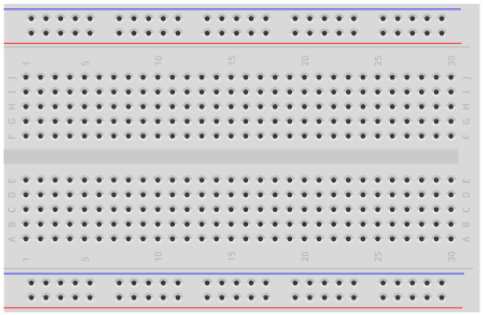
*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

NOME:_____

## O QUE É UMA PROTOBOARD

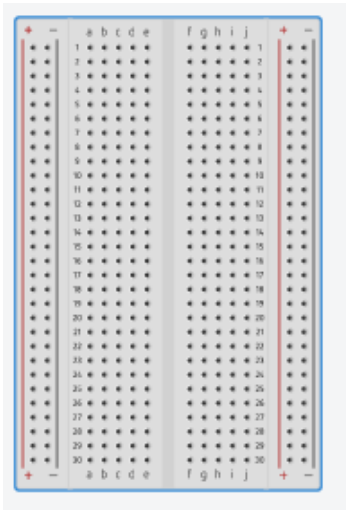
Matriz de contato, também chamada de protoboard, é um componente muitíssimo importante no mundo maker: ela te ajuda a criar protótipos sem a necessidade de uso de soldas. Vamos assim conhecer o seu funcionamento.

Na figura a seguir você confere a foto de uma matriz de contato[1].



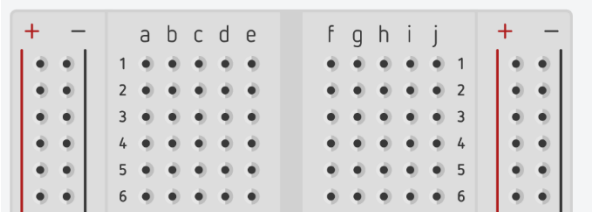
Fonte: Eletrogate

No tinkercad ela é chamada de “Placa de ensaio pequena”.



Fonte: Tinkercad

Observe as referências em cada linha e coluna. Veja isso num zoom na figura abaixo:




Fonte: Tinkercad

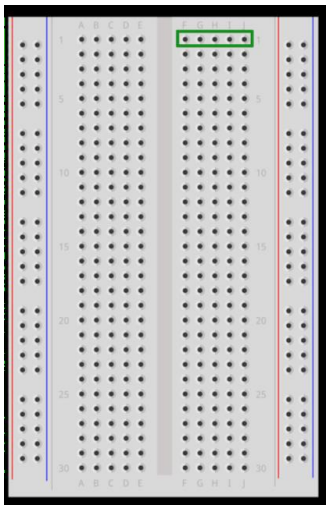
Veja que temos duas linhas verticais à esquerda e duas linhas verticais à direita. Internamente, todos os pinhos paralelis ao simblolo de + estão conectados como se fosse um único fio.

Os pinos paralelos à linha preta com o símbolo de – também estão conectados entre si, mas estão separados dos pinos com sinal de +. Além disso, os cinco pinos da esqueda (1a, 1b, 1c, 1d e 1e) estão conectados como se fossem um único fio bem como os pinos af, 1g, 1h, 1i e 1j.

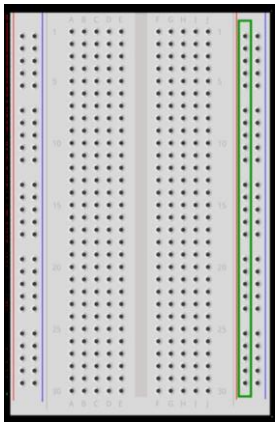
Veja como é uma protoboard internamente:

Fonte: Eletrogate

Veja nas duas figuras a seguir dois destaques de pinos que estão conectados entre si:



Fonte: Eletrogate



Fonte: Eletrogate

Legal, mas como faremos as conecções? A resposta é dada no próximo item.

[1] Importante: as figuras aqui presentes foram removidas dos materiais da ELETROGATE e todos os seus materiais podem ser obtidos gratuitamente no site https://www.eletrogate.com/pagina/apostilas.html

Tais apostilas serão minhas principais fontes de informação para montar este material.

*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## O QUE SÃO JUMPERS

No momento de aulas online, não fará muita diferença se você souber o que são jumpers, mas caso você tenha a oportunidade de fazer os experimentos em sua casa, por exemplo, já será importante saber o que são e quais os tipos.

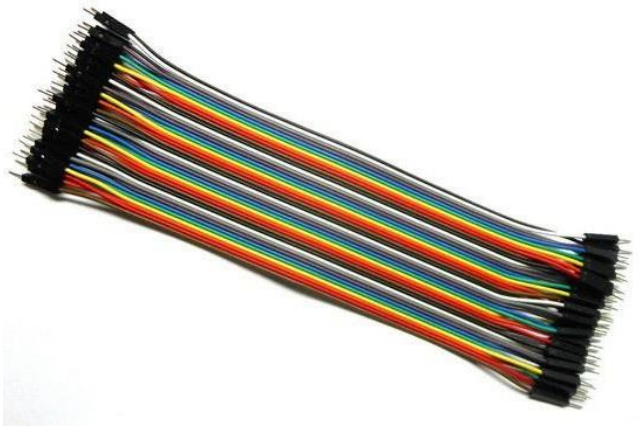
Existem basicamente dois tipos de conectores, o tipo M e o tipo F. Veja detalhes deles nas figuras s seguir.

Conector tipo M é o que possui uma ponta metálica.

Conectores tipo F possui um buraquinho. Note a semelhança com as tomadas de sua casa.

Com isso, nos podemos nomear os jumpers de acordo com os conectores em ambas as pontas.

Conectores tipo F x F por terem terminais tipo F em ambas as extremidades



Conectores tipo M x M por terem terminais tipo M em ambas as extremidades

Conectores tipo M X F possuem um lado tipo M e outro tipo F

No tinkercad, isso não terá importância, uma vez que as conexões serão representadas por linhas coloridas.

Detalhe dos conectores tipo M e tipo F.

Vamos praticar um pouco: vamos montar um circuito no inkercad.

*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## NOSSO PRIMEIRO CIRCUITO

Lembre-se que para entrar no Tinkercad você deve acessar o seguinte link:

https://www.tinkercad.com/joinclass/GTI4ZAB7Z1V3

Depois deve entrar com seu nick que compreende de seu promeiro nome (com todas as letras em minúsculo) e a primeira letra seguinte no seu nome. Por exemplo, se você se chama José da Silva então seu nick será joaod. Se você criou uma conta, então basta clicar no lonk acima e entrar com seu login normalmente.

Montemos o circuito a seguir. Você deverá fazer isso na sala de aula do tinkercad. Se não der para fazer isso junto com o professor, não se preocupe. O importante é que tente fazer isso sozinho(a) e faça até o final.

No tinkercad, selecione os seguintes elementos:

1. Placa de ensaio pequena
2. Bateria 9V
3. LED
4. Resistor

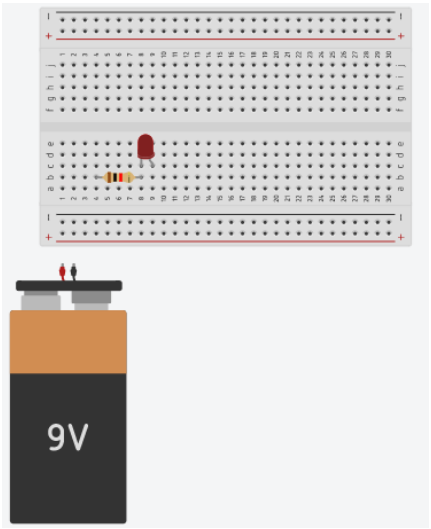
Rotacione a bateria e resistor para ficar conforme o esquema a seguir. Depois clique no resistor e mude o valor da resistência para 1000 Ω. Por ora, não se preocupe com o que cada coisa dessa significa, está bem?

Para rotacionar, clique no botão em cima e a esquerda:

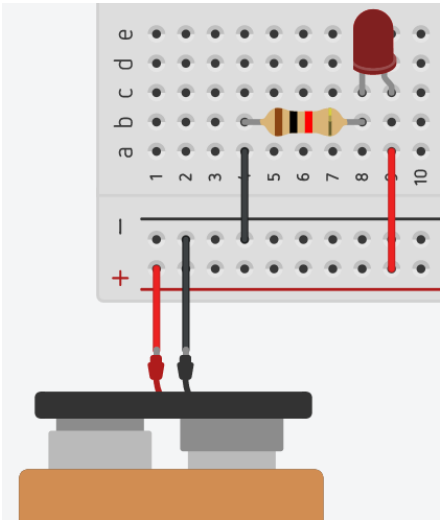
Para ajustar o valor da resistência, clique duas vezes no componente

e depois mude os dados conforme a figura a seguir


Agora, coloque os componentes conforme a imagem a seguir.

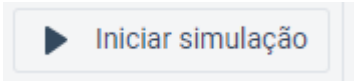



Por fim, conecte os fios conforme a figura abaixo, bastante ampliada para você conseguir ver as letras e números na protoboard (placa de ensaio).

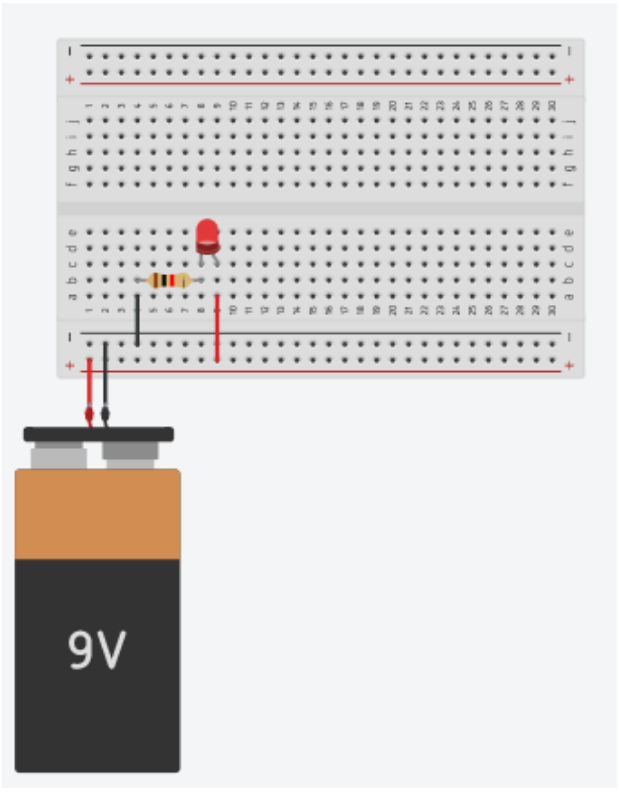


Detalhes das ligações. Note que a perninha torta do LED deve estar ligado no terminal vermelho da Bateria 9V. Se ligar ao contrário não funcionará e isso ocorre porque o LED é um elemento dito polarizado. O resistor, por outro nado, não é polarizado e por isso pode ter as ligações em seus terminais invertidas.

Ao concluir, clique no botão




e pronto, o seu primeiro circuito deverá funcionar.



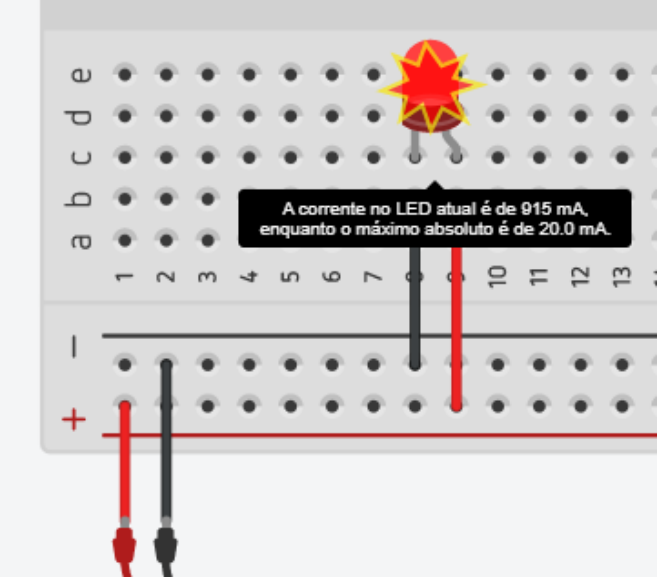

Circuito finalizado no Tinkercad. Note que ao iniciar a simulação, o LED vermelho irá ascender.

*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## O QUE SÃO RESISTORES E LEDs?

Estes dos componentes eletrônicos serão mais bem estudados ao longo do ano. Por hora, o que você precisa saber é que sem o resistor, no elemplo que acabamos de fazer, o LED irá queimar, isto é, ele deixará de funcionar na vida real.

Vamos montar o circuito anterior sem o LED para ver o que acontece.




Quando não conectamos um resistor para que o LED receba menos energia que o limite por ele suportado, o Tinkercad nos da uma mensagem representando que o LED será danificado e poderá deixar de fundionar se montar o circuito tal como foi simulado. Ao passar o mouse por cima do LED, um texto é exibido na tela para explicar o motivo.

Sendo assim, caso você consiga montar os esquemas apresentados em aula aí, na sua casa, com componentes de verdade, você deverá tomar muito cuidado para não danificar nenhum componente, caso contrário ele não mais fundionará.

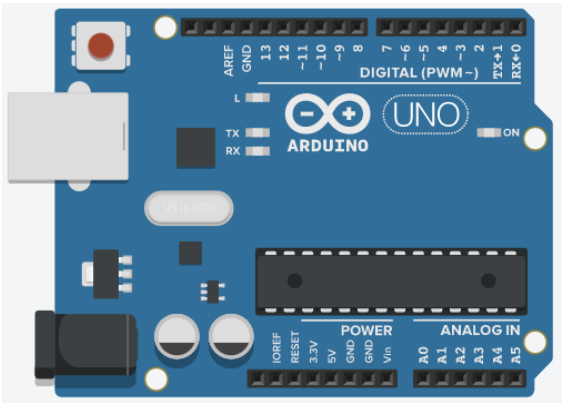
Vamos agora para o que interessa: o Arduíno.

## O QUE É O ARDUINO

O Arduino é uma placa que pode ser programada usando uma linguagem de programação apropriada. Ao ser programado, ele pode receber sinais elétricos (que chamaos de sinais de entrada) ou mesmo enviar sinais elétricos.

Ao receber sinais elétricos vindos, por exemplo, de um termômetro apropriado, ele [o Arduíno] poderá te informar a temperatura medida por aquele termômetro.

Ao enviar um sinal, o Arduino poderá, por exemplo, ligar um LED.

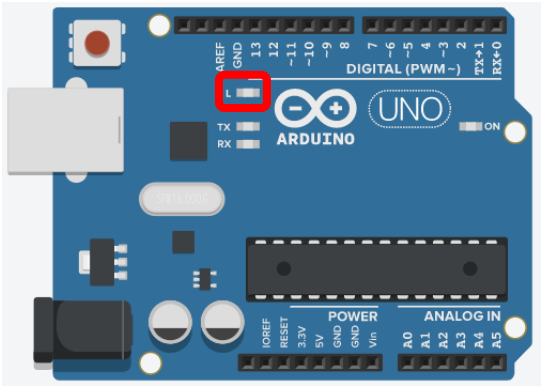



Fonte: Tinkercad

O Arduino pode muito mais que isso!!! Ao programá-lo, você dará um conjunto de instruções à ele para que ele possa efetuar um conjunto bastante longo e complexo de instruções como, por exemplo:

- Ligar o ar-condicionado de uma residência se a temperatura ambiente passar de 25°C;
- Medir a umidade do solo e se ele estiver muito seco, ligar uma bomba d'água para irrigar a vegetação nele contida;
- Verificar se está chovendo e acionar um toldo para que ele se abra ou feche de acordo com o que se deseja;
- Medir a velocidade de um carro ao se comunicar com um dispoditivo GPS no carro;
- Realizar todas as operações acima simultaneamente e muitas outras.

Ficou interessado(a)? Então, vamos criar um programa muito simples para ligar um LED que está incorporado ao Arduíno.



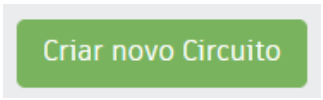

Detalhe do LED incorporado no Arduino. Veja como controlar este LED usando o Tinkercad.

*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## ASCENDENDO O LED INCORPORADO AO ARDUINO

Voltemos à nossa sala no tinkercad. Do lado do simbolo

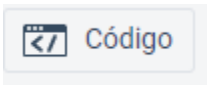
você pode escolher um nome para o seu circuit: que tal “meu primeir circuito”?. Ao clicar neste botão você volta para a página inicial onde é possível criar um novo circuito. Clique em



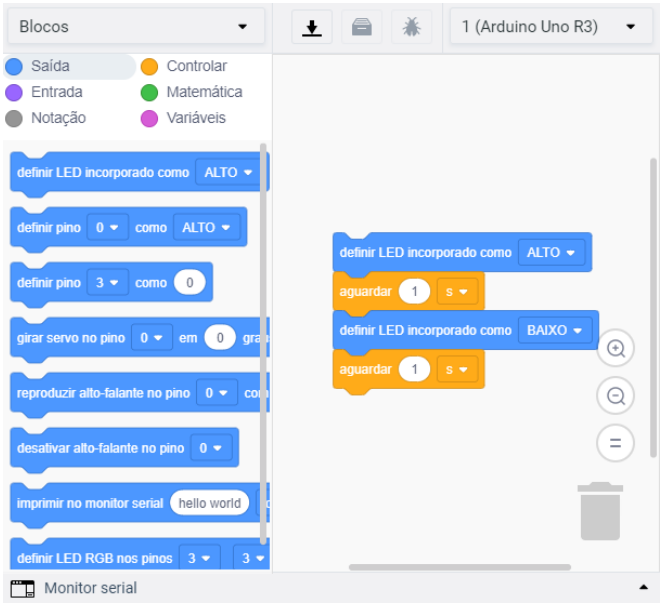

e siga os passos a seguir para criar um circuito que controla um LED.

Na barra lateral procure por:

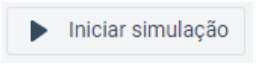

Clique neste elemento e clique na parte em branco da tela, no local onde você cria circuitos, para adicioná-lo. Na parte superior direita você encontra um botão chamado “Código”




que você deverá clicar. Após clicar, irá aparecer uma região na qual você pode criar programas para controlar o Arduino. Provavelmente o que você encontrará é algo como o que está representado abaixo:

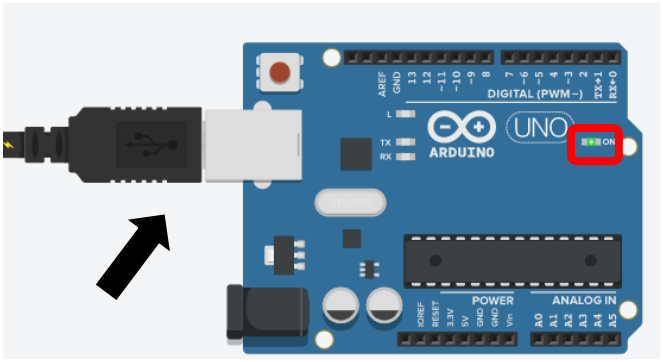



Como padrão, o Tinkercad já vem com um código. Vamos discuti-lo depois, pois agora o que você deve fazer é clicar em

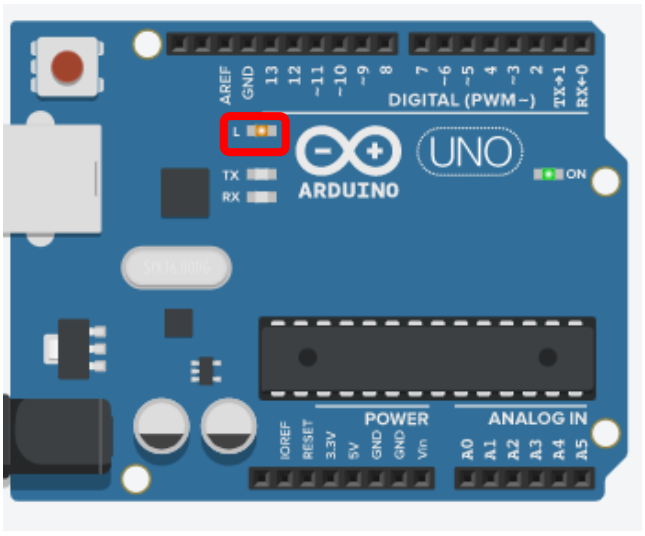



.

Se estiver tudo verto, você verá o LED incorporado do Arduino piscar.




Arduino conectado: a seta preta indica que o Arduino está conectado, isto é, está sendo simulada a situação em que você conecta o Arduino no PC e envia o programa para ele [Arduino]. Veja o LED destacado com o retângulo vermelho: ele representa que o Arduino está ligado e operando.



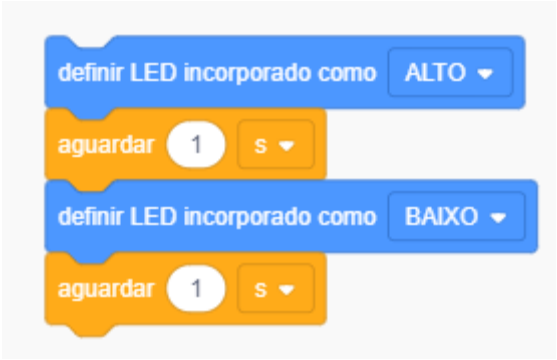

Observe na imagem acima que o LED incorporado à placa do Arduino está piscando.

Vamos analisar o código que faz isso com o Arduino.

*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## ANÁLISE DO CÓDIGO

Na figura a seguir vemos o código pré implementado para controlar o Arduino. Não precisamos fazer nenhuma programação por hora, pois ela já vem pronta para nós.




Veja que numeramos as linhas para ficar mais fácil. Vamos descrever o que cada linha do programa faz:

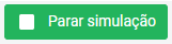
1. Liga o LED L na placa do Arduino;

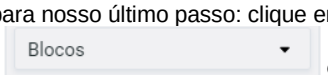
2. Espera por 1 segundo. Como o LED L está ligado, ele continuará asceso por este tempo;

3. Desliga o LED L;

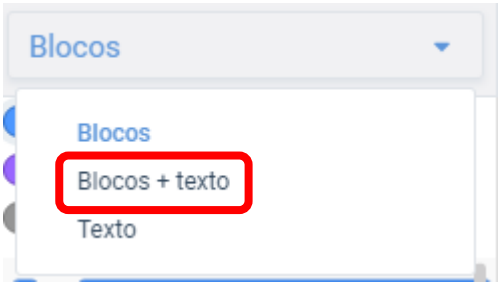
4. Espera por 1 segundo.

Note que depois que todo o código for executado, o programa se inicia e fica em um ciclo “infinito” e apenas para de executar se o Arduino for deligado.

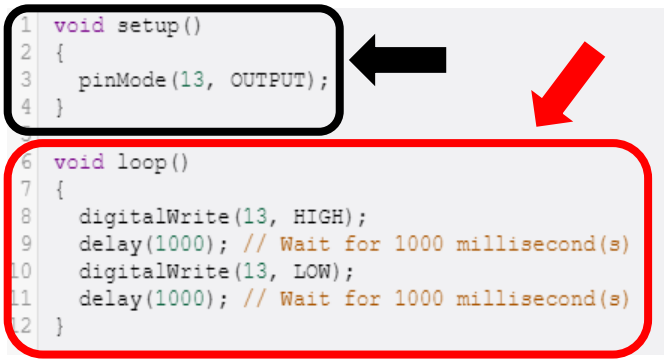
Vamos para nosso último passo: clique em 

e depois em 

e selecione “Bloco + Textos”:




Você verá o código deste programa na linguagem do Arduino



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

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Todo código em Arduino é sempre composto de pelo menos duas partes:

1. Void setup();

2. Void loop().

No primeiro, são feitas as configurações sobre quais os pinos do Arduino serão usados como entrada e quais serão usados como saídas. No segundo, é onde ficam os comandos propriamente ditos.

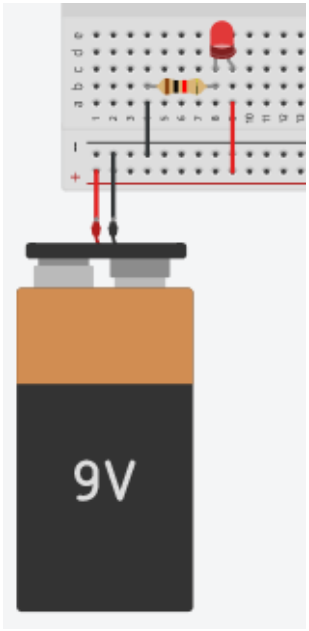
Vamos abordar isso melhor nas próximas aulas. Por enquanto é isso o que temos para a aula dessa semana.

**O professor irá verificar quem fez ou não fez os dois circuitos acima. Então estas atividades são obrigatórias.**

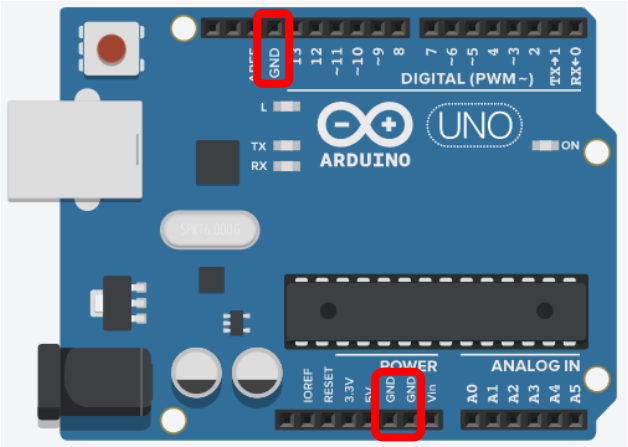
## ENTRADA E SAÍDA

### GND

Como vimos na aula passada, a energia elétrica deverá fluir do positivo ao negativo da bateria. No Arduino, o nagativo será chamado se GND (ou Ground ou ainda chão/terra).




Na figura abaixo venmos as portas com o negativo, tal como o menos (-) em uma bateria ou pilha.



*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## PORTAS DIGITAIS

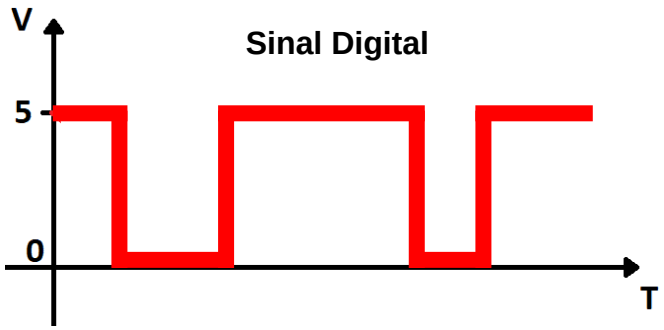
Os pinos de 0 a 13 no Arduino são chamados de portas digitais e os pinos de A0 até A5 são chamados de portas analógicas.

Os pinos de 0 a 13 podem fornecer 5 V e quando isso acontece dizemos que elas funcionam como SAÍDAS DIGITAIS. Você também pode conectar uma fonte de 5 V em uma destas portas fornecendo, portanto, energia elétrica para o Arduino (sem a intenção de alimentá-lo, é só um sinal) e neste caso dizemos que o pino que recebe 5 V está funcionando como uma ENTRADA DIGITAL.

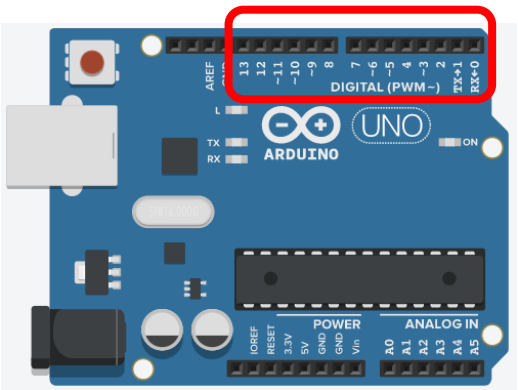
As portas de A0 até A5 são chamadas de portas analógicas e apenas recebem sinais, portanto são apenas ENTRADAS ANALÓGICAS. O Arduino não possui portas de saídas analógicas, no entanto ele é capaz de simular tais portas.

Antes de continuarmos devemos entender a diferença entre analógico e digital: se forma simplificada, imaginemos um sinal elétrico que possui valor mínimo igual a 0 V e máximo 5 V. Um sinal digital somente aceita os valores mínimo e máximo, portanto não temos valores intermediários. Costumamos pensar que só são possíveis dois sinais: ligado e desligado (outros preferem chamar de 0 ou 1 e justamente por isso que alguns botões possuem os símbolos 0 e 1 para representar que algo está ligado ou desligado.


Graficamente, podemos representar um sinal digital como se segue:




Veja a seguir quais são as portas digitais do Arduino.

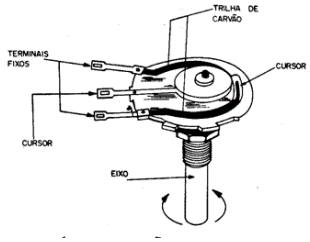



Em um programa no Arduino informaremos que uma porta deve esta ligada usando o número 1 ou a palavra *HIGH* que, em tradução livre, podemos dizer que significa alto ou ligado. Para indicarmos que uma saída digital está desligada, usamos o número 0 ou a palavra *LOW* que podemos entender como baixo ou desligado.

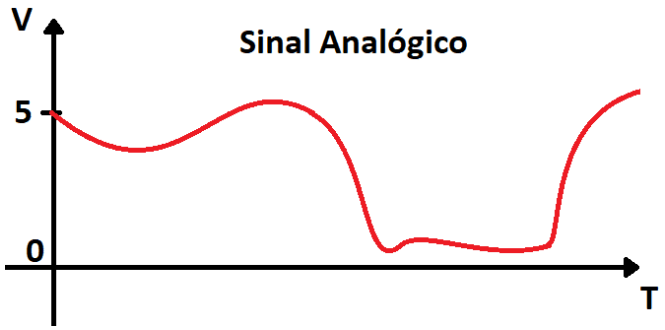
Se uma porta digital estiver sendo usada como entrada, usaremos comandos para ler estas portas e a leitura nos informará, de mesma maneira, 0 ou *LOW* se a porta digital estiver conectada ao GND e 1 ou *HIGH* se conectada a 5 V.

## PORTAS ANALÓGICAS

E a porta analógica? Ou melhor, e o sinal analógico, como funciona? Neste caso, não é só zeros e uns que ele aceita, mas também aceita outros valores. Ainda no exemplo de 0 a 5 V, podemos usar uma resistência variável associada à um botão para mudar a tensão continuamente conforme giramos o botão. Como ótimo exemplo, temos os botões que podemos girar para aumentar o volume de um rádio ou ainda o controle analógico de um vídeo game.

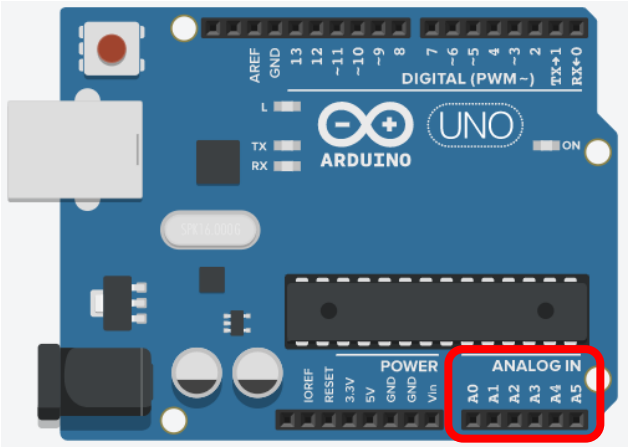



Graficamente podemos representar um sinal analógico como se segue:



*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

Na figura abaixo vemos as portas analógicas do Arduino.




Em um programa no Arduino, quando uma porta analógica for lida, esta fornecerá valores entre 0 e 255 sendo 0 quando estiver conectado ao GND e 255 quando conectado à 5 V.

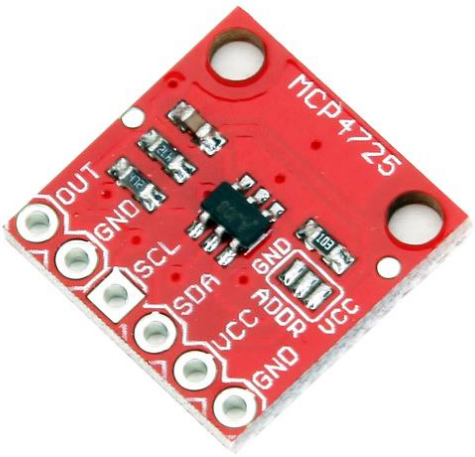
Recapitulando:

- No Arduino temos portas de entrada e saída digitais;
- No Arduino temos apenas portas de entrada analógica. Não existem portas de saída analógica.

## DAC

DAC, do inglês *Digital-to-Analog Converter*, que em português fica Conversor Analógico-Digital é um circuito eletrônico capaz de converter um sinal digital (por exemplo, enviado pelo Arduino) em um sinal analógico. Como vimos, o Arduino não possui saídas analógicas, mas não tem problema, pois iremos nos virar bem sem este conversor.

Existem muitos conversores no mercado, sendo um exemplo o DAC MCP4725 I2C mostrado abaixo.




Não se assute com os nomes e estes montes de números: eles possuem significados importantes, mas por enquanto imagine apenas que são nomes dados a estes componentes. Diferentes de nomes de pessoas, não é bom dar o mesmo nome a circuitos eletrônicos diferentes e como existem muitos circuitos elétricos com funcionalidades diversas, fica mais prático darmos nomes com especificações técnicas no próprio nome. Como exemplo, I2C é um tipo de comunicação que o Arduino é capaz de fazer para, por exemplom, mostrar uma informação em um display simples e neste tipo de comunicação teremos sempre os terminais SCL e DAS como visto na figura acima.

Resumindo: o Arduino fornece um número em formato digital e o DAC fornece uma tensão (voltagem) correspondente.

## PWM

Então, como simular um sinal analógico de saída no Arduino? A resposta está nas portas digitais PWM marcadas com um ~ no Arduino.

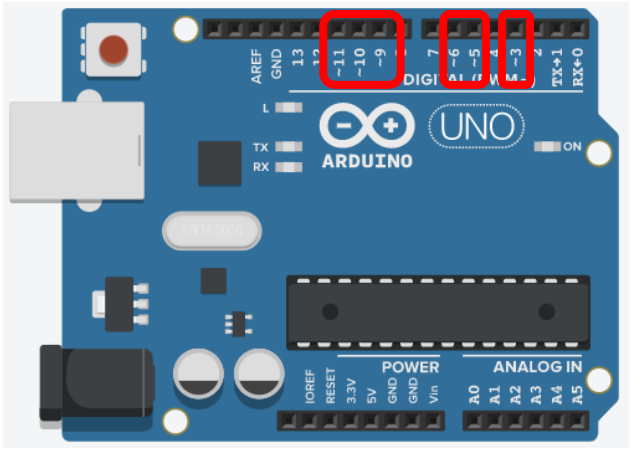
Para entender o que é PWM, recomendo que leiam o texto do seguinte link do qual extraí algumas imagens e informações:

https://www.embarcados.com.br/pwm-do-arduino/

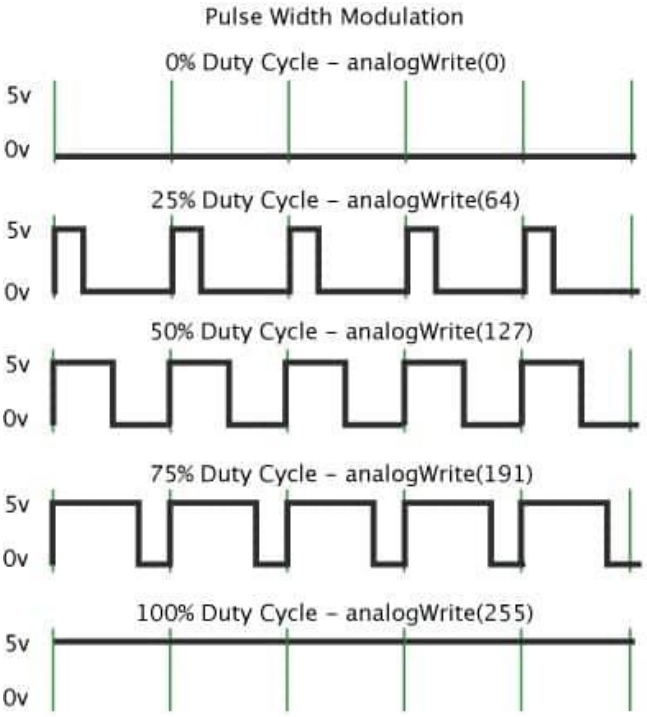
Na figura a seguir destacamos as portas PWM do Arduino UNO. São eslas as portas digitais 3, 5, 6, 9, 10 e 11.

Não precisa decorer, pois basta localizar onde estão as portas com o ~ ao lado. Note também que no próprio Arduino ele te lembra ao escrever “DIGITAL PWM ~”.

Abaixo da figura, transcrevo parte de um texto do site sugerido acima:




“PWM, do inglês *Pulse Width Modulation*, é uma técnica utilizada por sistemas digitais para variação do valor médio de uma forma de onda periódica. A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto. Esse tempo é chamado de *duty cycle*, ou seja, o ciclo ativo da forma de onda. No gráfico abaixo são exibidas algumas modulações PWM:”



Olha que interessante: se você ficar ligando e desligando um LED muito rapidamente com, digamos, 5 V, o LED poderá funcionar como se estivesse sendo alimentado com uma tensão menor. Por exemplo, se usarmos uma tensão de 5 V e um *Duty Cycle* de 50% então é como se alimentássemos o LED com 2,5 V. Esta tensão equivalente chamaremos de $V_{out}$ , a tensão máxima de alimentação (5 V no Arduino) de $V_{CC}$ então podemos ter a seguinte relação:

$$V_{out} = \frac{Duty\ Cycle}{100} \cdot V_{CC}.$$

As portas PWM podem ser usadas para controle de velocidade de motores, variação a luminosidade de LEDs, gerar sinais de áudio ou gerar sinais analógicos por razões diversas etc.

No Arduino, quando formos escrever o código, ao *Duty Cycle* não será indicado diretamente, isto é, não indicamos que o tempo de cclo ativo é um número que varia de 0% (totalmente desligado) até 100% (totalmente ligado): usamos valores que variam de 0 a 255. Assim, a fórmula acima pode ser reescrita como:

$$V_{out} = \frac{\text{Valor informado ao Arduino}}{255} \cdot V_{CC}.$$

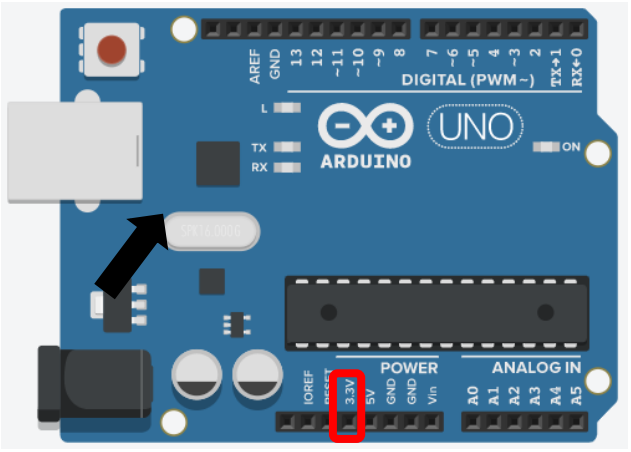
Como exemplo, se informarmos 0, então a saída ficará completamente desligada; informando 255 ficará totalmente ligada; informando 127, ficará desligado o mesmo tempo que ficaria desligado e assim por diante.

## FONTES DE ENERGIA

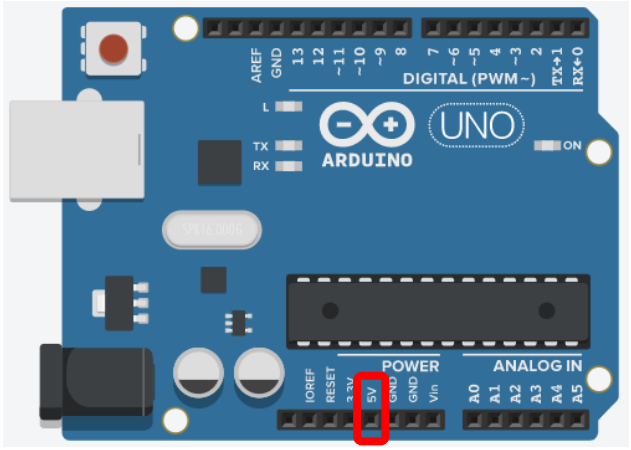
No Arduino o polo positivo poderá ser:

- 3.3V
- 5V
- Vin
- Portas digitais (de 0 a 13)

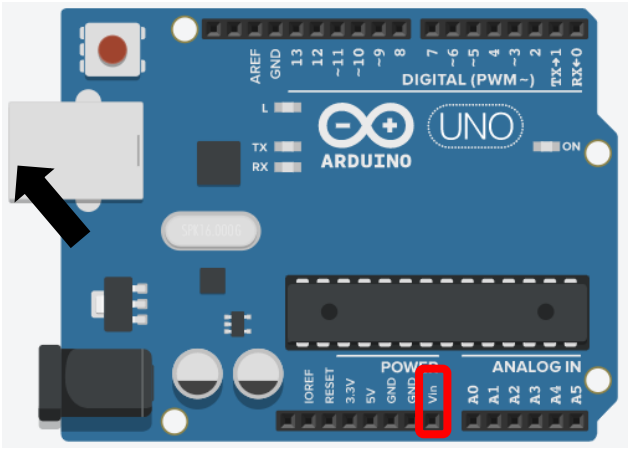
A primeira porta indicada acima está destacada na imagem abaixo e nos fornece 3,3 V. Alguns componentes eletrônicos irão funcionar com esta tensão, como módulos bluethoot.



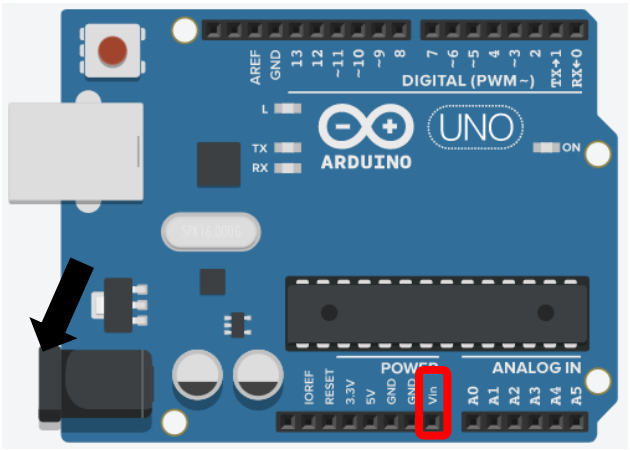

Veja que a segunda porta (5V) nos fornece 5 volts (como comparação, a bateria de seu celular fornece cerca de 4,7 V e a maioria dos periféricos conectados no PC funcionam com 5 V como mouse, teclado ou mesmo seu celular – a fonte do celular fornece 5 V). Veja na figura a seguir esta porta marcada.




Vin quer dizer algo como *V input*, isto é, o valor desta tensão depende da alimentação do Arduino: quando conectado no computador, ele fornece a tensão da porta USB conectada na porta indicada pela seta na figura abaixo.




O Arduino pode ter outra alimentação conforme indicado na figura abaixo e você pode conectar o Arduino à esta outra fonte mesmo se a USB já estiver conectada. Se fonte externa estiver conectada e o Arduino estiver ligado na USB, Vin será 5 V se a tensão de alimentação for menor que 7,4 V, mas se a tensão de entrada form maior que 7,4 V então Vin será igual à tensão fornecina no conector de alimentação (figura abaixo). Se somente uma das alimentações estiverem conectadas, então Vin será igual à tensão usada para alimentar o Arduino (nota: a tensão de alimentação recomendada é de 7 V até 12 V).




Ainda há mais o que estudar no Arduino, mas por enquanto isso já é o suficiente.

*PROFESSOR DANILO* *ROBÓTICA – 7 ANO – 31/09/2021*

## ATIVIDADE

Como atividade prática, vamos tentar fazer os seguintes circuitos:

1. Um circuito para acender e apagar um LED (saída digital);
2. Um circuito que lê o acionamento de um botão quando pressionado;
3. Um circuito que controla o brilho do LED usando PWM;
4. Um circuito que lê a tensão fornecida por um resistor variável;
5. Um circuito que junta todas as funcionalidades acima, ou seja, acende e apaga um LED usando um botão e controla o brilho de um LED usando um resistor variável.

## FUNÇÃO MAP

**O texto a seguir foi retirado de:**
https://www.arduino.cc/reference/pt/language/functions/math/map/

map()

[Math]

### Descrição

Remapeia um número **de** um intervalo **para** outro. Isto é, um valor de **deMenor** seria mapeado para **paraMenor**, um valor de **deMaior** para **paraMaior**, valores dentro de uma faixa para volores dentro da outra faixa etc.

Não restringe valores a ficar dentro do intervalo, porque valores fora do intervalo são as vezes úteis e pretendidos.

Note que os "limites mínimos" de cada intervalo podem ser maiores ou menores que os "limites máximos" tal que a função map() pode ser usada para reverter um intervalo de números, por exemplo

`y = map(x, 1, 50, 50, 1);`

A função também funciona bem com números negativos, tal que esse exemplo

`y = map(x, 1, 50, 50, -100);`

também é valido e funciona bem.

A função `map()` usa números inteiros e não irá gerar números fracionários, quando a matemática indicar que deveria. Resíduos fracionários são truncados e não são arredondados.

### Sintaxe

`=map(valor, deMenor, deMaior, paraMenor, paraMaior);`

### Parâmetros

`valor`: o número a ser mapeado

`deMenor`: o menor limite do intervalo atual do valor

`deMaior`: o maior limite do intervalo atual do valor

`paraMenor`: o menor limite do intervalo alvo

`paraMaior`: o maior limite do intervalo alvo

### Retorna

O valor mapeado para o novo intervalo.

### Código de Exemplo

```
/* Mapeia um valor analógico para 8 bits (0 a 255) */

void setup() {}


void loop() {
  int val = analogRead(0);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

## VAMOS PRATICAR USANDO BLOCOS

Felizmente, vamos trabalhar apenas com blocos, estilo Scratch. Nos próximos anos você vai se tornar capaz de programar em texto e ainda achará bem fácil fazer isso.

Bora codar?