

Instruções do Kit de Boas Vindas!

Parabéns! Acabaste de entrar em **Engenharia Eletrotécnica e de Computadores** no IST. Estás pronto/a para pôr as mãos na massa e descobrir o poder dos microcontroladores?

Neste guia vais conhecer um microcontrolador programável chamado **Arduino Uno** e como o podes usar para controlar periféricos como botões, luzes e sensores de luminosidade!

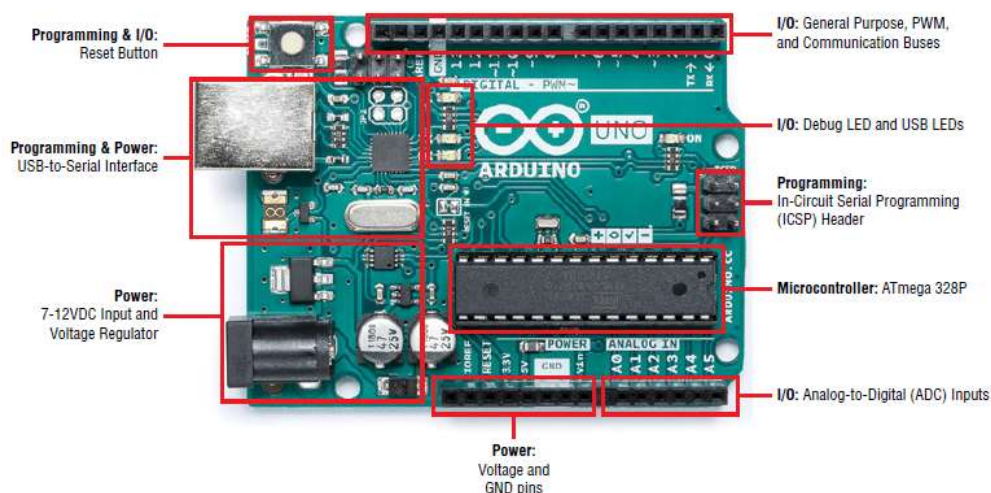


Aviso: Guarda este kit com carinho porque poderás vir a precisar dele para unidades curriculares do curso.

Introdução

O **Arduino Uno** é amplamente reconhecido como uma das placas de desenvolvimento de microcontroladores mais conhecidas em todo o mundo. Trata-se de uma *printed circuit board* (PCB) que integra um tipo especial de processador desenvolvido para computação física, denominado microcontrolador.

No centro desta placa encontra-se o microcontrolador ATmega328P, que confere ao Arduino uma vasta gama de funcionalidades e características.



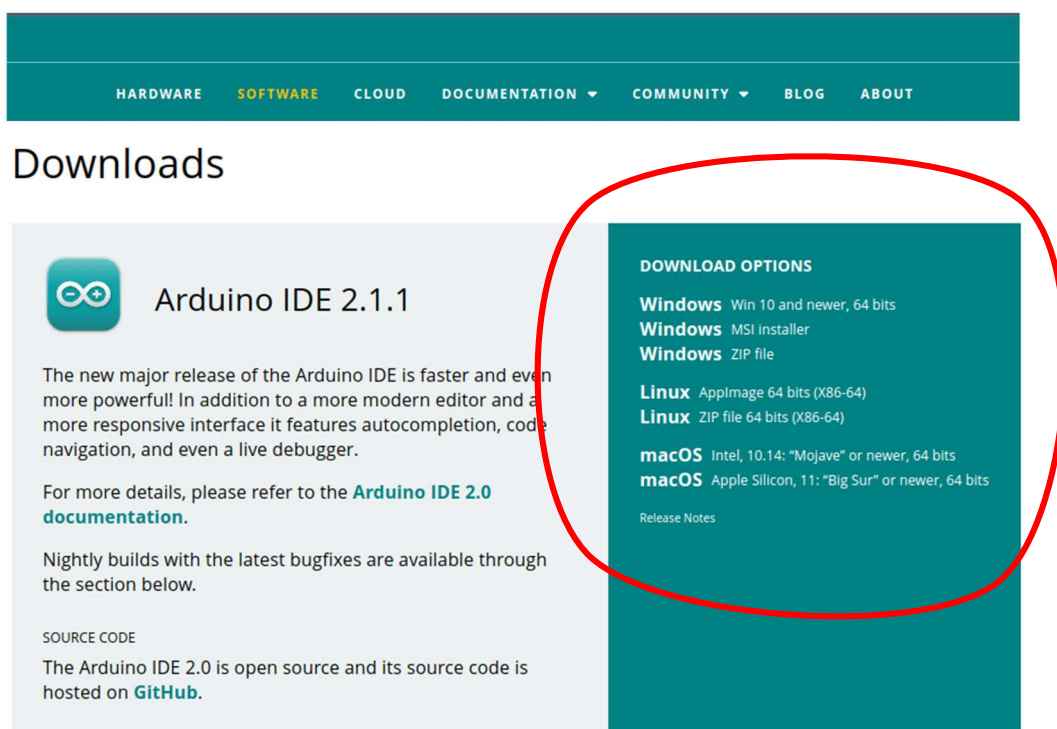
Microcontroladores são pequenas unidades de processamento que são úteis para controlar outro tipo de hardware. Podes encontrar exemplos de microcontroladores numa variedade de dispositivos, desde eletrodomésticos a equipamentos industriais e até em relógios inteligentes, demonstrando a sua versatilidade e presença abrangente.

Neste guia, vamos explorar o processo de programação do Arduino, utilizando a plataforma de desenvolvimento **Arduino IDE**. Esta ferramenta, conhecida pela sua simplicidade e facilidade de utilização, é responsável por converter o código escrito em linguagem C++ em instruções compreensíveis pelo Arduino. Este enfoque possibilita que mesmo aqueles com menos experiência em programação possam envolver-se de forma eficaz no processo de programação.

Instalação do ArduinoIDE

Para começar a utilizar o Arduino Uno, é necessário instalar o ArduinoIDE:

- Começa por abrir, no teu computador, o teu navegador de internet e escreve no campo do link: <https://www.arduino.cc/en/software>
- Nesta página seleciona o download correspondente para o teu sistema operativo da zona rodeada a vermelho na imagem abaixo.



- Assim que o download acabar procede à instalação do programa seguindo as instruções do instalador.
- Quando a instalação acabar ficarás com um ambiente de desenvolvimento de placas Arduino pronto a usar!

Breve explicação da interface

Ao abrires o programa pela primeira vez irás ver diversos separadores e muitos botões que podes clicar, por isso vamos explicar-te um bocadinho o que é que cada secção faz.



- Do lado esquerdo tens as tuas secções principais de desenvolvimento. Vendo de cima para baixo:
 - A primeira é a lista dos teus projectos de arduino;
 - A segunda é a secção de instalação de placas de desenvolvimento (deverás ver Arduino AVR Boards no topo, se não tiver instalado clica instalar);
 - A terceira é a secção das bibliotecas. Bibliotecas são bocados de código, já pré feitos, que desempenham diversas funções ou até mesmo implementam as funções necessárias para trabalhar com algum componente específico.
 - A quarta secção são mais algumas ferramentas de desenvolvimento e *debug* de código.
- Na barra de cima azul é possível de observares 3 botões e um campo de dropdown, o primeiro botão compila e valida o programa guardando o resultado no computador. O segundo botão tem como objectivo compilar o código e enviá-lo para a placa de desenvolvimento seleccionada. O terceiro corre o código de forma especial para ajudar ao debug do código. O campo de dropdown será onde aparecerá o teu Arduino caso já o tenhas ligado ao computador.

Existem ainda mais algumas funcionalidades que o Arduino IDE tem no seu conjunto de ferramentas, mas para o que irás fazer estas indicações chegaram.

Explora um pouco mais os menus e verás que o programa não é assim tão complexo e perceberás logo o que cada coisa faz.

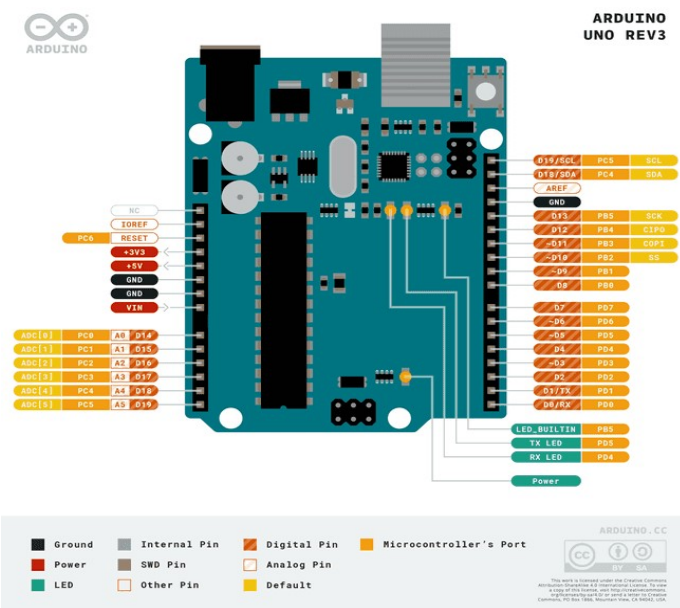
Arduino Uno

Antes de passares a programação do Arduino Uno convém perceber quais são as suas interfaces disponíveis.

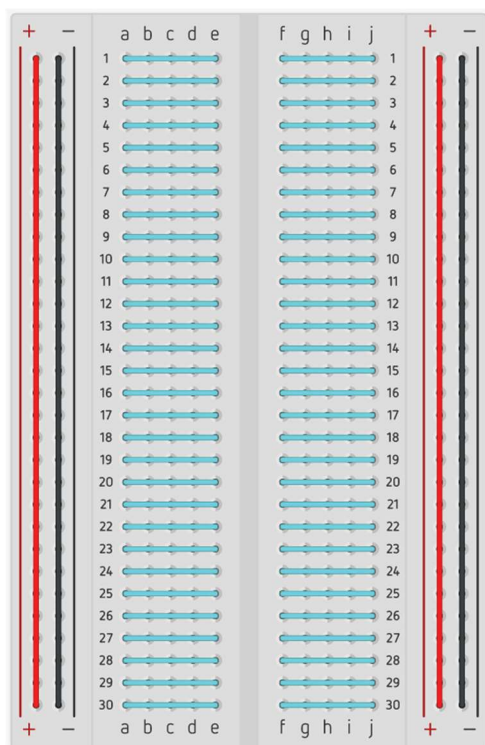
Tal como podes observar o Arduino Uno é composto por um total de 32 pins de entrada, cada um deles com uma função. Por exemplo todos os pins com etiqueta preta representam pins de ground, ou seja, negativos. Todos os outros pins à exceção dos pins de etiqueta vermelha, que têm como função fornecer o sinal positivo, têm como objetivo serem controlados através de código.

Para mais informações:

<https://docs.arduino.cc/retired/boards/arduino-uno-rev3-with-long-pins>



Breadboard



Outro elemento com características particulares é a breadboard. A imagem à esquerda ilustra o que efetivamente acontece no interior de uma placa de prototipagem. As quatro colunas mais exteriores (duas de cada lado) estão ligadas de "cima a baixo" e utilizam-se por norma para ligar a alimentação e o ground.

No meio da placa, cada linha de furos está ligada até à ranhura existente no centro. Estes caminhos fazem com que seja possível a ligação de vários componentes

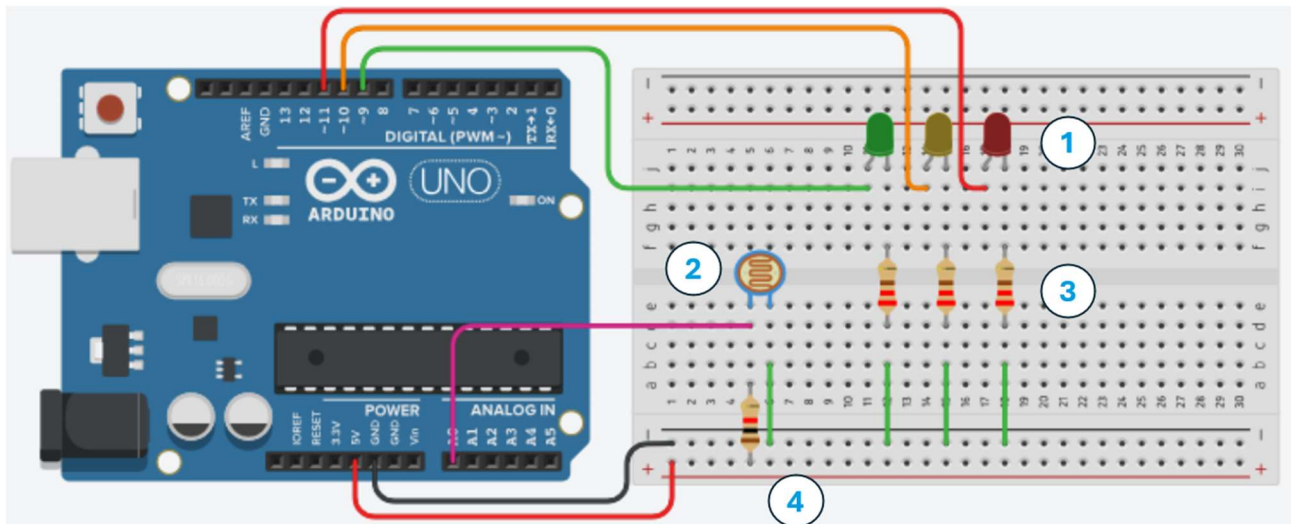
No teu kit também está disponível:

- 20 × cabos jumper macho-macho
- 1 × cabo USB
- 5 × LED vermelho
- 5 × LED amarelo
- 5 × LED verde
- 10 × Resistências 220 Ohm
- 10 × Resistências 1K
- 10 × Resistências 10K
- 2 × LDR (Utilizado e explicado mais a frente)
- 6 × Botão (2x brancos + 2x vermelhos + 2x azuis)
- 1 × Conector DC - 9V
- 1 × barra pinos 2.54mm

Medidor de intensidade de Luz

Programa 1 - Dificuldade fácil

Vamos agora começar a trabalhar com componentes físicos! Neste programa vamos trabalhar com um sensor de luminosidade. Para isso começamos com a montagem do circuito visível na imagem:



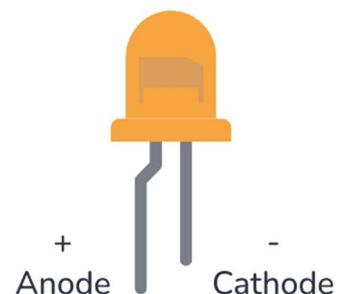
- 1 3 x LED's 2 1 x LDR 3 3 x Resistência 220 Ω 4 1 x Resistência 10k Ω

A montagem é composta por um LDR ou **Light Dependent Resistor**. Este componente varia a varia a quantidade de energia que por ele passa de acordo com a luminosidade a que ele é submetido.

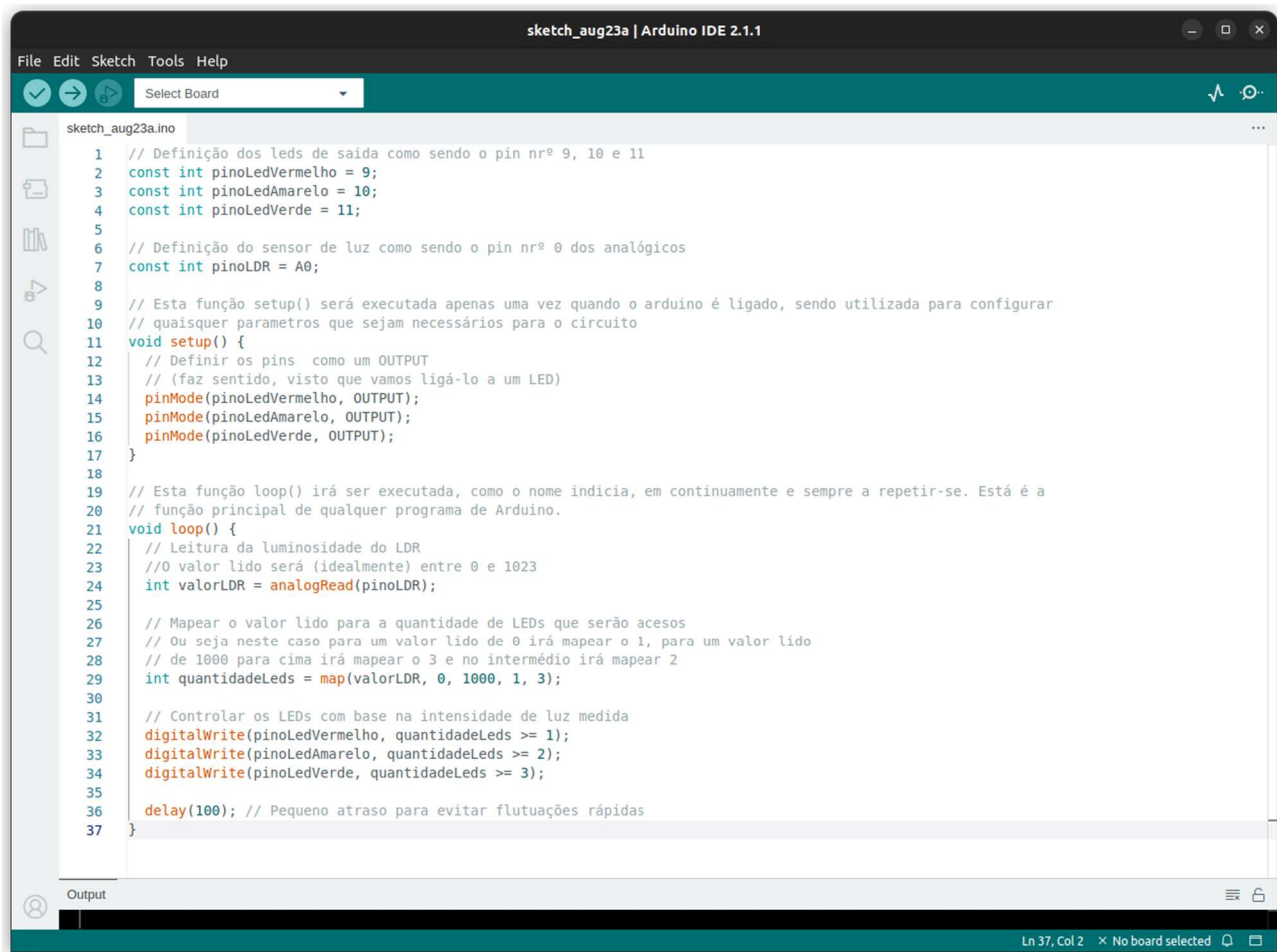
O programa depois de concluído permitirá observar a quantidade de luz que chega ao sensor através da quantidade de LED's que acender. Assim, se houver pouca luz os 3 leds acenderão, e caso houver muita luz apenas um acenderá.

Notas:

- Tem muita atenção na montagem pois alguns componentes como os LEDs têm uma perna específica para o positivo e outra para o negativo.
- No código da página seguinte existem linhas que apenas servem como comentários, estas linhas são todas aquelas iniciadas por "//" ou linhas abrangidas entre "/* */".



Código do programa 1: <https://neecist.org/atividade1.ino>

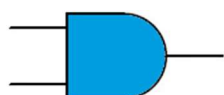


```
1 // Definição dos leds de saída como sendo o pin nº 9, 10 e 11
2 const int pinoLedVermelho = 9;
3 const int pinoLedAmarelo = 10;
4 const int pinoLedVerde = 11;
5
6 // Definição do sensor de luz como sendo o pin nº 0 dos analógicos
7 const int pinoLDR = A0;
8
9 // Esta função setup() será executada apenas uma vez quando o arduino é ligado, sendo utilizada para configurar
10 // quaisquer parametros que sejam necessários para o circuito
11 void setup() {
12     // Definir os pins como um OUTPUT
13     // (faz sentido, visto que vamos ligá-lo a um LED)
14     pinMode(pinoLedVermelho, OUTPUT);
15     pinMode(pinoLedAmarelo, OUTPUT);
16     pinMode(pinoLedVerde, OUTPUT);
17 }
18
19 // Esta função loop() irá ser executada, como o nome indicia, em continuamente e sempre a repetir-se. Está é a
20 // função principal de qualquer programa de Arduino.
21 void loop() {
22     // Leitura da luminosidade do LDR
23     // O valor lido será (idealmente) entre 0 e 1023
24     int valorLDR = analogRead(pinoLDR);
25
26     // Mapear o valor lido para a quantidade de LEDs que serão acesos
27     // Ou seja neste caso para um valor lido de 0 irá mapear o 1, para um valor lido
28     // de 1000 para cima irá mapear o 3 e no intermédio irá mapear 2
29     int quantidadeLeds = map(valorLDR, 0, 1000, 1, 3);
30
31     // Controlar os LEDs com base na intensidade de luz medida
32     digitalWrite(pinoLedVermelho, quantidadeLeds >= 1);
33     digitalWrite(pinoLedAmarelo, quantidadeLeds >= 2);
34     digitalWrite(pinoLedVerde, quantidadeLeds >= 3);
35
36     delay(100); // Pequeno atraso para evitar flutuações rápidas
37 }
```

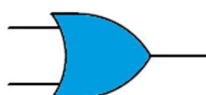
Portas Lógicas ou Booleanas

Programa 2 - Dificuldade média

Portas ou circuitos lógicos são dispositivos que operam e trabalham com um ou mais sinais lógicos de entrada para produzir uma e somente uma saída, dependente da função implementada no circuito.



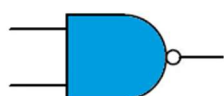
AND



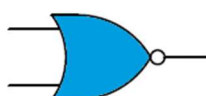
OR



XOR



NAND



NOR

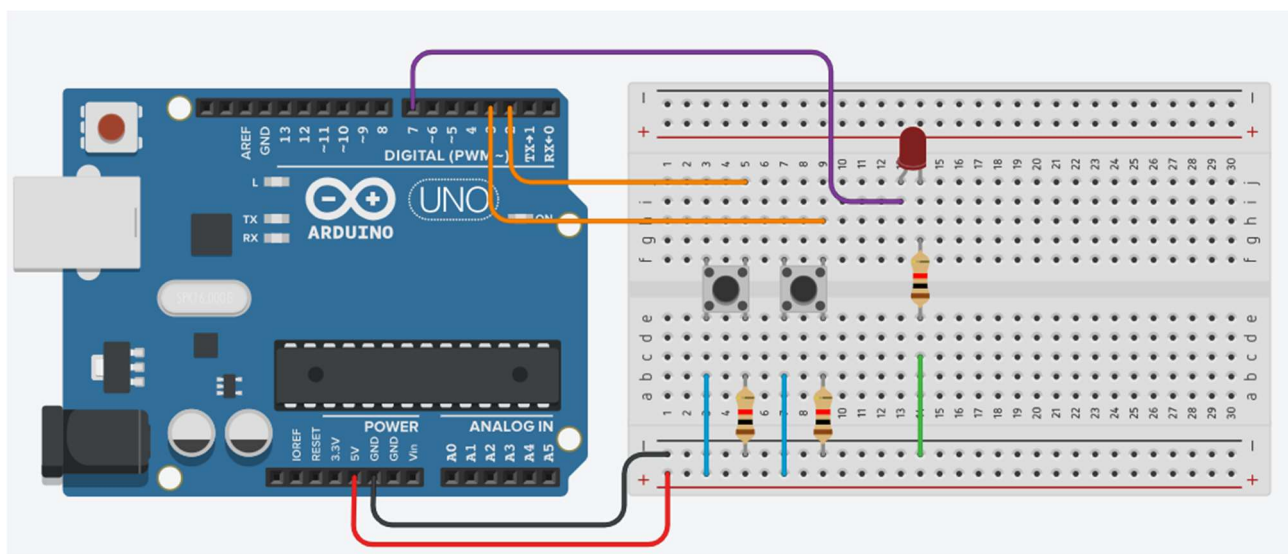


XNOR

São geralmente usados em circuitos eletrônicos, devido às condições que os sinais nesses circuitos podem apresentar: presença de sinal, ou "1", e ausência de sinal, ou "0".

Estas portas lógicas será algo que irás aprender numa das cadeiras de primeiro semestre, chamada de Sistemas Digitais e, portanto, o que achas de ver a lógica booleana já em ação?

À semelhança do exercício anterior, começa por seguir a montagem visível na seguinte imagem:



Neste programa, o código apresentado em baixo tem algumas das suas linhas comentadas que teram de ser descomentadas uma de cada vez de forma a ver o seu funcionamento:

- Começa por tentar perceber e copiar o código abaixo para o teu IDE.
- De seguida retira os “//” da linha correspondente à linha 44 desta forma estas a atribuir à variável “resultado” o resultado da porta lógica AND, tendo como input os dois botões.
- Compila o programa e envia-o para o Arduino.

Código: <https://neecist.org/atividade2.ino>



```
sketch_jul31a.ino
1 // Definição dos botões de input como sendo os pins nº 2 e 3
2 int pbA = 2;
3 int pbB = 3;
4
5 // Definição do led de saída como sendo o pin nº 7
6 int led = 7;
7
8 // Esta função setup() será executada apenas uma vez quando o arduino é ligado, sendo utilizada para configurar
9 // quaisquer parametros que sejam necessários para o circuito
10 void setup()
11 {
12   // Definir os pins dos botões como INPUTS
13   // (faz sentido, visto que vamos ligá-lo a um botão [que é um input])
14   pinMode(pbA, INPUT);
15   pinMode(pbB, INPUT);
16
17   // Definir o pin como um OUTPUT
18   // (faz sentido, visto que vamos ligá-lo a um LED)
19   pinMode(led, OUTPUT);
20   digitalWrite(led, LOW);
21 }
22
23 // Esta função loop() irá ser executada, como o nome indicia, em continuamente e sempre a repetir-se. Está é a
24 // função principal de qualquer programa de Arduino.
25 void loop()
26 {
27   // Leitura do estado dos botões
28   // 0 - Sem estar a clicar
29   // 1 - A clicar
30   int inA = digitalRead(pbA);
31   int inB = digitalRead(pbB);
32
33   // Variável para guardar o resultado da porta lógica
34   bool resultado = false;
35   /*
36    NOT : !
37    AND : &
38    OR  : |
39    XOR : ^
40   */
41
42   // Botão A [E] (and) botão B
43   // Botão A * Botão B
44   // resultado = inA & inB;
45
46   // Botão A [OU] (or) botão B
47   // Botão A + Botão B
48   // resultado = inA | inB;
49
50   // Botão A [OU Exclusivo] (xor) botão B
51   // Botão A # Botão B
52   // resultado = inA ^ inB;
53
54   // NOT(Botão A [E] (and) botão B)
55   // !(Botão A * Botão B)
56   // resultado = !(inA & inB);
57
58   /*
59    .
60    .
61    .
62    .
63   */
64
65   digitalWrite(led, resultado);
66   delay(500);
67 }
68
```

- Assim que o upload acabar experimenta clicar nos dois botões e verás o led acender. Se clicares apenas num deles, o led irá permanecer desligado. Este é o comportamento da porta AND, todos os seus inputs teram de ser 1 (true) para a saída ser 1 (true).

Volta a colocar os “//” na linha 44 e experimenta fazer os mesmos passos de cima para as linhas 48, 52 e 56 observando sempre se os resultados obtidos correspondem à porta suposta. Utiliza as tabelas de verdade da página seguinte para confirmar os teus resultados.

Tabelas de verdade

De forma resumida, uma tabela de verdade é como um mapa que apresenta todas as respostas possíveis para uma proposição que pode ser verdadeira ou falsa. Neste caso estamos a considerar portas lógicas com duas entradas e uma saída.

AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

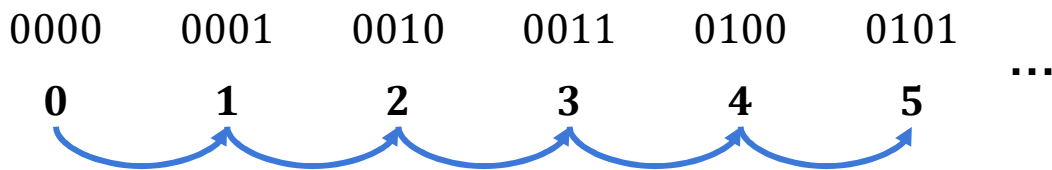
- Porta AND (E): Saída verdadeira apenas quando ambas as entradas são verdadeiras.
- Porta OR (OU): Saída verdadeira se pelo menos uma das entradas é verdadeira.
- Porta XOR (OU Exclusivo): Saída verdadeira quando as entradas são diferentes, e falsa quando são iguais.

Numeração Binária

Programa 3 - Dificuldade média

Nesta terceira montagem vais criar um circuito que ilustra o sistema de numeração utilizado pelos computadores, o binário. Nos computadores quando falamos do seu funcionamento a baixo nível, tudo é representado por uma sequência de 1's e 0's (bits), por exemplo o número 5 é representado por 101.

Neste circuito irás poder observar os bits no seu estado 1 (led ligado) e 0 (led desligado) enquanto incrementas ou decrementas a contagem.



Apesar de haver um padrão que se pode seguir, quando os números ficam grandes de mais é mais fácil pensar de forma matemática ou seja pensar que cada um dos bits é uma potência de 2 ou seja:

$$1 = 0001 = 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 1$$

$$2 = 0010 = 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 0$$

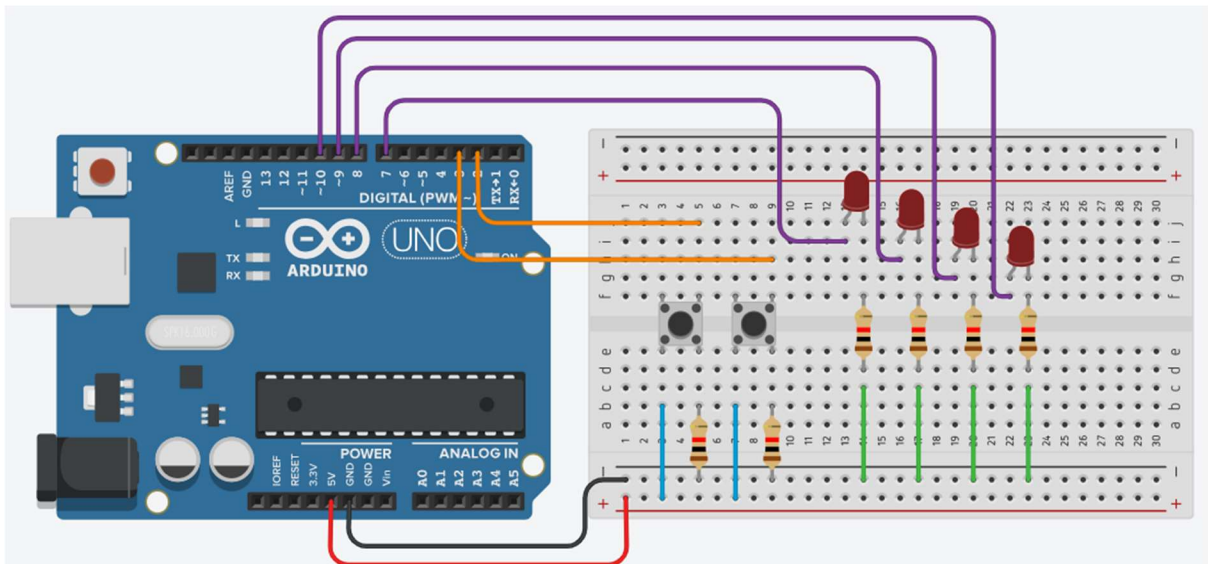
$$3 = 0011 = 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1$$

...

$$14 = 1110 = 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 0$$

$$15 = 1111 = 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1$$

À semelhança dos programas anteriores, abaixo encontra-se a montagem deste circuito e na página seguinte o código para que possas testá-lo:



Código: <https://neecist.org/atividade3.ino>

```
sketch_jul31a | Arduino IDE 2.1.1
File Edit Sketch Tools Help
Select Board

sketch_jul31a.ino
1 // Definição dos botões como sendo os pins nº 2 e 3
2 int botao_aumentar = 2;
3 int botao_diminuir = 3;
4
5 // Definição dos leds de saída como sendo os pins nº 7, 8, 9, 10
6 int led4 = 7;
7 int led3 = 8;
8 int led2 = 9;
9 int led1 = 10;
10
11 int valor_atual[] = {0, 0, 0, 0};
12
13 // Esta função setup() será executada apenas uma vez quando o arduino é ligado, sendo utilizada para configurar
14 // quaisquer parametros que sejam necessários para o circuito
15 void setup()
16 {
17     // Definir os pins como INPUTS
18     // (faz sentido, visto que vamos ligá-los a perifericos de input)
19     pinMode(botao_aumentar, INPUT);
20     pinMode(botao_diminuir, INPUT);
21
22     // Definir os pins como um OUTPUT
23     // (faz sentido, visto que vamos ligá-los a LEDs)
24     pinMode(led4, OUTPUT);
25     pinMode(led3, OUTPUT);
26     pinMode(led2, OUTPUT);
27     pinMode(led1, OUTPUT);
28 }
29
30 void loop()
31 {
32     // Alterar o estado dos leds para o valor correspondente a cada bit
33     digitalWrite(led1, valor_atual[0]);
34     digitalWrite(led2, valor_atual[1]);
35     digitalWrite(led3, valor_atual[2]);
36     digitalWrite(led4, valor_atual[3]);
37
38     delay(1000);
39     // Leitura dos botões
40     int inA = digitalRead(botao_aumentar);
41     int inB = digitalRead(botao_diminuir);
42
43     // Se o botão de incrementar foi clicado
44     if(inA){
45         // Se o bit mais a direita é igual a 1 então passa a 0 e continua
46         if(valor_atual[0] == 1){
47             valor_atual[0] = 0;
48             // Caso contrario se for igual a 0 passa a ser 1 e sai
49         }else{
50             valor_atual[0] = 1;
51             return;
52         }
53         // E assim sucessivamente
54
55         if(valor_atual[1] == 1){
56             valor_atual[1] = 0;
57         }else{
58             valor_atual[1] = 1;
59             return;
60         }
61
62         if(valor_atual[2] == 1){
63             valor_atual[2] = 0;
64         }else{
65             valor_atual[2] = 1;
66             return;
67         }
68
69         if(valor_atual[3] == 1){
70             valor_atual[3] = 0;
71         }else{
72             valor_atual[3] = 1;
73             return;
74         }
75     }
76 }
```

Output

Downloading index: library_index.tar.bz2

Ln 52, Col 6 × No board selected