PROFESSOR DANILO *ROBÓTICA – 9 ANO – 30/03/2021*

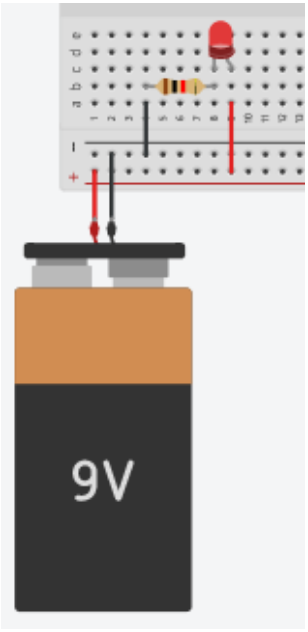
NOME:_____

AULA 04

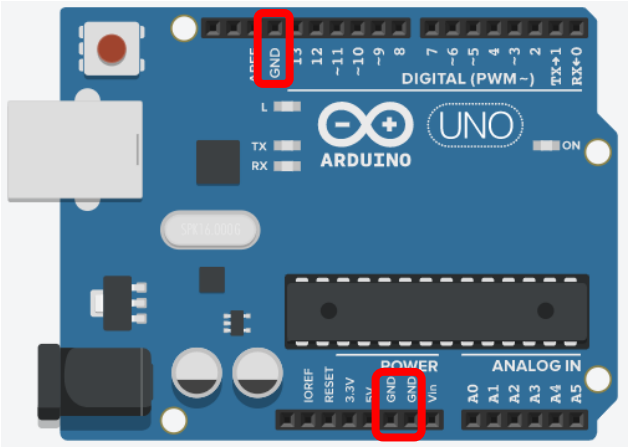
ENTRADA E SAÍDA

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.



PORTAS DIGITAIS

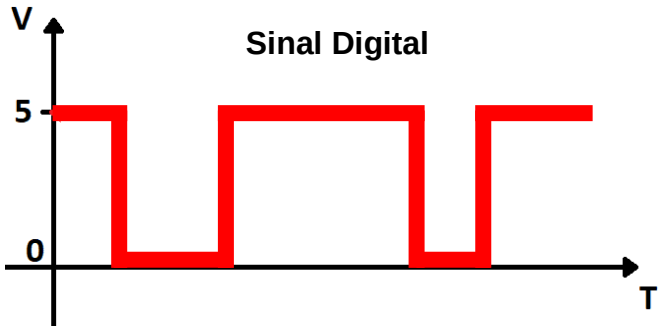
Os pinos de 0 a 13 no Arduino são chamados de portas digitais e os pinos de A0 até A5 são chamados de portas analógicas.

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.

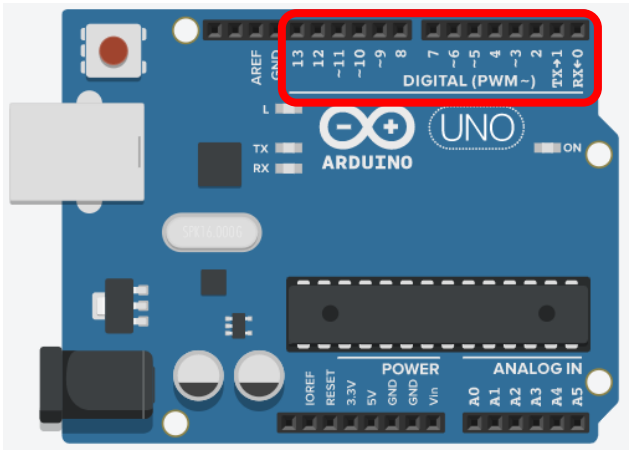
As portas de A0 até A5 são chamadas de portas analógicas e apenas recebem sinais, portanto são apenas ENTRADAS ANALÓGICAS. O Arduino não possui portas de saídas analógicas, no entanto ele é capaz de simular tais portas.

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.



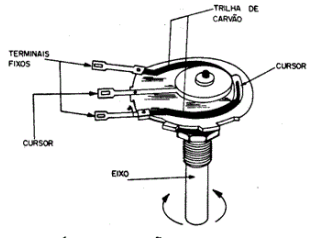
Em um programa no Arduino informaremos que uma porta deve esta ligada usando o número 1 ou a palavra *HIGH* que, em tradução livre, podemos dizer que significa alto ou ligado. Para indicarmos que uma saída digital está desligada, usamos o número 0 ou a palavra *LOW* que podemos entender como baixo ou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

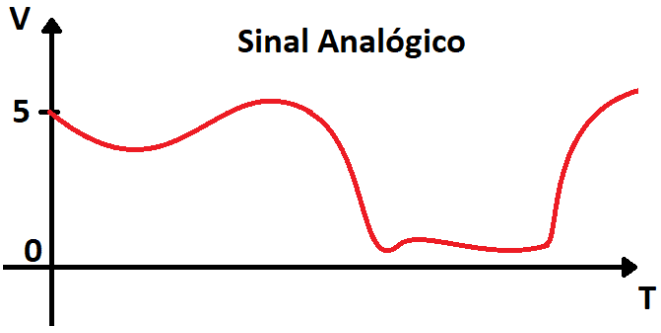
PROFESSOR DANILO *ROBÓTICA – – 9 ANO – 30/03/2021*

PORTAS ANALÓGICAS

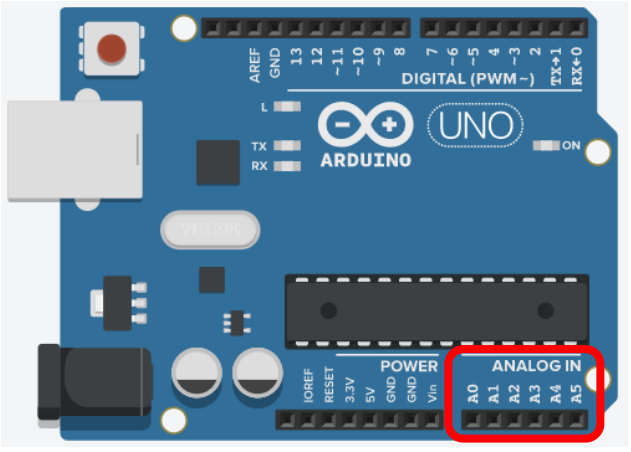
E a porta analógica? Ou melhor, e o sinal analógico, como funciona? Neste caso, não é só zeros e uns que ele aceita, mas também aceita outros valores. Ainda no exemplo de 0 a 5 V, podemos usar uma resistência variável associada à um botão para mudar a tensão continuamente conforme giramos o botão. Como 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:



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.

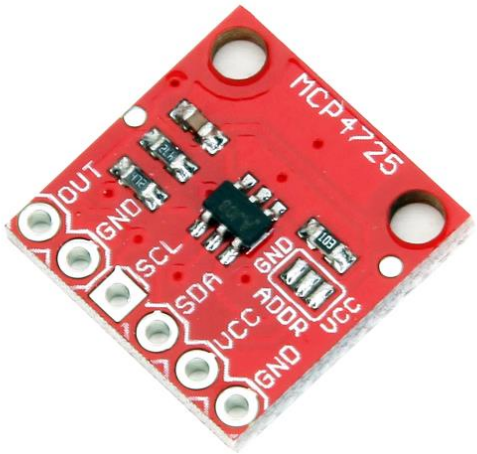
Recapitulando:

- 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:

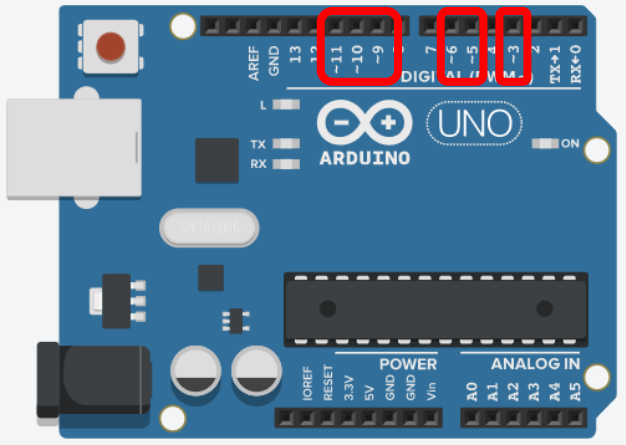
https://www.embarcados.com.br/pwm-do-arduino/

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

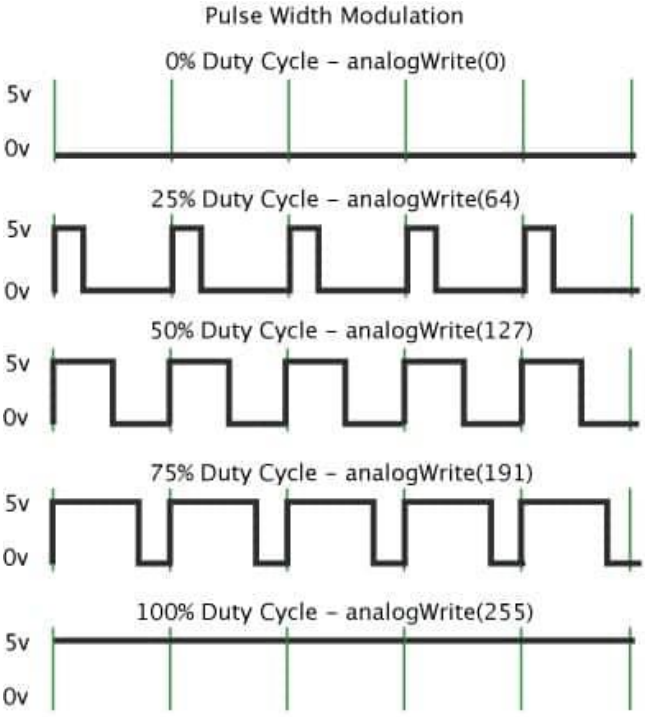
Não precisa decorer, pois basta localizar onde estão as portas com o ~ ao lado. Note também que no próprio Arduino ele te lembra ao escrever “DIGITAL PWM ~”.

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

PROFESSOR DANILO *ROBÓTICA – – 9 ANO – 30/03/2021*



"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:

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

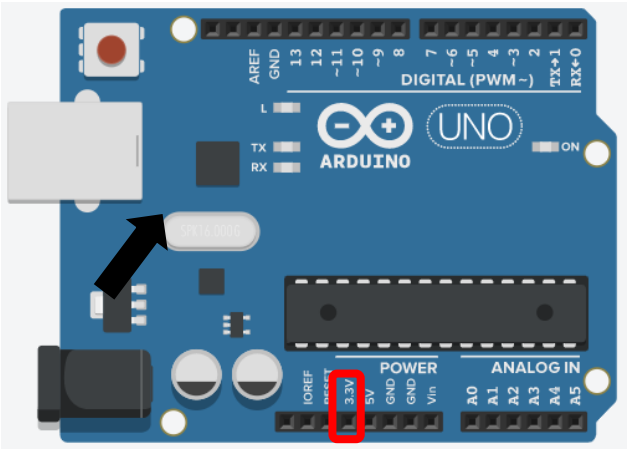
Como exemplo, se informarmos 0, então a saída ficará completamente desligada; informando 255 ficará totalmente ligada; informando 127, ficará desligado o mesmo tempo que ficaria desligado e assim por diante.

FONTES DE ENERGIA

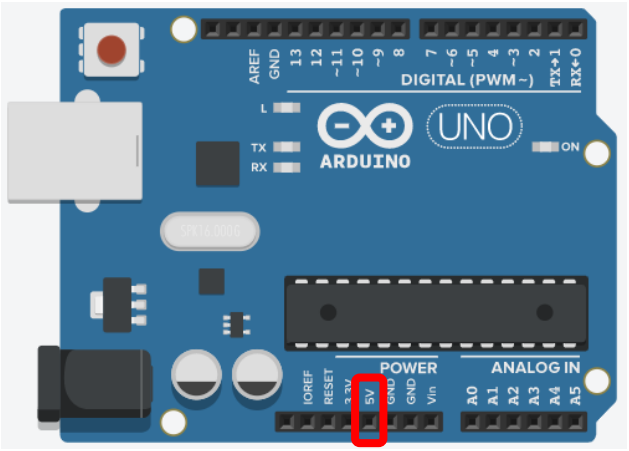
No Arduino o polo positivo poderá ser:

- 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.

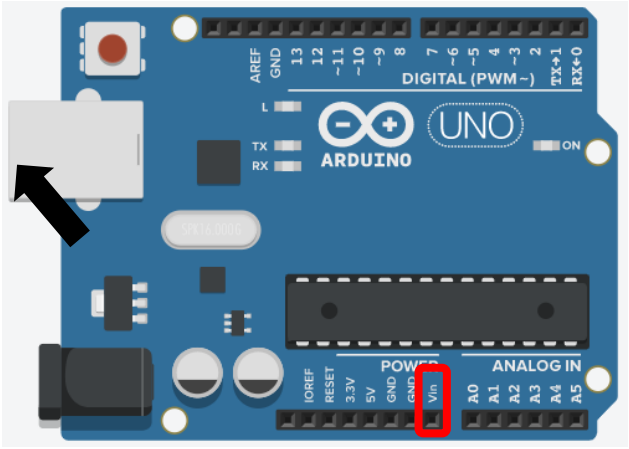


Veja que a segunda porta (5V) nos fornece 5 volts (como comparação, a bateria de seu celular fornece cerca de 4,7 V e a maioria dos periféricos conectados no PC funcionam com 5 V como mouse, teclado ou mesmo seu celular – a fonte do celular fornece 5 V). Veja na figura a seguir esta porta marcada.

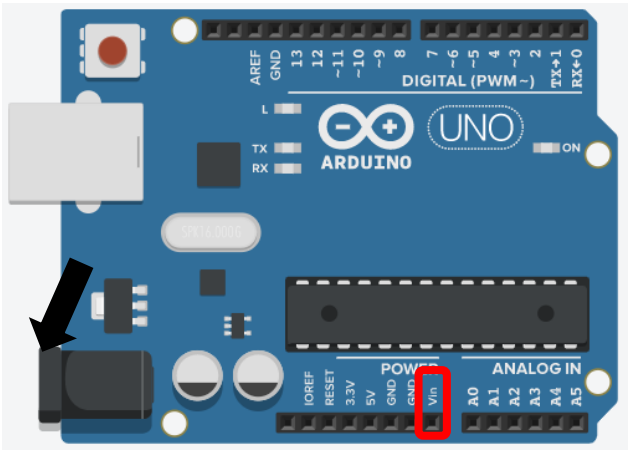


Vin quer dizer algo como *V input*, isto é, o valor desta tensão depende da alimentação do Arduino: quando conectado no computador, ele fornece a tensão da porta USB conectada na porta indicada pela seta na figura abaixo.

PROFESSOR DANILO *ROBÓTICA – – 9 ANO – 30/03/2021*



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.

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. A função constrain() pode ser usada tanto antes como depois dessa função, se limites para os intervalos são desejados.

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);
}
```