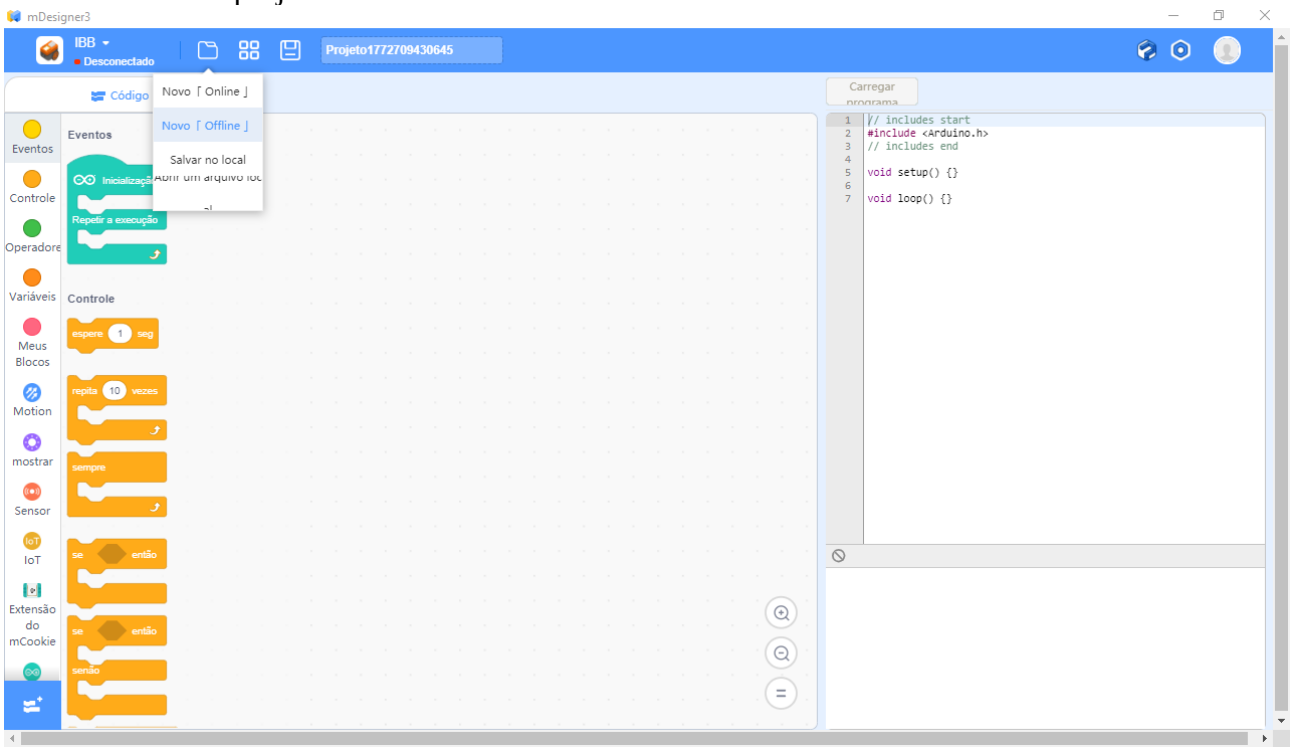
Trabalhando com o mDesigner e o Arduino

ATENÇÃO – Se algo não der certo, não desista. Tente de novo. Até pegar o ‘jeitinho’ tem que ter um pouco de paciência. Em muitos casos digitar simultaneamente as teclas ‘CTRL’ e ‘Z’ poderá voltar a última ação realizada. Se travar, reinicie o programa ou o equipamento. Peça ajuda se precisar. Tenha calma. Não pule etapas, preste atenção no que está acontecendo para ter domínio do ambiente e dos seus comandos. E, … divirta-se.

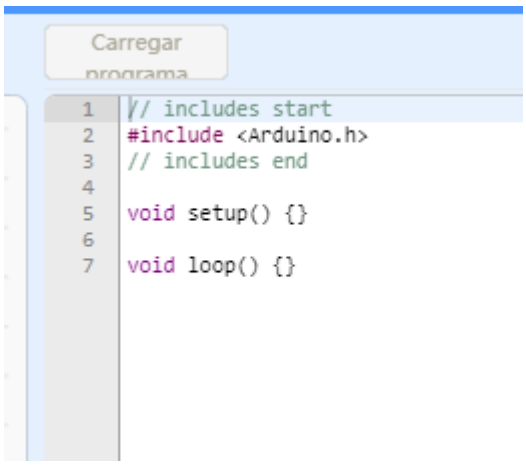
1 – Identifique o ícone do ‘mDesigner’ e acione-o

2 – Selecione um projeto ‘off-line’



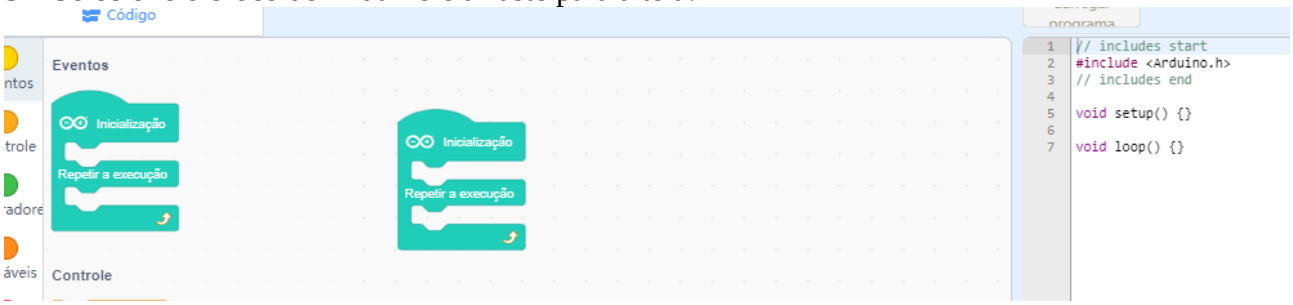
3 – Procure nos eventos o bloco do Arduino

4 – Veja que há um código sendo gerado

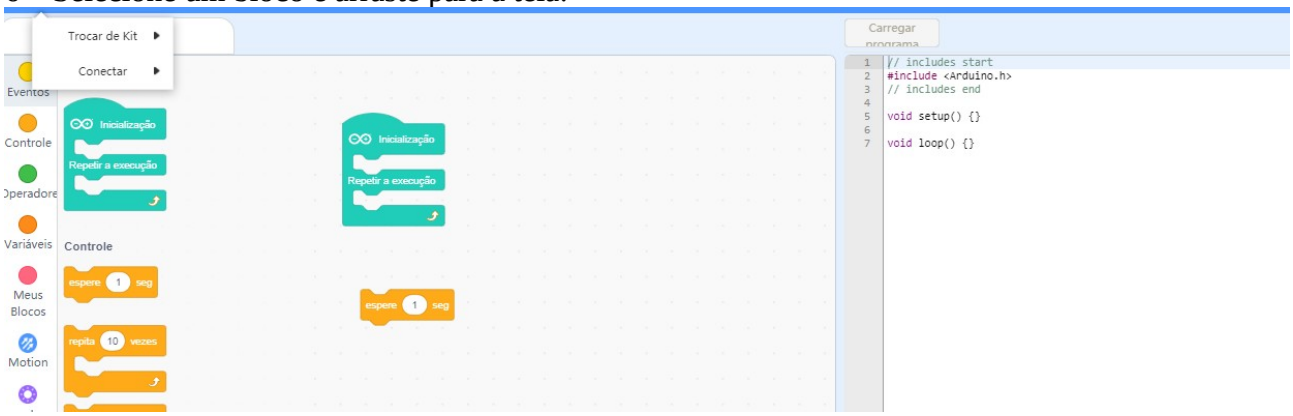


- ‘#include <Arduino.h>’ é uma instrução para que o program procure os comandos em uma biblioteca de códigos (a ‘ Arduino.h’);
- ‘void setup()’ é uma função que será executada uma única vez, quando o Arduino for ligado ou receber uma nova programação, e serve para explicarmos algumas alternativas ao Arduino, como por exemplo quais locais estamos utilizando.
- ‘void loop()’ é uma função de repetição. Todos os comandos colados dentro dela serão executados sequencialmente, e serão repetidos desde o primeiro quando o último acabar. Isto é chamado de ‘loop’, ou laço, ou repetição;
- as chaves, ‘{‘ e ‘}’, formam um bloco de comandos.

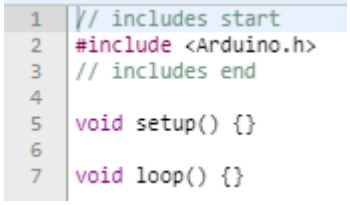
5 – Selecione o bloco do Arduino e arraste para a tela:



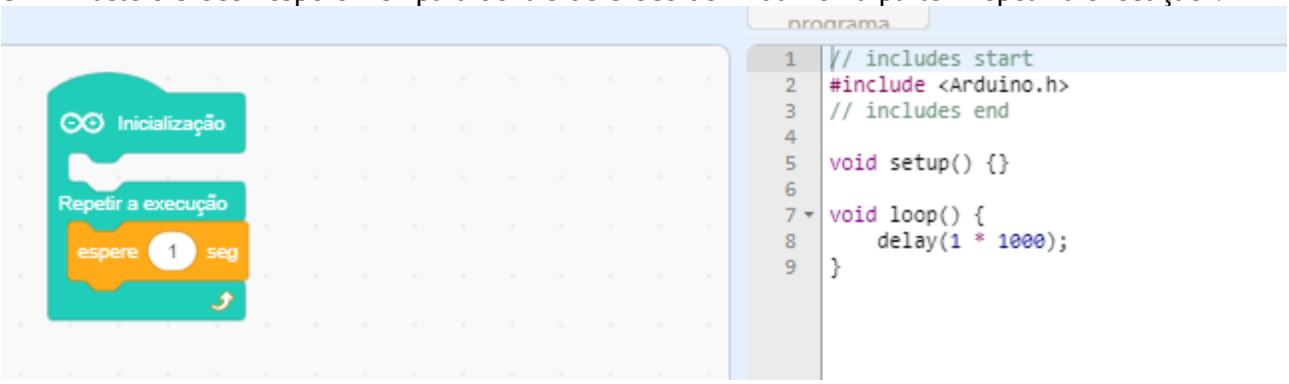
6 – Selecione um bloco e arraste para a tela:



7 – Note o código gerado – não mudou:



8 – Arraste o bloco “espere 1 s” para dentro do bloco do Arduino na parte ‘Repetir a execução’:



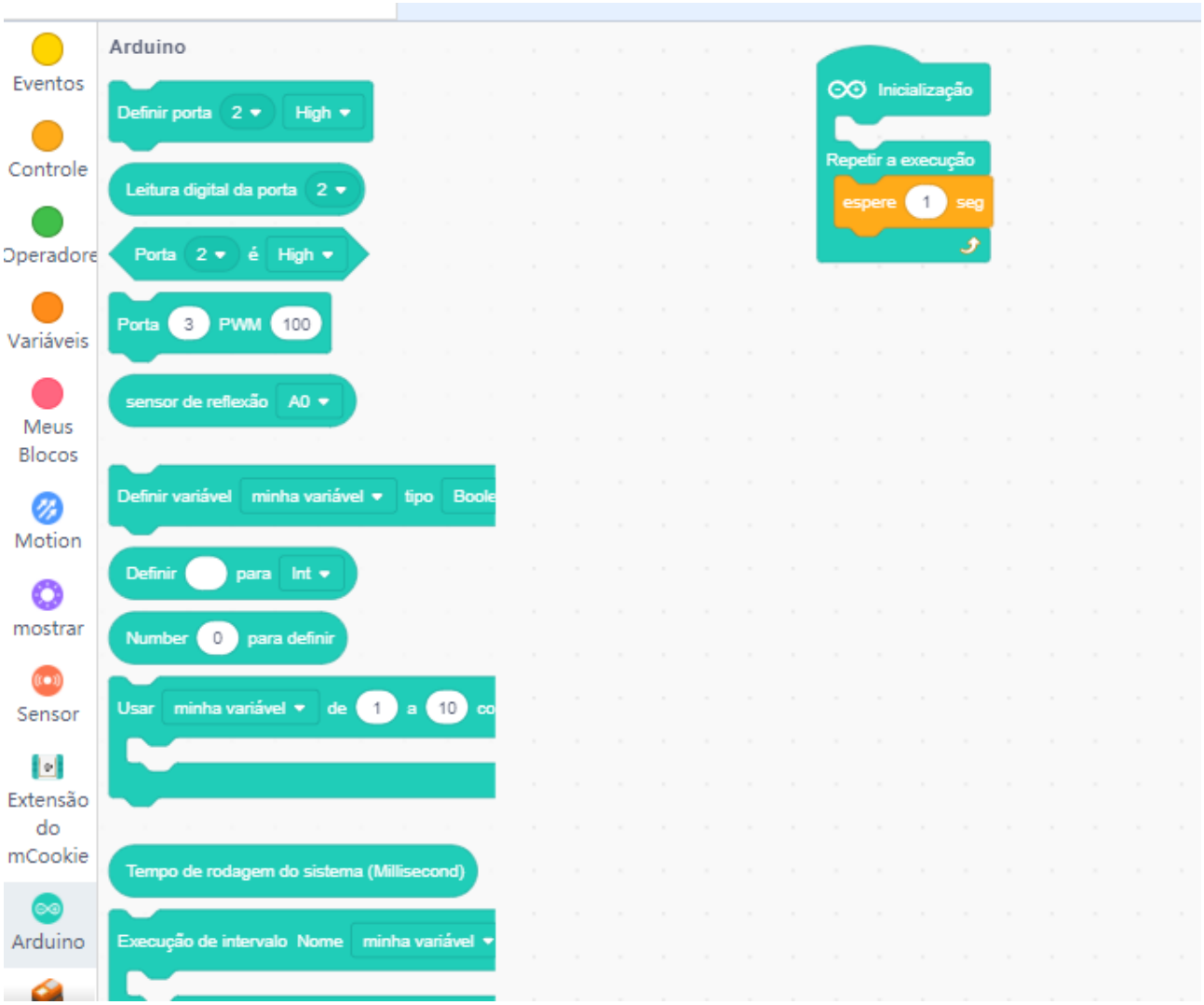
9 – Observe que agora o código foi modificado, pois há um sentido naquilo que foi inserido no bloco de repetição:

```
// includes start
#include <Arduino.h>
// includes end

void setup() {}

void loop() {
    delay(1 * 1000);
}
```

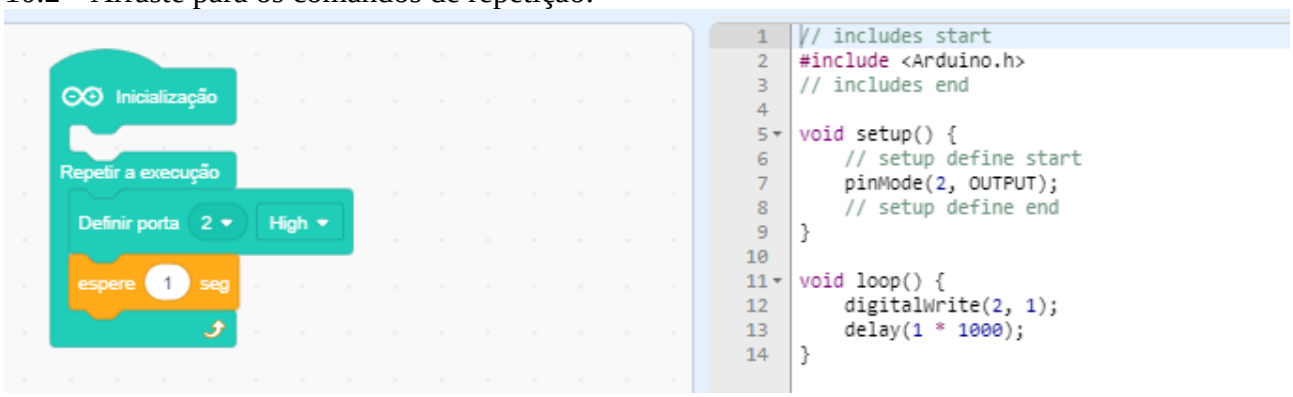
- os códigos estão sendo gerados em ‘linguagem C’ ou ‘linguagem C++’;
- os códigos utilizam comandos em Inglês;
- ao invés de utilizarmos um programa ao estilo Scratch, como o mDesigner que estamos utilizando, poderíamos utilizar qualquer editor de textos para criar o programa que o ambiente está criando para nós;
- o programa que está sendo ‘digitado’ pelo ambiente é chamado de **programa fonte**;
- em algum momento precisaremos enviar este programa para o Arduino. Mas, o Arduino não conhece nosso programa fonte, ele só fala a linguagem dele, formada por ‘1s’ e ‘0s’, chamada de linguagem binária. Então, teremos que **compilar** estes nossos comandos para a linguagem dele. O ambiente fará isso para nós.

10 – Vamos continuar. Procure no bloco do Arduino o bloco ‘Definir porta’; arraste-o para o bloco de comandos de repetição:,

10.1 – procure no bloco do Arduino



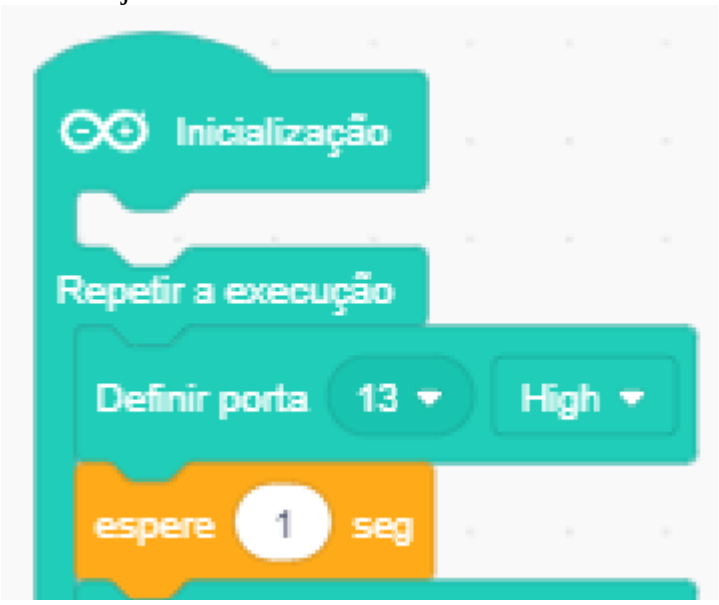
10.2 – Arraste para os comandos de repetição:



11 – Troque o número da porta:

- os locais de controle utilizados pelo Arduino são identificados por ‘portas’, que podem ser de entrada ou de saída;
- estamos mudando a porta para o número ‘13’;
- esta ‘porta’ corresponde ao LED interno da placa do Arduino (LED é um tipo de diodo capaz de emitir luz quando percorrido pela corrente elétrica; existem de várias cores, em sua placa poderá ser vermelho, ou verde, ou azul…).

11.1 – Veja como ficou:



- note que o código já foi ajustado!

11.2 – Código ajustado:

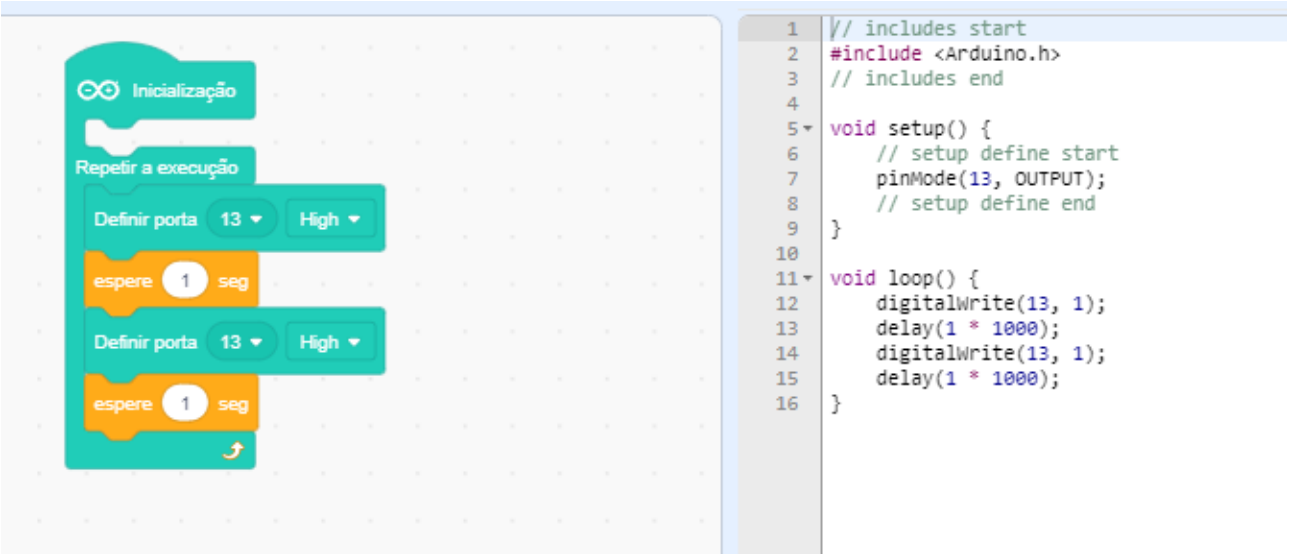
```
5  void setup() {
6      // setup define start
7      pinMode(13, OUTPUT);
8      // setup define end
9  }
10
11 void loop() {
12     digitalWrite(13, 1);
13     delay(1 * 1000);
```

- o comando ‘pinMode’ explica ao Arduino que queremos utilizar a porta 13 como saída (OUTPUT);
- o comando ‘digitalWrite’ explica ao Arduino que queremos enviar um sinal de nível alto (1) para a porta 13;
- o comando ‘delay’ faz com que o Arduino espere uma certa quantidade de milisegundos antes de executar o próximo comando. Neste exemplo ele vai esperar 1 x 1000 = 1000 ms, ou seja, 1 segundo.

12 – Vamos duplicar os comandos. Clique com o botão do lado direito do seu mouse sobre o bloco de comandos de definição da porta 13, e selecione ‘Duplicar’:

12.1 – Selecione o ‘Duplicar’:

12. 2 – Resultado:

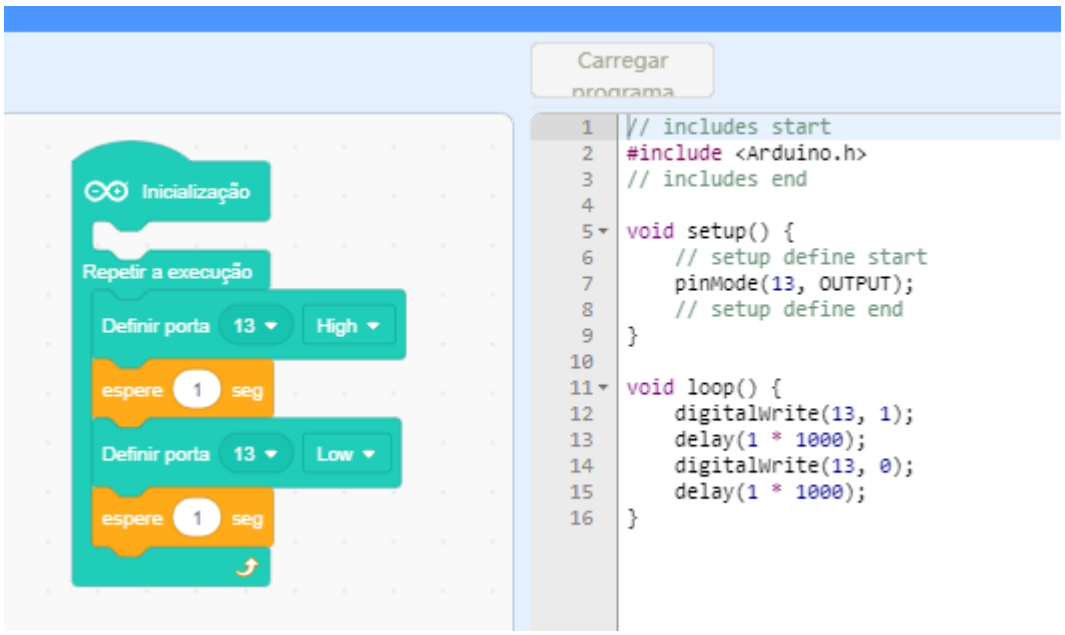


- note o código.

13 – Mude o segundo valor enviado para a porta 13:

13.1 – Mude o valor:

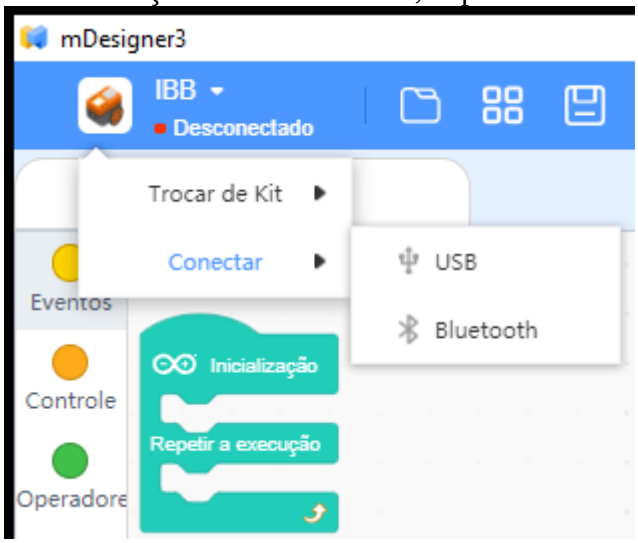
13.2 – Veja como ficou:



- agora a sequência de comandos vai enviar para a porta 13 um valor alto (1, o LED vai acender), esperar um segundo (delay), depois enviar para a porta 13 um valor baixo (0, o LED vai apagar) e esperar mais um segundo. E vai ficar repetindo isso enquanto houver energia.

14 – Conecte seu Arduino. Ligue o cabo USB na plaquinha do Arduino e conecte a outra extremidade em seu computador. Depois, selecione a conexão USB:

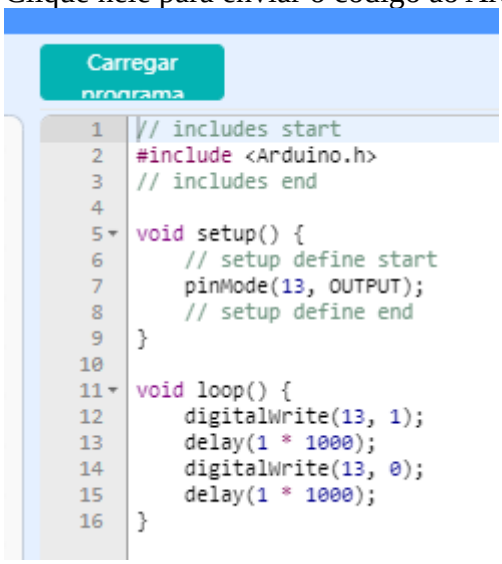
14.1 – Seleção da conexão USB, depois de conectar os cabos:



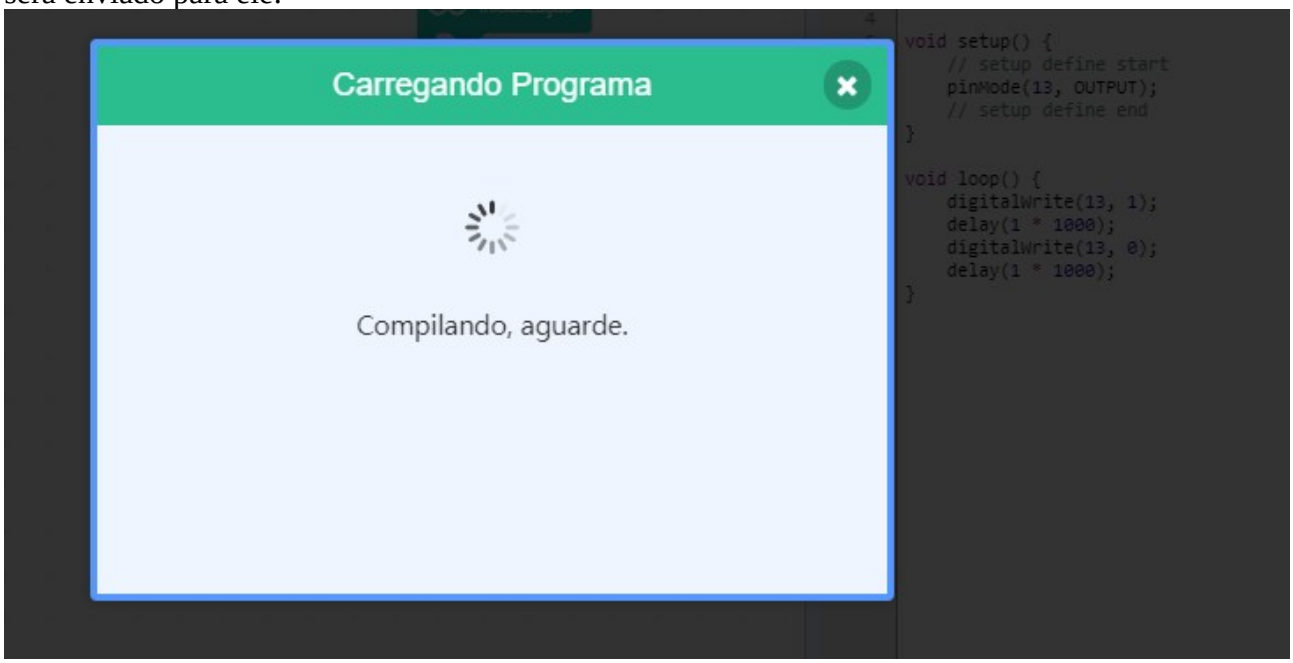
- o mDesigner vai procurar seus dispositivos conectados e exibir uma lista. Se tiver somente o Arduino conectado ele será p único na lista e é só clicar em conectar e aguardar (o mDesigner vai procurar o local USB em que o Arduino está conectado e ‘conversar’ com ele para verificar se a comunicação está ok);
- NOTA – quando você conecta o Arduino no computador a energia para a plaquinha do Arduino será fornecida pelo computador. O programa será transferido e o Arduino poderá funcionar ‘off line’, desconectado do computador, mas você terá que fornecer energia para isto acontecer, por exemplo por meio de uma bateria.

15 – Enviar o código:

15.1 – Note que quando o Arduino está conectado o botão de ‘Carregar programa’ fica disponível. Clique nele para enviar o código ao Arduino:



15.2 – Compilando… O código será convertido da nossa linguagem para a linguagem do Arduino, e será enviado para ele:



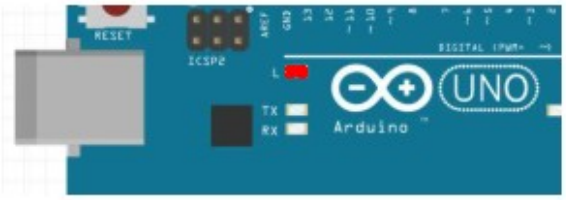
- o primeiro envio pode demorar um pouquinho.

Parabéns. O LED vai piscar.

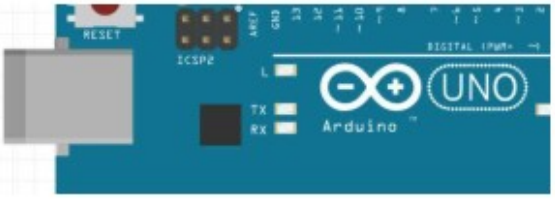
(se algo deu errado, lembre-se: tente de novo!)

A posição do LED varia de fabricante para fabricante.

LED da placa ligado



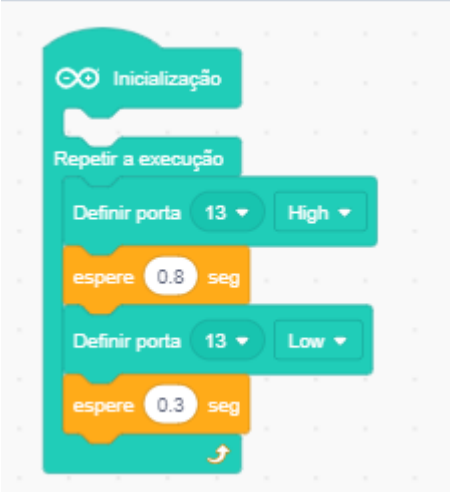
LED da placa desligado



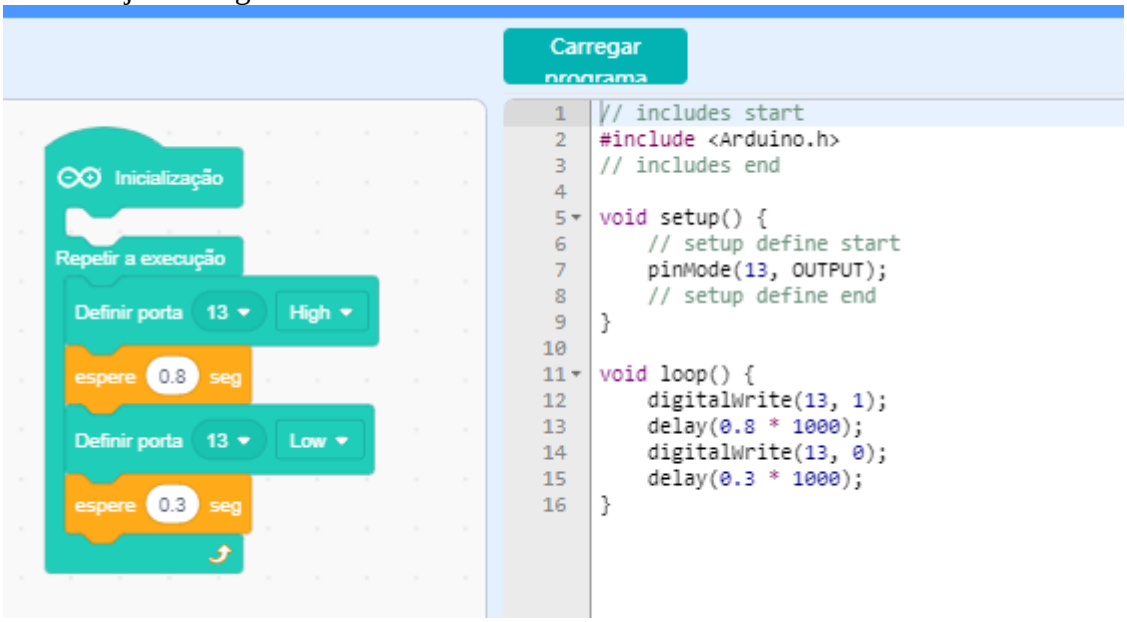
O LED vai ficar ligado e desligado de acordo com os tempos que você definiu usando o comando delay.

16 – Mude o tempo:

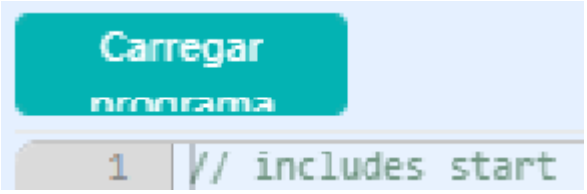
16.1 – Mude o valor do tempo (eu coloquei 0.8 e 0.3, você pode testar outros valores):



16.2 – Veja o código:

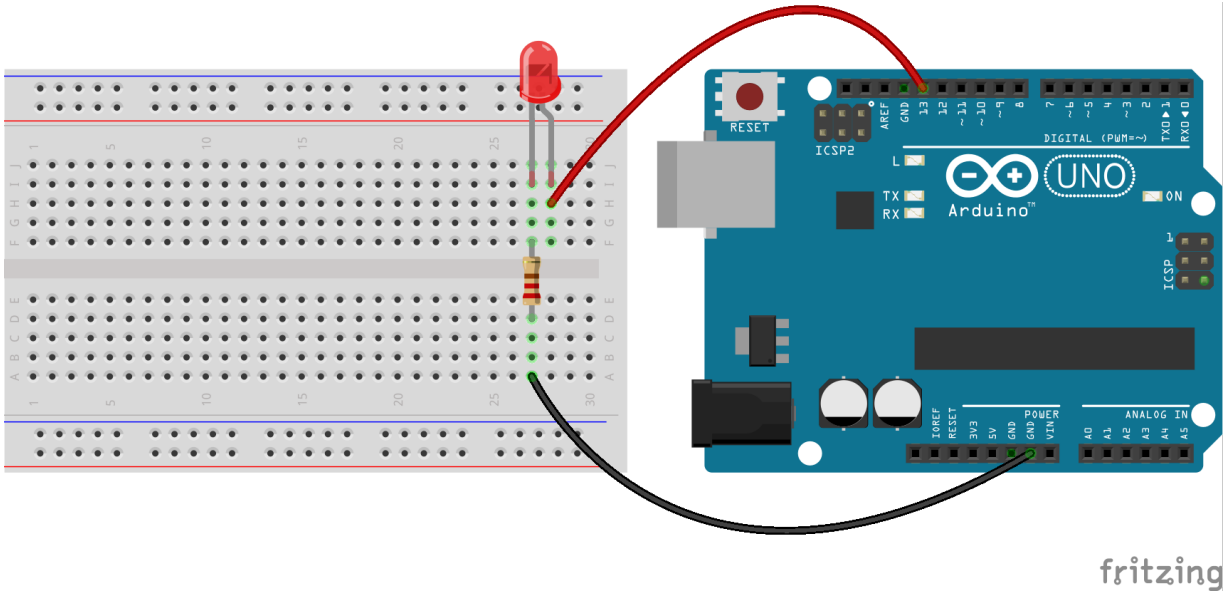


16.3 – Carregue novamente o programa:



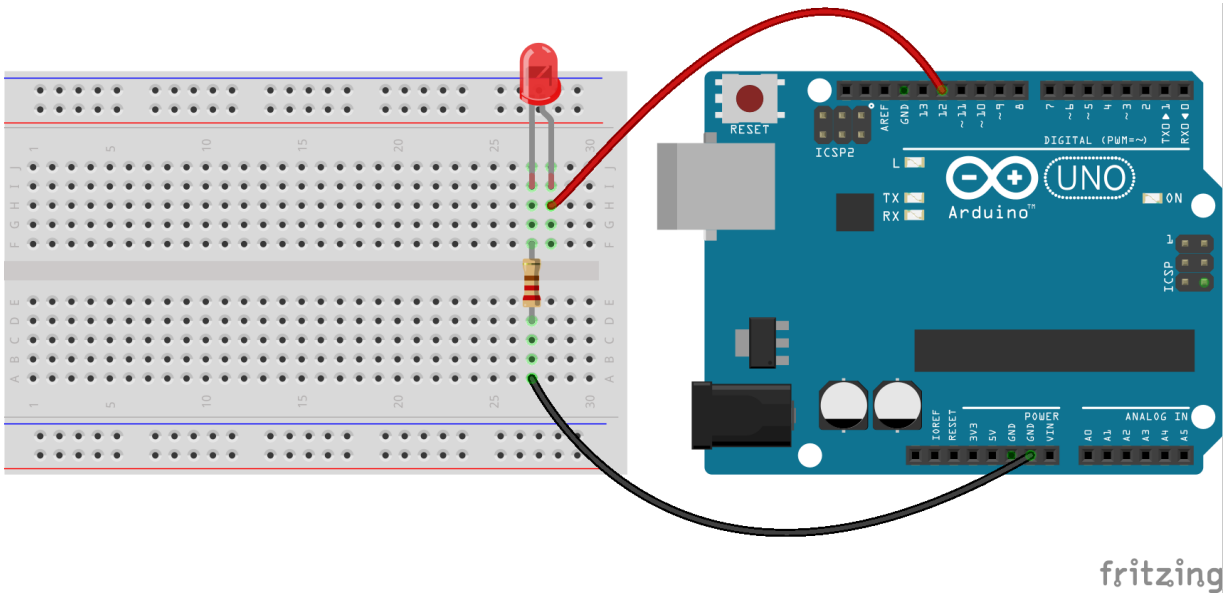
- SEMPRE que você mudar o programa você tem que enviar novamente para o Arduino (se você desconectar o Arduino ou houver outra situação o botão de envio não estará disponível. Se for o caso, volte ao passo ‘14’ e conecte o Arduino novamente).

17 – Ligue um LED externo ao Arduino:



- você vai precisar de uma placa de experimentação (protoboard), um LED (qualquer cor), dois fios (qualquer cor, eu usei vermelho e preto) e um resistor. O resistor tem valor específico que deve ser respeitado. Por exemplo {vermelho, vermelho, marrom, que dará 220 ohms – que é a unidade de medida de resistência elétrica}, ou {laranja, laranja, marrom, 330 ohms}, ou {amarelo, violeta, marrom, 470 ohms}, ou {marrom, preto, vermelho, 1000 ohms}. Estas são sugestões de valores comuns. Você pode utilizar outros: se o valor for mais alto o brilho do LED será menor (por ter mais resistência – se for muito alto ele não acende. Se o valor for muito baixo a corrente será muito alta e o Arduino pode queimar. Não ligue os LEDs sem usar um resistor! Você pode procurar na internet ‘códigos de cores de resistores’ para saber qual o valor do resistor que você tem disponível. O LED é polarizado - o lado maior é positivo (e vai na porta 13) e o lado menor é negativo (e vai no GND).

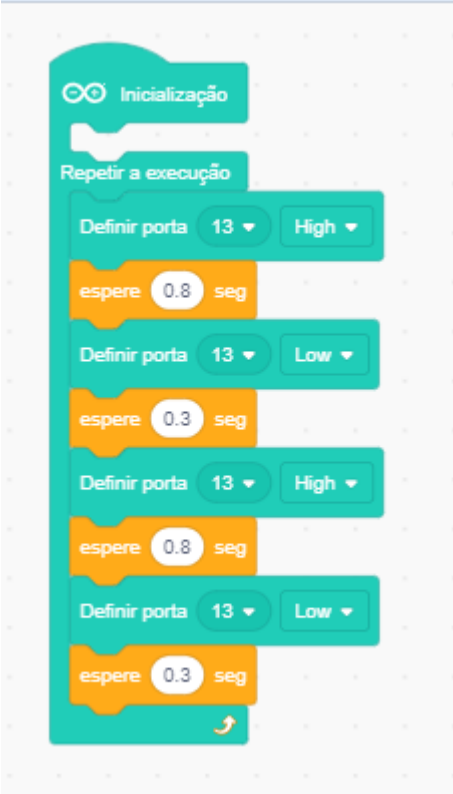
18 – Mude a porta:



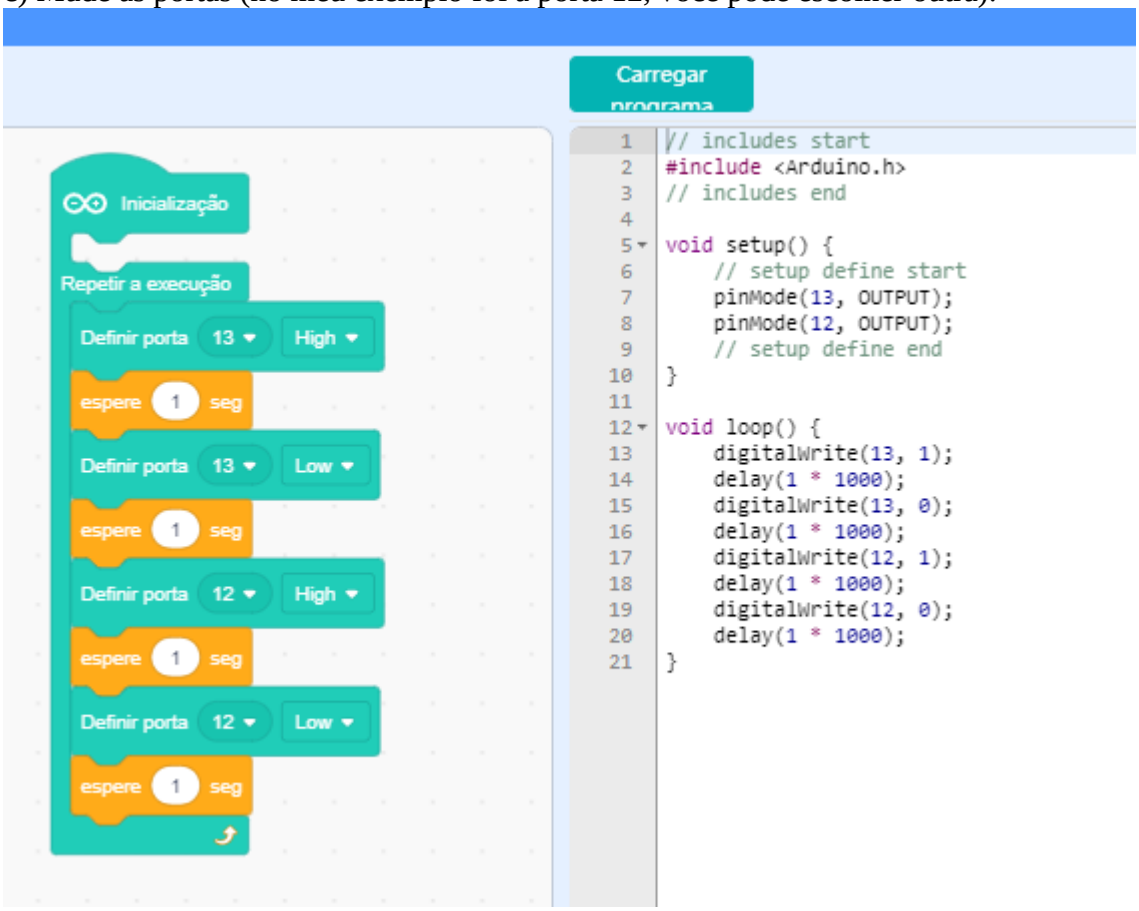
- se você somente mudar a porta o LED não acende. Por quê? Porque você não mudou o código…

- mude o código:
a) duplique os comandos

b) Resultado da duplicação



c) Mude as portas (no meu exemplo foi a porta 12, você pode escolher outra):



- carregue o programa. O LED da placa vai acender e depois apagar, daí o LED da porta 12 vai acender e depois apagar, e isto vai se repetir enquanto houver energia.

EXPERIÊNCIAS:

- Faça os LEDs da placa e externo acenderem e apagarem ao mesmo tempo.

- Faça com que quando um LED estiver aceso o outro esteja apagado, e vice-versa.

- Teste com mais um LED; depois, faça um sinaleiro com eles (cada LED acende sozinho na sequência). Depois, aumente seu sinaleiro para 6 LEDs e fala com que, quando um estiver ‘vermelho’ o outro fique ‘verde’ – pense na LÓGICA que você está utilizando. Construa um ALGORITMO que represente seu pensamento.

Em caso de dúvidas, pergunte aos seus colegas e ao instrutor que está na sala te ajudando.

Se tiver dúvidas, críticas e/ ou sugestões pode enviar por e-mail para simao@ufpr.br.

Bom divertimento. Boa aprendizagem.