

## *Programando com a Arduino IDE*

Como já dito, um dos motivos da popularização do Arduino é a sua facilidade de ser programado, seja pela linguagem amigável baseada em C/C++ (e com diversas bibliotecas para fácil comunicação e controle de sensores e atuadores), seja pelo modo eficiente e simples de se carregar o programa desenvolvido para a placa. Um dos grandes responsáveis por isso é a Arduino IDE, nascida do projeto de adaptação do Processing e integração com Wiring. Existem diversas IDEs que podem ser utilizadas para programar e realizar upload dos códigos para a placa (como Visual Studio Code, Eclipse, Ardublock, etc), mas a Arduino IDE sem dúvidas é a mais leve, limpa e simples. Nesta disciplina, esta será a nossa IDE padrão e é interessante conhecer algumas de suas características antes de efetivamente começarmos a programar o Arduino.

### *Instalando e configurando o ambiente de trabalho*

A Arduino IDE possui versões para Windows, Linux e Mac, contando com instaladores simples e diretos, sem necessidade de configurações adicionais. O primeiro passo é acessar a página de Downloads, escolher a versão mais adequada ao nosso sistema operacional e realizar o download: <https://www.arduino.cc/en/software/>

### Downloads



 **Arduino IDE 1.8.13**


The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

**SOURCE CODE**

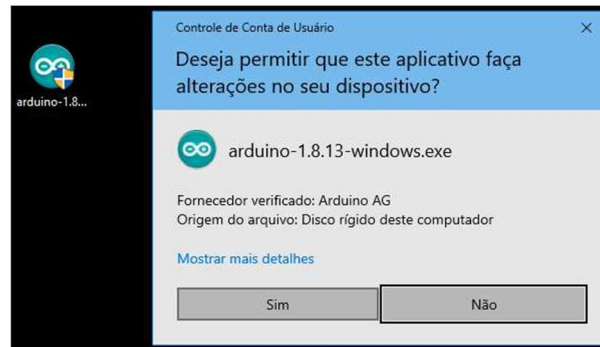
Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

**DOWNLOAD OPTIONS**

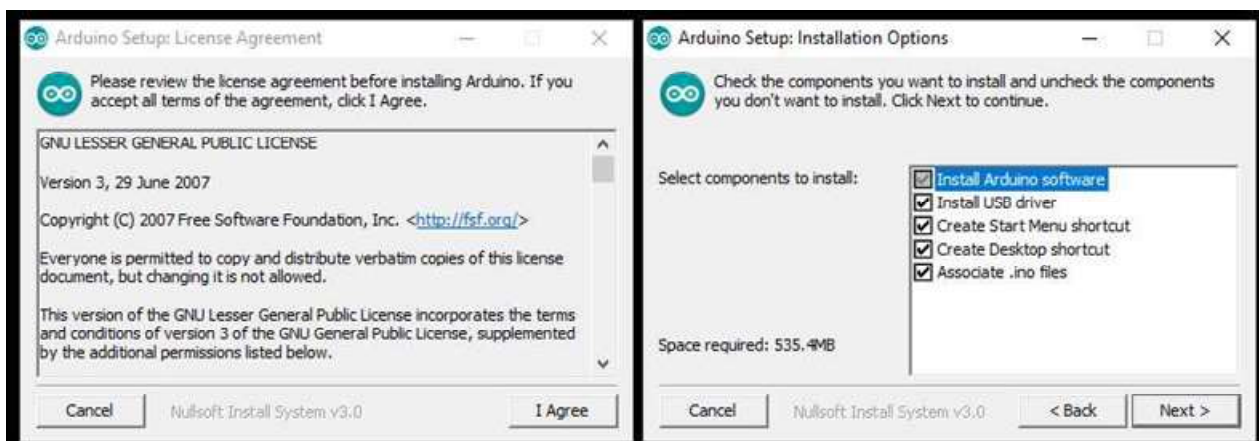
- Windows** Win 7 and newer
- Windows** ZIP file
- Windows app** Win 8.1 or 10 
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits
- Mac OS X** 10.10 or newer

[Release Notes](#) [Checksums \(sha512\)](#)

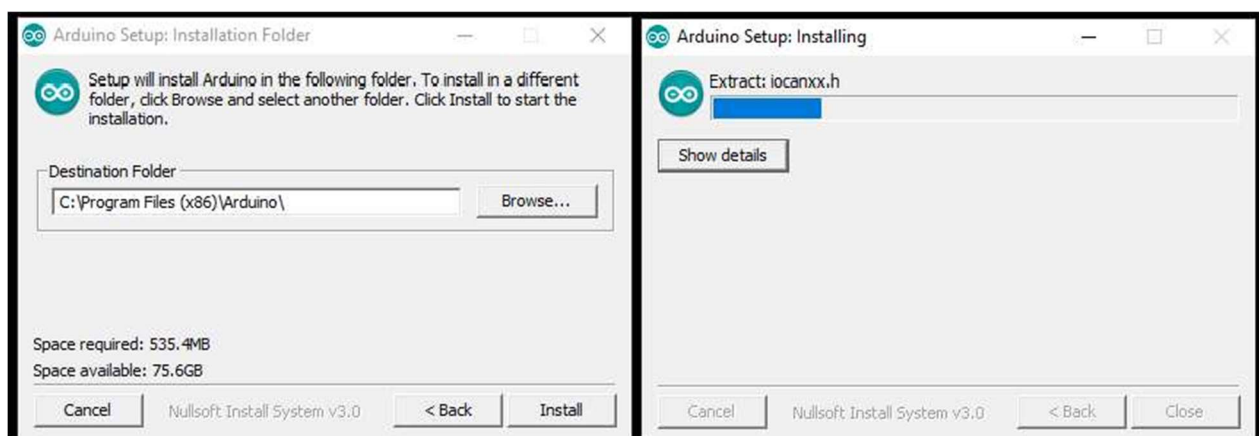
Após o download, um duplo clique no arquivo inicia a instalação. Dependendo do sistema operacional, uma mensagem diferente de confirmação é exibida, bastando clicar em Sim ou Ok. Neste exemplo, estamos realizando a instalação no Windows.



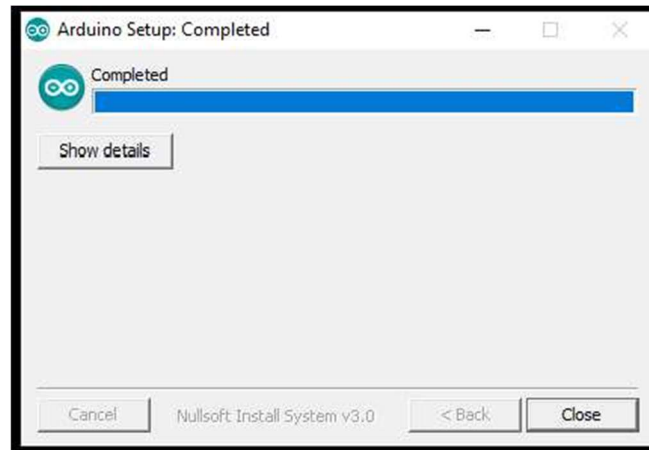
A próxima janela aberta traz as informações da licença de uso. Após lê-la, caso aceite, clique no botão "I Agree". Posteriormente, na tela de seleção de componentes a serem instalados, podemos deixar todos marcados, como já acontece por padrão, e clicar em Next.



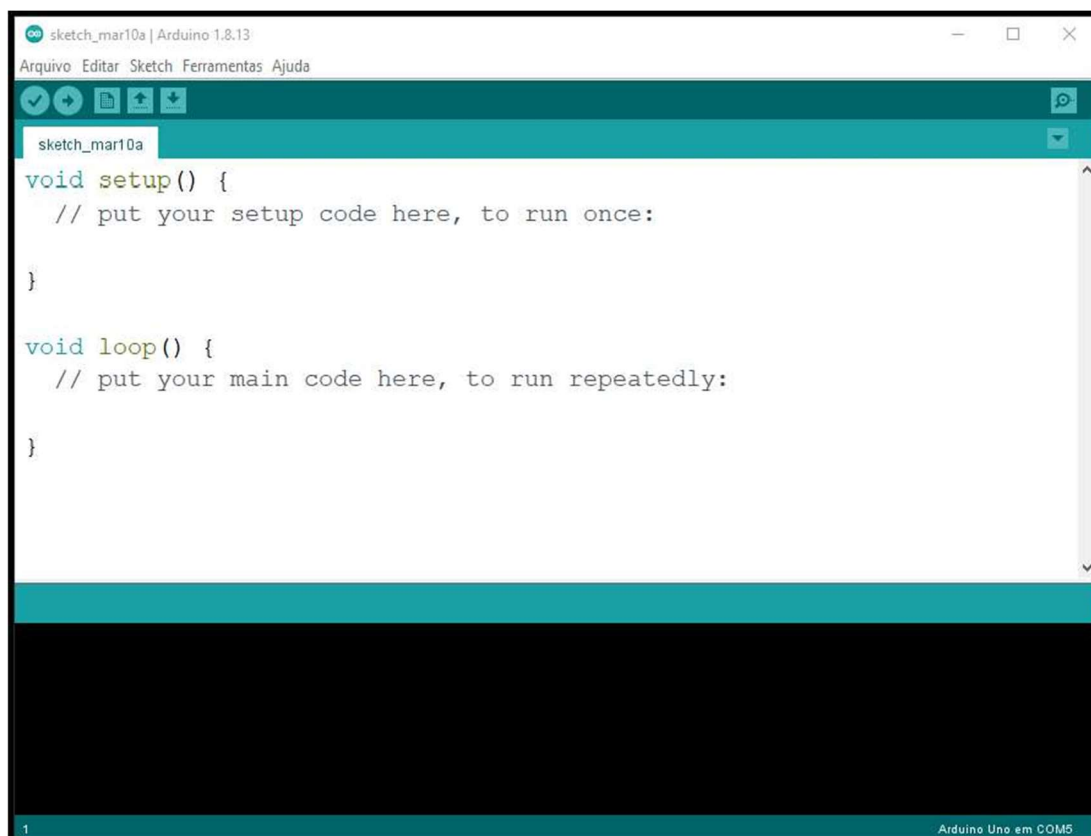
A próxima tela permite personalizar o local de instalação. Caso queira alterar o local padrão, clique em Browse. Para confirmar o local, clique em Next.



Após a instalação concluída, será exibida a última janela, mostrando que a execução do setup está completa. Basta então clicar em Close.



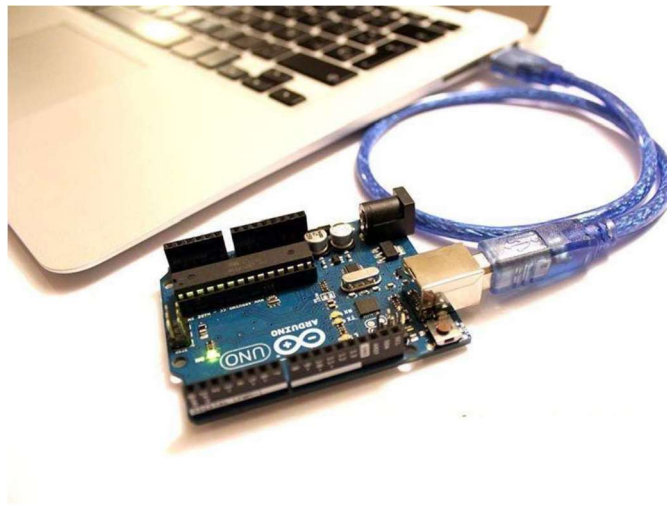
Para verificar se a instalação aconteceu de forma correta, localize o ícone do Arduino IDE (ou pesquisa através das ferramentas de seu sistema operacional) e clique sobre ele.



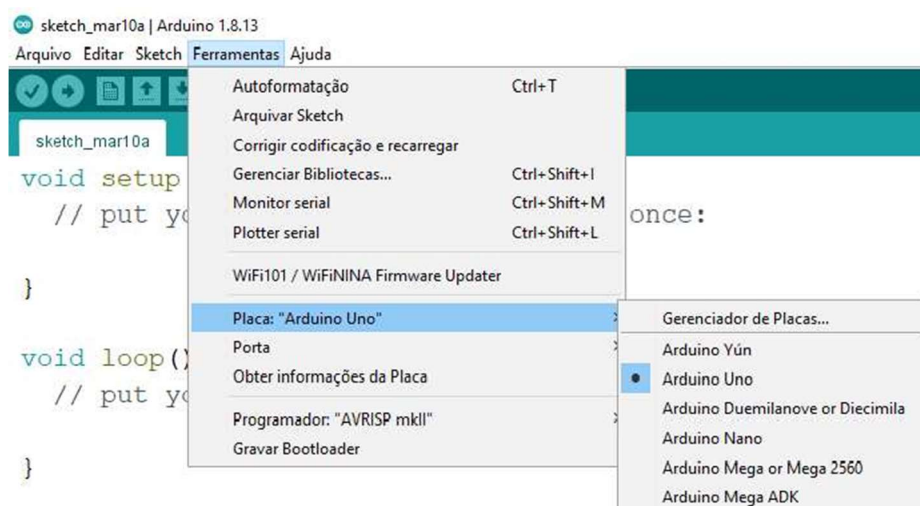
TUDO CERTO! Agora finalmente temos a ferramenta necessária para iniciar a programação das nossas placas Arduino.

### *Configurações iniciais*

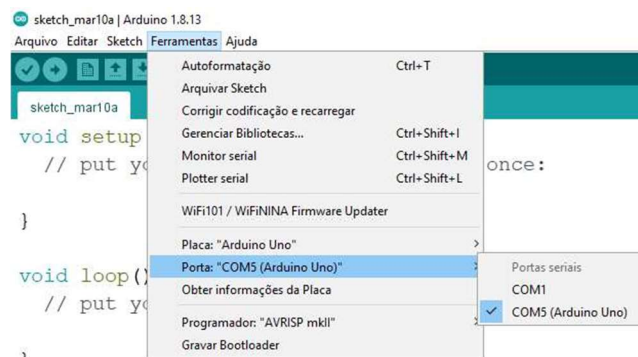
Com a IDE instalada, três configurações são necessárias. Primeiramente, devemos ligar a nossa placa ao computador através da porta USB. Ao fazer isso pela primeira vez, o sistema operacional deve reconhecê-la como um novo periférico USB. O led ON (à direita da palavra Uno) deve acender indicando que a placa está energizada e o led TX (à esquerda da palavra Arduino) deve piscar indicando que o Arduino consegue enviar mensagens ao PC existindo, portanto, comunicação entre os dois.



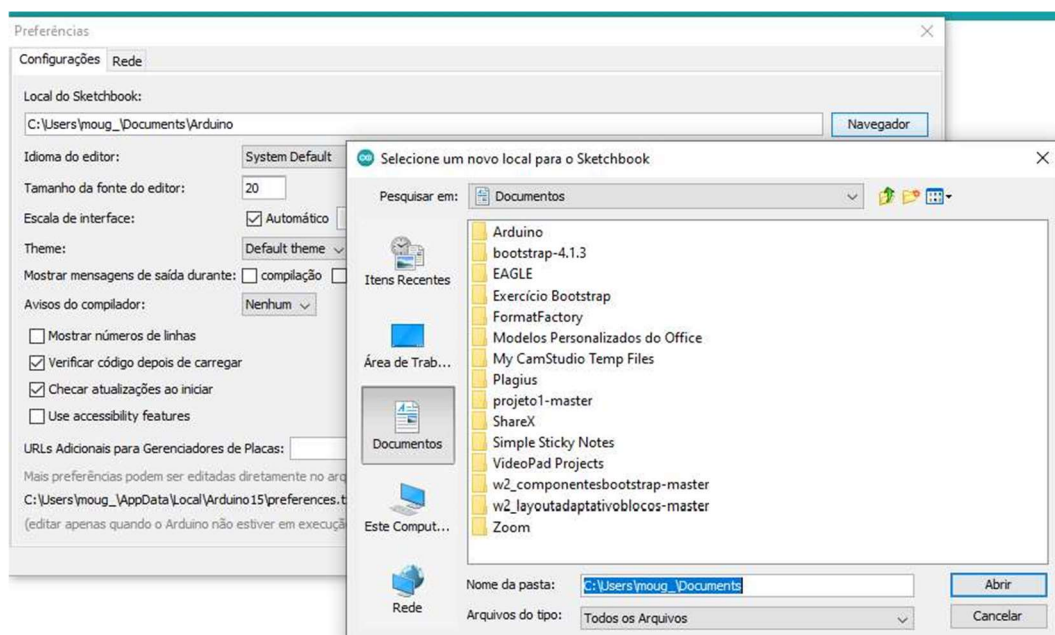
Com o Arduino já conectado, devemos informar à Arduino IDE qual o modelo da placa, para que sejam realizadas configurações internas importantes para a interpretação do código e o seu upload através do bootloader. Para isso, clique no menu Ferramentas e após na opção Placa. Muito provavelmente, a IDE já percebeu se tratar de nosso Arduino (no caso, o Uno) e sugeriu esse modelo de placa. Mas caso o modelo esteja configurado diferentemente disso, podemos ajustar clicando na opção correta.



A segunda configuração inicial necessária é informar à qual porta COM do PC nosso Arduino está conectado. Para isso, basta clicar no menu Ferramentas e ir até a opção Porta. Aparecerão todas as portas COM que naquele momento estão em funcionamento. Se o Arduino foi reconhecido e o melhor driver foi instalado para ele pelo sistema operacional, a identificação será fácil pois o modelo estará indicado ao lado da porta. Caso o sistema operacional tenha instalado apenas o driver USB padrão, o uso da placa ainda acontecerá sem problemas, porém o modelo não será indicado ao lado da COM. A alternativa é ver todas as COM apresentadas, remover o Arduino e verificar novamente as COM que sobraram. Aquela que sumiu é então a COM à qual a placa estava conectada.

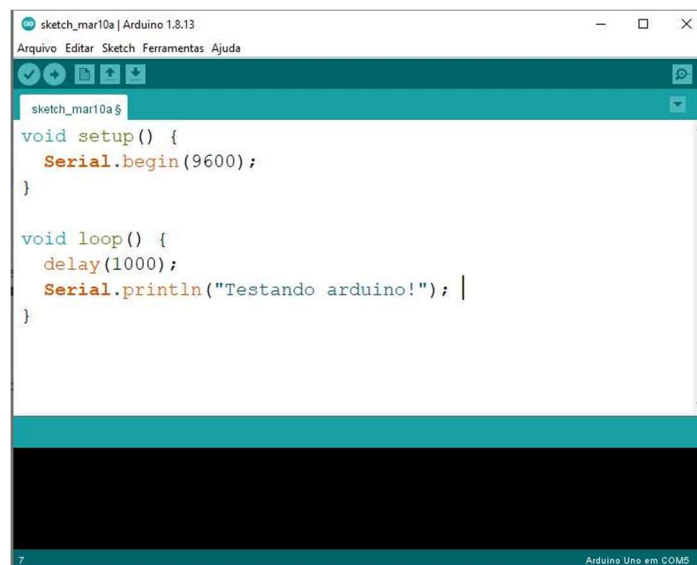


A terceira e última configuração inicial é dizer à nossa IDE a pasta base utilizada para o salvamento de nossos projetos. No Windows, por padrão, é criada uma pasta Arduino dentro da pasta Documentos do usuário. Para alterar, basta clicar no menu Arquivo e após na opção Preferências. Na janela aberta, podemos então alterar o caminho.



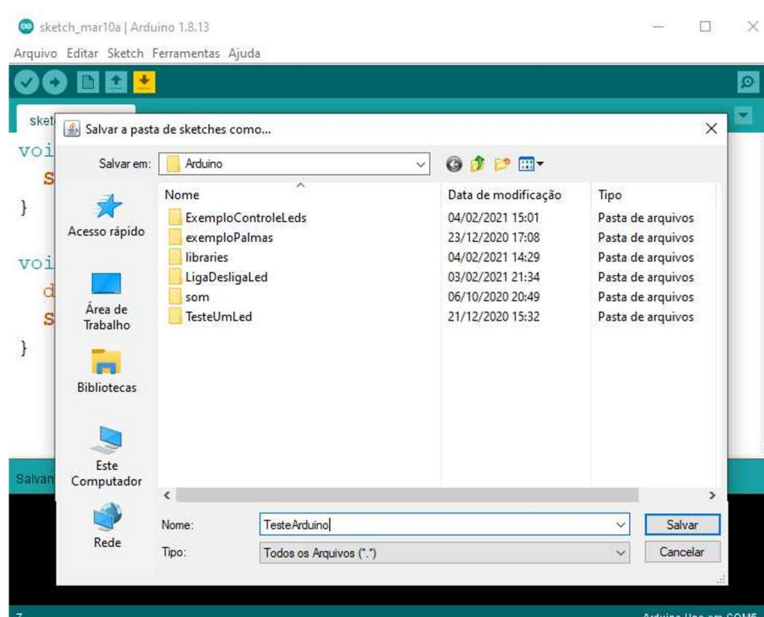
### *Testando a compilação, o upload e a execução do código*

Para termos certeza que tudo está funcionando corretamente, vamos criar um pequeno código, compilá-lo, enviá-lo para a placa e verificar a sua execução. Não cabe aqui explicar de forma detalhada este código (pois isso será feito no próximo capítulo) e o importante neste momento é entender o processo. Então, com o Arduino IDE aberto, digite as seguintes linhas dentro do método setup e dentro do método loop:



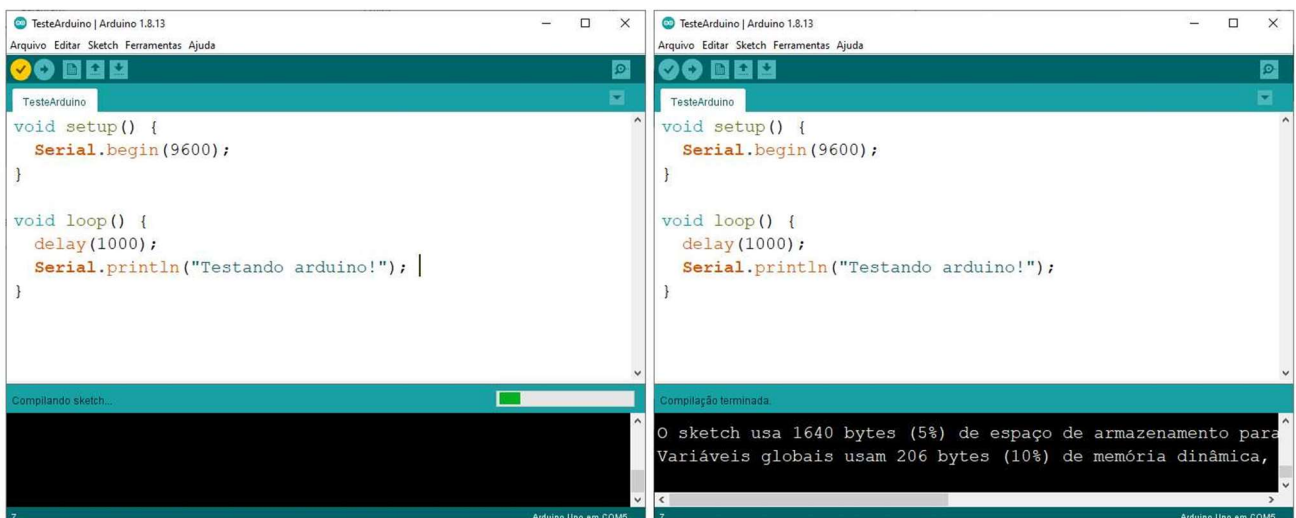
```
sketch_mar10a$  
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  delay(1000);  
  Serial.println("Testando arduino!");  
}
```

Com nosso código pronto, precisamos agora salvar o projeto. Para isso, devemos clicar no menu Arquivo e após na opção Salvar (ou utilizar o atalho Ctrl + S). Podemos então dar um nome ao nosso projeto (neste exemplo, TesteArduino) e selecionar um local para o salvamento (sendo que o diretório padrão configurado anteriormente será indicado).

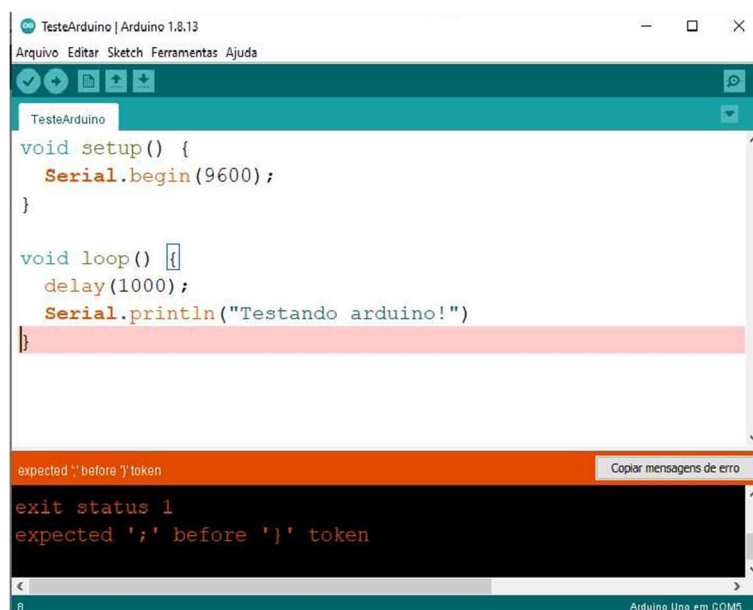




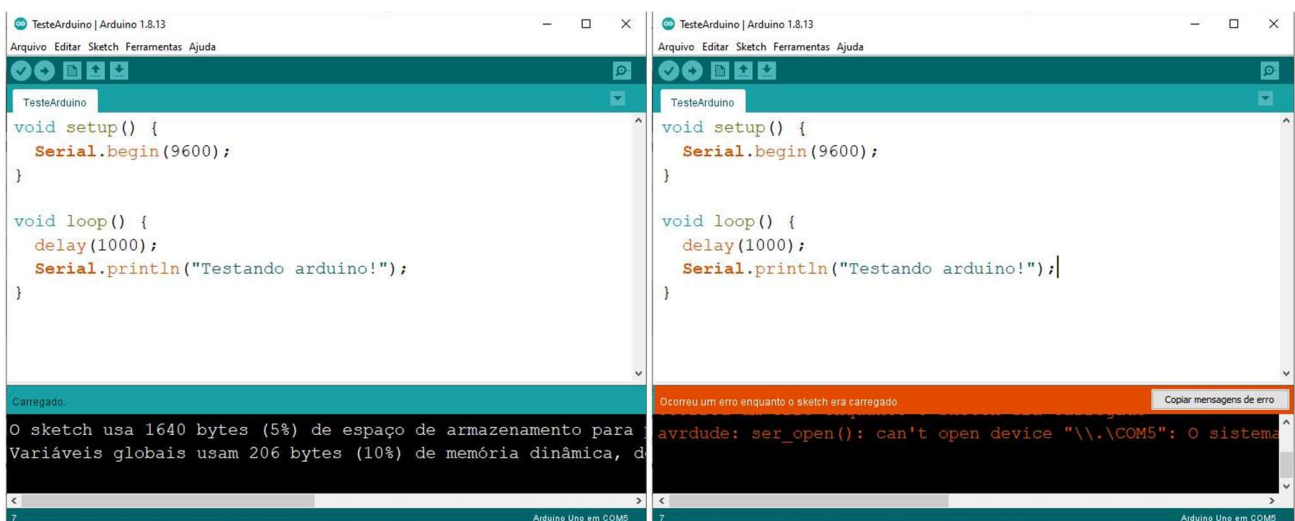
Com o projeto salvo, chega a hora de realizarmos a compilação e vermos se existe algum erro em nosso código. Para isso, basta clicar no botão **Verificar**, o primeiro à esquerda (logo abaixo do menu). Durante a verificação, aparecerá na área de mensagens (abaixo) a indicação que o Sketch (rascunho) está sendo compilado, juntamente com uma barra de progresso. Quando a compilação terminar, a indicação de compilação terminada será acompanhada de mensagens informativas sobre o tamanho do Sketch e o quando o mesmo ocupará da memória disponível na placa.



Caso algum erro for encontrado no código durante a compilação (para conseguir dar um exemplo, removemos propositalmente o ponto e vírgula do final do comando da penúltima linha) a barra inferior se tornará laranja e dentro da área escura o erro encontrado será indicado.



Consertados eventuais erros identificados e compilado com sucesso o código, é hora de realizar o seu upload para a memória da placa, através do seu bootloader. Este processo também é extremamente simples: basta clicar no segundo botão à esquerda (Carregar, que possui uma seta) e o código será novamente compilado e enviado para a placa. Quando isso ocorrer, repare que o led RX do Arduino piscará algumas vezes, indicando que a placa recebeu algo de fora (no caso, de nosso PC). Quando o upload estiver concluído, a IDE mostrará na parte inferior uma mensagem indicando que o carregamento ocorreu com sucesso. Se alguma falha ocorrer (a placa não estiver conectada por exemplo), a IDE irá apontar que o carregamento para o Arduino não teve sucesso.

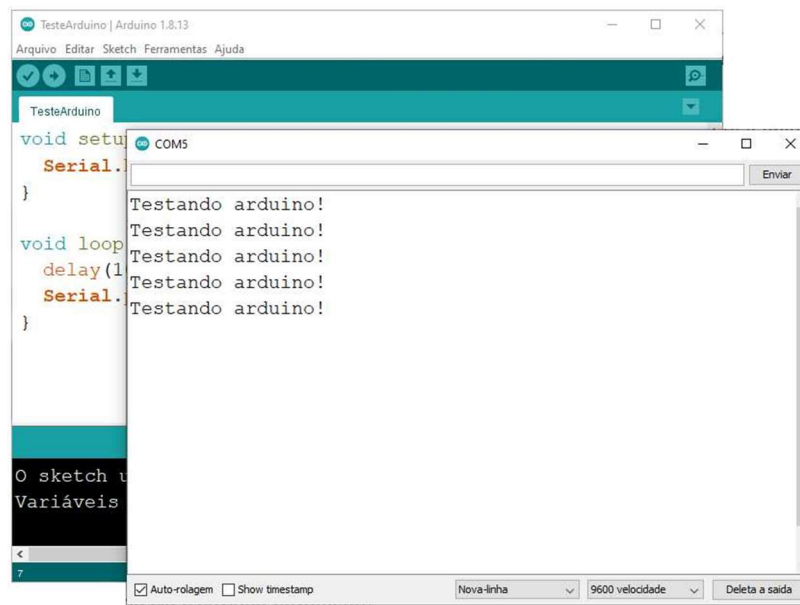


Após o carregamento do código para a placa, o mesmo já está pronto para começar a ser executado pelo Arduino. Não importa se a alimentação se der via USB (como está acontecendo), via conector de energia ou via pino Vin (utilizando fontes externas nestes dois últimos casos): enquanto estiver em funcionamento, a placa está rodando o último código carregado nela. E o interessante é que como essa memória que recebe o programa não é uma memória volátil, não importa se o Arduino for desligado por dias. Ao voltar a ser ligado, o código volta a ser executado pois está gravado dentro da placa.

Vamos testar então se esta execução está ocorrendo com sucesso. O comando `Serial.println` utilizado no exemplo, faz com que o Arduino envie via Serial (para quem estiver conectado através da USB a ele, neste caso, nosso PC), a mensagem “Testando arduino”, em um loop que se repete a cada 1 segundo. A Arduino IDE possui uma ferramenta que permite monitorar tudo que é recebido e enviado através da porta COM (Serial). Então, através dela, se o código está sendo executado corretamente e mensagem está sendo enviada com sucesso, conseguiremos visualizá-la. Esta ferramenta se chama



Monitor Serial e é acessada através do menu Ferramentas. Então, com o Arduino ainda ligado ao PC e já com o programa carregado para a sua memória, abra o monitor serial e verifique se a velocidade (frequência) de funcionamento que ele está utilizando é a mesma com a qual configuramos o envio Serial da placa no código (no nosso caso, 9600, que é a velocidade padrão). Deste modo, a mensagem deverá ser visualizada e perceberemos que a cada 1 segundo ela é enviada novamente.



Caso queiramos apagar o código que está na memória da placa, interrompendo definitivamente a execução, basta clicar e segurar por 3 segundos o botão Reset (ao lado da entrada do cabo USB da placa). O led L (ao lado do led TX) irá piscar algumas vezes confirmando que a memória foi limpa e que não existe mais código sendo executado pelo Arduino.

Com estes passos concluídos temos certeza que a IDE está bem instalada, o Arduino foi reconhecido pela Serial, existe comunicação com o PC, o upload está acontecendo com sucesso através do bootloader e a placa está funcionando corretamente e executando o código carregado em sua memória interna. Podemos então, efetivamente, começar a trabalhar com o Arduino.

