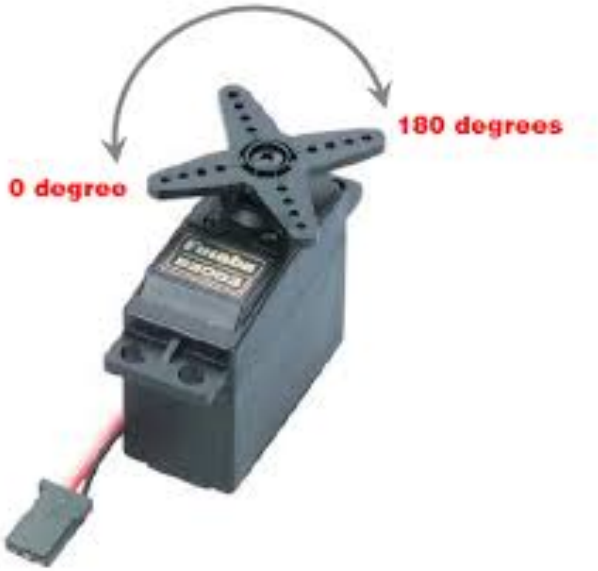
- É possível manipular a direção do giro através de controle de ponte H de transistores ou placa controladora de motor pronta!
- É possível saber se o motor está ou não rodando e quanto através de sistemas de feed-back;

Globalcode

Servomotor

elétron livre

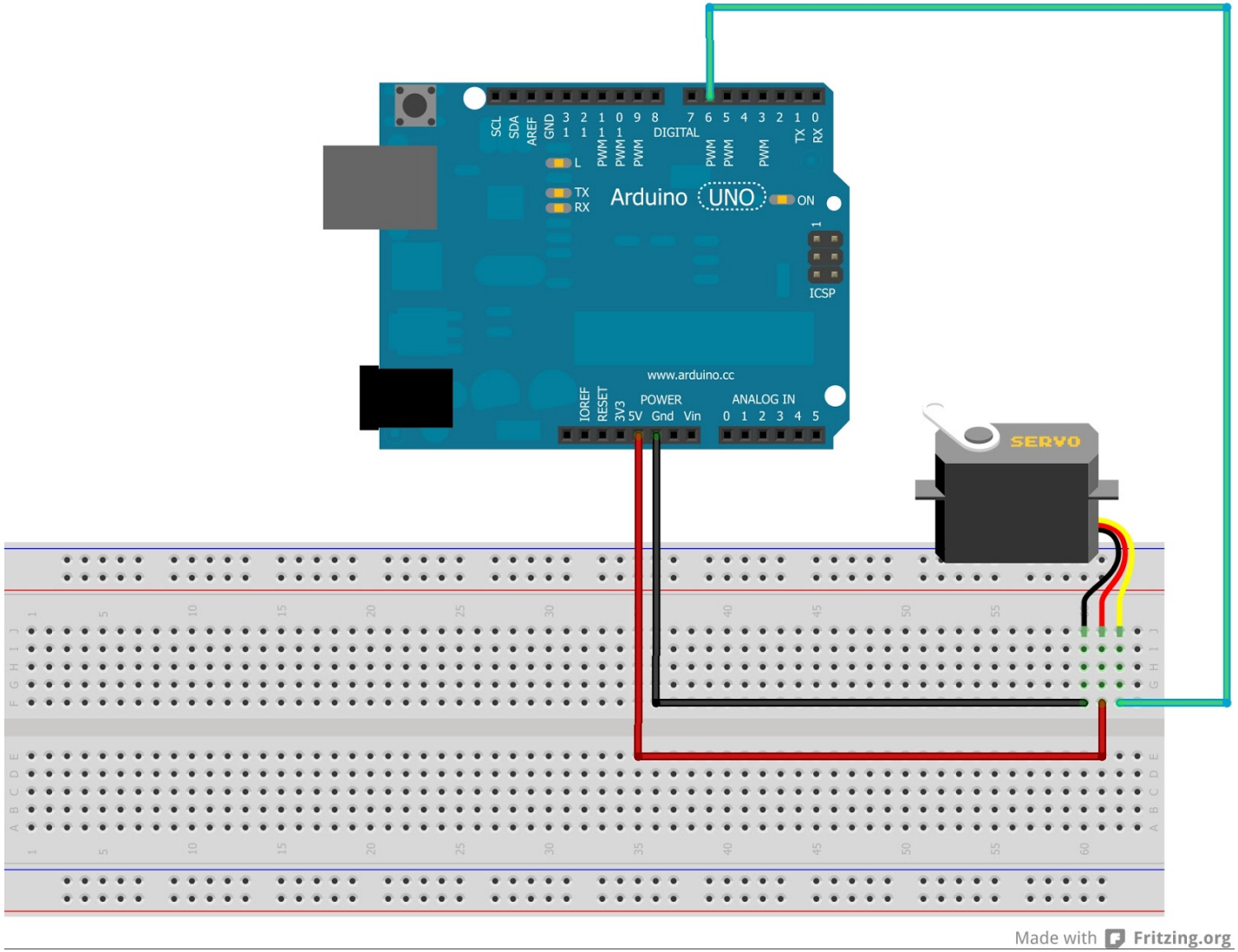
- É um conjunto de um motor DC + caixa de redução + sistema de feed-back que permite o posicionamento com precisão entre 0 e 180 graus;



Globalcode

Servomotor + Arduino

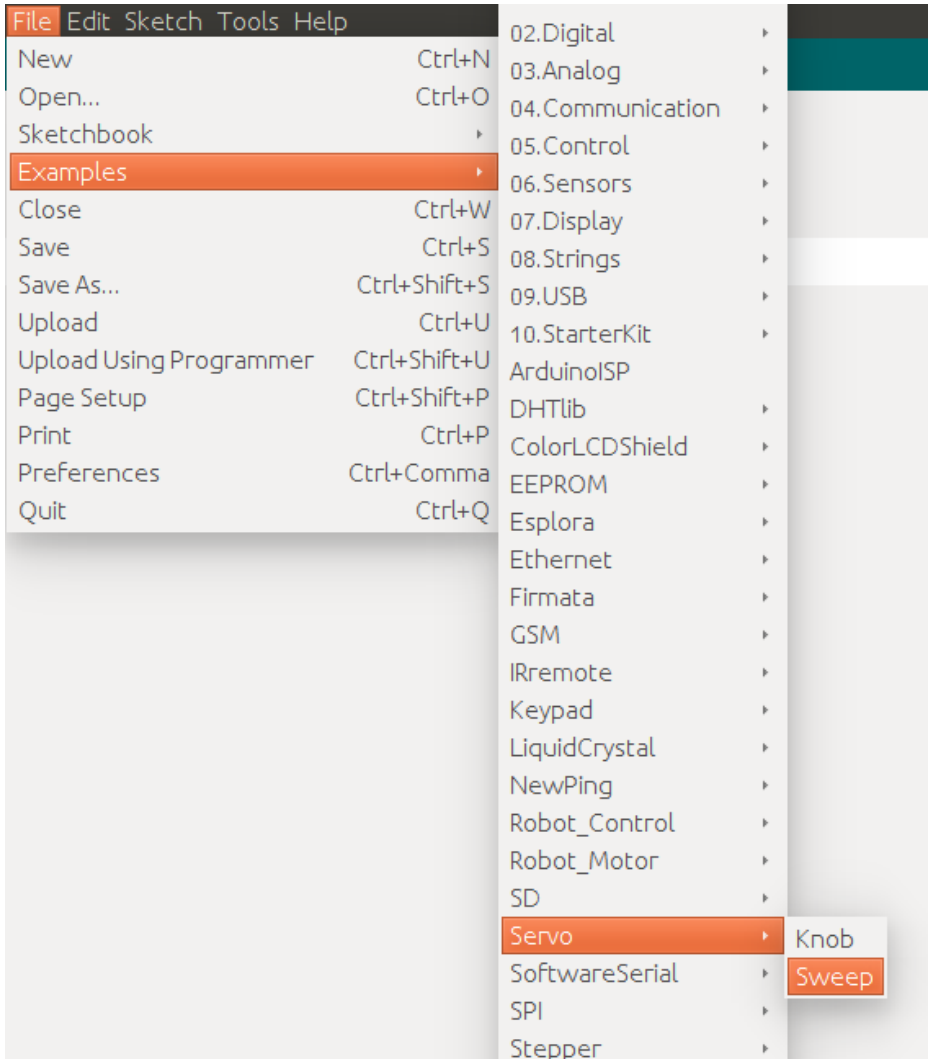
elétron livre



Globalcode

Arduino Servo: exemplo

elétron livre



Globalcode

Servomotor: código


```
#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
  {                                  // in steps of 1 degree
    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

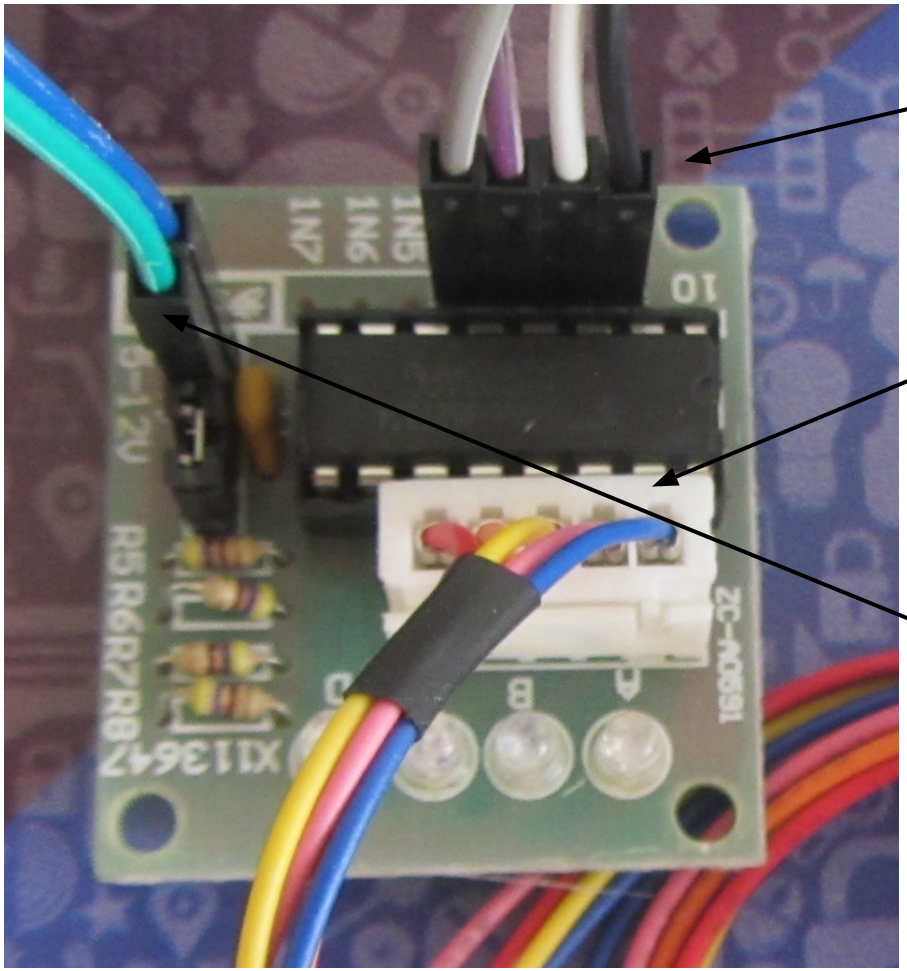
elétron livre

- 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;

Globalcode

Driver motor de passo

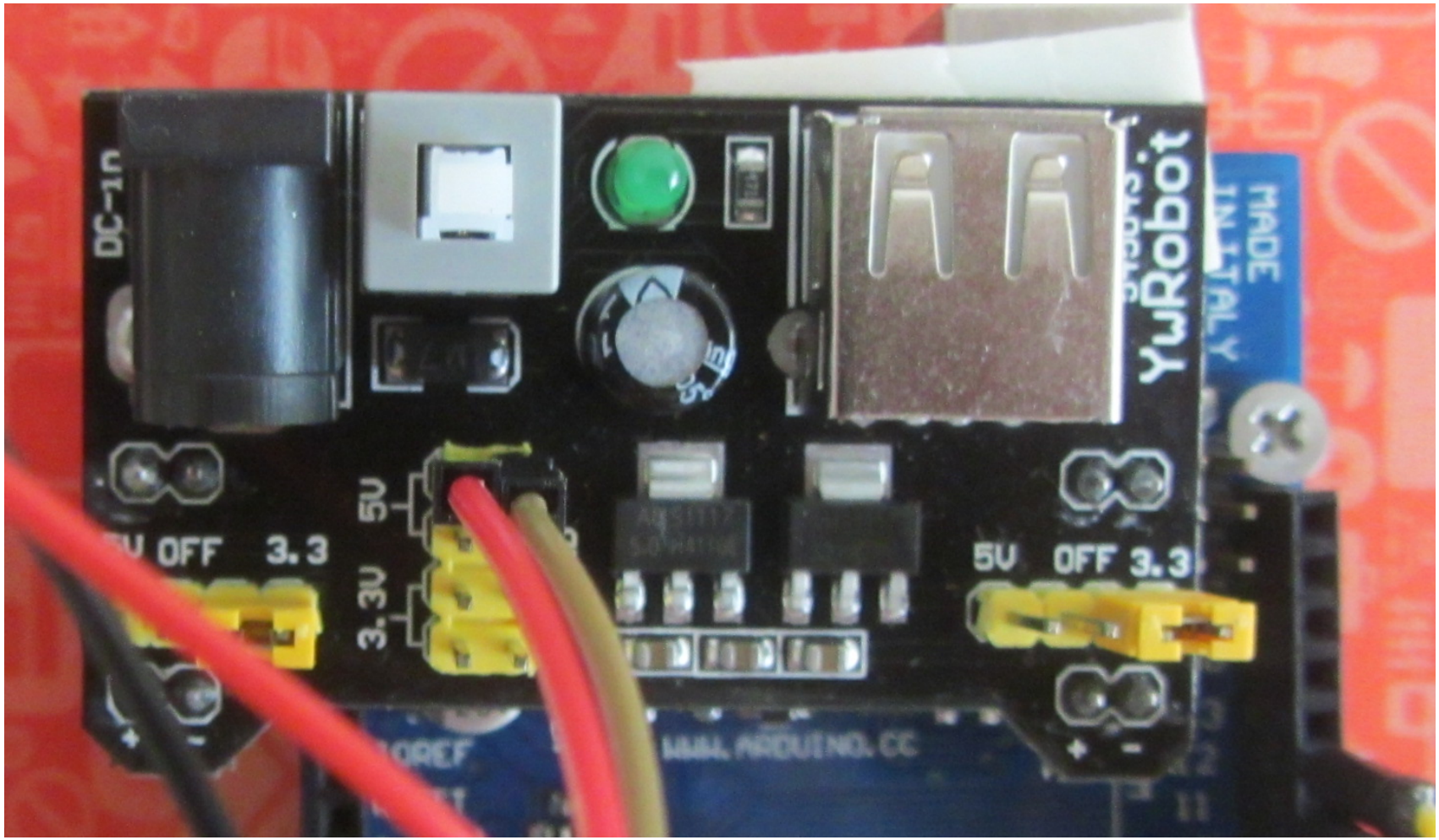
elétron livre



Globalcode

Fonte Externa

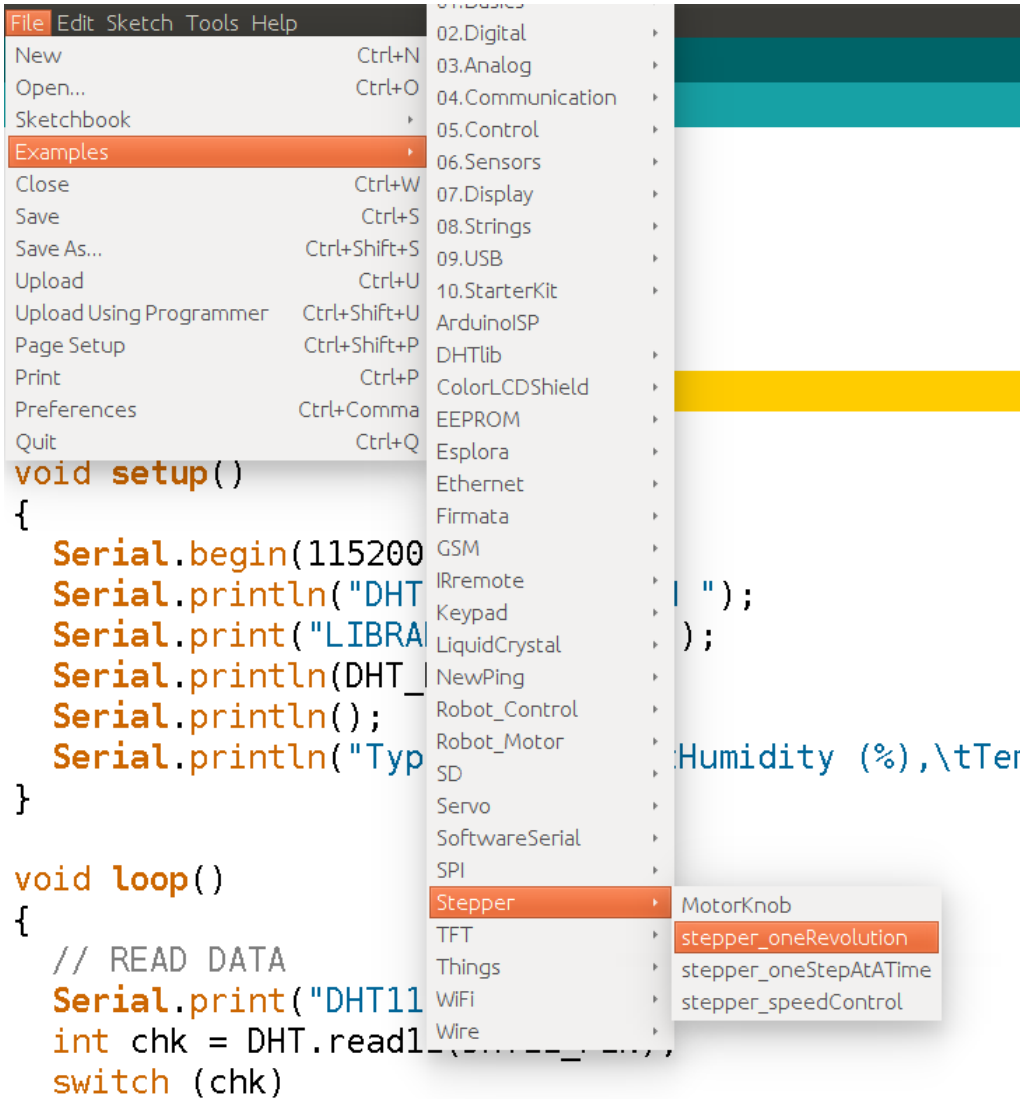
elétron livre



Globalcode

Motor de passo: código

elétron livre



Globalcode

Motor de passo: código

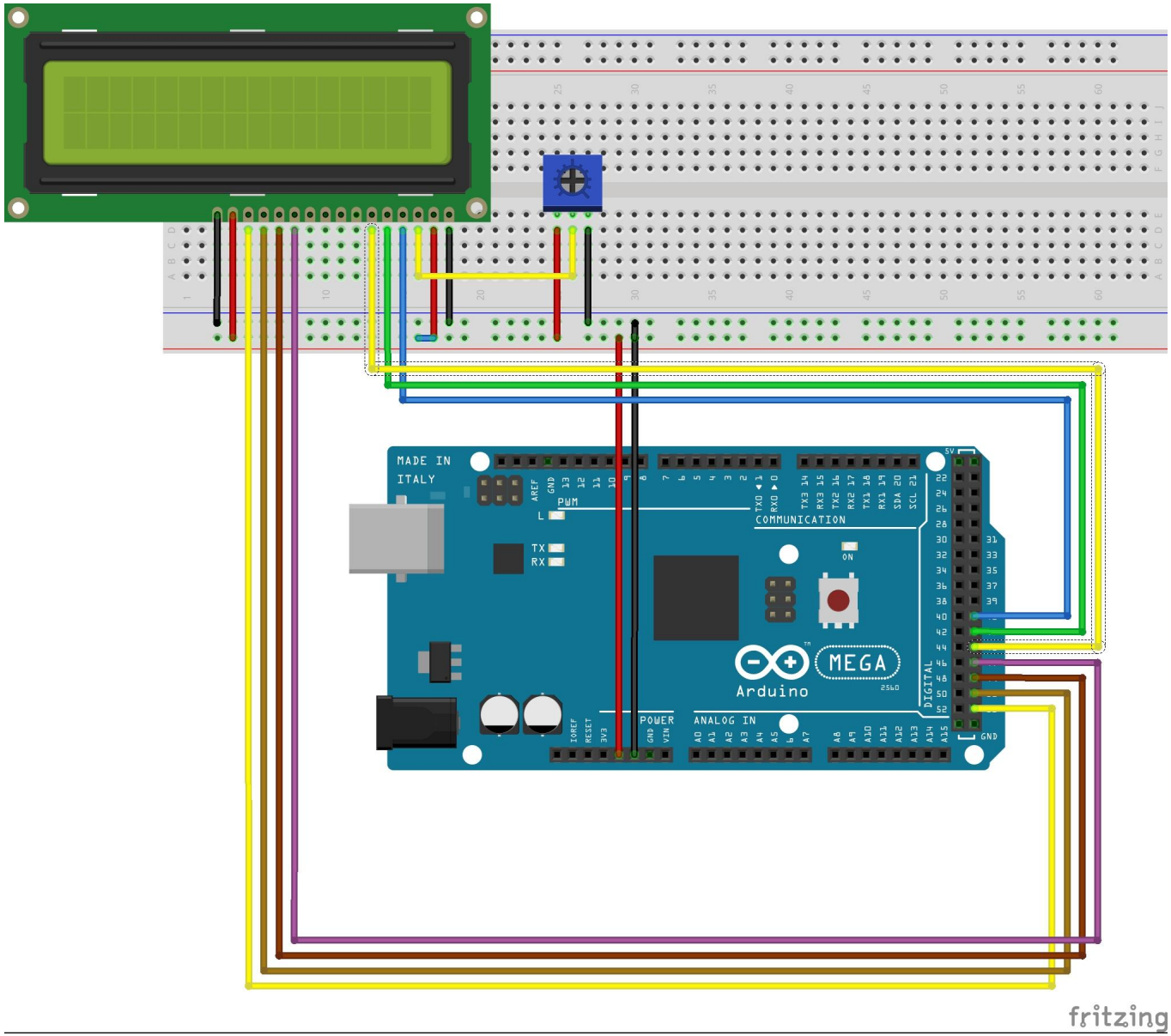
elétron livre

```
#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fi
                                     // 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);
```

Globalcode

Display LCD

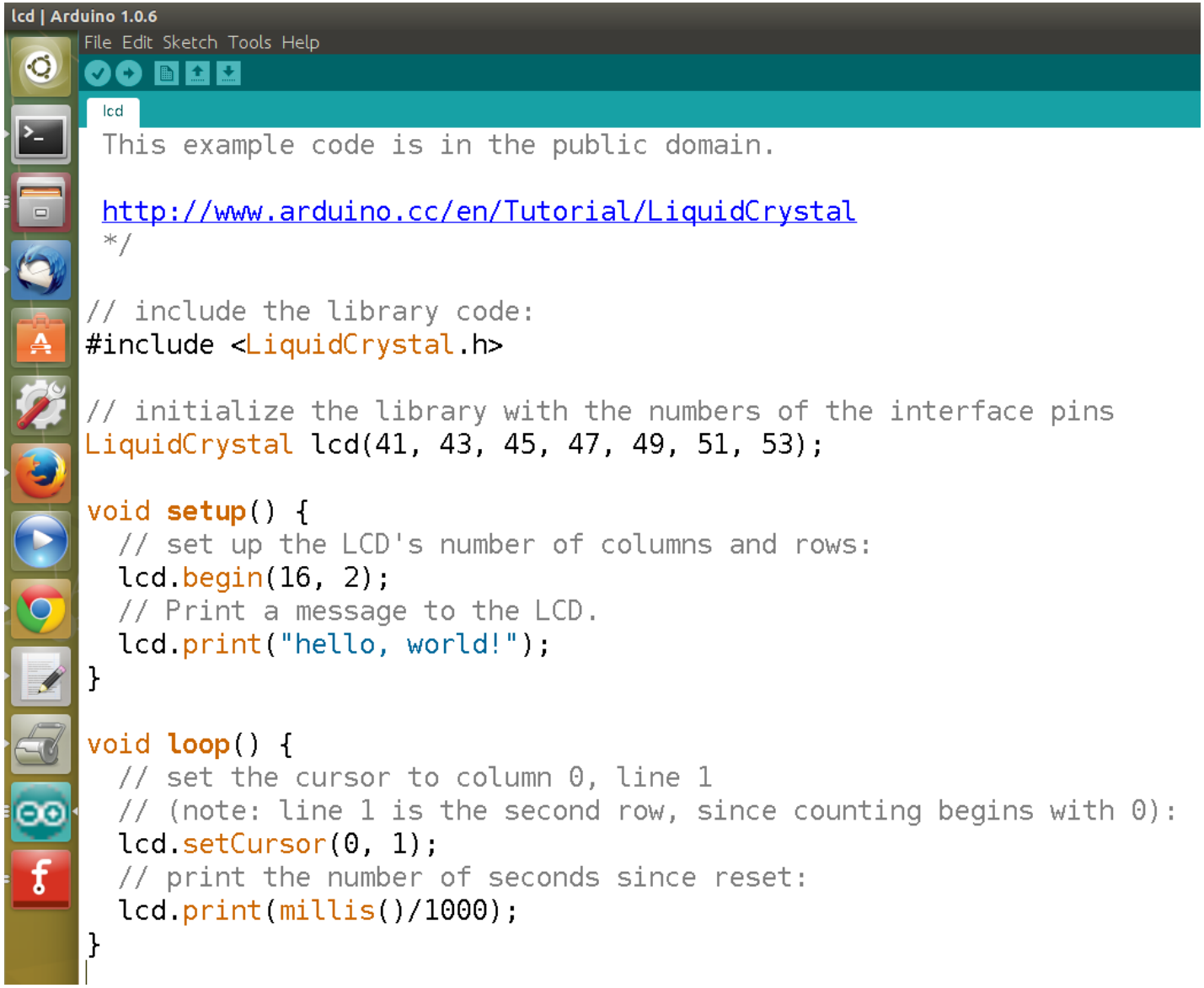
elétron livre



Globalcode

Display LCD

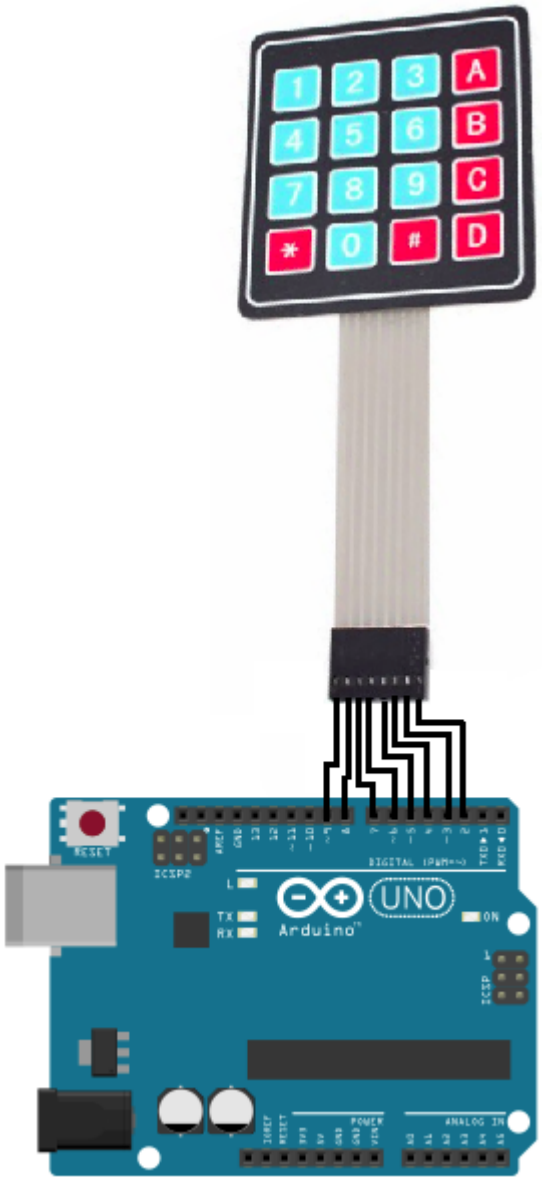
elétron livre



Globalcode

Teclado

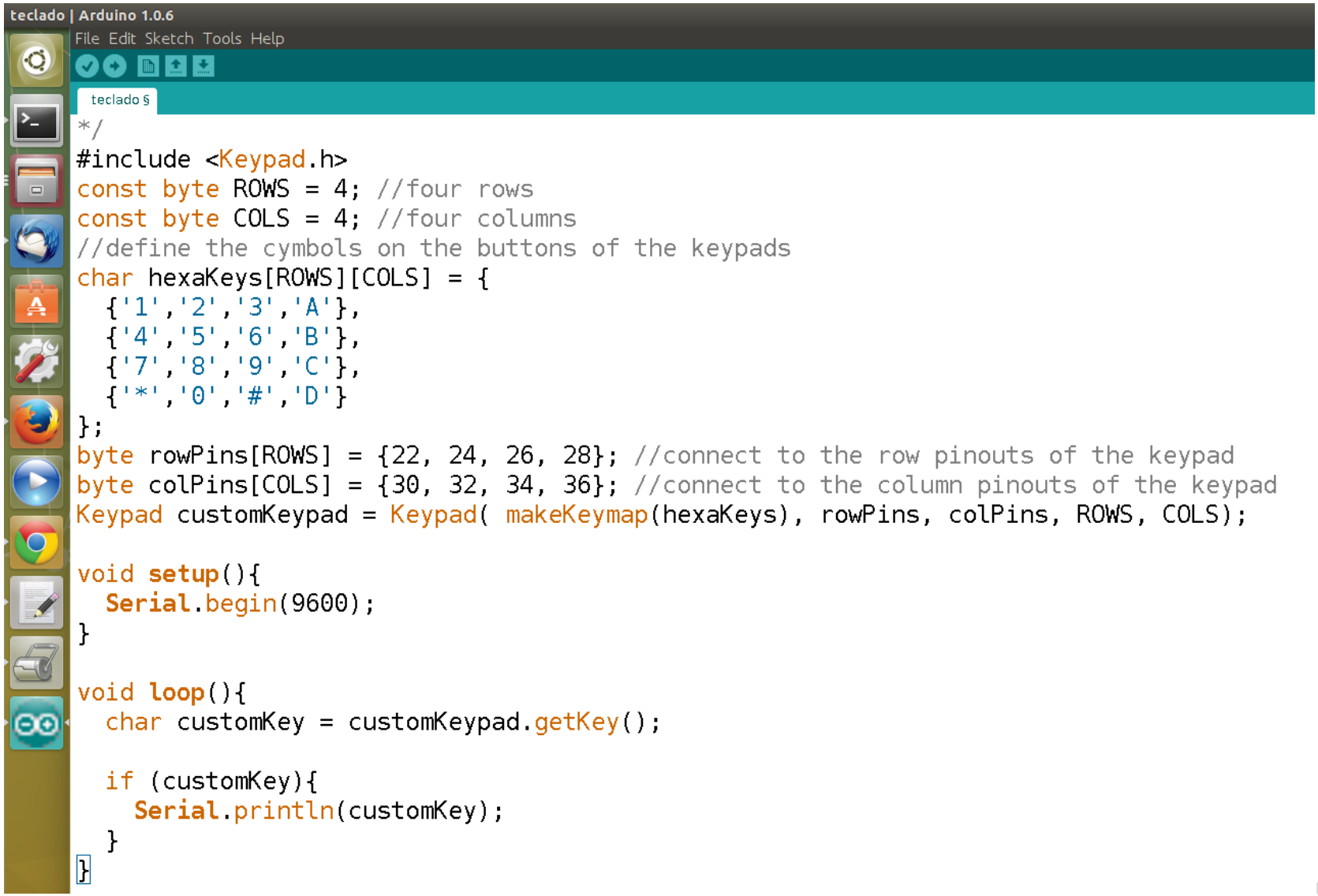
elétron livre



Globalcode

Teclado: código

elétron livre




```
*/
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the cymbols 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);
  }
}
```

Globalcode