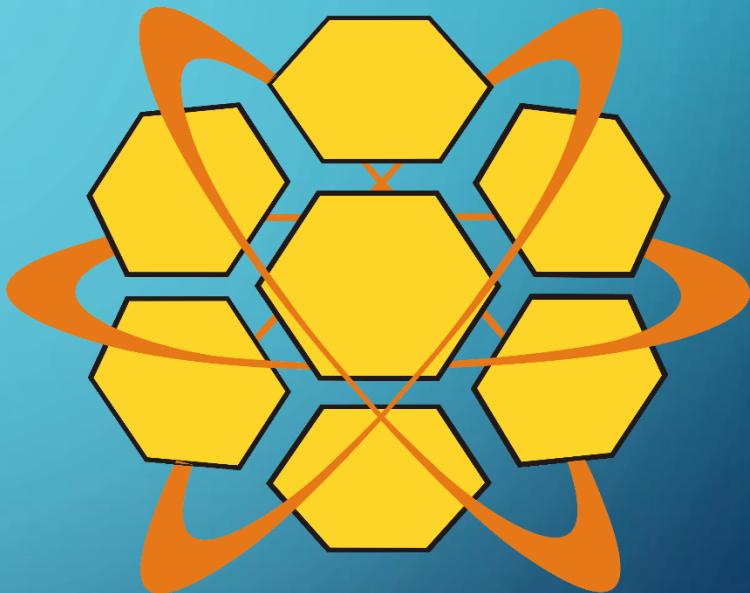


MINICURSO ARDUINO BÁSICO

Ministrantes:

- Henrique Wippel Parucker da Silva
 - Murilo de Oliveira do Nascimento
 - Ícaro Cristofolini



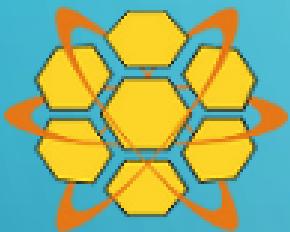
COLMÉIA



UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

JOINVILLE
CENTRO DE CIÊNCIAS
TECNOLÓGICAS

COLMÉIA

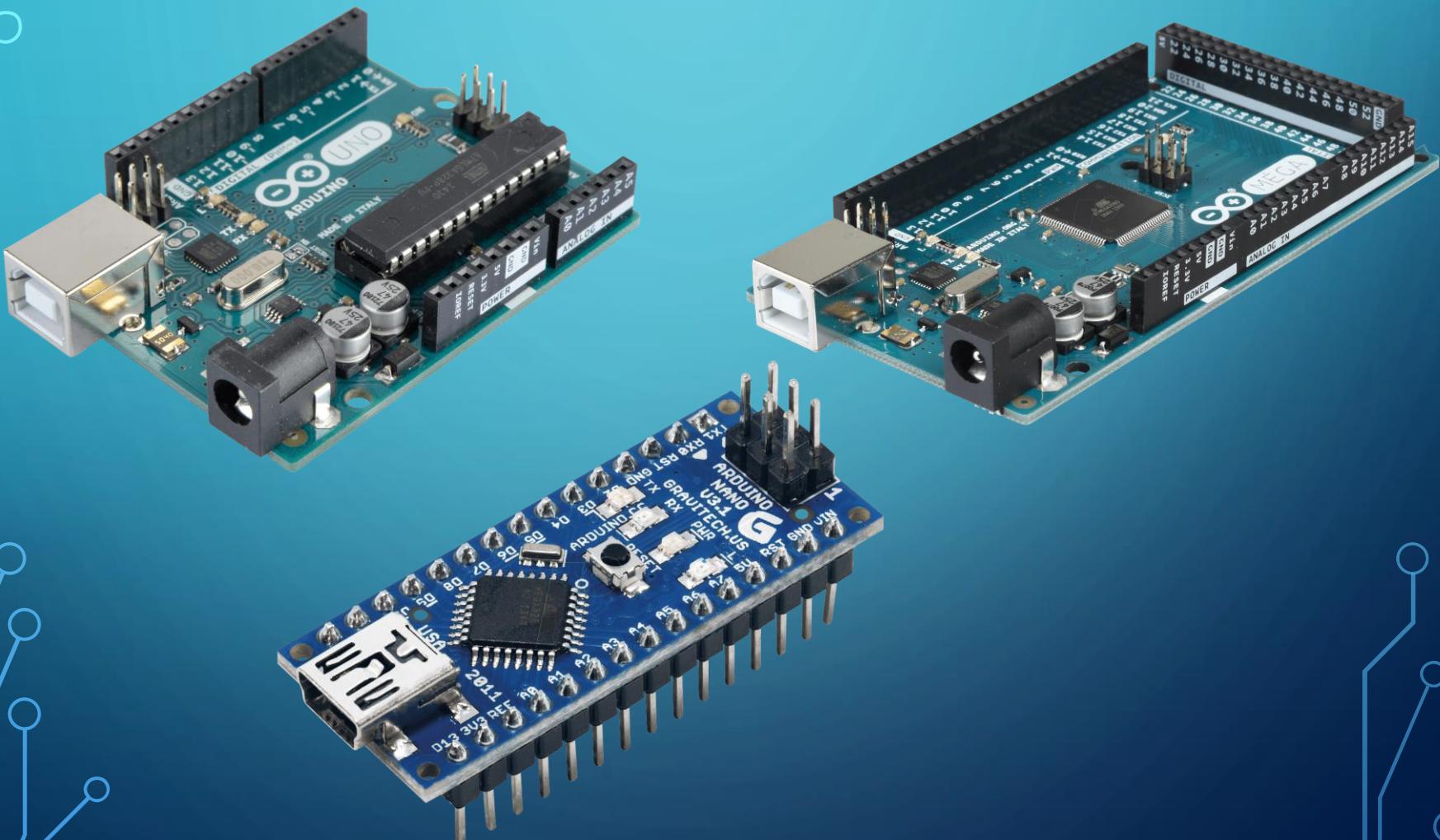


COLMÉIA
Grupo de Pesquisa em Software Livre

Quem somos?

O que fazemos?

PLATAFORMA DE PROTOTIPAGEM ARDUINO



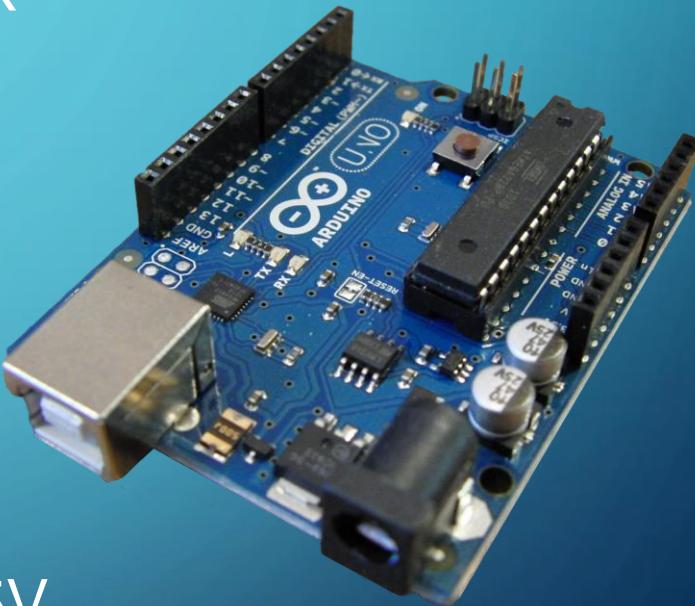
ARDUINO

- Projeto iniciado em 2005.
- Alternativa barata e simples de usar.
- Conjunto de ferramentas para desenvolvimento de aplicações diversas.
- Diversos modelos (universal).
- Hardware Livre.

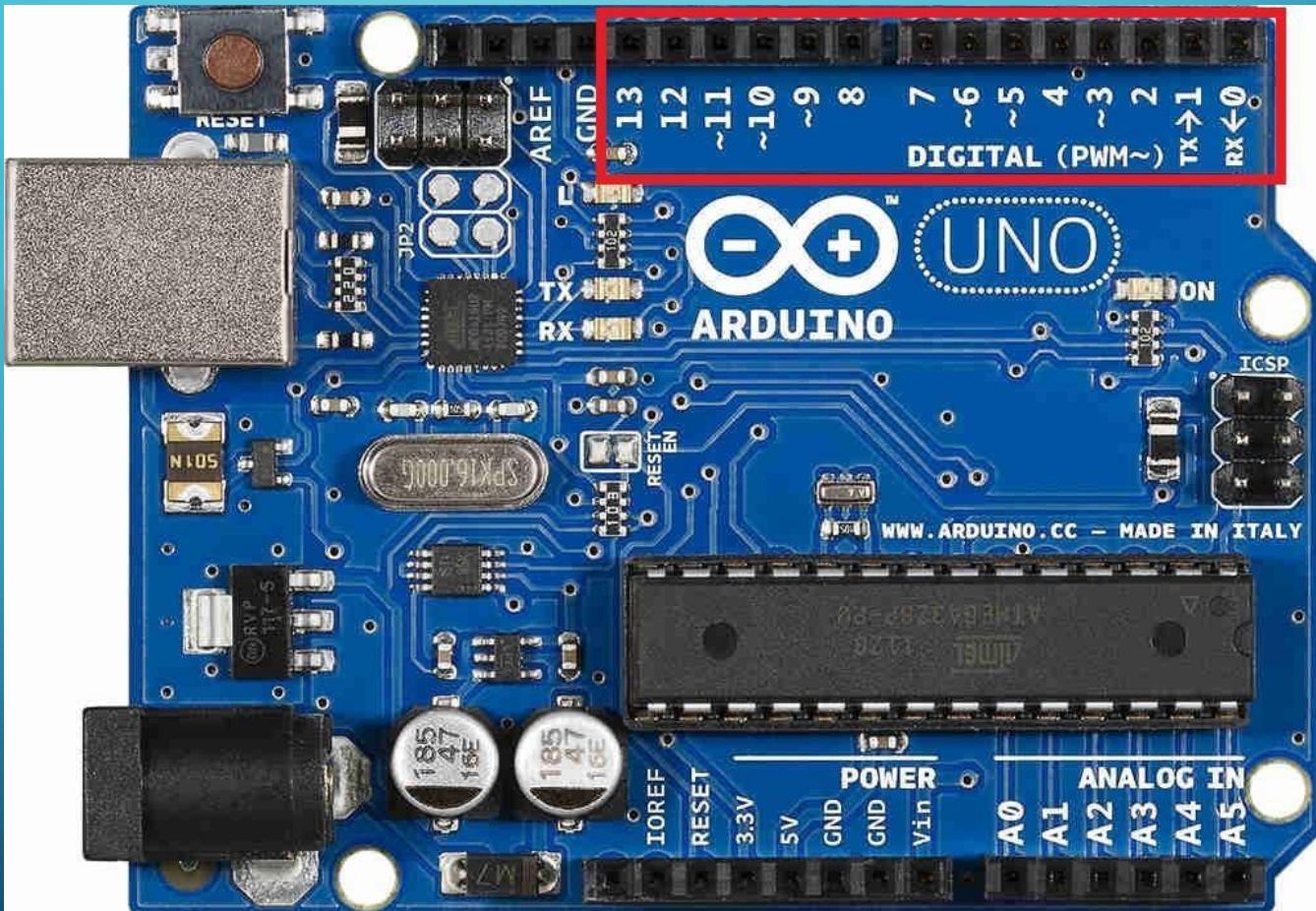


ARDUINO UNO

- Controlador: ATmega328 (32k flash)
- 14 pinos digitais
 - 6 pinos PWM
- 6 pinos de entrada analógica
- 2 pinos alimentação
- 1 pino reset
- Entrada USB e alimentação 5V

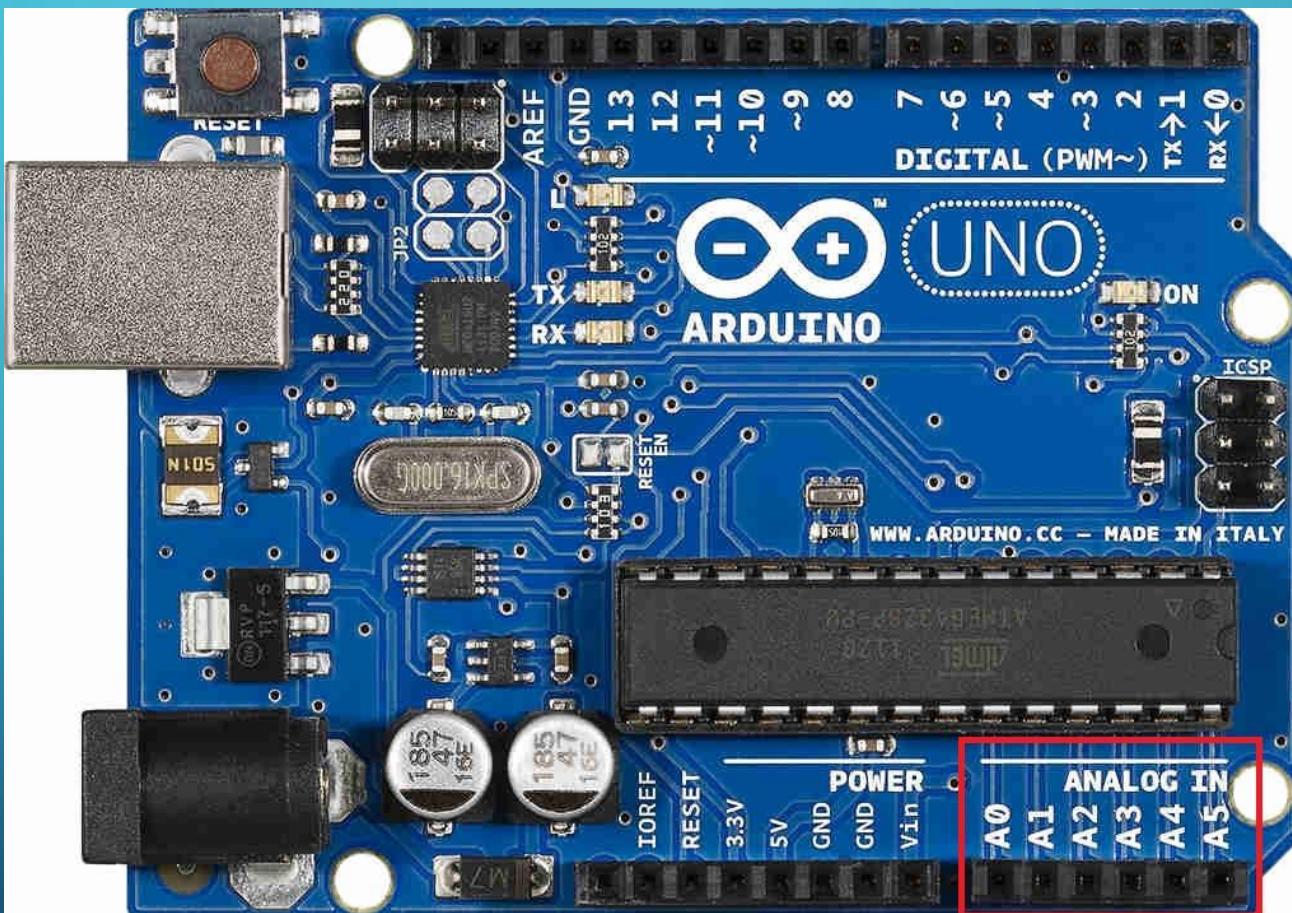


PINOS DIGITAIS



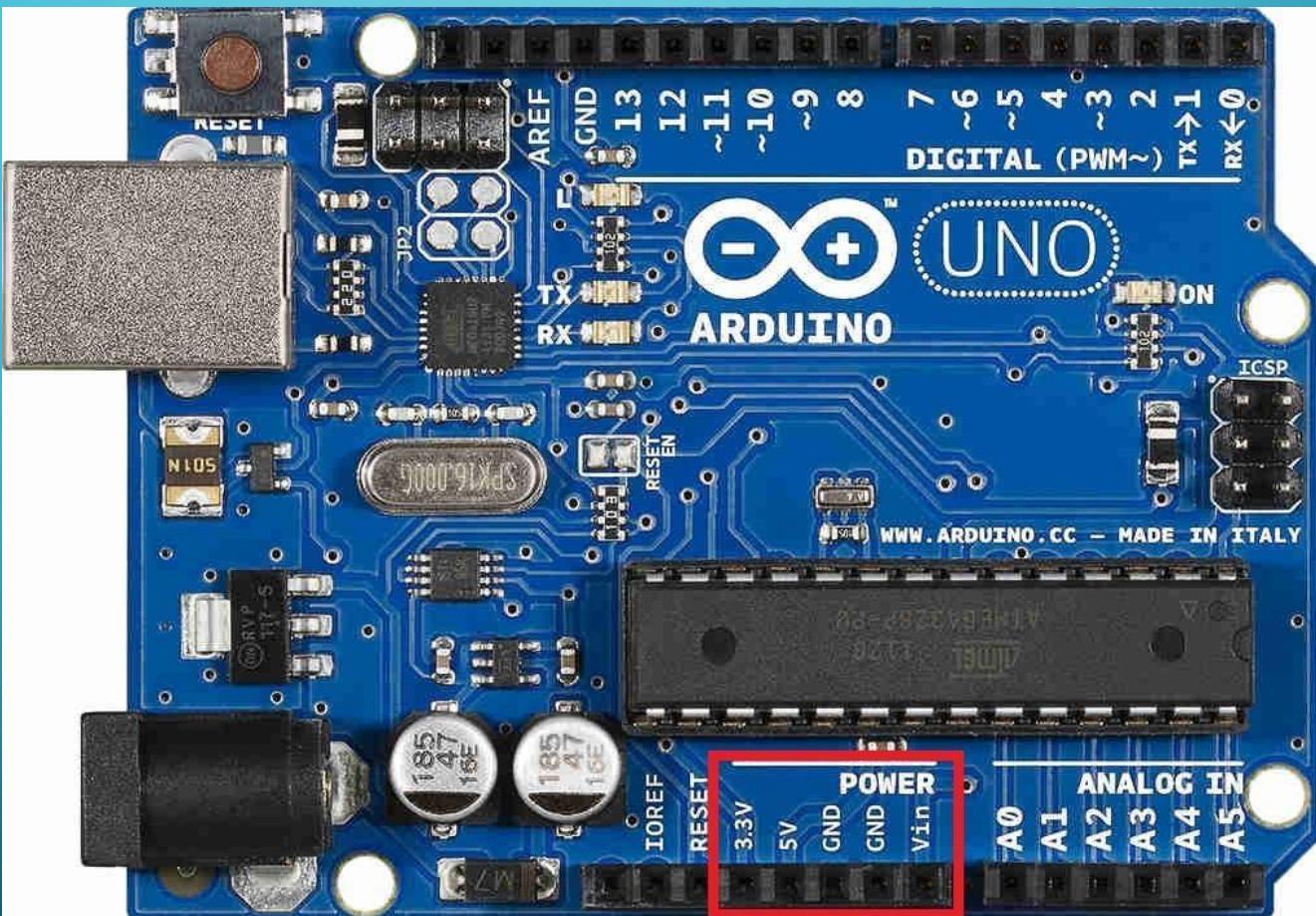
Fonte: <http://robotechshop.com/shop/arduino/arduino-board/arduino-uno/>

PINOS ANALÓGICOS



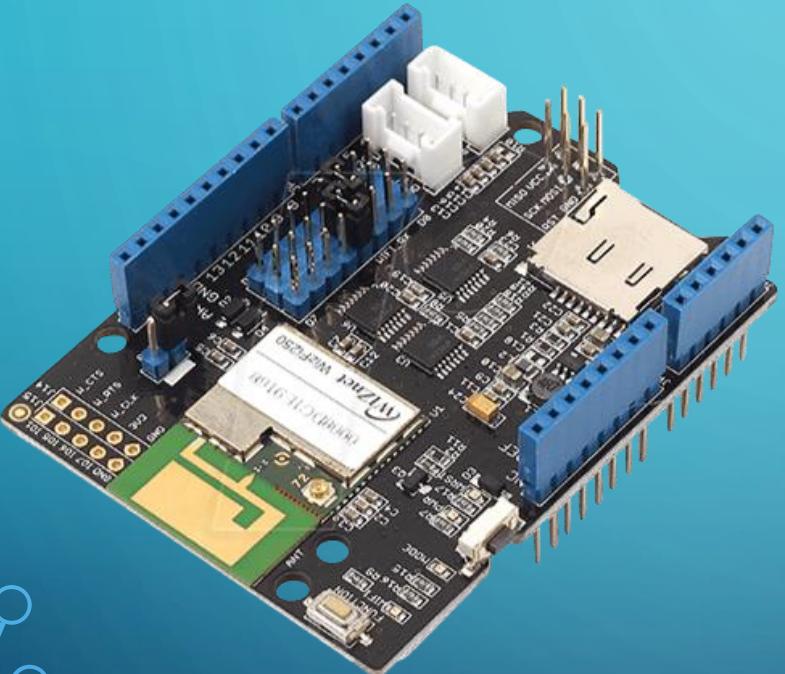
Fonte: <http://robotechshop.com/shop/arduino/arduino-board/arduino-uno/>

PINOS ALIMENTAÇÃO

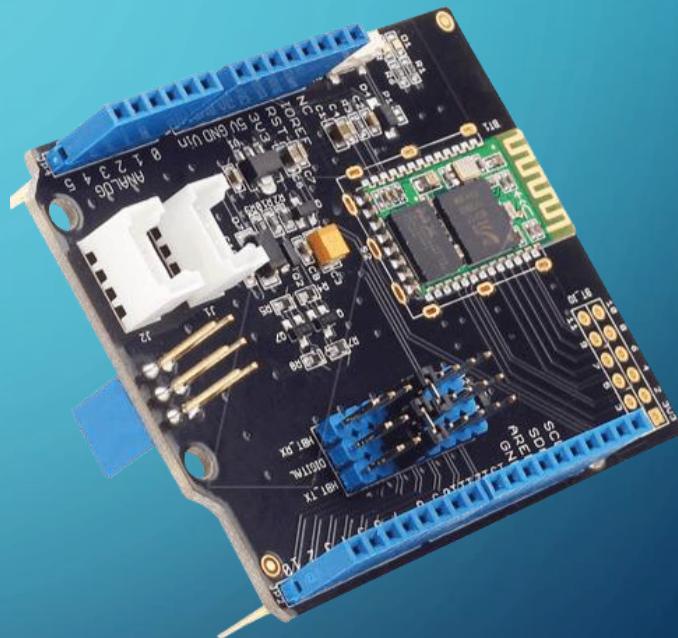


Fonte: <http://robotechshop.com/shop/arduino/arduino-board/arduino-uno/>

SHIELDS



Wifi

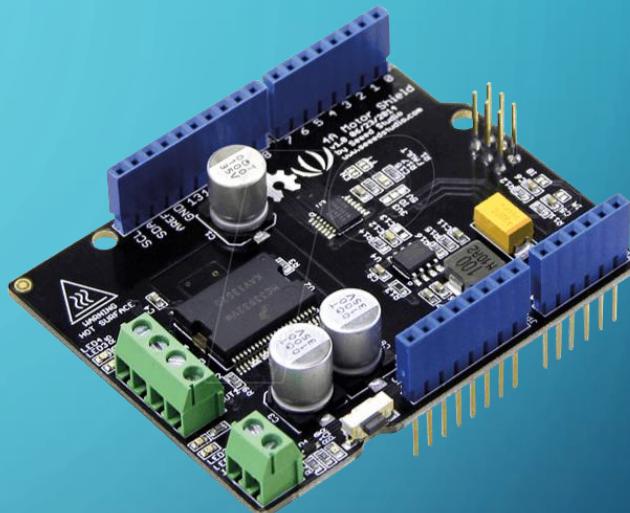


Bluetooth

SHIELDS



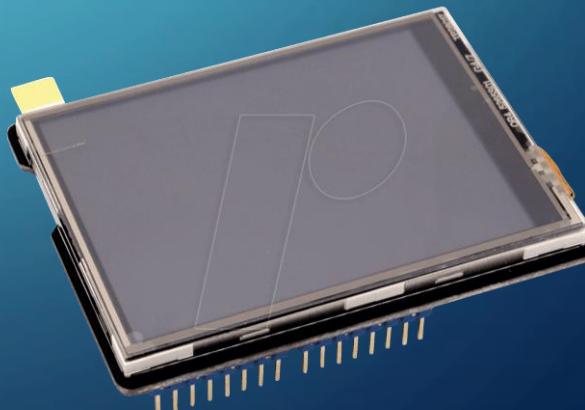
Câmera 640x480



Motor



40 LEDs

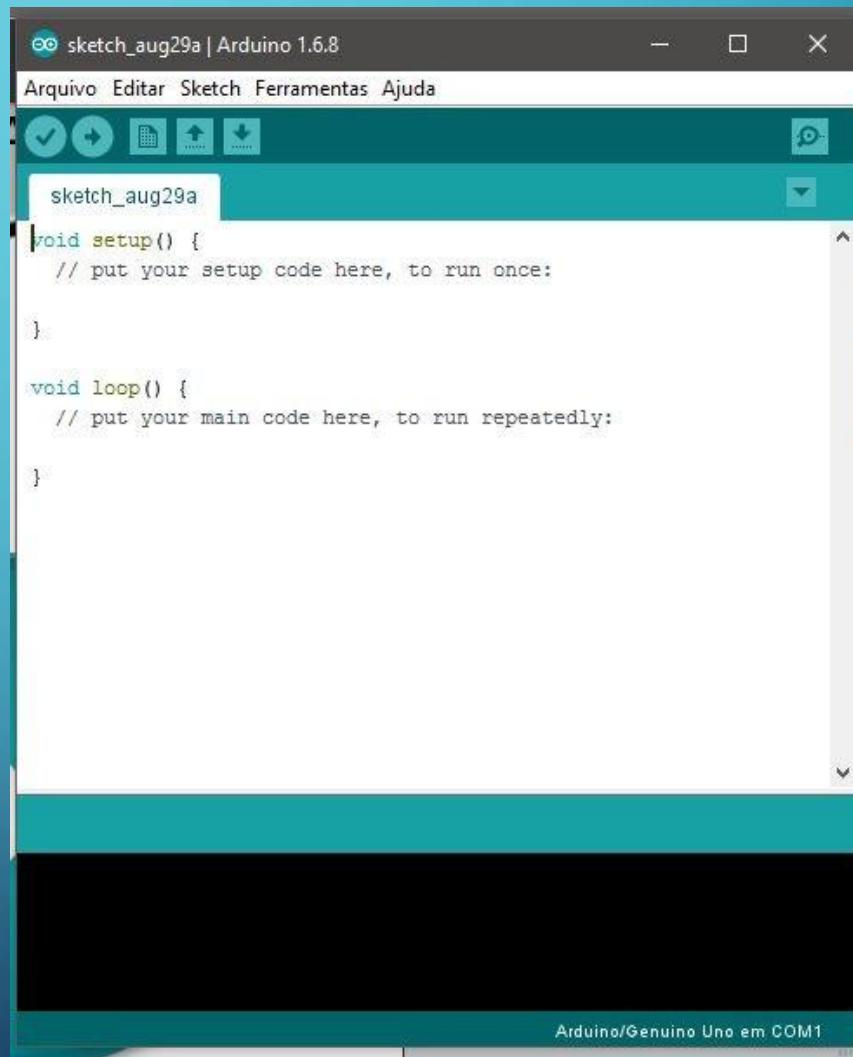


Touch Screen

ARDUINO IDE



Fonte: arduino.cc



The screenshot shows the Arduino IDE version 1.6.8. The window title is "sketch_aug29a | Arduino 1.6.8". The menu bar includes Arquivo, Editar, Sketch, Ferramentas, and Ajuda. The toolbar has icons for file operations and a magnifying glass. The code editor contains the following sketch:

```
sketch_aug29a

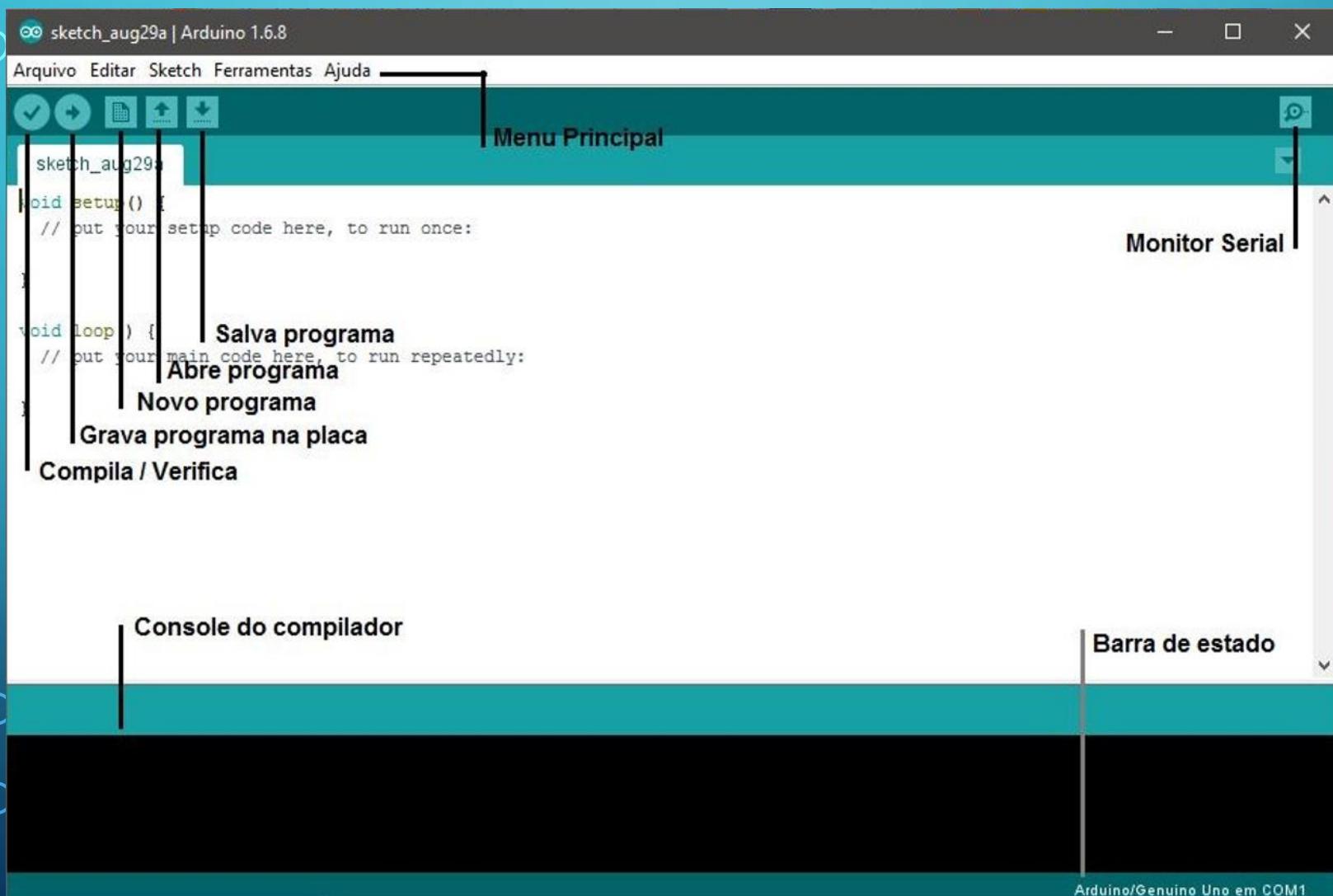
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

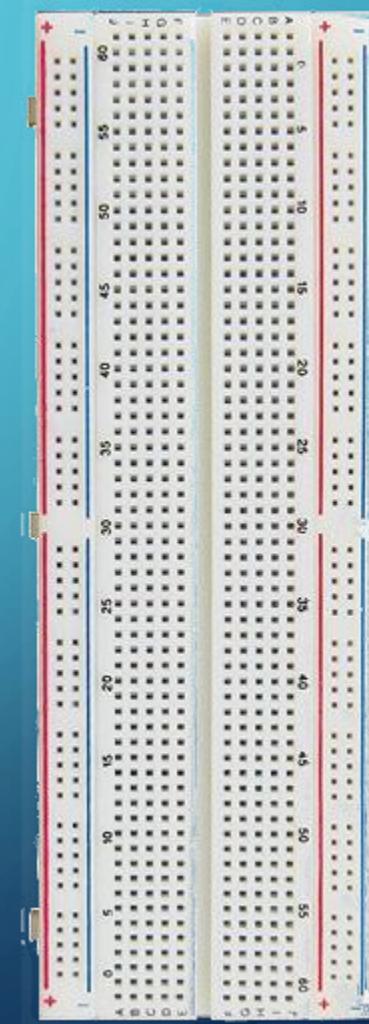
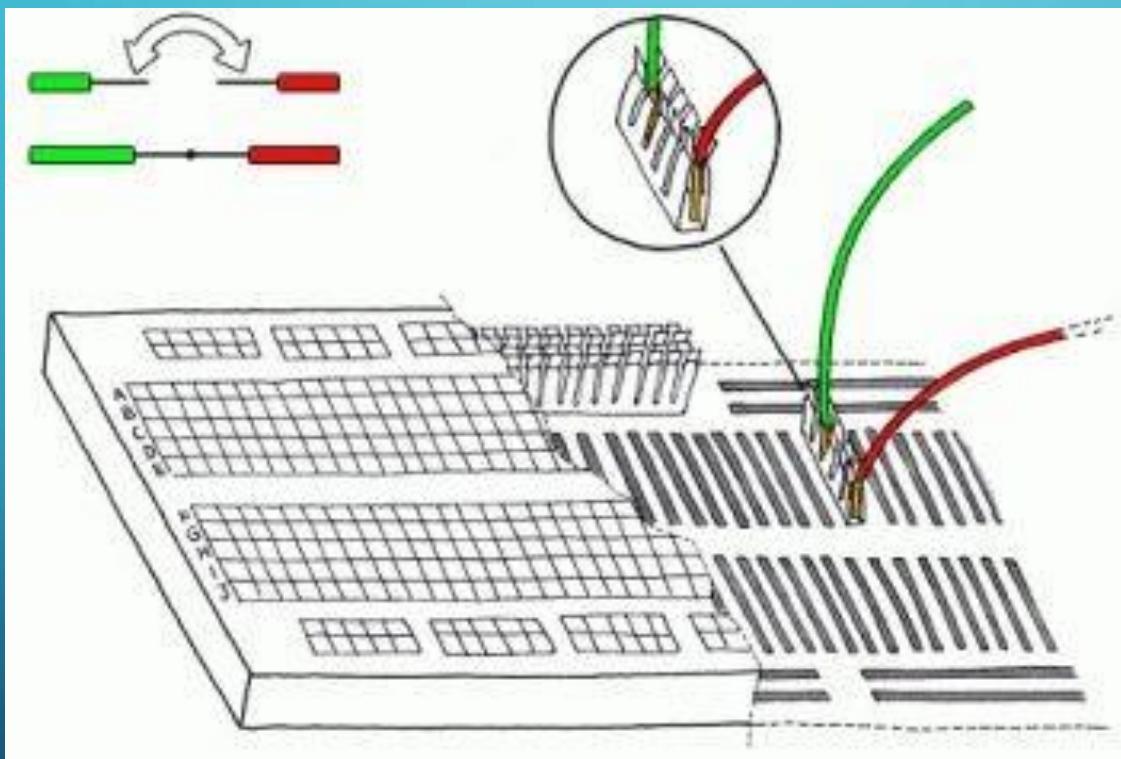
The status bar at the bottom indicates "Arduino/Genuino Uno em COM1".

IDE Arduino 1.8.4

DETALHES DA IDE

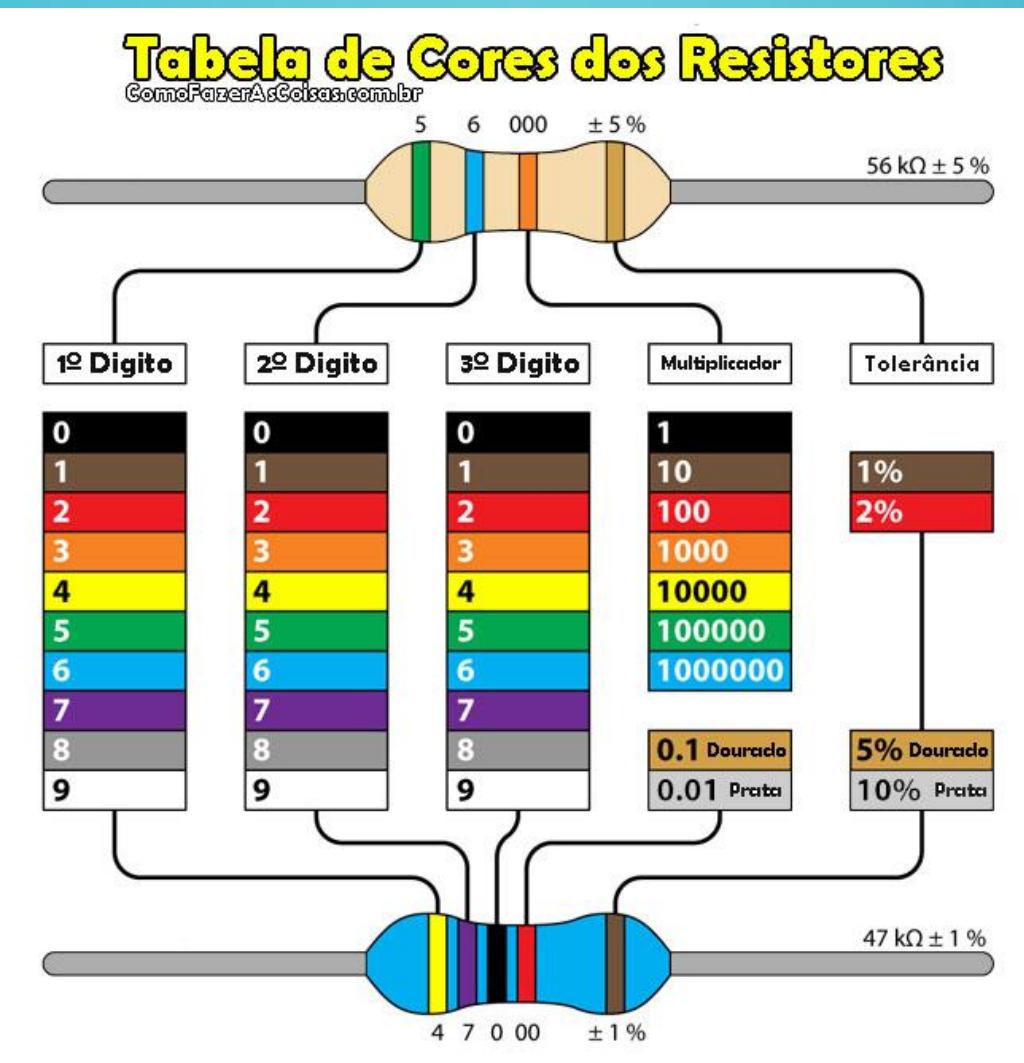


PROTOBOARD



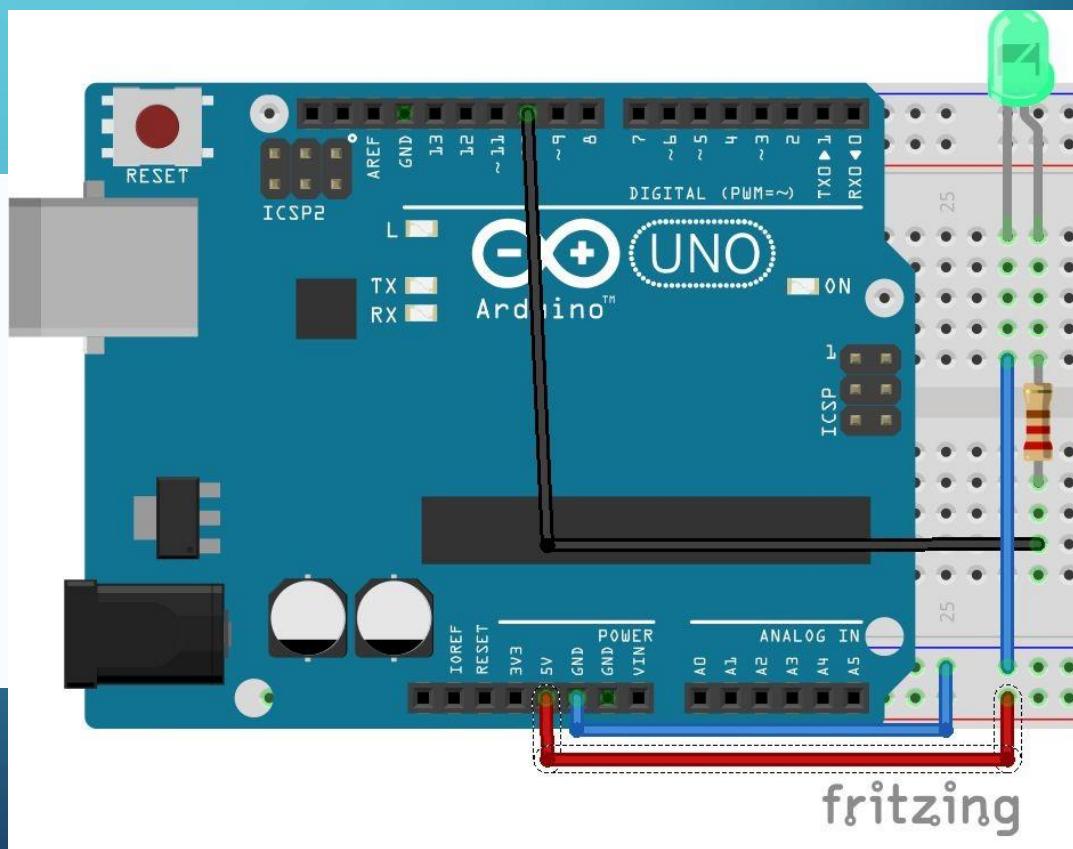
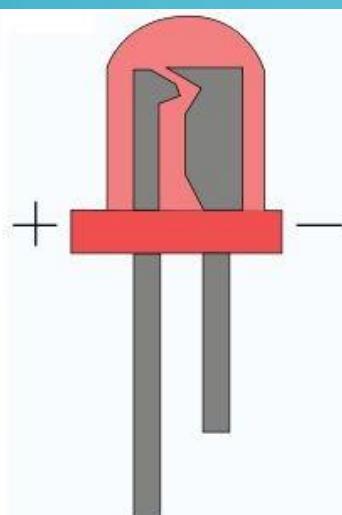
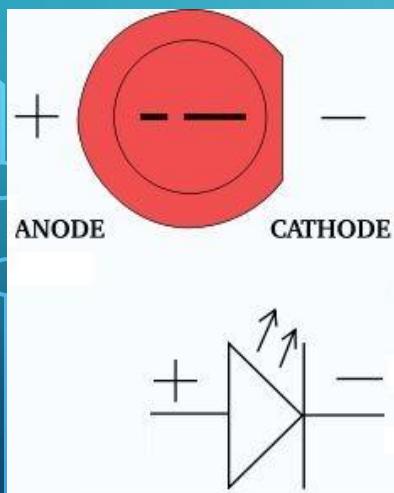
Fonte: <https://arduinowordpress.com/2016/01/16/protoboard/>

RESISTORES



PROJETO 1: PISCA LED

Objetivo: fazer um LED piscar.



SINTAXE DO PROGRAMA

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

Configuração dos pinos de entrada ou saída.
Executado apenas uma vez na inicialização
do programa

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

Rotina a ser executada pelo programa.

Linguagem baseada em C / C++, logo sua
programação é Case Sensitive.

PROJETO 1 - PISCA LED

- 1^a etapa: declarar variáveis;
- 2^a etapa: configurar saídas (`void setup()`);
- 3^a etapa: lógica do programa (`void loop()`);
- 4^a etapa: compilar e enviar para a placa.

PROJETO 1 - PISCA LED

```
int pino = 10;    // configura pino do LED  
  
void setup() {    // configura  
    pinMode(pino, OUTPUT); //  pino de saida para LED  
}  
  
void loop() {    // programa em loop  
    digitalWrite(pino, HIGH); // liga LED  
    delay(1000); // espera 1s  
    digitalWrite(pino, LOW); // desliga LED  
    delay(1000); // espera 1s  
}
```

FUNÇÕES

- **pinMode(pino, modo)** : configura um pino específico para se comportar como entrada ou saída. modo: **INPUT**; **OUTPUT**.
- **digitalWrite(pino, modo)** : escreve um pino digital como saída alta ou baixa. Modo: **HIGH**, **LOW**.
- **delay(int tempo_ms)** : Pausa o programa por um certo tempo em milissegundos.

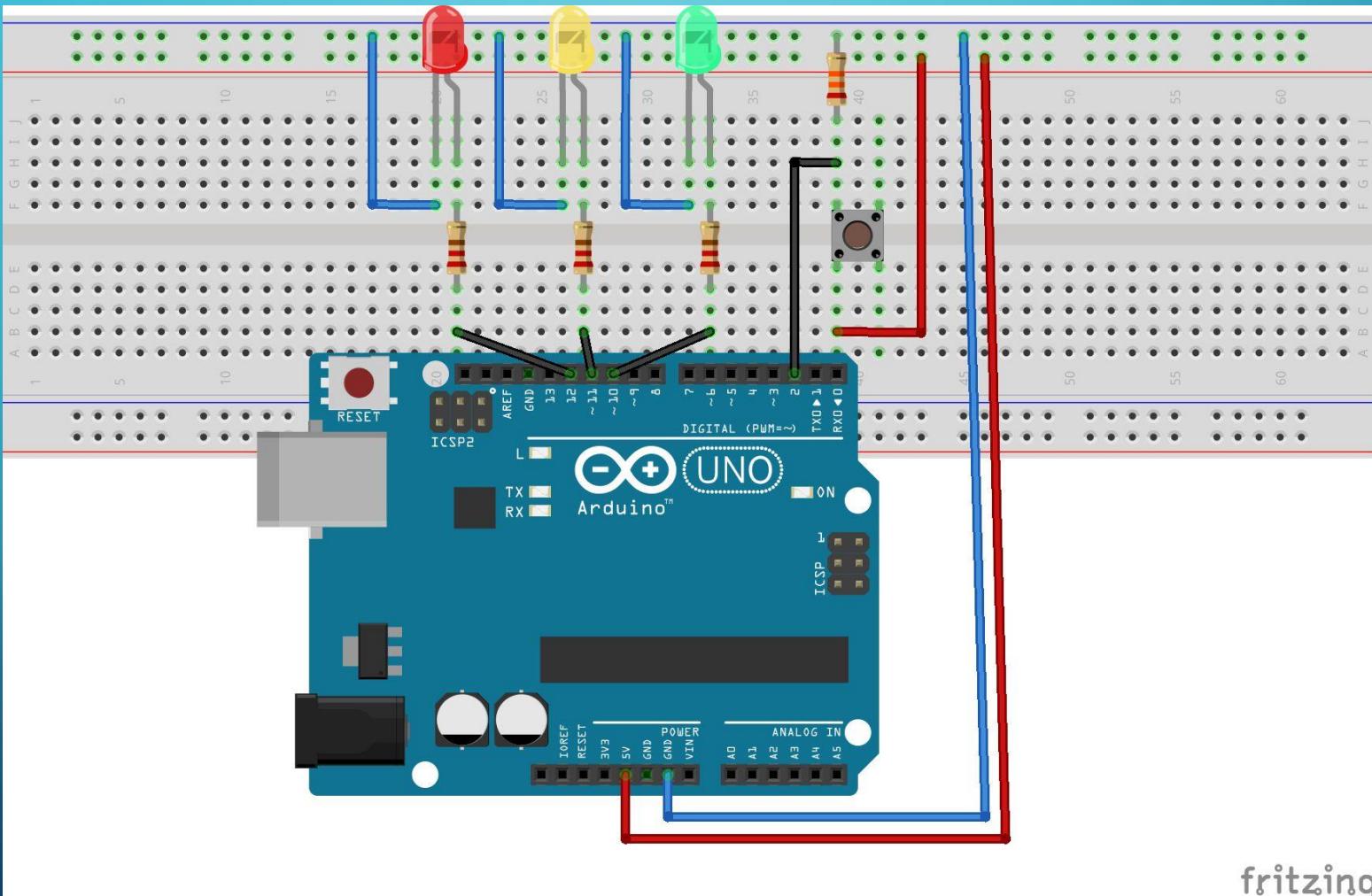
FUNÇÕES

- `attachInterrupt(canal, função, modo)` : Função que habilita algum evento (função) com uma interrupção externa.
- modo: (**FALLING, RISING, LOW, CHANGE**).

PROJETO 2 - SEMÁFORO

Objetivo: projetar um semáforo com três LEDs e um botão para que os pedestres possam atravessar.

PROJETO 2 - SEMÁFORO



fritzing

PROJETO 2 - SEMÁFORO

- 1^a etapa: declarar variáveis;
- 2^a etapa: configurar as entradas e saídas assim como o attachInterrupt do botão.
- 3^a etapa: configurar as subrotinas do programa. 4^a etapa: logica do programa.
- 5^a etapa: compilar e enviar para a placa.

PROJETO 2 - SEMÁFORO

```
int BOTAO = 2;  
  
int RED = 12, YELLOW = 11, GREEN = 10;  
  
int passo = 1;  
  
void setup(){  
  
    attachInterrupt(0, irVermelho, FALLING); // configura o botão com  
                                            // a função irVermelho  
  
    pinMode(BOTAO, INPUT);  
    pinMode(RED, OUTPUT);  
    pinMode(YELLOW, OUTPUT);  
    pinMode(GREEN, OUTPUT);  
}  
}
```

PROJETO 2 - SEMÁFORO

```
void irVermelho(){  
    if(passo == 1)  
        amarelo();  
  
}  
  
void amarelo(){  
    digitalWrite(RED, LOW);  
    digitalWrite(YELLOW, HIGH);  
    digitalWrite(GREEN, LOW);  
}
```

PROJETO 2 - SEMÁFORO

```
void vermelho(){  
    digitalWrite(RED, HIGH);  
  
    digitalWrite(YELLOW, LOW);  
  
    digitalWrite(GREEN, LOW);  
}  
  
void verde(){  
    digitalWrite(RED, LOW);  
  
    digitalWrite(YELLOW, LOW);  
  
    digitalWrite(GREEN, HIGH);  
}
```

PROJETO 2 - SEMÁFORO

```
void loop(){
    if(passo == 1){
        amarelo();
        passo++;
        delay(1500);
    }
    if(passo == 2){
        vermelho();
        passo++;
        delay(5000);
    }
    if( passo == 3){
        verde();
        passo = 1;
        delay(5000);
    }
}
```

COMUNICAÇÃO SERIAL

Realizar o debug do programa

Envia e recebe dados no formato serial



The screenshot shows the Arduino IDE interface with the title bar "sketch_aug29a | Arduino 1.6.8". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". Below the menu is a toolbar with icons for file operations. The main code editor window displays the following code:

```
sketch_aug29a

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

A "Monitor Serial" window is visible on the right side of the interface.

COMUNICAÇÃO SERIAL

`Serial.begin(taxa);`

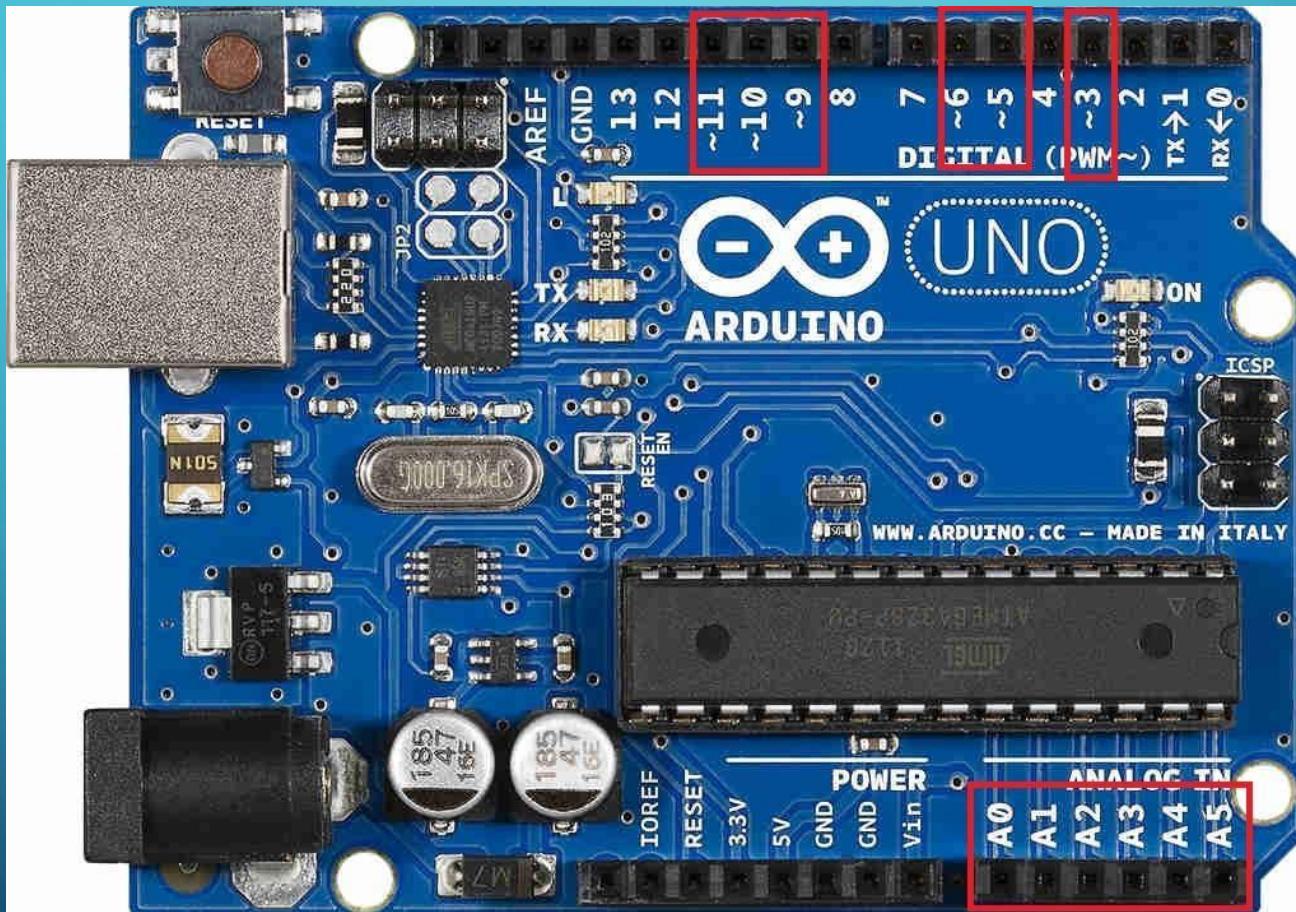
taxa geralmente é usado o valor 9600.

`Serial.print()` e `Serial.println();`

`Serial.read()` e `Serial.write();`

`Serial.available();`

ENTRADAS E SAÍDAS ANALÓGICAS



Fonte: <http://robotechshop.com/shop/arduino/arduino-board/arduino-uno/>

SAÍDAS ANALÓGICAS

O Arduino simula uma saída digital usando as portas PWM;

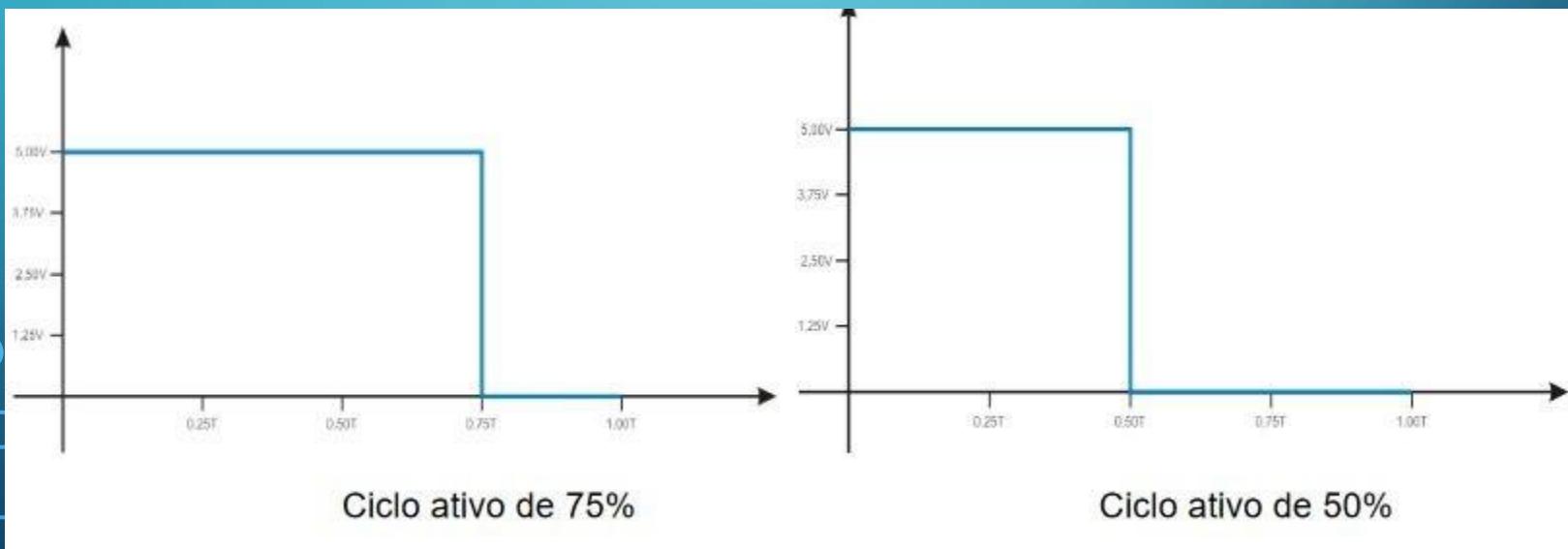


Imagen: Autoria Própria

SAÍDAS ANALÓGICAS

`analogWrite(pino, valor);`

pino: pino de saída PWM;

valor: inteiro de 0(0%) a 255(100%), que determina a porcentagem em que o sinal ficara ativo.

Obs.: Não é preciso declarar os pinos PWM como saída usando `pinMode()`.

ENTRADAS ANALÓGICAS

`analogRead(pino);`

pino indica a entrada usada.

Lê valores de 0V a 5V.

A saída é um valor de 0 (0V) a 1023 (5V).

FUNÇÃO

`map(valor, minimoIN, maximoIN, minimoOUT, maximoOUT);`

valor: valor que deseja transformar;

minimoIN: valor mínimo de entrada;

maximoIN: valor máximo de entrada;

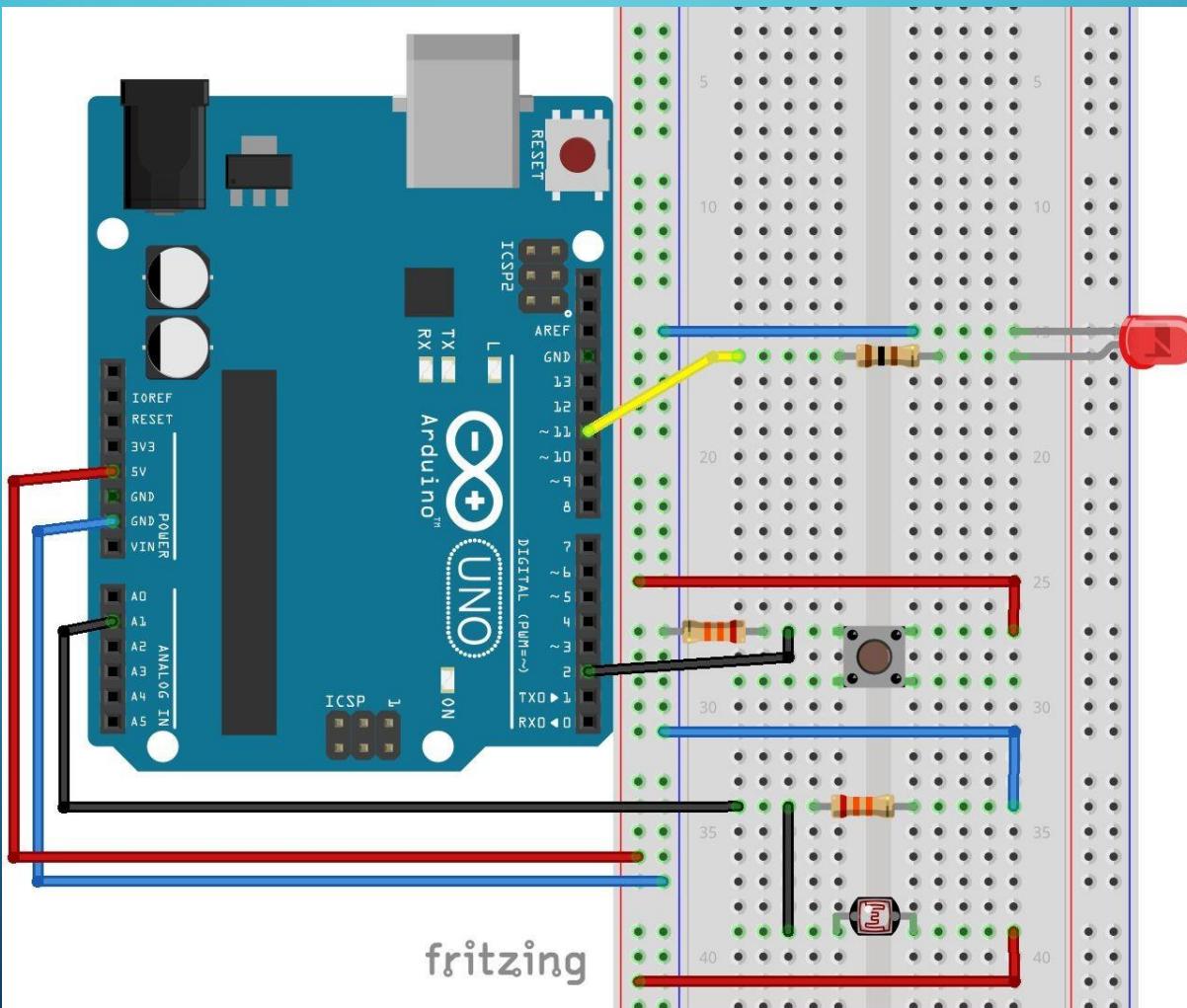
minimoOUT: valor mínimo de saída;

maximoOUT: valor máximo de saída.

PROJETO 3 - CONTROLE DE LUMINOSIDADE

Objetivo: controlar a luminosidade de um LED de acordo com a luminosidade do ambiente.

PROJETO 3 - CONTROLE DE LUMINOSIDADE



PROJETO 3 - CONTROLE DE LUMINOSIDADE

- 1º passo: Descobrir a faixa de leitura do LDR utilizando o monitor serial.
- 2º passo: Ajustar os valores lidos para escrevê-los na saída desejada.
- 3º passo: Fazer a rotina do programa.

PROJETO 3 - CONTROLE DE LUMINOSIDADE

```
int BOTAO = 2;  
  
int LDR = 4;  
  
int LED = 11;  
  
int LDRmin = 250;  
  
int LDRmax = 900;  
  
void setup(){  
    attachInterrupt(0, leitura, FALLING);  
    Serial.begin(9600);  
    pinMode(BOTAO, INPUT);  
}  
}
```

PROJETO 3 - CONTROLE DE LUMINOSIDADE

```
void leitura(){  
    Serial.println("Valor lido: ");  
    Serial.println(analogRead(LDR));  
    Serial.println(map(analogRead(LDR), LDRmin, LDRmax, 0,  
    255));  
}
```

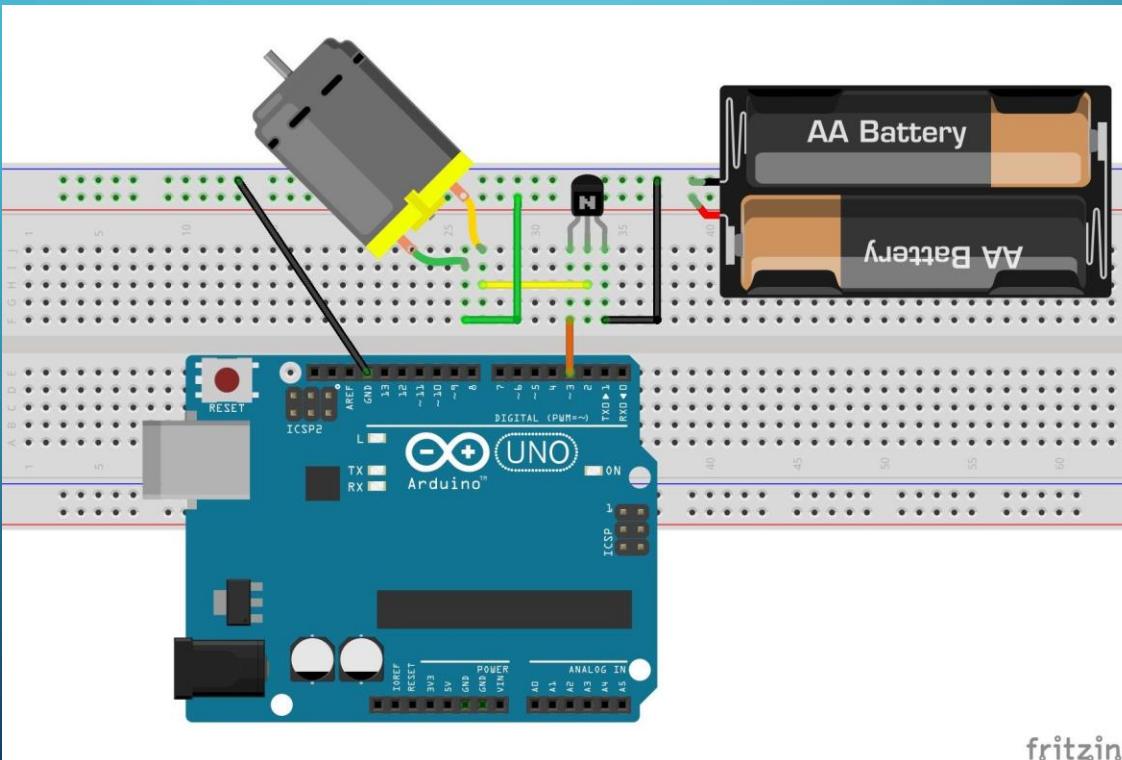
PROJETO 3 - CONTROLE DE LUMINOSIDADE

```
void loop(){  
  
    int luz = map(analogRead(LDR, LDRmin, LDRmax, 0, 255));  
  
    //Serial.println(analogRead(LDR));  
  
    if(( luz > 50 ) && ( luz < 255 ))  
        analogWrite(LED, map(analogRead(LDR, LDRmin,  
                                         LDRmax, 0, 255)));  
  
    else if( luz >= 255)  
        analogWrite(LED, 255);  
  
    else  
        analogWrite(LED, 0);  
}
```

CONTROLE DE MOTORES DC

Corrente máxima suportada: 40mA;

Necessidade de um circuito auxiliar;



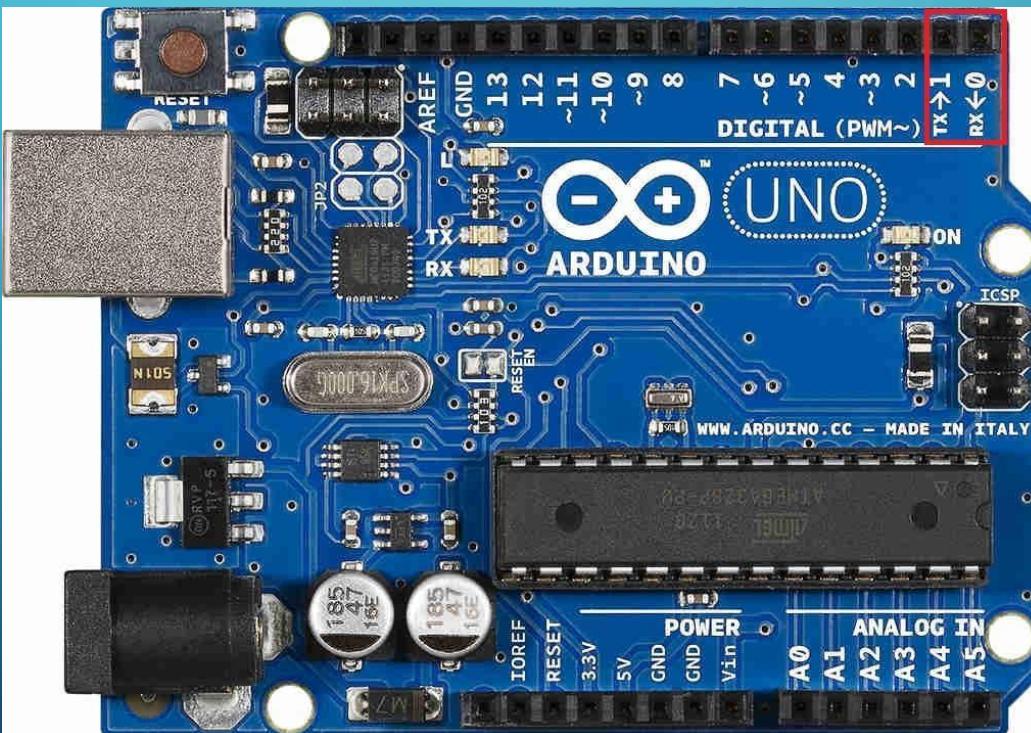
COMUNICAÇÃO ENTRE ARDUINOS

Para realizar comunicação entre dois Arduinos utiliza-se as portas seriais.

Dentre as formas de comunicação possíveis estão as transmissões por fio, ou por radio além de outras formas.

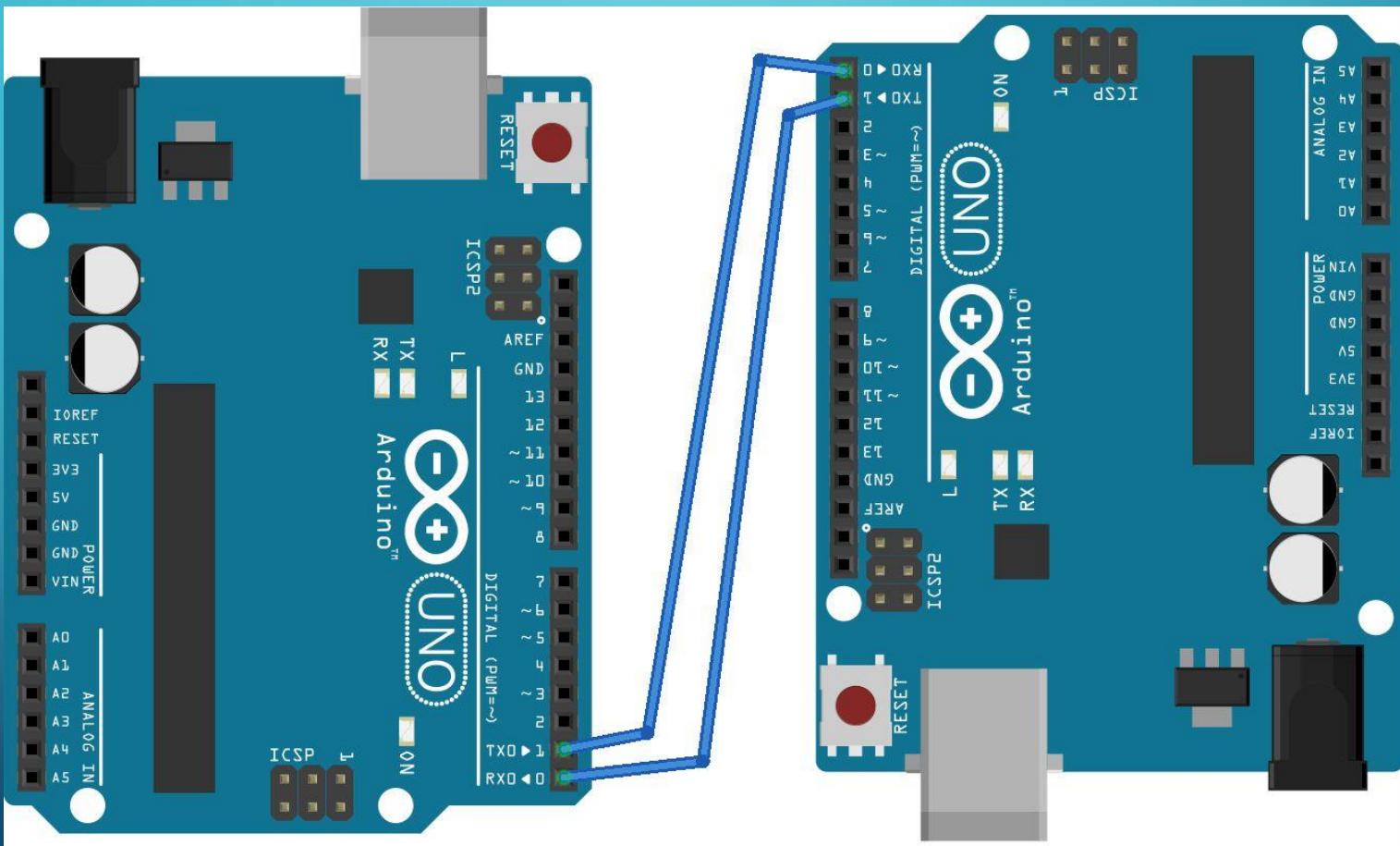
COMUNICAÇÃO ENTRE ARDUINOS

A comunicação entre arduinos por fio é realizada ligando as portas seriais dos Arduinos



Fonte: <http://robotechshop.com/shop/arduino/arduino-board/arduino-uno/>

COMUNICAÇÃO ENTRE ARDUINOS



fritzing

BIBLIOGRAFIA

arduino.cc

- Site oficial Arduino

vimeo.com/31389230

- Documentário Arduino

fritzing.org

- Software de desenho de circuitos

shieldlist.org

- Lista de Shields documentados

PERGUNTAS?

Site : www.colmeia.udesc.br

E-mail Murilo: murilo.on@hotmail.com

E-mail Ícaro: icaroaj@hotmail.com

E-mail Henrique: henriquewps@gmail.com

Obrigado!