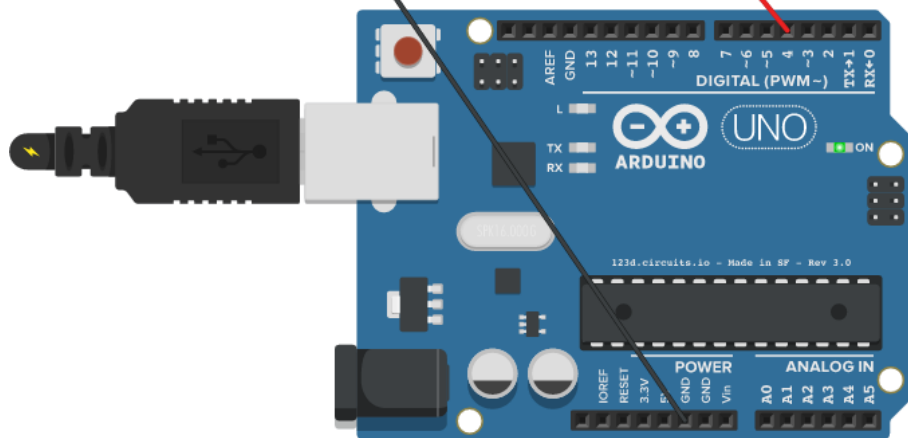
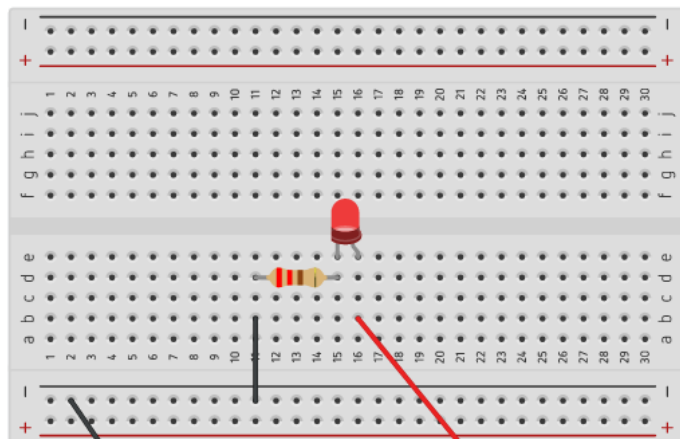


PROJETO ARDROBOTICA


PROGRAMAÇÃO COM ARDUINOS




Atividade 2 – Led a piscar (Blink)



Material necessário:

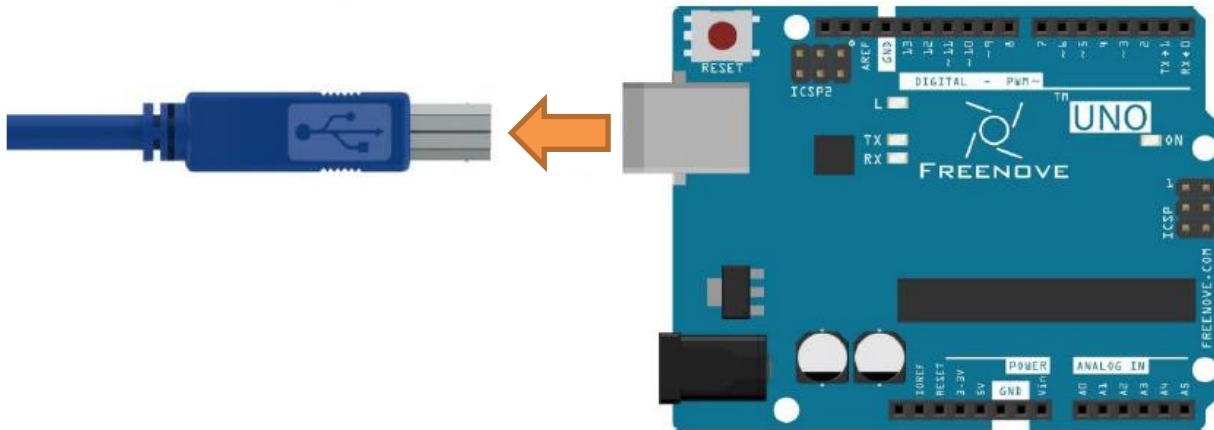
Led  (Output Digital)

Resistência 220 Ω 

Fios 

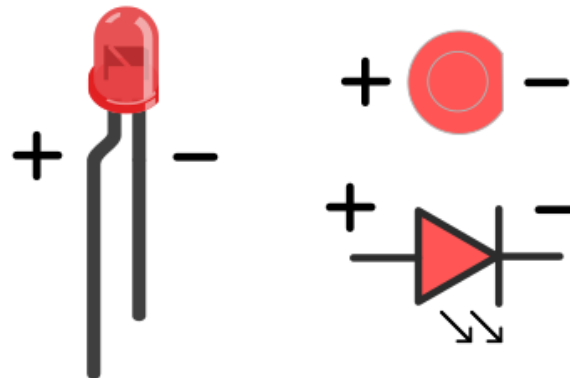
Atividade 2 – Led a piscar (Blink)

- Nas suas montagens de hardware certifique-se que o Arduino está desligado (desligue o cabo USB entre o computador e o Arduino)



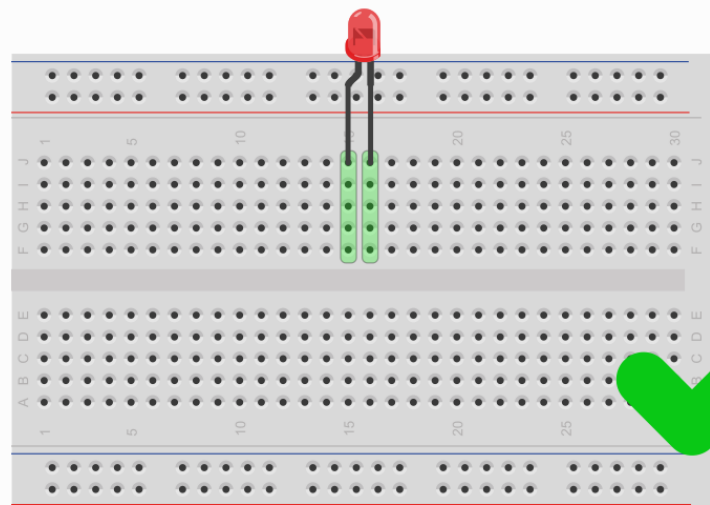
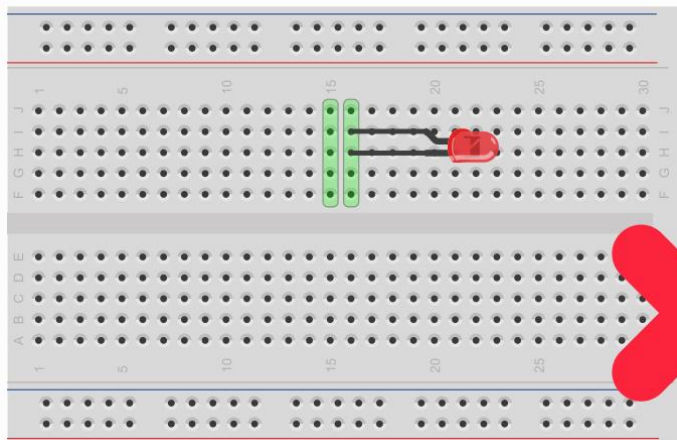
Atividade 2 – Led a piscar (Blink)

- O led tem polaridade, só emite luz se corretamente ligado.
- Certifique-se que a linha da placa de ensaio onde está ligada a perna mais longa do led (ânodo) está ligada a um pino digital do Arduino.
- A perna mais curta (cátodo) do led deve ser ligada à resistência e daí o circuito fecha no terminal GND (terra).



Atividade 2 – Led a piscar (Blink)

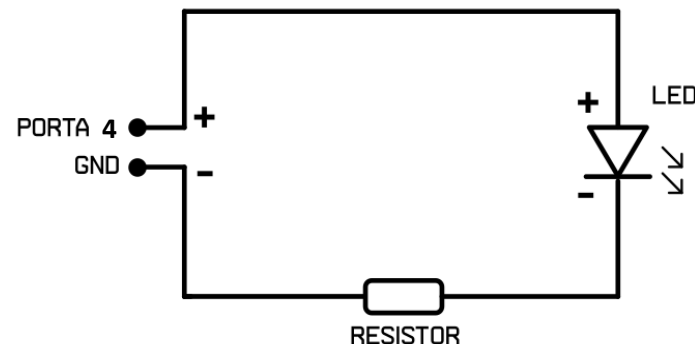
- Os dois terminais do LED não podem ficar na mesma fila da placa de ensaio. Este princípio de montagem aplica-se a outros componentes.



Atividade 2 - Led a piscar (Blink)

Como decidir qual o valor da resistência a usar?

- Os pinos digitais do Arduino trabalham com tensões entre 0V (OFF) e 5V (ON) e correntes até 40mA.
- No datasheet dos leds é especificado que produzem uma queda de tensão (forward voltage) de 1,8V e uma corrente máxima de 20mA para se obter o brilho máximo.
- Precisamos, portanto, de reduzir a corrente dos 40mA para 20mA. Para tal, precisamos de colocar uma resistência em série com o LED.



Atividade 2 - Led a piscar (Blink)

Resistência – qual utilizar?

- Tensão de saída de uma porta digital do Arduino = 5v
- Tensão de funcionamento do led vermelho = 1,8v
- Corrente máxima do led = 0,02 A (20mA)

Lei de Ohm:
$$R = \frac{V}{I} = \frac{5V - 1,8V}{0,02} = 160\Omega$$

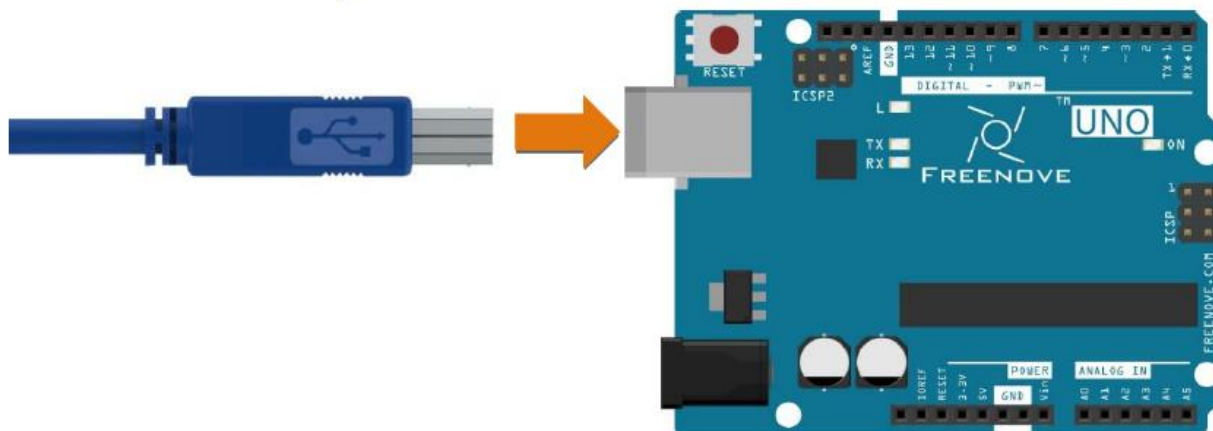


Não se deve utilizar uma resistência MENOR do que o valor recomendado, pois o led pode queimar.

Caso se utilize um resistência MUITO elevada, o led não acenderá.

Atividade 2 – Led a piscar (Blink)


- Quando estiver certo da correção das ligações pode ligar o Arduino ao PC, através do cabo USB.



Atividade 3 – Semáforo

Material necessário:

3 Leds  (Output Digital)

3 Resistências (220Ω) 

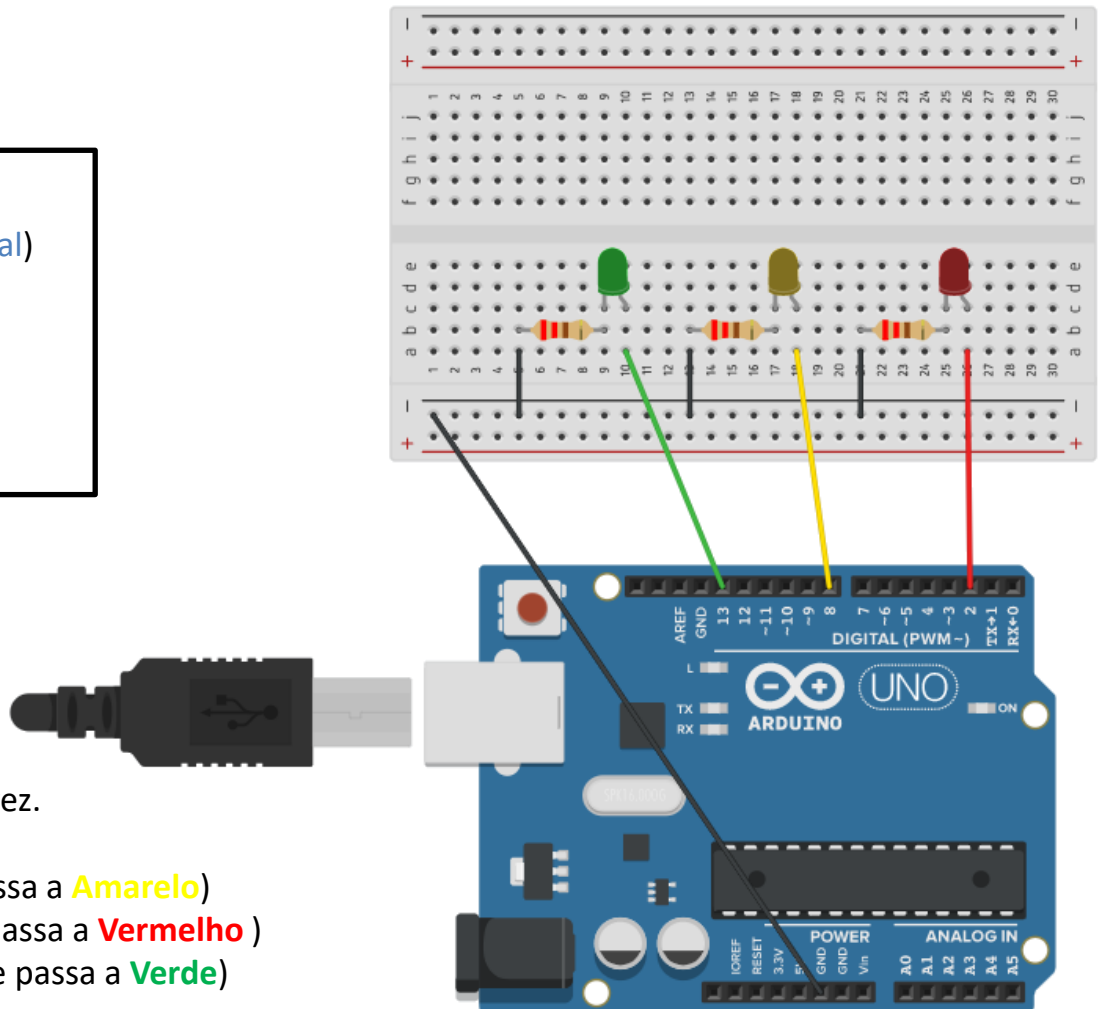
Fios 

Regras de funcionamento do semáforo:

Só pode estar uma cor acesa de cada vez.

Sequência de cores:


- Acende **Verde** (5 segundos e passa a **Amarelo**)
- Acende **Amarelo** (1 segundo e passa a **Vermelho**)
- Acende **Vermelho** (5 segundos e passa a **Verde**)



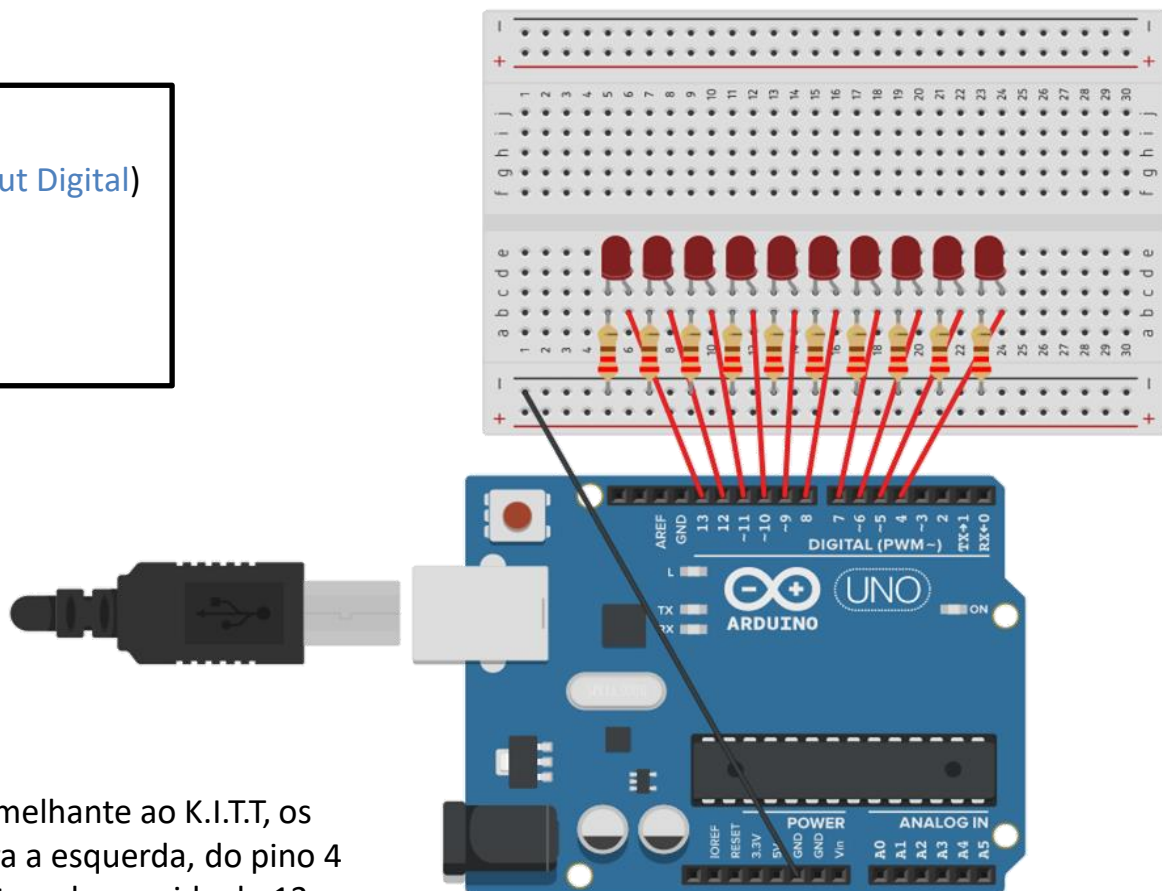
Atividade 4 – Arduino Knight Rider (K.I.T.T)

Material necessário:

10 Leds  (Output Digital)

10 Resistências (220Ω) 

Fios 



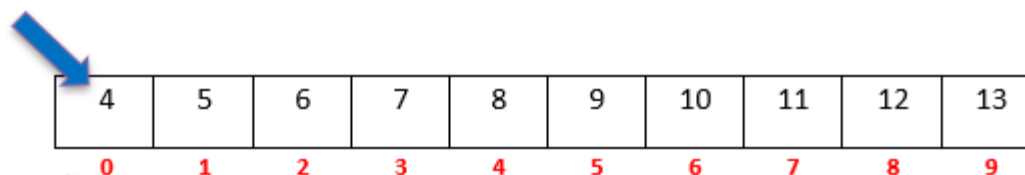
Nota:

Para se conseguir o efeito luminoso semelhante ao K.I.T.T, os leds começam por piscar da direita para a esquerda, do pino 4 até ao 13 fazendo o primeiro varrimento e de seguida do 13 até ao 4 efetuando o segundo varrimento.

Atividade 4 – Arduino Knight Rider (K.I.T.T)

- 10 leds significa termos de declarar 10 variáveis para representar as respectivas portas digitais;
- De forma a evitar ter de criar tantas variáveis, podemos optar por utilizar uma estrutura de dados que permita armazenar os números correspondentes às portas digitais: o **Array**
- Um **Array** é uma lista de valores de um mesmo tipo de dados que se podem posteriormente alterar ou ler, através do seu índice. Os índices, correspondem a cada posição do array e começam no valor zero.

Conteúdo do array em cada posição (portas digitais)



Representação gráfica do Array

Índice (posição do array)

Atividade 4 – Arduino Knight Rider (K.I.T.T)

- Com uma única estrutura podemos manipular os valores correspondentes aos pinos dos vários leds.

Declaração de um array constante de inteiros:

```
const int ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

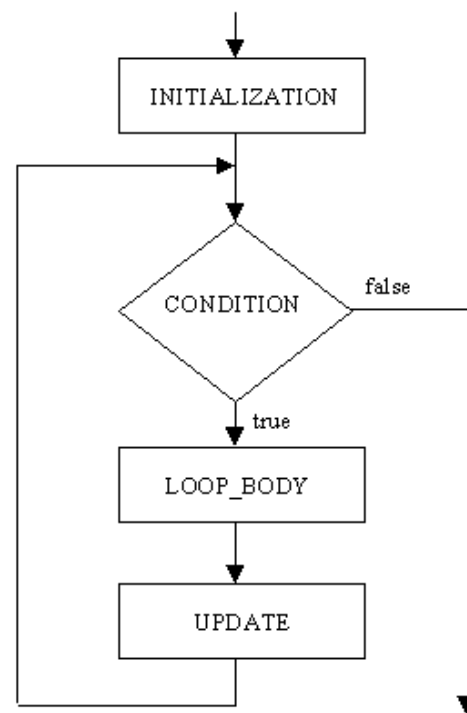
- **const** significa que a variável (array) será constante, ou seja, não poderá ser alterada ao longo da execução do programa.
- O tipo de dados **int** permite representar números inteiros negativos ou positivos entre -32768 e 32767.
- A expressão **ledPin[]** identifica o nome do array e os valores dentro das chavetas permitem inicializar e atribuir o número das portas digitais a cada posição do array.

Atividade 4 – Arduino Knight Rider (K.I.T.T)

- Para podermos percorrer o Array de valores, vamos recorrer a uma Estrutura de Repetição – o ciclo **for**.
- O ciclo **for** é utilizado para repetir um bloco de expressões de código.

```
for (i = 0; i < 10; i++) {  
    ... //bloco de instruções a repetir  
}
```

- É necessário definir uma variável e o seu respetivo valor inicial, a condição de paragem do ciclo e a atualização da variável após cada iteração do ciclo.



Tipos de Dados

- Os tipos de dados que podemos utilizar são os seguintes:

tipo de dados	RAM	Gama de Valores
void	N/A	N/A
boolean	1 byte	0 to 1 (true or false)
byte	1 byte	0 a 255
char	1 byte	-128 a 127
unsigned char	1 byte	0 a 255
int	2 byte	-32768 a 32767
unsigned int	2 byte	0 a 65535
word	2 byte	0 a 65535
long	4 byte	-2147483,648 a 2147,483,647
unsigned long	4 byte	0 a 4294967295
float ou double	4 byte	-3.4028235E+38 a 3.4028235E+38
string	1 byte + x	arrays de chars
array	1 byte + x	vector de variáveis

Operadores Aritméticos

Símbolo	Função
%	Resto
*	Multiplicação
+	Adição
-	Subtração
/	Divisão
=	Operador de atribuição

Operadores de Comparação

Símbolo	Função
<code>!=</code>	diferente de
<code><</code>	menor que
<code><=</code>	menor que ou igual a
<code>==</code>	igual a
<code>></code>	maior que
<code>>=</code>	maior que ou igual a

Operadores Lógicos

Símbolo	Função
!	Negação
&&	E (lógico)
	OU (lógico)

Sessão 2



The End