**-+-Exercício Prático 3**

**Laboratório de ac2**

**Objetivo:**

**Construir uma Unidade Lógica e Aritmética (ULA) de 1 bit e implementar no Arduino.**

**Parte 1 (O Arduino)**

Estaremos neste exercício construindo uma ULA de 1 bit e que irá evoluir para os 4 bits nos próximos exercícios.

O primeiro passo é conhecer como a comunicação entre a sua máquina e o Hardware externo irá ocorrer. Nesse semestre iremos utilizar a plataforma Arduino para tal.

As seguintes ações deverão ser realizadas para essa aula:

1. Para este exercício será necessário o Tinkercad. +
2. Em uma pasta conhecida da sua máquina, baixar e descompactar o arquivo EP03\_2024\_1\_Arduino.zip.
3. Abrir o arquivo Introducao\_Arduino.pdf que estará dentro da pasta descompactada.

Para os exercícios a seguir tenha em mente os seguintes Leds conectados ao arduino (esta montagem já está pronta no usuário “pucminas lab” e a montagem é a “Placa\_Lab\_1”:

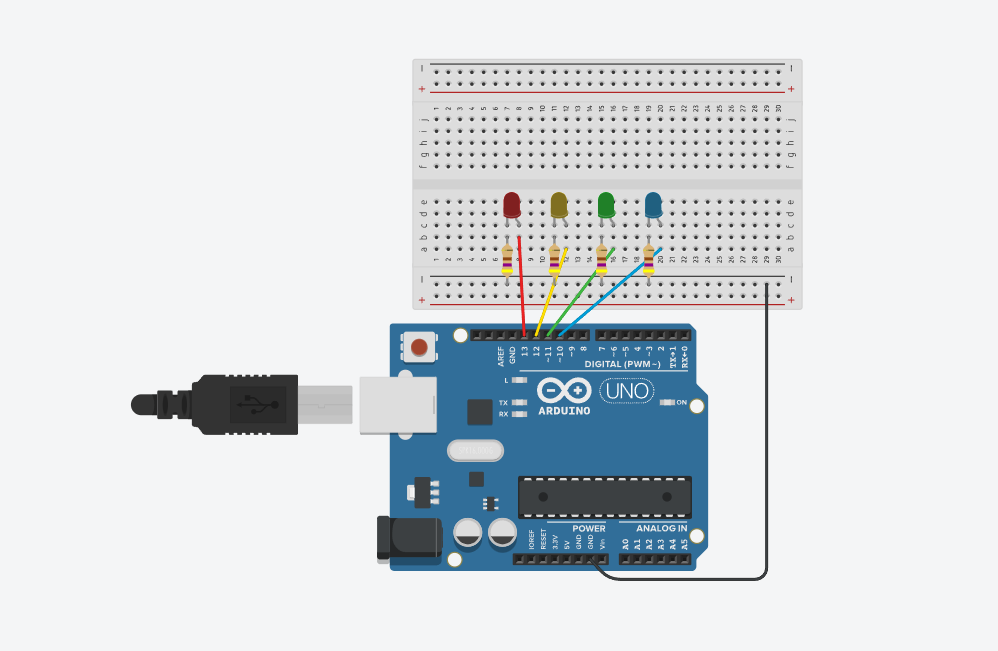
**Exercício 1**

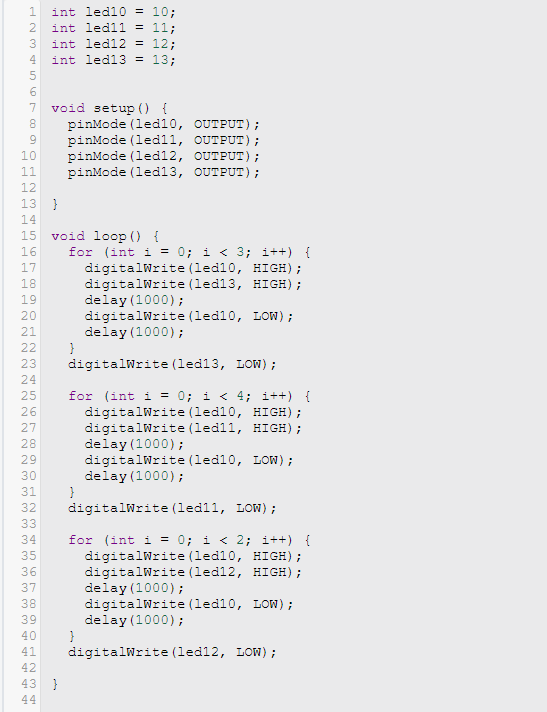
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Saída do Arduino** | **13** | **12** | **11** | **10** |
| **Cor do Led** | Verm. | Amar. | Verde | Azul |

Com a tabela acima, você deverá elaborar um programa que simule um semáforo temporizado.

O led Azul deverá sempre piscar de um em um segundo.

Durante os primeiros 3 ciclos de azul ligado, apenas o vermelho deverá estar ligado, em seguida durante 4 ciclos de azul ligado, apenas o verde ficará ligado e finalmente durante 2 ciclos de azul ligado, apenas o amarelo ficará ligado. O ciclo continua indefinidamente.





**Exercício 2**

Você deverá simular uma unidade lógica de 1 bit no arduino. Sua unidade lógica deverá executar 4 operações diferentes:

|  |  |
| --- | --- |
| **Op. Code (Operation)** | **Instrução (Result)** |
| 0 | AND (a,b) |
| 1 | OR (a,b) |
| 2 | NOT (a) |
| 3 | SOMA(a,b) |

Através da comunicação serial três valores deverão ser passados ao Arduino. O valor de a, b e o OP. Code.

Ao passarmos os valores, deveremos passar os três números de uma vez, observe os exemplos a seguir:

* se enviarmos 100 significa que a=1, b=0 e queremos a operação 0 ou and(a.b), a saída será 0 e vai1 também 0.
* se enviarmos 012 significa que a=0, b=1 e queremos a operação 2 ou not(a), a saída será 1 e vai1 também 0.
* se enviarmos 113 significa que a=1, b=1 e queremos a operação 3 ou soma(a,b), a saída será 0 e vai1 será 1.

Utilize a seguinte configuração para mostrar as entradas e saídas:

O valor de a no led vermelho (saída 13)

O valor de b no led amarelo (saída 12)

O valor da saída da ULA no led verde (saída 11)

O valor de vai1 no led azul (saída 10)

Considere o seguinte programa a ser executado em uma Unidade Lógica e Aritmética (ULA) considerando números de 1 bit:

***Inicio:***

A=0; ( ou A=0) B=1; ( ou B=1)

AND(A,B); ( esta é a operação bit a bit entre A e B)

B=0; ( ou B=0)

A=1; ( ou A=1) OR(A,B);

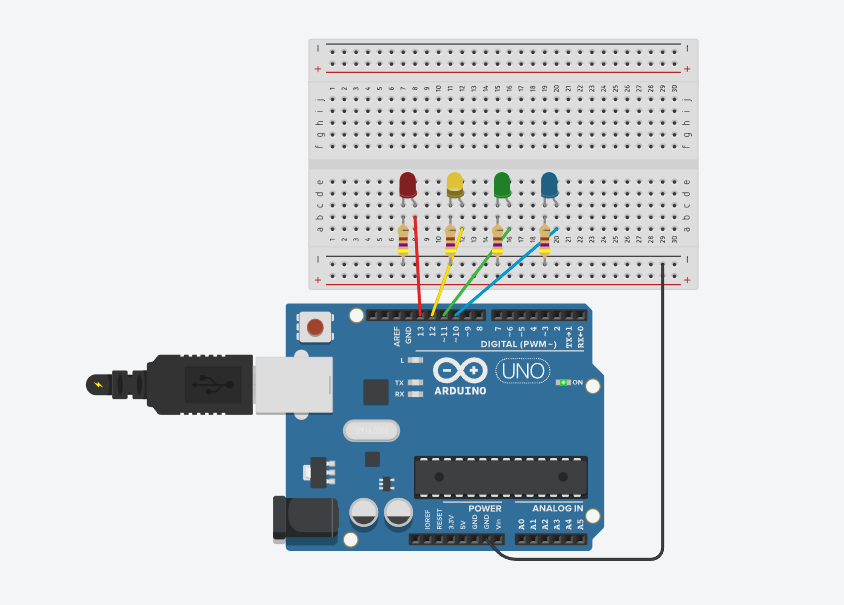
SOMA(A,B); (esta é a operação aritmética da soma entre A e B)

A=0; ( ou A0)

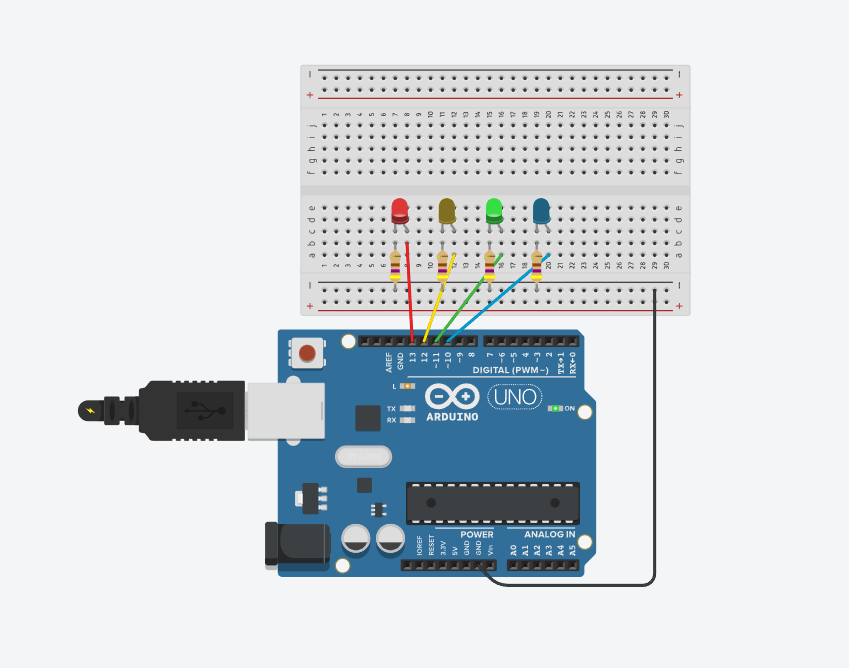
NOT(A);

B=1; ( ou B=1) AND(B,A); ***Fim.***

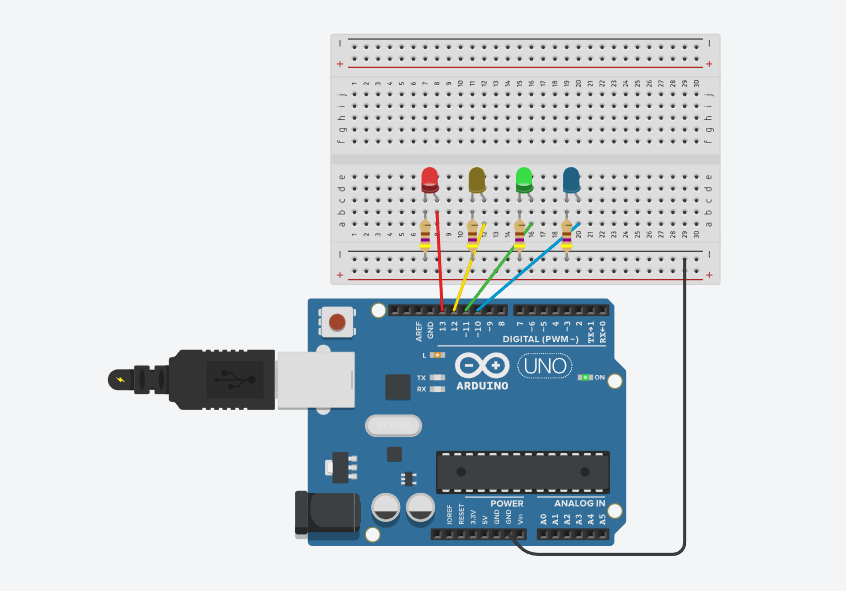
010



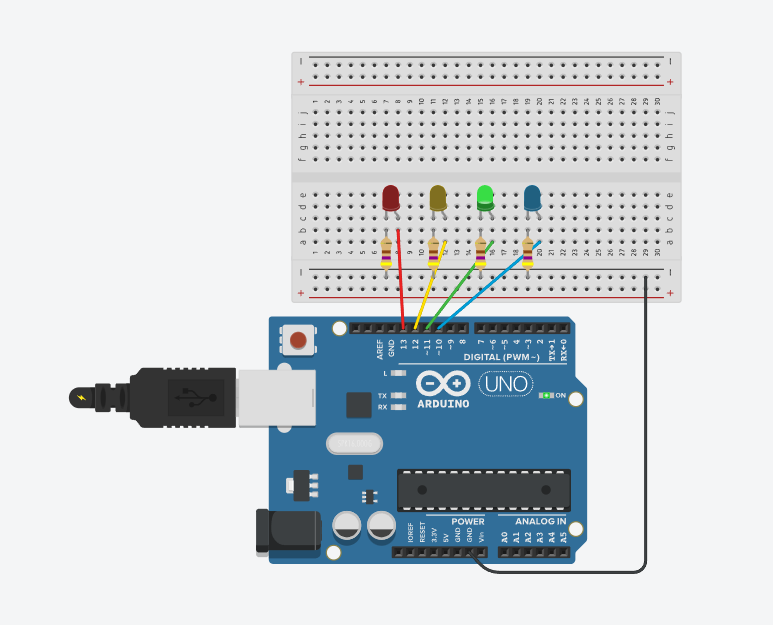
101



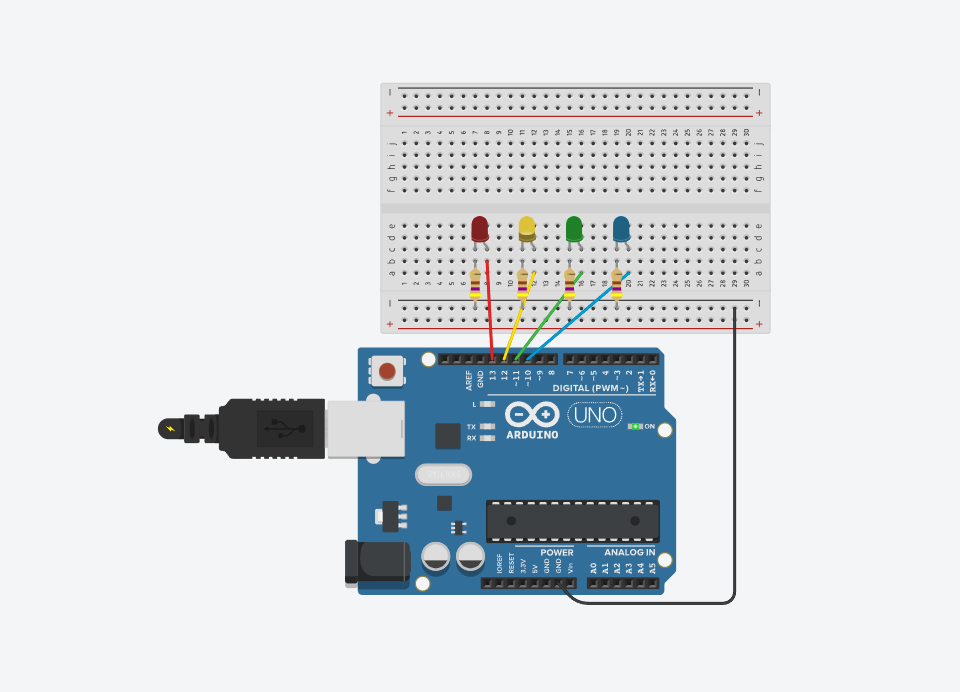
103

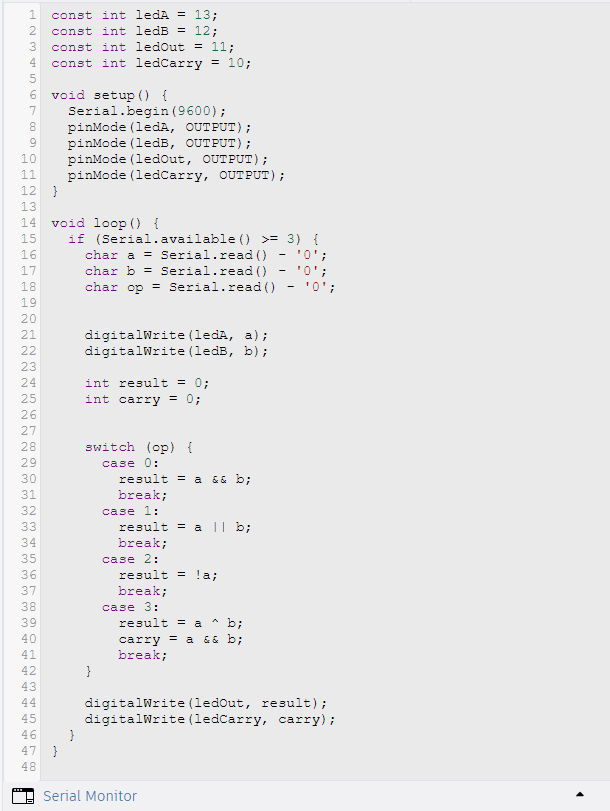


002



010





Para o programa de teste acima, preencher a tabela a seguir considerando que cada linha corresponderá à execução de uma instrução (a primeira linha já foi realizada, observe que a palavra de código deverá conter 4 bits, para escrevermos em hexa 0x na frente do número):

|  |  |  |  |
| --- | --- | --- | --- |
| Instrução realizada | Binário (A,B,Op.code) | Valor em Hexa (0x …) | Resultado em binário |
| **AND(A,B)** | 0 1 00 | 0x4 | 0 |
| **OR(A,B)** | 10 01 | 0x9 | 1 |
| **SOMA(A,B)** | 1 0 11 | 0Xb | 1 |
| **NOT(A)** | 0 0 10 | 0X2 | 1 |
| **AND(B,A)** | 0 1 00 | 0X4 | 0 |

**O que apresentar nesse exercício:**

**Exercício 1**

* **Um print da tela do Tinkercad mostrando a montagem e o programa.**

**Exercício 2**

* **Preencher a tabela com as instruções e os resultados.**
* **Um print da tela do Tinkercad mostrando a montagem e o programa. A seguir executar o teste (o mesmo programa de teste usado para a parte 1**

**para a unidade de 1 bit) e mostrar o resultado para cada uma das instruções executadas.**

**Mostrar os prints apenas da execução da instrução, não é necessário prints quando apenas uma atribuição de valores às variáveis for realizada.**