



ETAPA *1*

O QUE É ARDUINO ?



Segundo o site WIKIPEDIA, Arduino é:

É um microcontrolador de placa única e um conjunto de software para programá-lo. O hardware consiste em um projeto simples de hardware livre para o controlador, com um Processador Atmel AVR e suporte embutido de entrada/saída. O software consiste de uma Linguagem de programação padrão e do bootloader que roda na placa.

É um pequeno computador que permite uma programação embarcada para controlar dispositivos de entrada e saída.



O primeiro ARDUINO foi criado em janeiro de 2005 no Instituto de Interatividade e Design, escola de Artes Visuais de Ivrea, Itália.

Criado a partir da ideia de dois professores:

David Cuartelle.

Massimo Banzi.

O objetivo é permitir que pessoas que não são especialistas no assunto pudessem desenvolver projetos no curso de Arte e Design.



Junto com outros especialistas criaram um Ambiente de Desenvolvimento Integrado, um software que permite a programação através de uma linguagem de alto nível como C converto o programa para linguagem de máquina que o hardware entende.

Todo o projeto segue o princípio do Open Source, ou seja de domínio público.

Com base nisso qualquer pessoa pode produzir as placas e alterar o software de acordo com sua necessidade ou para produzir novos resultados.

Esses estudos fazem parte do que hoje é conhecido como **COMPUTAÇÃO FÍSICA**.



ESTRUTURA DA PLACA DO ARDUINO

Baseado na linha de microcontroladores ATMEL - AVR.

AVR → Advanced Virtual RISC ou Alf and Vegard RISC.

RISC – Reduced Instruction Set Computer

ARDUINO	Diecimila	Duemilanove168	Duemilanove328	Mega
Processador	ATmega8	ATmega168	ATmega328	ATmega1280
Memória flash	8 K	16 K	32 K	128 K
Memória RAM	1 K	1 K	2 K	8 K
Memória EEPROM	512 bytes	512 bytes	1 K	4 K
Pinos digitais	14	14	14	54
Pinos analógicos	6	6	6	16
Saídas PWM	3	6	6	14

Os principais Arduinos e seus microcontroladores

ARDUINO BÁSICO



ESTRUTURA DA PLACA DO ARDUINO

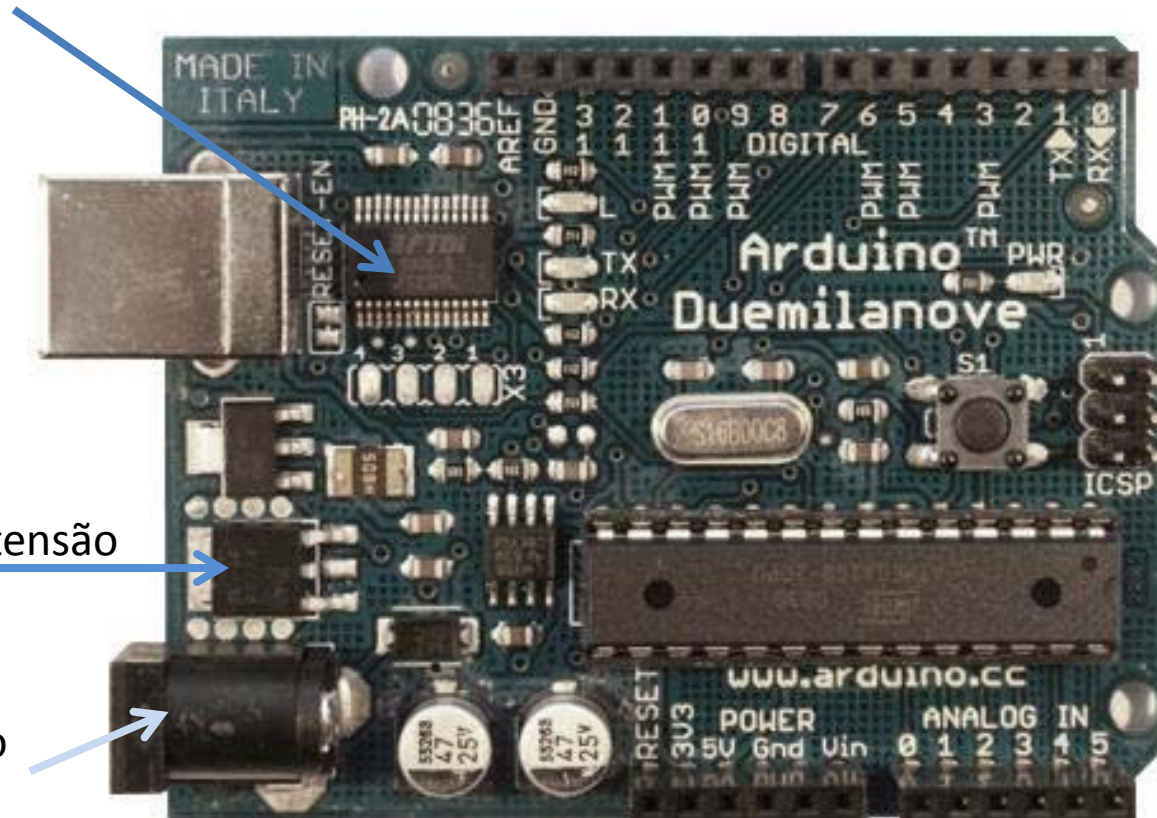


ARDUINO BÁSICO



ESTRUTURA DA PLACA DO ARDUINO

Conversor USB—Serial RS-232



Regulador de tensão
7805

Entrada de
Alimentação
Externa
12 Volts



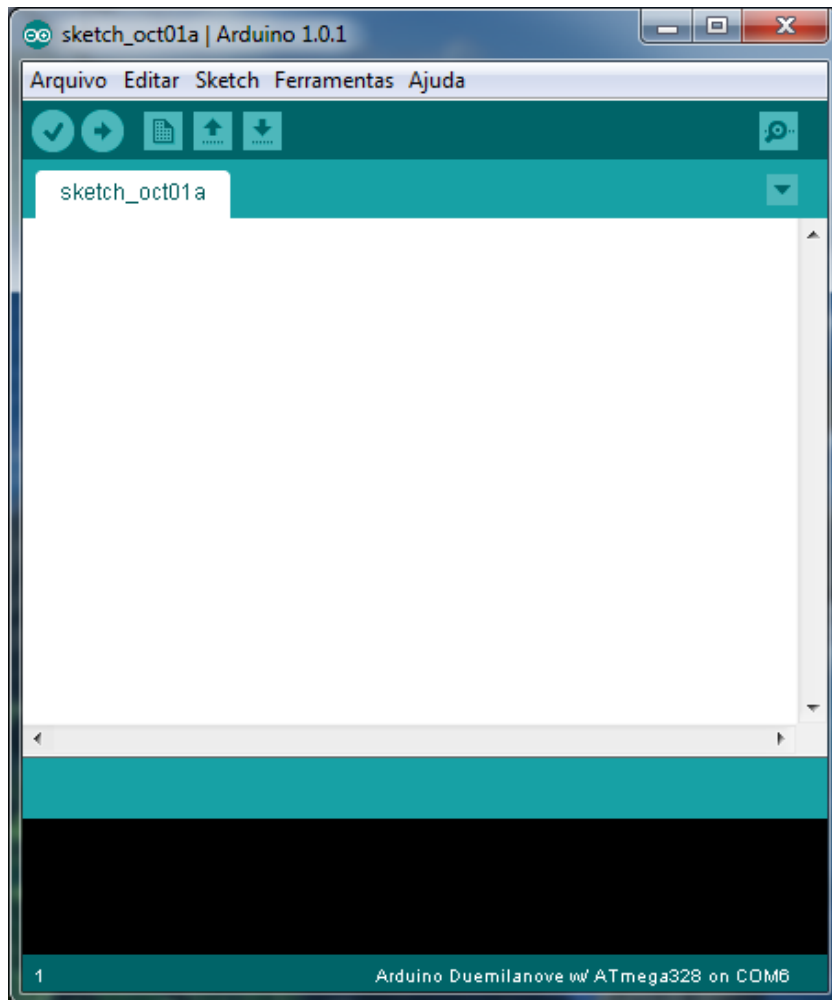
ESTRUTURA DA PLACA DO ARDUINO

- DIGITAL
 - Padrão TTL:
 - 0 a 0,8 V = 0
 - 2 a 5 V = 1
- ANALÓGICA
 - Conversor A/D de 10 bits:
 - 0 – 0 V
 - 1023 – 5V

ARDUINO BÁSICO



AMBIENTE DE DESENVOLVIMENTO INTEGRADO - IDE

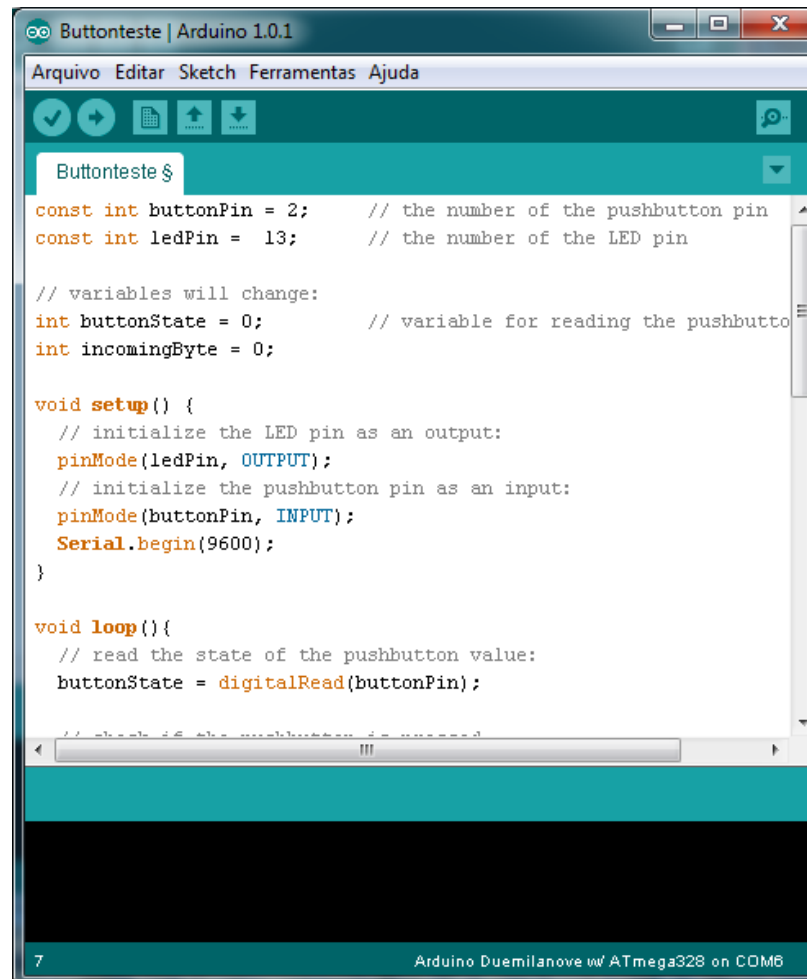


O ambiente de desenvolvimento do Arduino pode ser baixado do site: <http://arduino.cc/>
Não há a necessidade de ser instalado, apenas descompacte o arquivo no padrão ZIP.



A Linguagem – Baseada em C

Os programas do Arduino são conhecidos como sketch ou rascunho.

A screenshot of the Arduino IDE interface. The window title is 'Buttonteste | Arduino 1.0.1'. The menu bar includes 'Arquivo', 'Editar', 'Sketch', 'Ferramentas', and 'Ajuda'. The toolbar shows icons for opening, saving, and running. The main text area contains the following C++ code:

```
Buttonteste $
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton state
int incomingByte = 0;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed
  if (buttonState == HIGH) {
    // turn the LED on (HIGH)
    digitalWrite(ledPin, HIGH);
  } else {
    // turn the LED off (LOW)
    digitalWrite(ledPin, LOW);
  }
}
```

The status bar at the bottom indicates '7' and 'Arduino Duemilanove w/ ATmega328 on COM6'.



A Linguagem – Baseada em C

Elementos básicos de programação (**Constantes**):

Booleanas

Correspondem a valores lógicos True/False ou 0 / 1, usados tanto para os pinos digitais de entrada e saída.

High/Low

Definem o nível de tensão nos pinos do Arduino. High (nível alto) 5 Volts ou Low (nível baixo) 0 Volt.

Output/Input

Usadas com a função pinMode() definem se o pino especificado será configurado como saída (Output) ou entrada (Input).



A Linguagem – Baseada em C

Elementos básicos de programação (**FUNÇÕES**):

Todo programa deve ter duas funções obrigatórias:

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

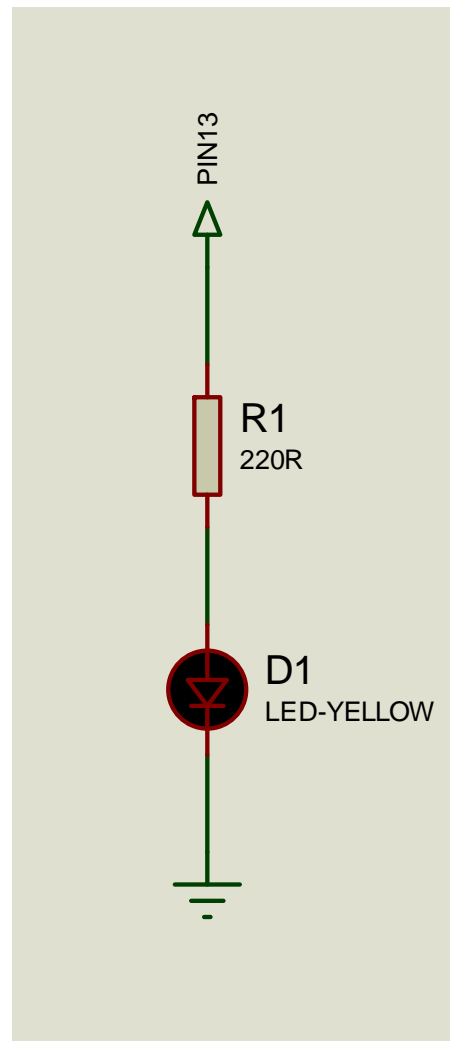
Esta função é responsável pelas configurações iniciais das portas e recursos do Arduino.

```
void loop() {  
    lógica do programa  
}
```

Na seção loop fica toda a parte lógica do seu projeto, esta função é executada indefinidamente.

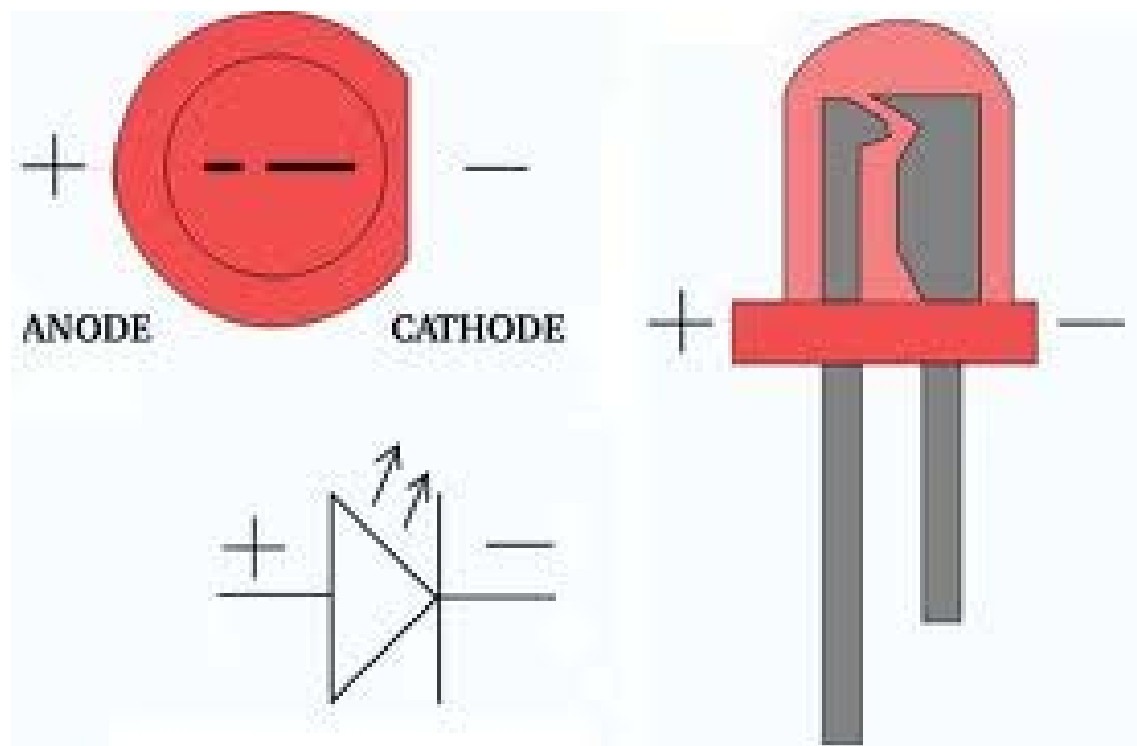
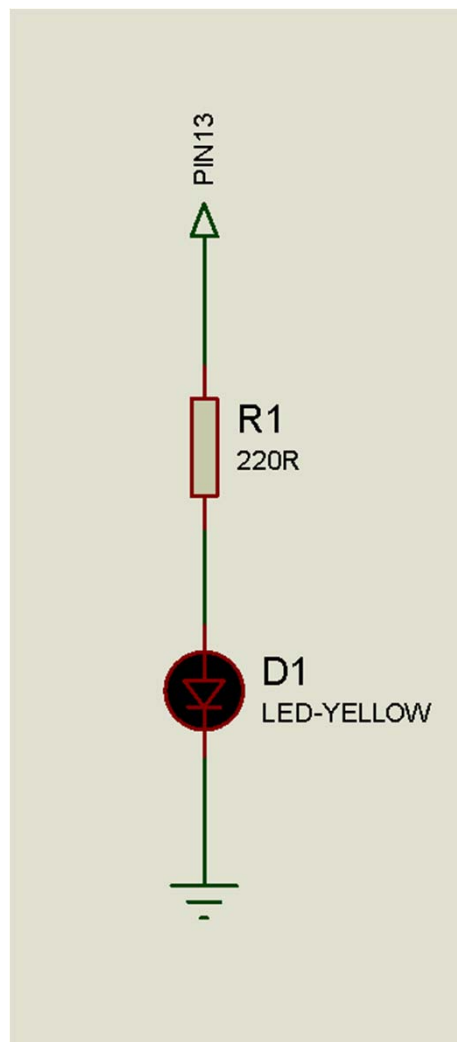


Componentes básicos – Iniciando os projetos



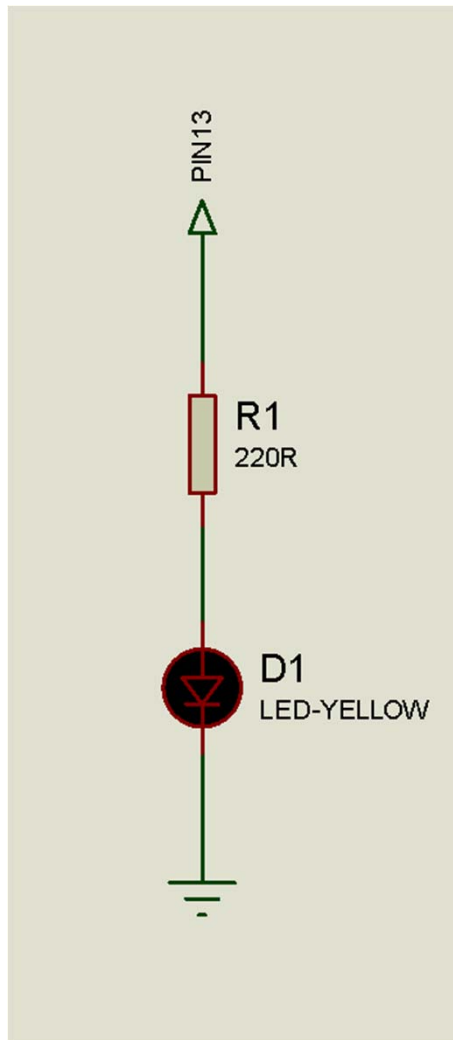


Componentes básicos – Iniciando os projetos – LED



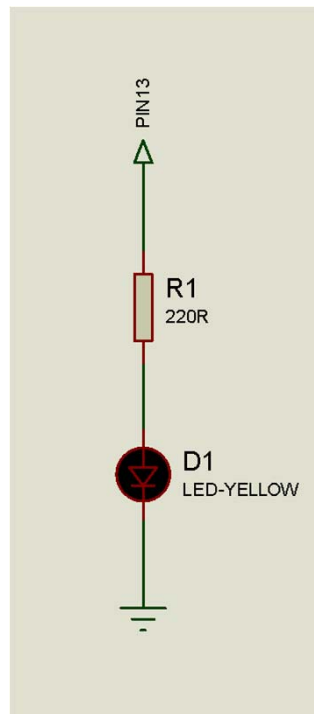


Componentes básicos – Iniciando os projetos – RESISTOR





Componentes básicos – Iniciando os projetos – RESISTOR



Cálculo de Resistores
Versão 1.1
Powered by Zanon

Cores das Faixas	1ª Faixa	2ª Faixa	3ª Multiplicador	4ª Tolerância
Preto	0	0	x 1	2%
Marrom	1	1	x 10	
Vermelho	2	2	x 100	
Laranja	3	3	x 1.000	
Amarelo	4	4	x 10.000	
Verde	5	5	x 100.000	
Azul	6	6	x 1.000.000	
Lilás	7	7		
Cinza	8	8		
Branco	9	9		
Ouro			x 0,1	5%
Prata				10%

Cálculo de valor do resistor pelas cores

1ª Faixa: Vermelho 2
2ª Faixa: Vermelho 2
3ª Faixa: Marrom x 10
4ª Faixa:

Calcular Limpar

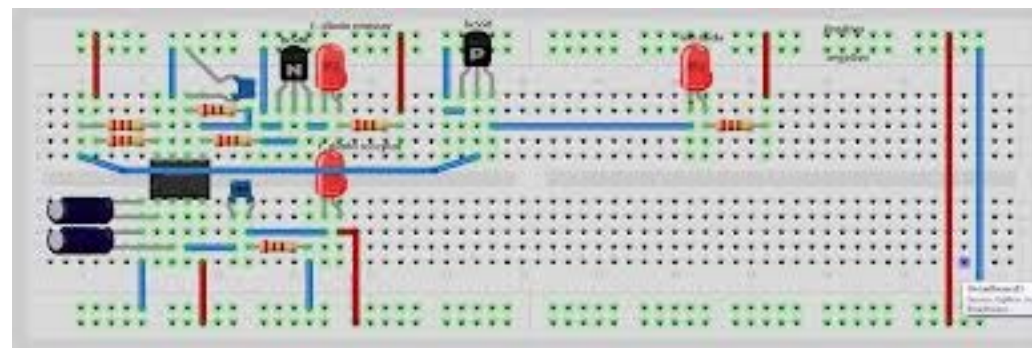
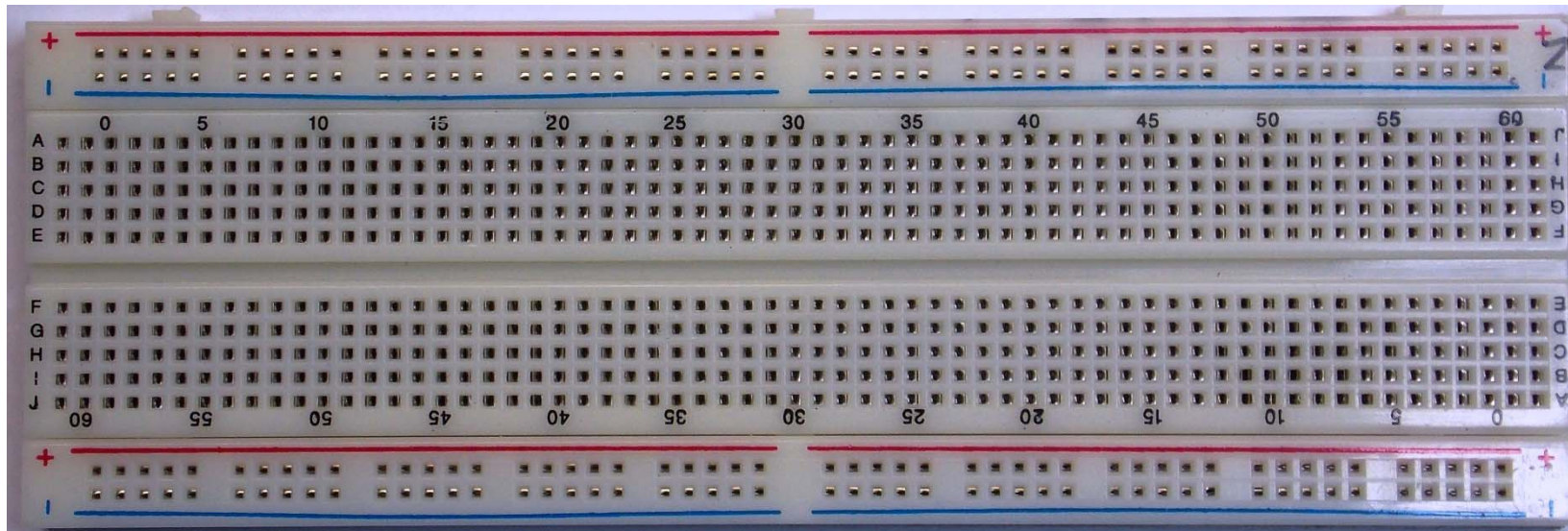
220 Ohms
0,22 Kohms
0,00022 Mohms

Verificando a cor de cada faixa pelo valor do resistor
220

Informe o valor do resistor em Ohms
2-2
Ver Cores



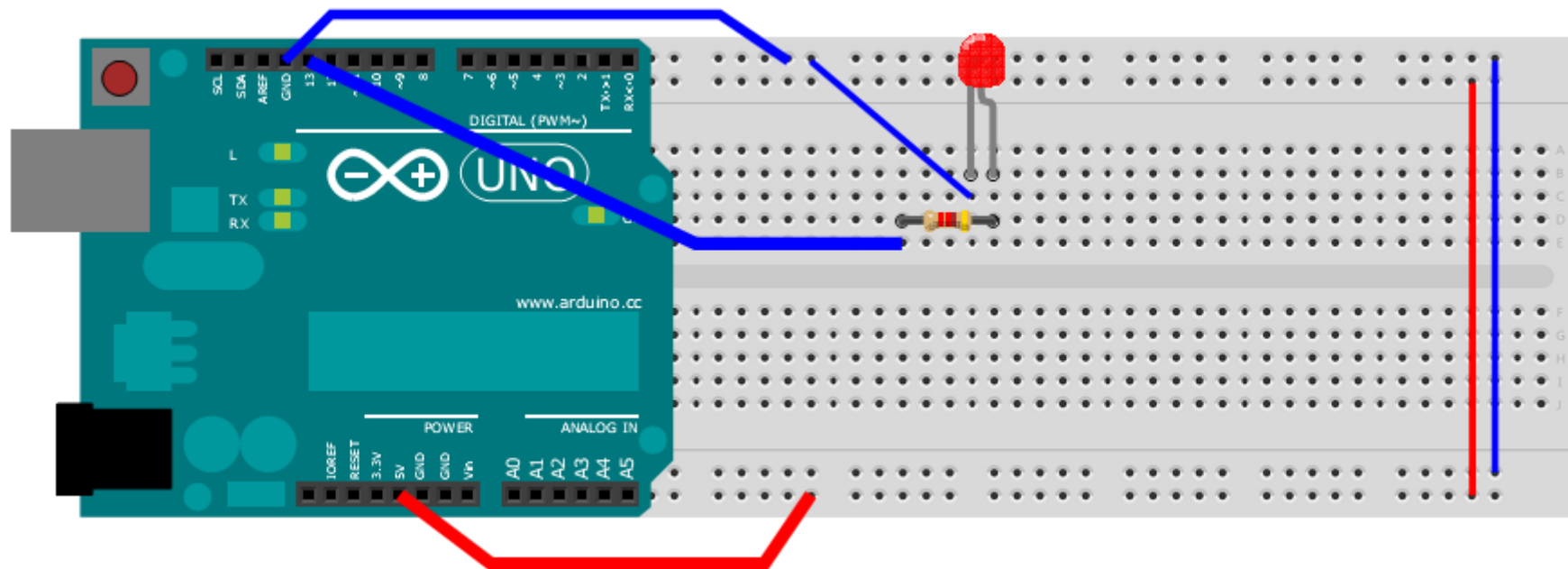
Componentes básicos – Iniciando os projetos – PROTOBOARD



ARDUINO BÁSICO



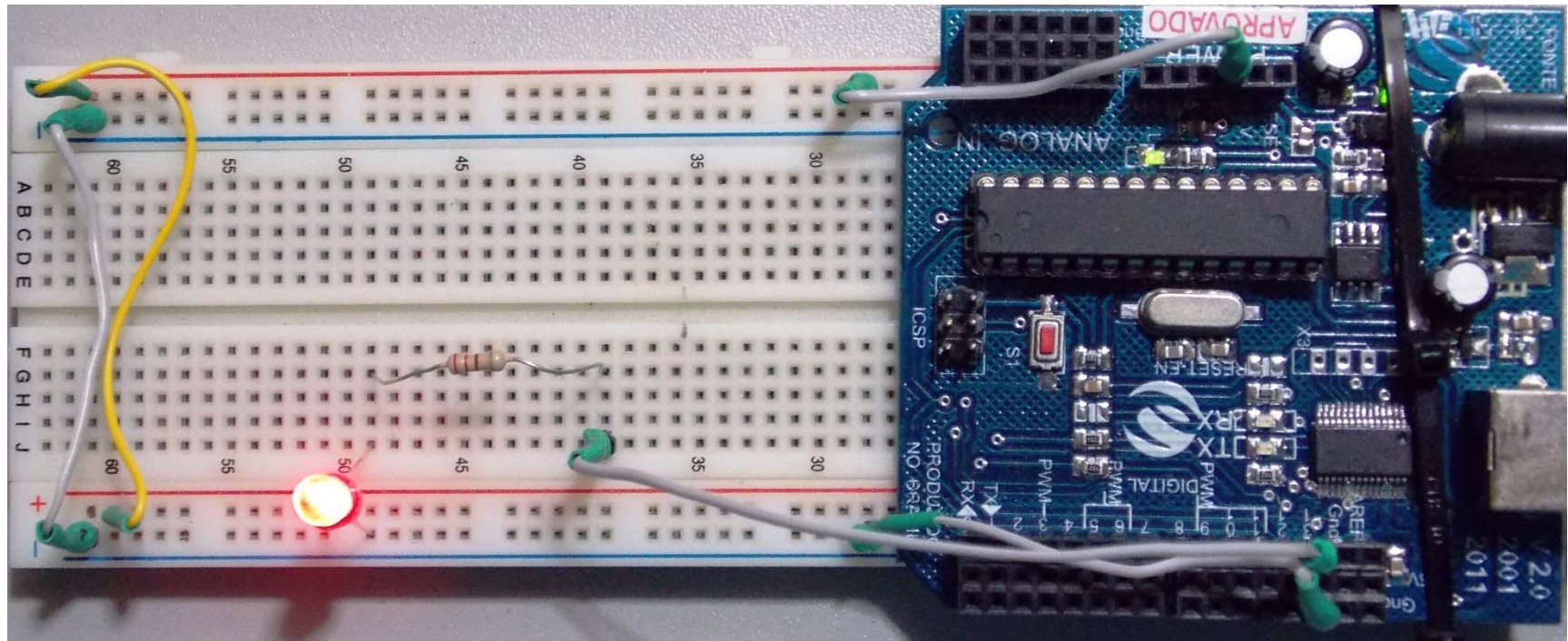
Componentes básicos – Iniciando os projetos – PROTOBOARD



ARDUINO BÁSICO



Componentes básicos – Iniciando os projetos – PROTOBOARD



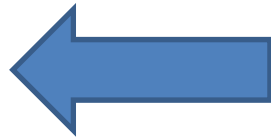


A Linguagem – Baseada em C

Elementos básicos de programação (**Declaração de variáveis e constantes**):

As variáveis e constantes de um programa devem ser definidas antes da função setup.

```
const int pino = 13;
```



```
void setup(){  
  pinMode(pino, OUTPUT);
```

```
}
```

```
void loop() {  
  digitalWrite(pino, HIGH);
```

```
}
```




A Linguagem – Baseada em C

Elementos básicos de programação (**Declaração de variáveis e constantes**):

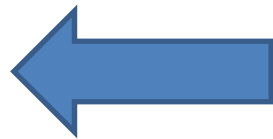
As variáveis e constantes de um programa devem ser definidas antes da função setup.

```
//Declaração das constantes e variáveis
```

```
const int pino = 13;
```

```
const int botao = 2;
```

```
int EstadoBotao = 0;
```



```
void setup(){
```

```
  pinMode(pino, OUTPUT);
```

```
  pinMode(botao, INPUT);
```

```
}
```

```
void loop() {
```

```
  digitalWrite(pino, HIGH);
```

```
}
```



A Linguagem – Baseada em C

Elementos básicos de programação (**Temporizadores**):

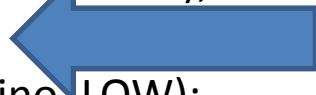
Os temporizadores permitem controlar pausas na execução dos programas.

```
const int pino = 13;  
const int botao = 2;
```

```
int EstadoBotao = 0;
```

```
void setup(){  
  pinMode(pino, OUTPUT);  
  pinMode(botao, INPUT);  
}
```

```
void loop() {  
  digitalWrite(pino, HIGH);  
  delay(1000);  
  digitalWrite(pino, LOW);  
  delay(1000);  
}
```





A Linguagem – Baseada em C

Elementos básicos de programação (**Temporizadores**):

Os temporizadores permitem controlar pausas na execução dos programas.

```
const int pino = 13;  
const int botao = 2;
```

```
int EstadoBotao = 0;
```

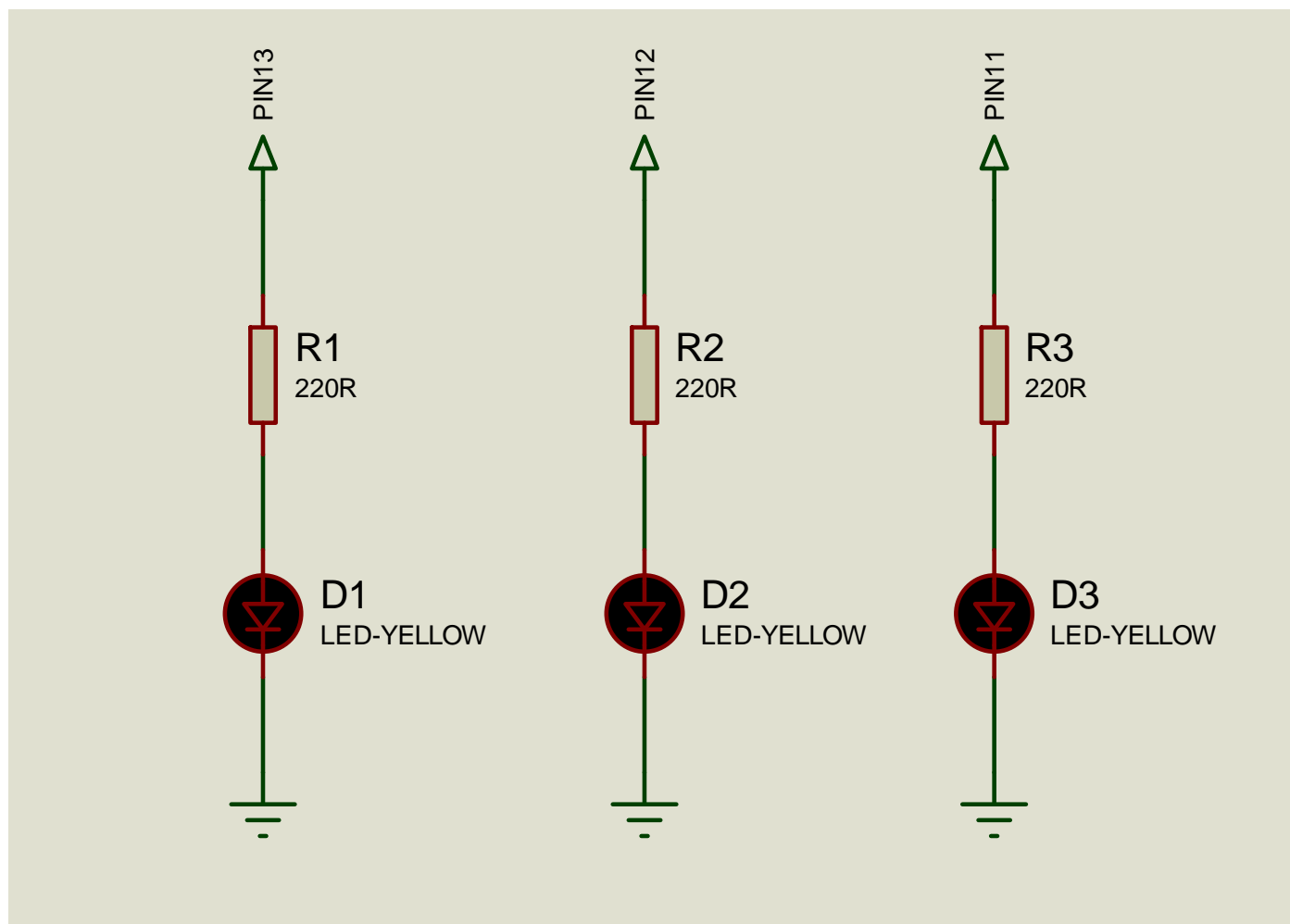
```
void setup(){  
  pinMode(pino, OUTPUT);  
  pinMode(botao, INPUT);  
}
```

```
void loop() {  
  digitalWrite(pino, HIGH);  
  delay(1000);  
  digitalWrite(pino, LOW);  
  delay(1000);  
}
```

Faça algumas experiências
mudando o tempo do
temporizador.

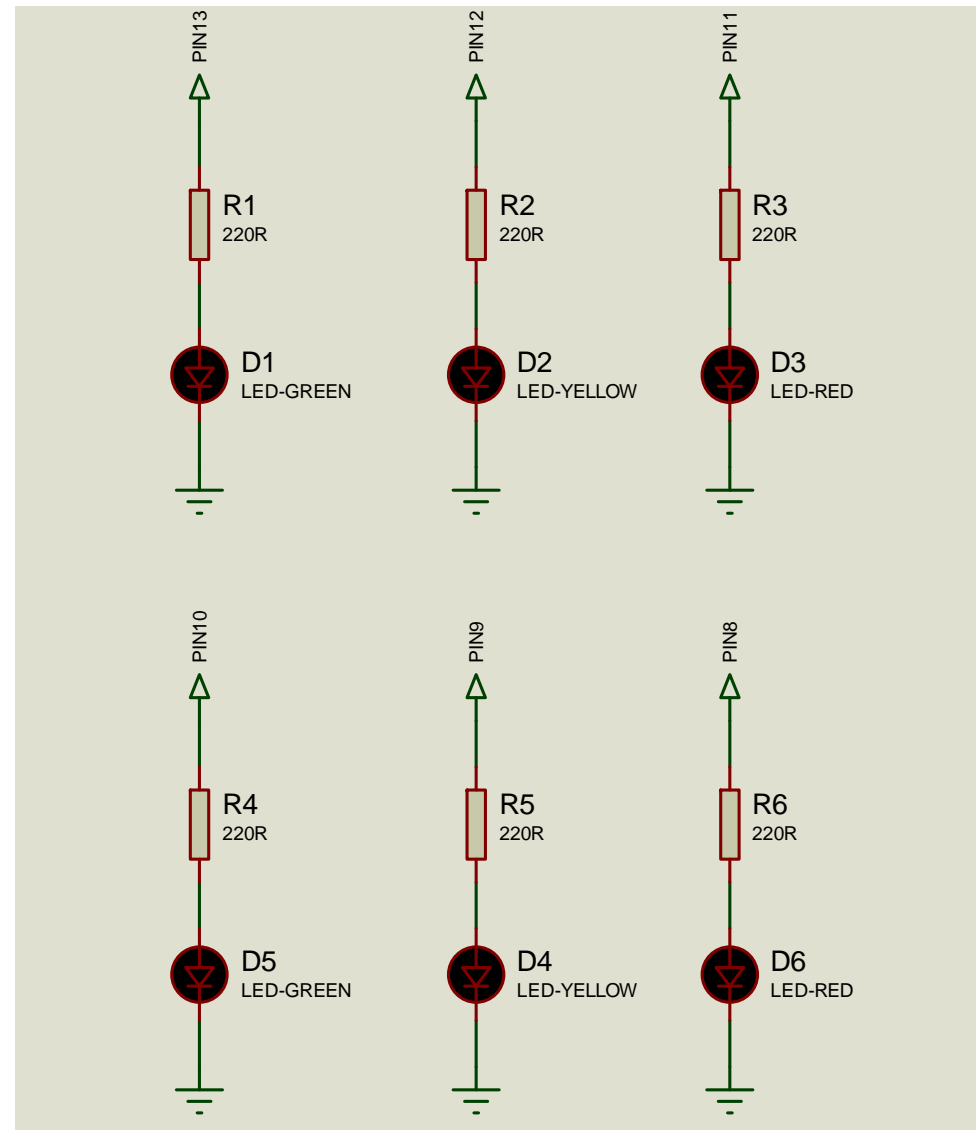


PROJETO PRÁTICO





PROJETO PRÁTICO – Desenvolva um sistema de semáforo

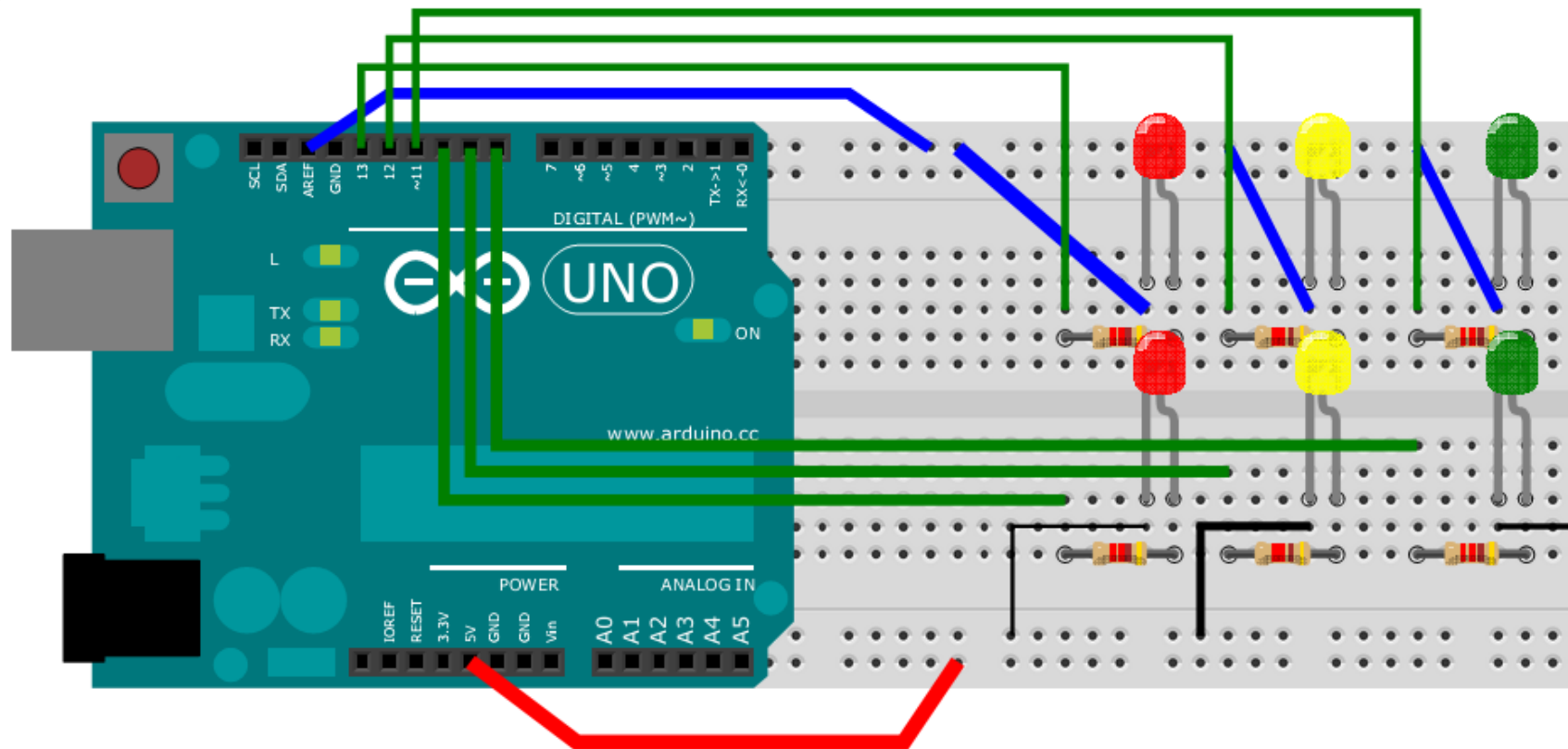




PROJETO PRÁTICO – Desenvolva um sistema de semáforo

Temporizar de maneira que facilite o seu entendimento.

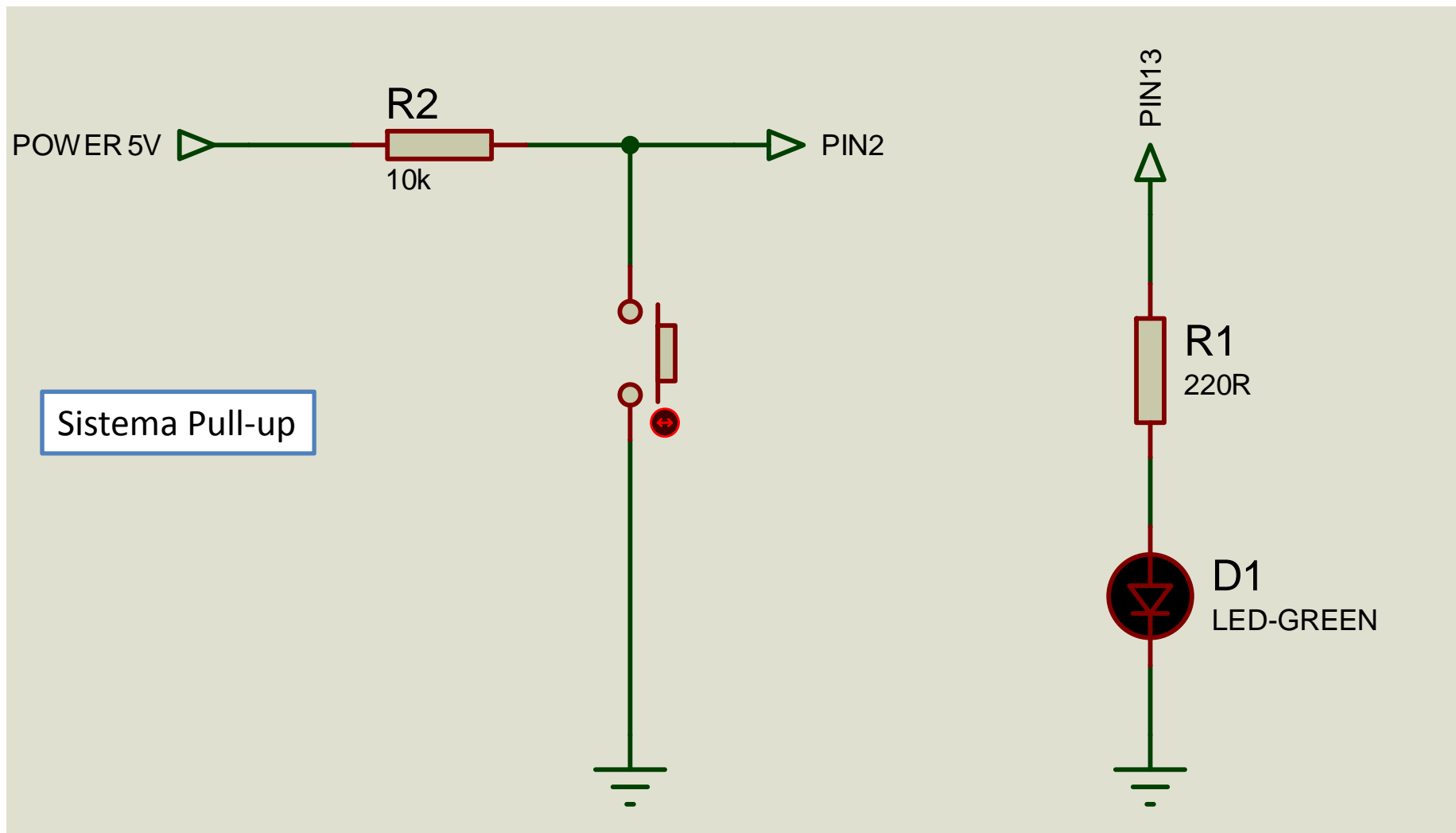
Quando um dos sinais está em verde, deve-se passar primeiro para amarelo e em seguida para o vermelho.



ARDUINO BÁSICO



Trabalhando com entrada de sinal digital – Monte o circuito abaixo

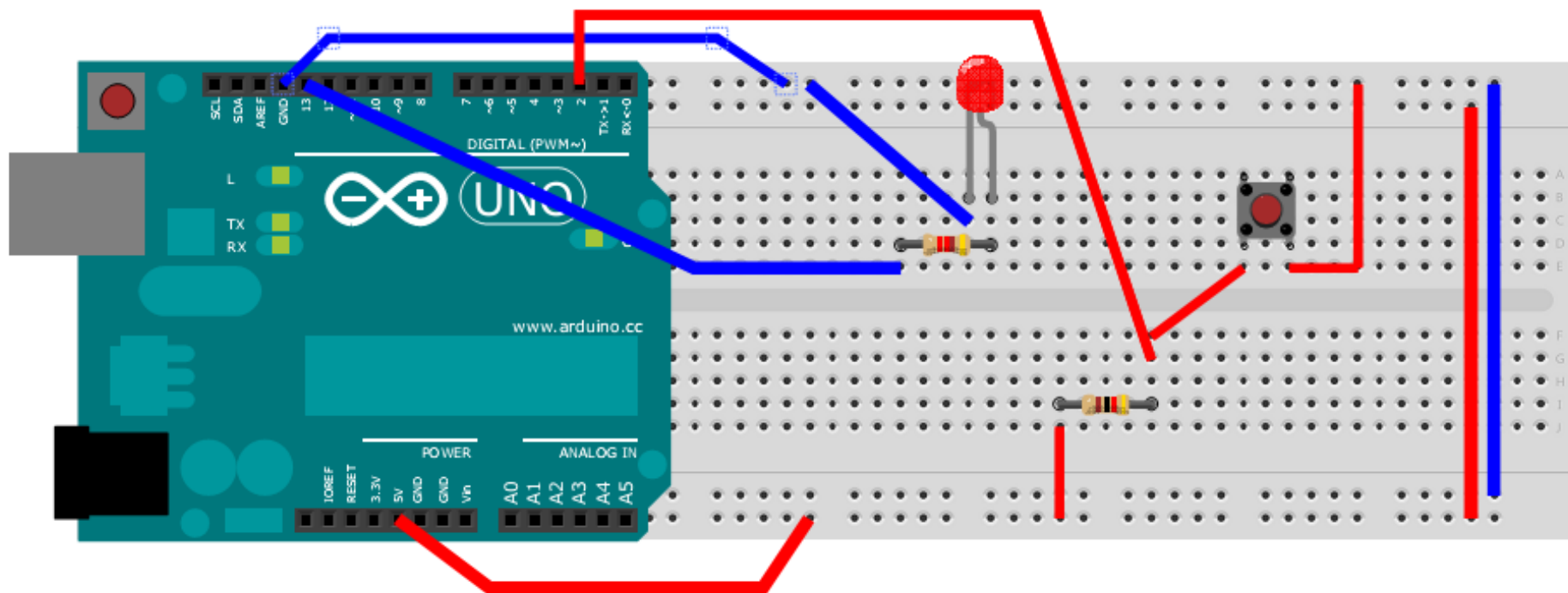


ARDUINO BÁSICO



Trabalhando com entrada de sinal digital

Sistema Pull-up





Trabalhando com entrada de sinal digital

```
//Declaração das constantes e variáveis
```

```
const int pino = 13;
```

```
const int botao = 2;
```

```
int EstadoBotao = 0;
```

```
void setup(){
```

```
  pinMode(pino, OUTPUT);
```

```
  pinMode(botao, INPUT);
```

```
}
```

```
void loop() {
```

```
  EstadoBotao = digitalRead(botao);
```

```
  if (EstadoBotao == LOW) {
```

```
    digitalWrite(pino,HIGH);
```

```
  }
```

```
  else {
```

```
    digitalWrite(pino,LOW);
```

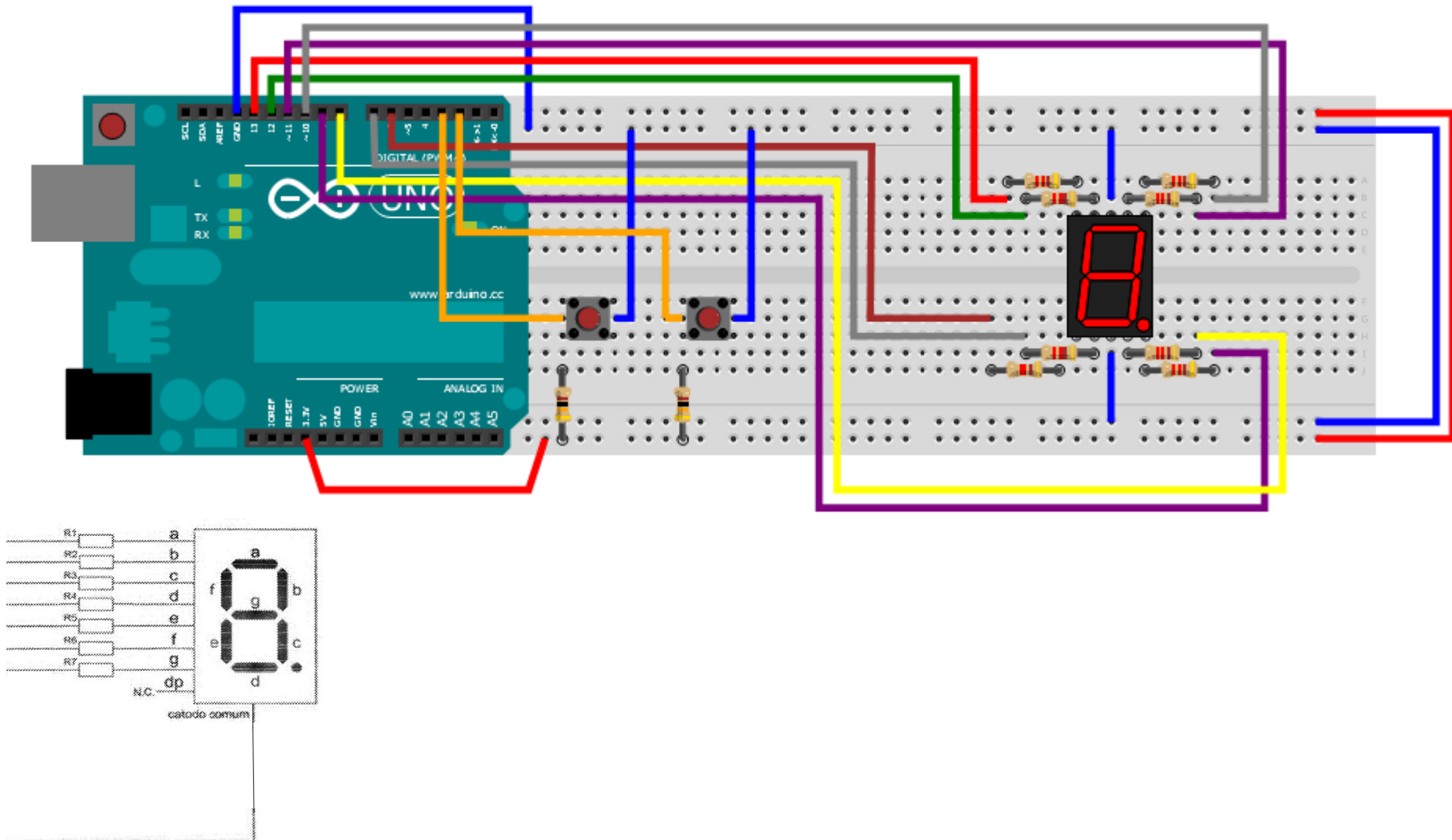
```
  }
```

```
}
```

ARDUINO BÁSICO



DISPLAY DE 7 SEGMENTOS





ETAPA

2

PORTAS ANALÓGICAS



Portas analógicas permitem o controle de sinais que variam de 0 a 5 V.

Utilizando-se de um conversor analógico/digital A/D de 10 bits teremos então uma variação do valor da porta conforme a tabela abaixo:

0 V \rightarrow 0

2,5 V \rightarrow 512

5 V \rightarrow 1023

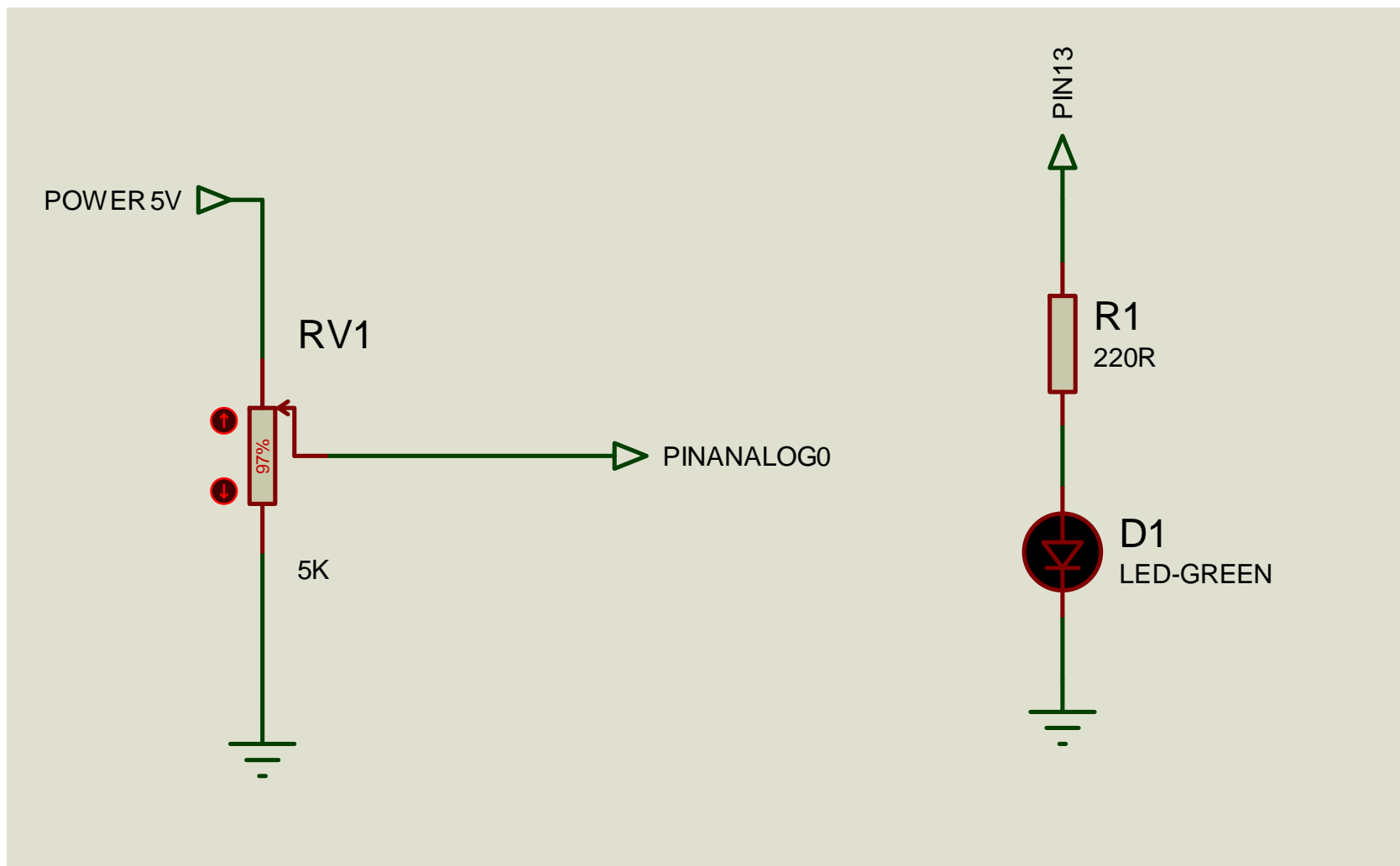
Ou seja, cada unidade lido na porta corresponde a

5 V / 1024 ou

0,0048828125 V por unidade \rightarrow Aproximadamente 4 mV por unidade lida



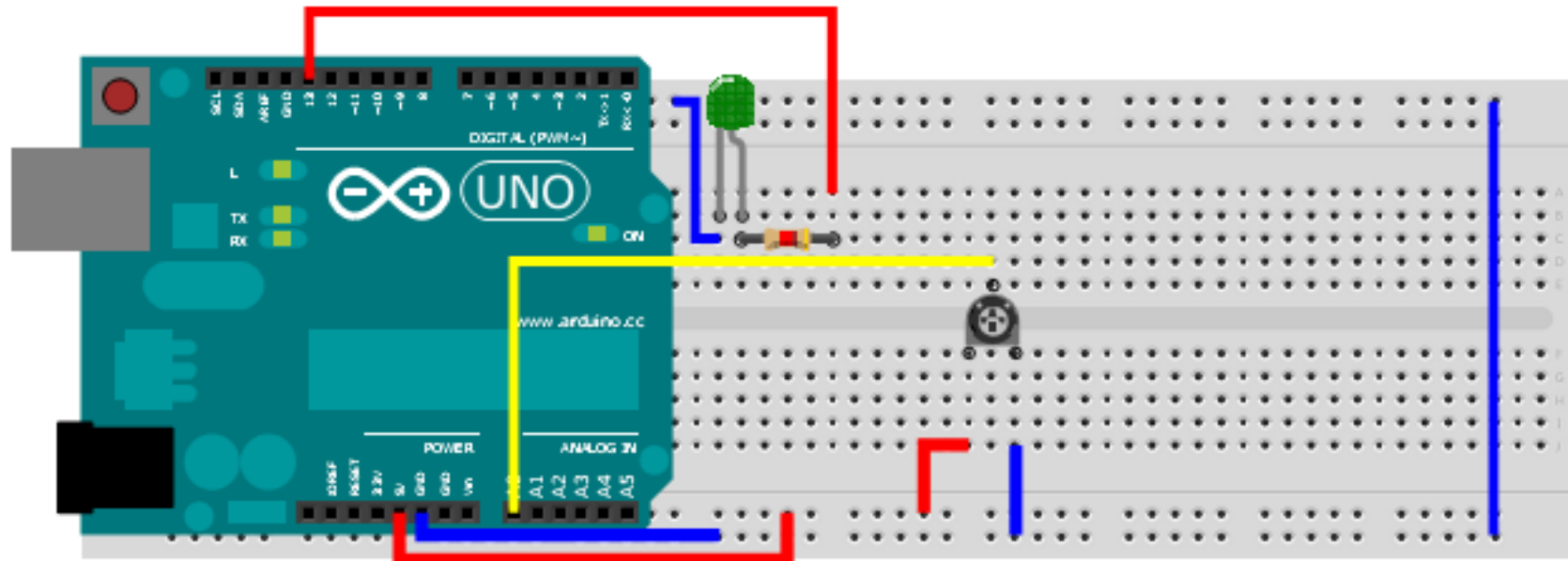
Trabalhando com sinal analógico



ARDUINO BÁSICO



Trabalhando com sinal analógico





Trabalhando com sinal analógico
– Acendendo o Led quando a
tensão superar 2,5V.

```
const int pino = 13;
int Entrada = A0;
int ValorPorta = 0;
float tensao = 0;

void setup(){
    pinMode(pino,OUTPUT);
}

void loop() {
    ValorPorta = analogRead(Entrada);
    tensao = 0.00488758 * ValorPorta;
    if (tensao > 2.5) {
        digitalWrite(pino,HIGH);
    }
    else
    {
        digitalWrite(pino,LOW);
    }
    delay(100);
}
```



Referências bibliográficas.

Banzy, Massimo (2009). Getting Started with Arduino, 118p. 1.ed. Sebastopol-CA-EUA: O'Reilly.

McRoberts, Michael (2011). Arduino Básico, 453p. 1.ed. São Paulo-SP-Brasil: Novatec.