

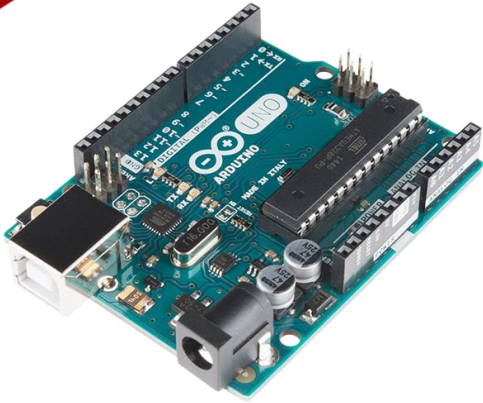


Internet das Coisas

Aula 09 – Introdução ao Arduino

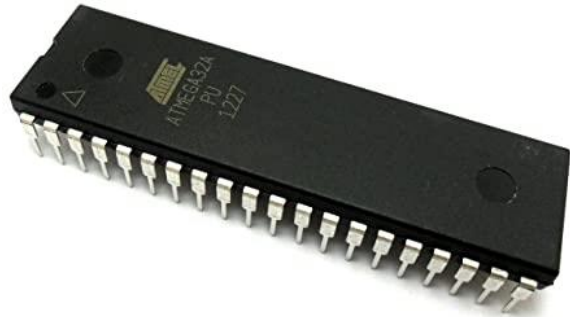


Introdução ao Arduino



O Arduino é a placa de prototipagem mais conhecida do mundo, sendo uma das primeiras a levar ao extremo o conceito de simplificar e popularizar a eletrônica. Seus principais objetivos era ser barata, funcional, simples de programar e simples de conectar a outros dispositivos, como sensores e atuadores. Foi criada em 2005 na Itália por 5 pesquisadores: Massimo Banzi, David Cuartilles, Tom Igoe, David Mellis e Gianluca Martino.

Introdução ao Arduino



Foi criada utilizando um microcontrolador Atmel, muito popular na época por possuir portas analógicas e digitais, portas de tensão, portas de comunicação e uma série de outras portas especiais, porém extremamente difícil de programar com as plataformas comumente utilizadas (como linguagem Basic ou mesmo Assembly).

Introdução ao Arduino

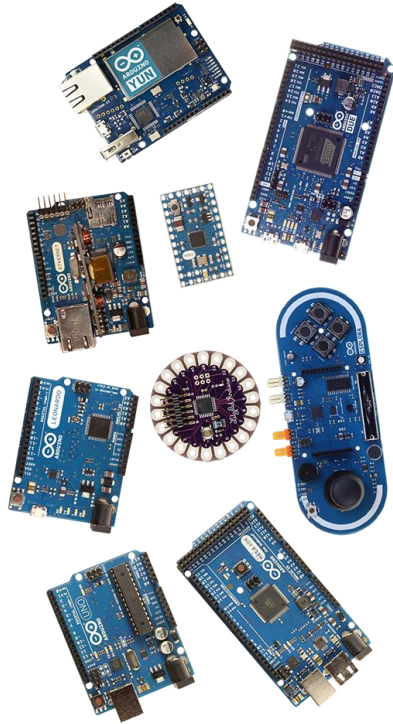


+



Aproveitando-se de uma IDE desenvolvida por um colega de Massimo (Casey Reis), o Processing (que tornava muito fácil a criação de descrições visuais e com uma linguagem de alto nível muito simples de aprender), foi desenvolvido o Wiring. Hernando Barragán, desenvolvedor do Processing, trabalhou ativamente com o time do Arduino para modificar a sua linguagem e dar origem a uma interface simples, amigável e poderosa.

Tipos de Arduino



A popularização do Arduino fez com que a placa tivesse uma série de variantes diferentes, com características diferentes que tornassem o uso de uma ou de outra mais adequado a diferentes tipos de projetos. Essa flexibilidade foi um dos motivos que ajudou ainda mais na popularização da plataforma e na sua incorporação em massa em projetos tanto acadêmicos, quanto do mundo Maker.

Tipos de Arduino

Uno



O primeiro Arduino comercial e, sem dúvida alguma, o mais popular. Alia a simplicidade de uso com um dimensionamento que o torna adequado à maioria dos projetos, sendo, ao mesmo tempo, um dos mais baratos e acessíveis de toda a família. Utiliza um microcontrolador Atmega328, com 14 portas digitais (sendo 6 delas PWM) e 6 portas analógicas, além de contar com portas de comunicação I2C (SDA e SCL) e outras portas especiais. A alimentação recomendável via USB é 5 V e via entrada de energia entre 7 V e 12 V (sendo as saídas de 5 V).

Tipos de Arduino

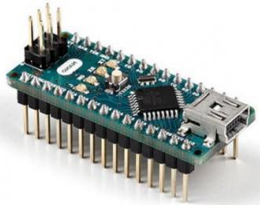
Mega



Desenvolvido para grandes projetos que necessitem conectar um grande número de sensores e atuadores. Tem entre os Arduinos o maior número de portas analógicas e digitais. Utiliza um microcontrolador Atmega2560 e possui 54 portas digitais (sendo 15 delas PWM) e 16 portas analógicas, além de contar com portas de comunicação I2C (SDA e SCL) e outras portas especiais. A alimentação é idêntica ao Arduino Uno com saídas também de 5 V. Outra diferença é a grande memória interna em comparação com o modelo anterior (256 KB contra 32 KB).

Tipos de Arduino

Nano



Um Arduino compacto, o Nano foi uma das primeiras tentativas de desenvolver uma placa que primasse pelas pequenas dimensões, acomodando todo o projeto em um encapsulamento relativamente pequeno (inclusive, conectado diretamente em uma protoboard, facilitando ainda mais o uso). Podemos dizer que ele é uma versão em miniatura do Arduino Uno pois utiliza o mesmo microcontrolador e possui as mesmas características dele, com exceção da conexão USB do tipo micro para acompanhar o seu tamanho enxuto.

Tipos de Arduino

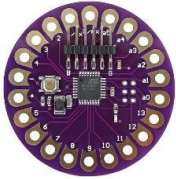
Pro Mini



O menor dos Arduinos, o Pro Mini foi desenvolvido para ser o mais adequado para projetos. Porém esta redução de tamanho cobrou um preço: ele perdeu sua porta de comunicação USB, sendo necessário conectá-lo através dos pinos RX e TX a outro Arduino para permitir o upload do código na placa. Outra perda foi da porta de alimentação, necessitando-se soldar o negativo da fonte a um dos seus GND e o positivo à sua porta Vin. Utiliza microcontrolador Atmega168 e possui uma versão com saídas de 5 V e outra versão com saídas de 3,3 V.

Tipos de Arduino

Lilypad



O mais fino dos Arduinos, o Lilypad é projetado para o desenvolvimento de projetos para wearables (vestíveis), permitindo a sua fácil incorporação sem fazer volume, de modo a ser facilmente escondido e com fios em forma de linhas que podem ser costuradas a ele e à roupa. Utiliza microcontrolador Atmega328P, contando com 14 portas digitais (6 deles PWM) e 6 portas analógicas, operando com uma alimentação entre 2,7 V e 5,5 V (e saídas 3,3 V), possuindo ainda um pino especial para a ligação de baterias de Íon-Lítio para sua alimentação.

Tipos de Arduino

Leonardo



O primeiro Arduino a utilizar o microcontrolador ATmega32u4, que dá a ele uma característica especial: pode ser reconhecido pelo dispositivo ao qual está conectado via USB como um mouse, teclado ou joystick. Isto permite que a placa envie sequências binárias interpretadas como caracteres, como comandos ou como ordens de movimentação do ponteiro do mouse, o que, na prática, possibilita o controle do dispositivo ao qual está conectado. Quanto a portas, alimentação e saídas, ele se equipara ao Arduino Uno.

Tipos de Arduino

Due



Esta é a primeira placa da família Arduino a ser equipada com um microcontrolador ARM (no caso, um ARM Cortex-M3), sendo, até o momento, o mais rápido e com maior poder de processamento pelo seu chip de 32 bits. Isso o torna especialmente útil em projetos onde necessitamos de rapidez, processamento poderoso e tomada de decisões rápidas. Possui 54 portas digitais (12 delas PWM), 12 portas analógicas, 4 portas seriais adicionais, mas apesar de tensões típicas de entrada da maioria dos Arduinos, suas saídas são de 3,3 V.

Tipos de Arduino

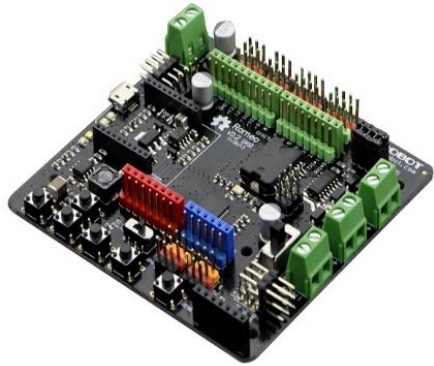
Esplora



Arduino desenvolvido especialmente para os entusiastas do movimento maker, amantes de games e pessoas que necessitam do desenvolvimento de interfaces de controle (de hardware ou software) usando o Arduino. Possui integrados componentes de controle, como acelerômetro, joystick analógico, push buttons, botões de ajuste, conectores, além de led RGB, buzzer e motor de vibração. Também possui uma interface de conexão que permite a fácil associação de displays coloridos.

Tipos de Arduino

Romeo



O Arduino Romeo foi especialmente projetado para o desenvolvimento de robôs e projetos de automação com atuadores mecânicos. O microcontrolador ATmega32u4 é montado em um encapsulamento que oferece uma placa com até 4 controladores lógicos para motores de passo ou servo motores, 3 pontes H para inversão polarizada em motores DC, conexão para módulo de comunicação via bluetooth ou rádio FM, entre outros. Conta com 20 portas digitais (sendo 7 delas PWM), 12 portas analógicas e algumas portas especiais.

Tipos de Arduino

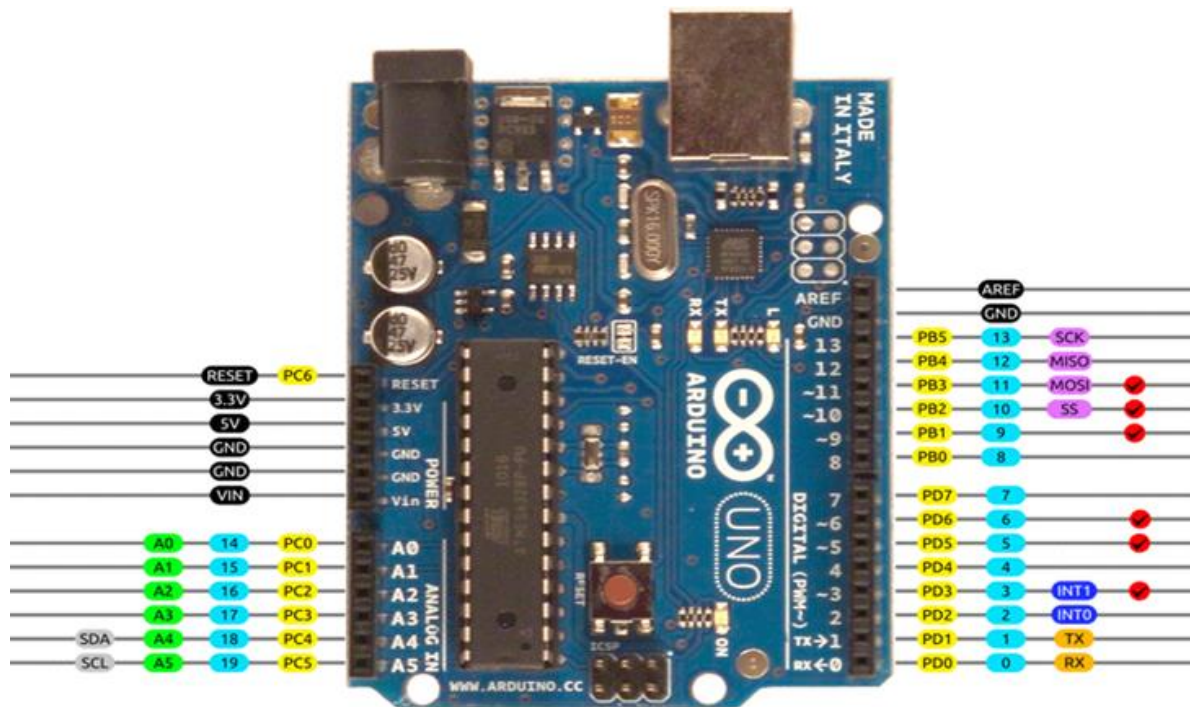
	UNO	MEGA	NANO	PRO MINI	LILYPAD	LEONARDO	DUE	ESPLORA	ROMEO
Microcontrolador	ATmega328	Atmega2560	ATmega328	ATmega328P	ATmega328P	ATmega32u4	ArmCortex-M3	ATmega32u4	ATmega32u4
Dimensões	75x55mm	108x55mm	43x15mm	33x18mm	50x50mm	75x55mm	102x54mm	165x61mm	89x85mm
Clock	16MHz	16MHz	16MHz	8MHz	8MHz	16MHz	84MHz	16MHz	16MHz
Vin	7 à 12 V	7 à 12 V	7 à 12 V	3,3 à 12 V	2,7 à 5,5 V	7 à 12 V	7 à 12 V	5V	7 à 12 V
Vout	5 V	5 V	5 V	3,3 V ou 5 V	3,3 V	5 V	3,3 V	5V	5 V
Memória flash	32Kb	256Kb	32Kb	16Kb	16Kb	32Kb	512Kb	32Kb	32Kb
Digitais	14 (6 PWM)	54 (15 PWM)	14 (6 PWM)	14 (6 PWM)	14 (6 PWM)	20 (7 PWM)	54 (12 PWM)	0	20 (7 PWM)
Analogicas	6	16	6	6	6	12	12	0	12
Seriaís (UART)	1	4	1	1	1	2	4	1	1
Interface progr.	USB	USB	Micro USB	FTDI	FTDI	USB	USB	Micro USB	Micro USB



Portas do Arduino

Os Arduinos possuem uma série de portas diferentes, cada uma dedicada a um determinado fim e em números diferentes. Ao planejar um projeto, conhecer essas portas é fundamental para dimensionar nosso circuito, verificar a oferta das portas necessárias e, principalmente, saber fazer o melhor uso possível de cada uma. Nesta disciplina, utilizaremos o Arduino Uno R3 e por isso analisar a sua pinagem (Pinout) é fundamental.

Portas do Arduino



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT

Portas do Arduino

Portas Digitais

As portas digitais do Arduino, trabalham com sinal digital, tanto para leitura quanto para escrita. Resumidamente, elas têm a capacidade de trabalhar com uso do sistema binário, onde um bit 0 é a porta não energizada e o bit 1 é a porta energizada (no caso do Arduino Uno, com uma tensão aproximada de 5 V). No caso da porta atuando como uma entrada (recebendo sinais de sensores e outros dispositivos), de tempos em tempos é realizada uma leitura dessa tensão e a presença ou ausência da mesma é convertida em um conjunto de 0 e 1.



Portas do Arduino

Portas Digitais

De forma análoga, o mesmo ocorre quando as portas digitais funcionam com portas de saída, escrevendo na porta 0 ou 1 (ou seja, sem tensão ou com tensão) de tempos em tempos. Estas portas são indicadas para trabalhar com sensores e atuadores que também utilizem sinal digital, mas também como portas de controle, ligando e desligando dispositivos externos (seja um led, seja um relé). Na leitura de dados (de um sensor, por exemplo) é possível receber diretamente o valor exato da grandeza medida de forma binária.

Portas do Arduino

Portas PWM

As portas PWM (Pulse Width Modulation) são portas digitais especiais com a capacidade de transformar sinais digitais em modularizações de sinal que permitam, na prática, regular a saída modificando a sua tensão (e consequentemente a sua corrente). Imaginando um led ligado a uma porta digital normal, o máximo que poderemos fazer com ele é ligá-lo ou desligá-lo. Porém, se essa porta digital for uma PWM, podemos regular a sua saída e, deste modo, alterar a luminosidade do led, fazendo que o mesmo brilhe mais forte ou mais fraco.

Portas do Arduino

Portas Analógicas

As portas analógicas trabalham com sinais que podem variar entre 0V e 5V. Assim, um sensor analógico é especialmente desenvolvido para, ao medir a grandeza ou o fenômeno desejado, variar a tensão emitida. Por exemplo, um sensor de temperatura analógico que mede temperaturas entre 0 e 100 graus celsius, é construído para ao medir 0°C (o primeiro valor possível) gerar uma tensão de 0 V e ao medir uma temperatura de 100°C (o último valor possível) gerar uma tensão de 5 V. Qualquer temperatura acima de 0°C e abaixo de 100°C, irá gerar uma tensão diferente (acima de 0 V e abaixo de 5 V).



Portas do Arduino

Portas Analógicas

Estes sensores são geralmente mais baratos e rápidos que os digitais (por não necessitam de conversões binárias para devolver o valor final pronto) mas, por isso mesmo, têm uma maior dificuldade no uso, já que sempre necessitamos fazer um mapeamento entre a tensão medida e o quanto ela representa dentro da grandeza medida.

Portas do Arduino

Portas de Alimentação

O Arduino Uno pode ser alimentado de 3 formas diferentes: pela sua porta USB (com uma entrada de 5V), pelo seu conector de alimentação (permitindo carregadores ou pilhas com uma tensão recomendada entre 7V e 12V) e pela sua porta Vin (soldando uma fonte ou bateria com entrada também entre 7V e 12V). Alimentada a placa, ela garante uma saída de 5V nas suas portas digitais e de até 5 V nas suas portas analógicas. Porém, para isso, teríamos que configurar as portas (via código) como portas de saída e escrever na sua saída que estão em estado alto (como se fosse um estado “ligado”).

Portas do Arduino

Portas de Alimentação

Para simplificar então a alimentação de sensores, atuadores e outros dispositivos externos, o Arduino Uno oferece algumas portas com tensão fixa e que estão sempre ativas: uma com tensão de 5V, outra com tensão de 3.3V. Além disso, o V_{in} , caso não utilizado para alimentar o Arduino, entregará uma tensão igual à tensão de alimentação da placa (se o Arduino está alimentado por uma fonte de 12V no seu conector, o V_{in} oferecerá uma tensão de saída fixa também de 12V, por exemplo). Além disso, temos 2 portas GND que são o terra (negativo).



Portas do Arduino

Porta Reset

A porta reset serve para, quando acionada, resetar a nossa placa, apagando o código carregado nela pelo bootloader. Podemos também fazer isso manualmente, pressionando o botão reset presente na placa (ligado à mesma porta). Mas é interessante esse acionamento programaticamente para situações extremas. Por exemplo, se detectamos um comportamento inesperado de nosso sistema ou na execução de nosso código que pode trazer riscos à segurança do usuário. Caso não consigamos parar essa execução de outra forma (em um loop travado, por exemplo), resetamos a placa e forçamos a interrupção de qualquer execução.



Portas do Arduino

Porta RX e TX

Os pinos digitais 0 e 1 do Arduino Uno, são dedicados às portas RX e TX, de comunicação serial direta, capazes de enviar comandos e informações (TX = Transmitter) ou de receber comandos e informações (RX = Receiver) que são interpretadas pela placa. Isso permite desde a programação do Arduino por outro Arduino (como o Pro Mini que não tem porta de comunicação USB), de um sensor ou atuador pelo Arduino (configurar os parâmetros de uma rede WiFi em um módulo ESP8266) ou mesmo para que Arduinos possam trocar informações durante o funcionamento (mensagens, valores de variáveis, medições, etc).



Portas do Arduino

Portas de Interrupção

Os pinos digitais 2 e 3 do Arduino Uno, são dedicados às portas de interrupção. Como o Arduino utiliza microcontroladores com apenas um núcleo, temos um monoprocessamento, que na prática fará com que quando entrarmos em uma tarefa muito pesada (um laço que executa várias vezes, um método que demora muito para ser executado), enquanto essa tarefa não terminar, ficamos “travados” na linha de código.



Portas do Arduino

Portas de Interrupção

Agora imagine um sistema crítico que realiza tarefas pesadas mas, caso um sensor detecte uma situação de perigo, seja necessário interromper a execução. Se este sensor estiver ligado a uma das portas de interrupção, ele poderá ser lido em paralelo, de forma independente, mesmo que uma tarefa pesada esteja sendo realizada e a execução de nosso código esteja parada. Assim, conseguimos implementar uma camada de segurança ao nosso sistema.

Portas do Arduino

Portas I2C (SDA e SCL)

Os pinos analógicos A4 e A5 do Arduino Uno, são dedicados às portas SDA (Serial Data) e SCL (Serial Clock), utilizadas para permitir o uso do protocolo de comunicação I2C. O I2C (Inter-Integrated Circuit), é na verdade uma comunicação serial, porém com uma característica muito interessante: trabalha com endereçamento. Isso permite que possamos ligar quantos sensores e atuadores I2C quisermos na mesma porta, pois na hora de ler um valor ou enviar um valor, o endereço deles (que deve ser único) será utilizado para auxiliar no direcionamento dos pacotes.



Portas do Arduino

Portas SPI

De forma análoga ao I2C, as portas SPI (Serial Peripheral Interface) são uma comunicação serial com endereçamento mas para a ligação entre placas ou entre a placa e periféricos externos. No Arduino Uno, são implementadas nas portas digitais 10, 11, 12 e 13.



Portas do Arduino

Portas AVR

As portas AVR nada mais são do que as portas que podem ser controladas via programação no Arduino. Incluem todas as portas digitais, todas as portas analógicas, além da porta reset.

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 [Get](#)

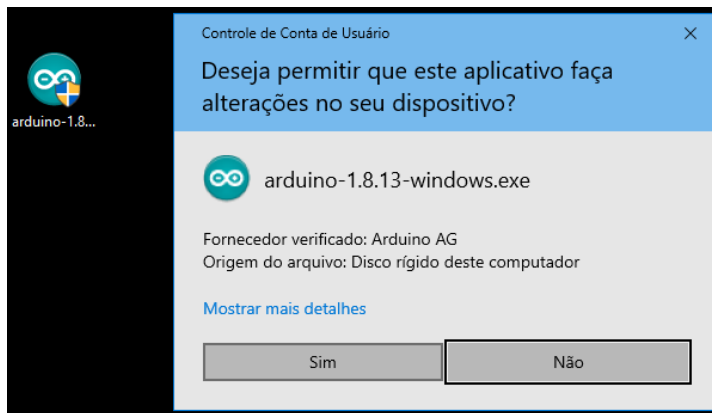
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Mac OS X 10.10 or newer

Release Notes Checksums (sha512)

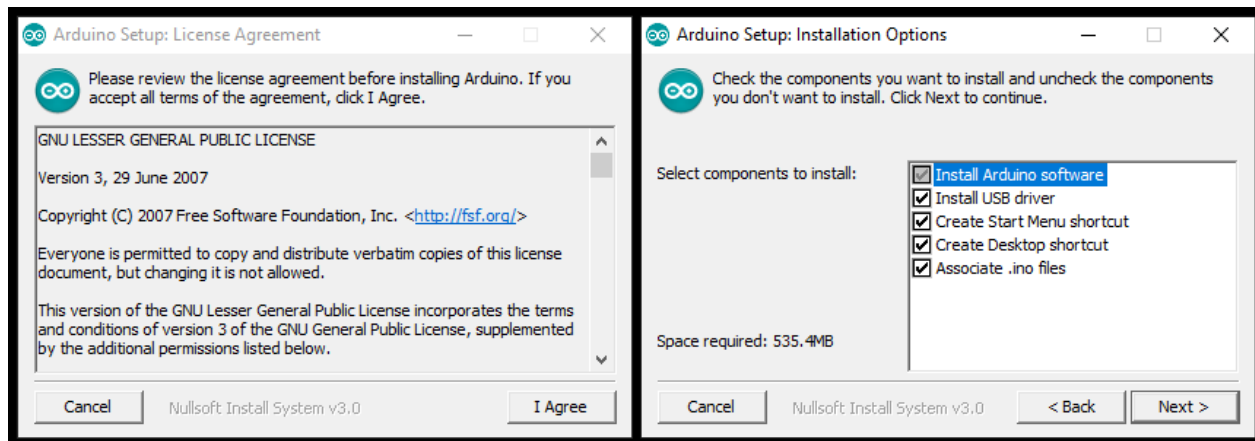
Configurando o ambiente de trabalho

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.



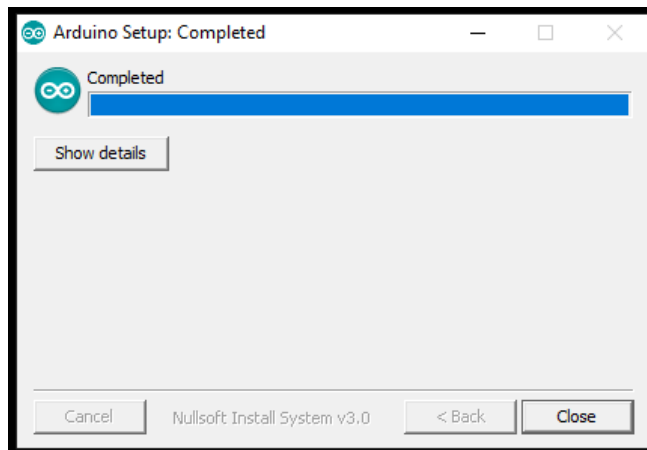
Configurando o ambiente de trabalho

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.



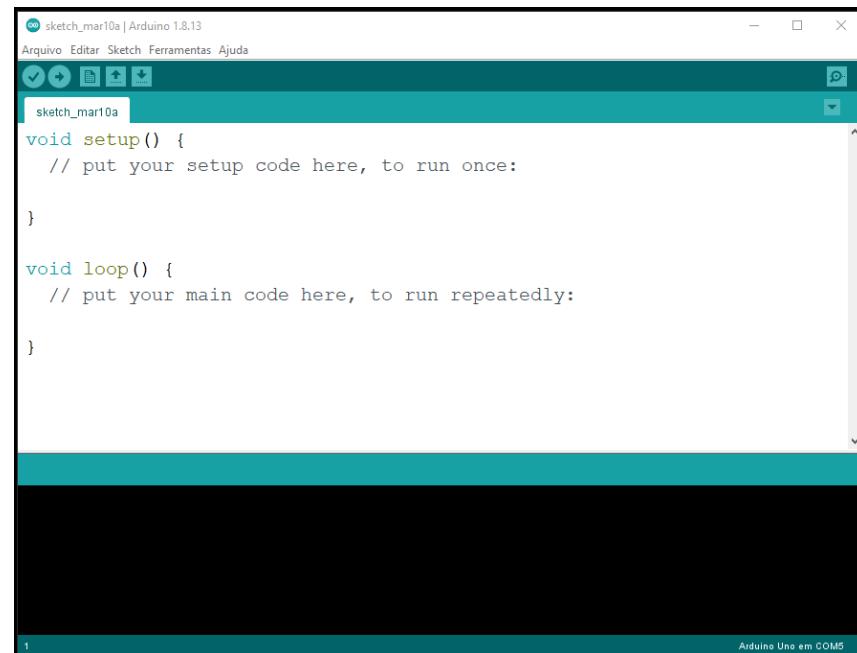
Configurando o ambiente de trabalho

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



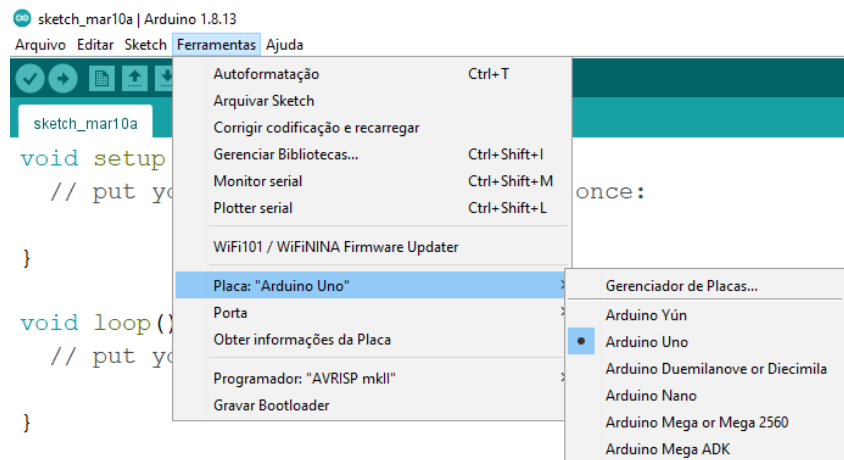
Configurando o ambiente de trabalho

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.



Configurando o ambiente de trabalho

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.



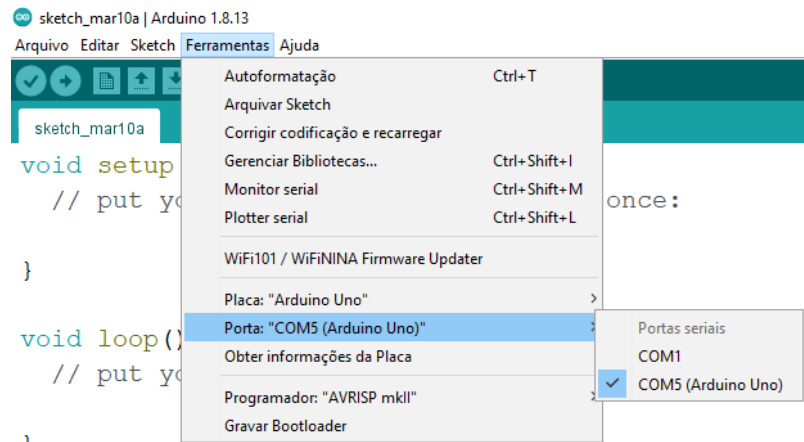


Configurando o ambiente de trabalho

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.

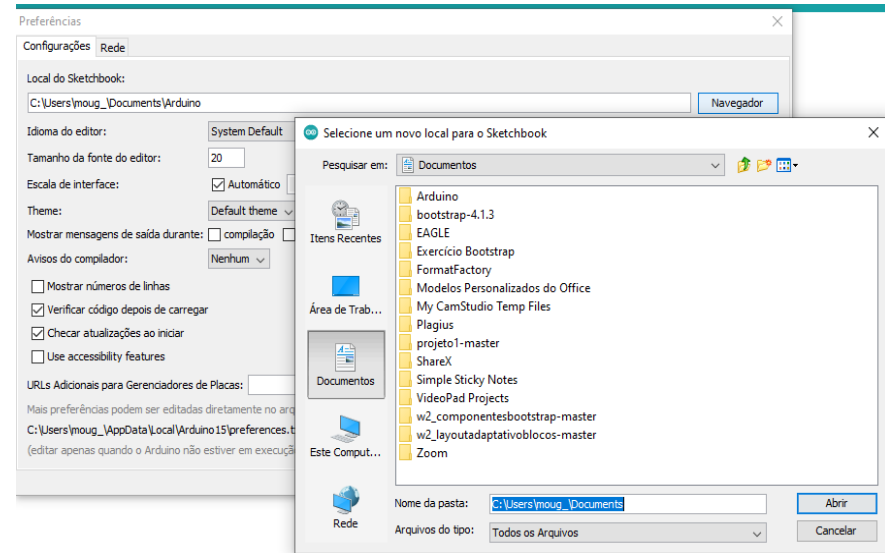
Configurando o ambiente de trabalho

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.



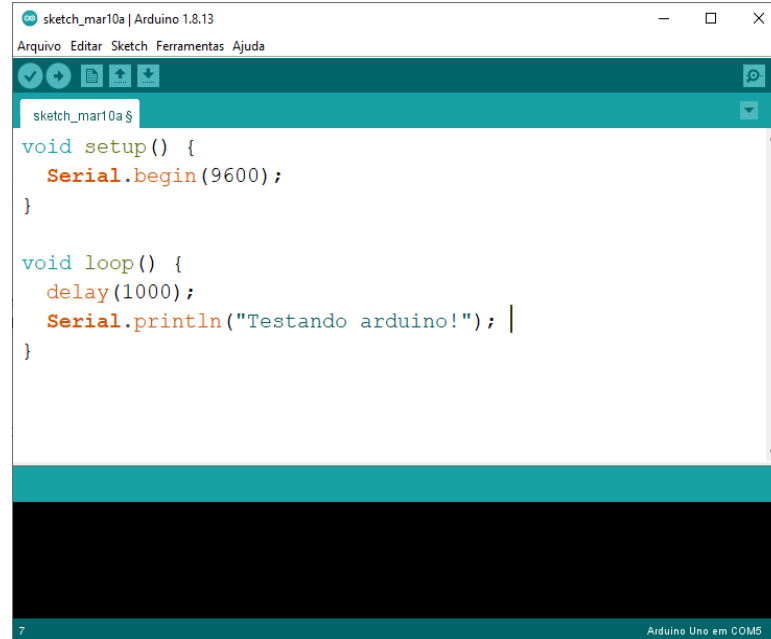
Configurando o ambiente de trabalho

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.



Configurando o ambiente de trabalho

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:

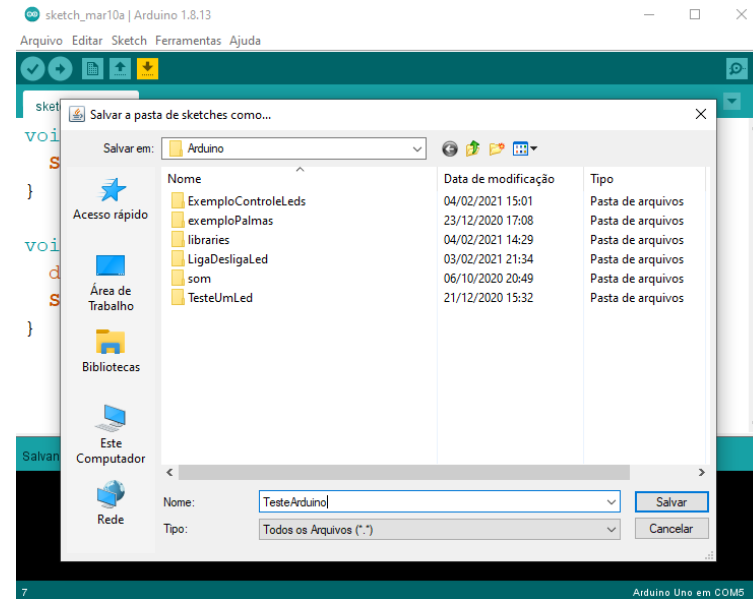
A screenshot of the Arduino IDE interface. The title bar reads 'sketch_mar10a | Arduino 1.8.13'. The menu bar includes 'Arquivo', 'Editar', 'Sketch', 'Ferramentas', and 'Ajuda'. Below the menu bar is a toolbar with icons for opening files, saving, compiling, and uploading. The main text area contains the following code:

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

The status bar at the bottom indicates '7' and 'Arduino Uno em COM5'.

Configurando o ambiente de trabalho

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).



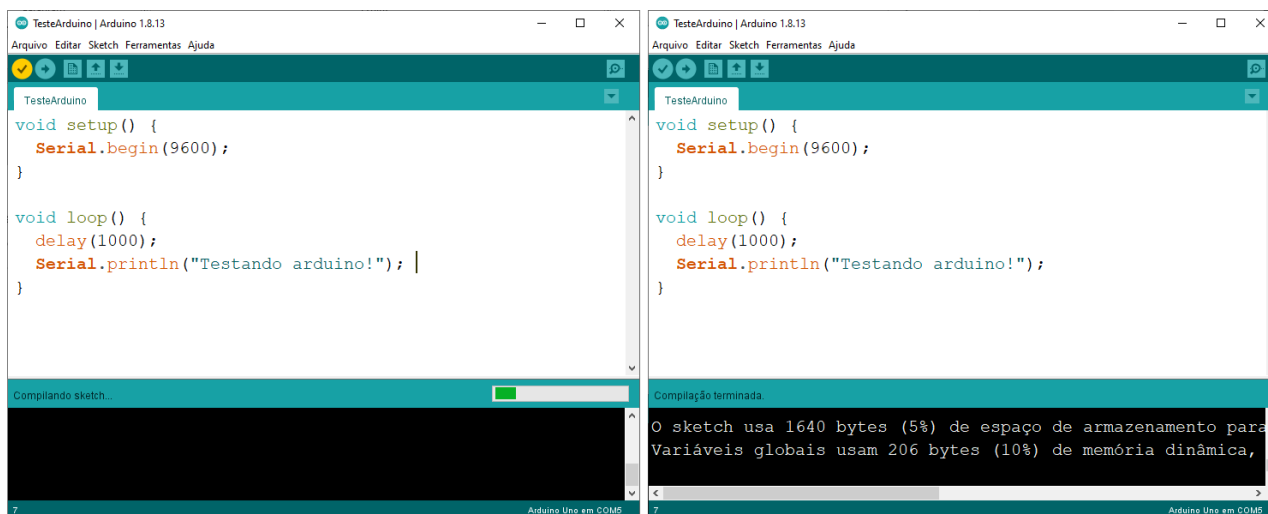


Configurando o ambiente de trabalho

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.

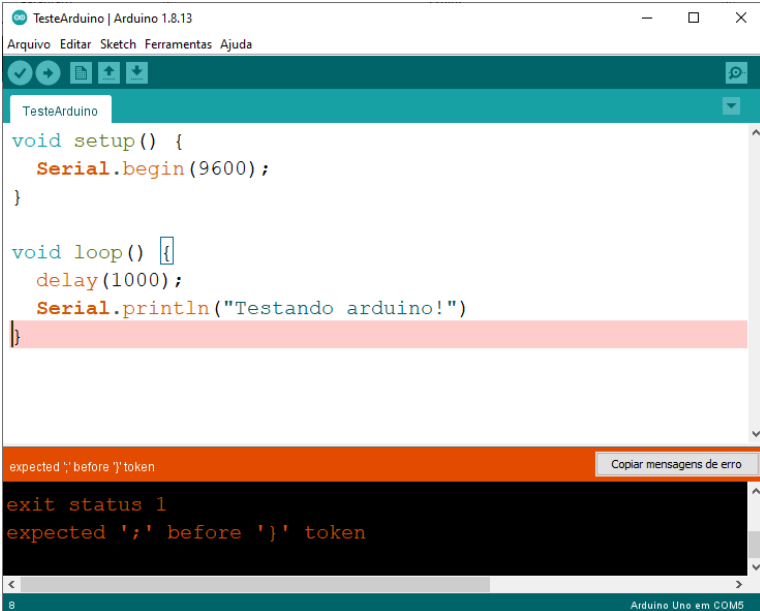
Configurando o ambiente de trabalho

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.



Configurando o ambiente de trabalho

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.

A screenshot of the Arduino IDE interface. The title bar reads "TesteArduino | Arduino 1.8.13". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar shows icons for opening files, saving, and running. The code editor displays the following code:

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

The line containing the `Serial.println` statement is highlighted in pink. Below the code editor, the status bar is orange, indicating an error. The error message area shows:

```
expected ';' before ')' token  
exit status 1  
expected ';' before ')' token
```

A button labeled "Copiar mensagens de erro" is visible next to the error message. The bottom status bar shows "8" and "Arduino Uno em COM5".

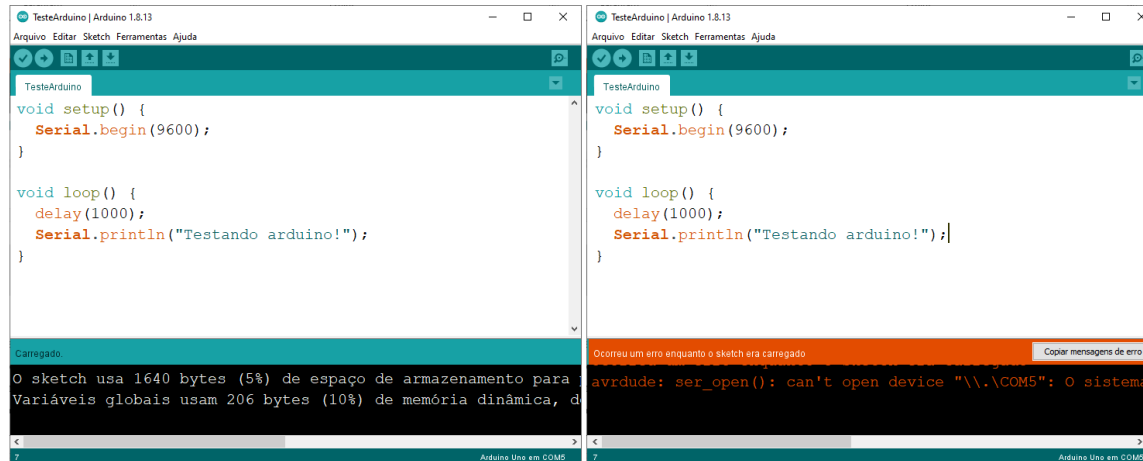


Configurando o ambiente de trabalho

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).

Configurando o ambiente de trabalho

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.





Configurando o ambiente de trabalho

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.



Configurando o ambiente de trabalho

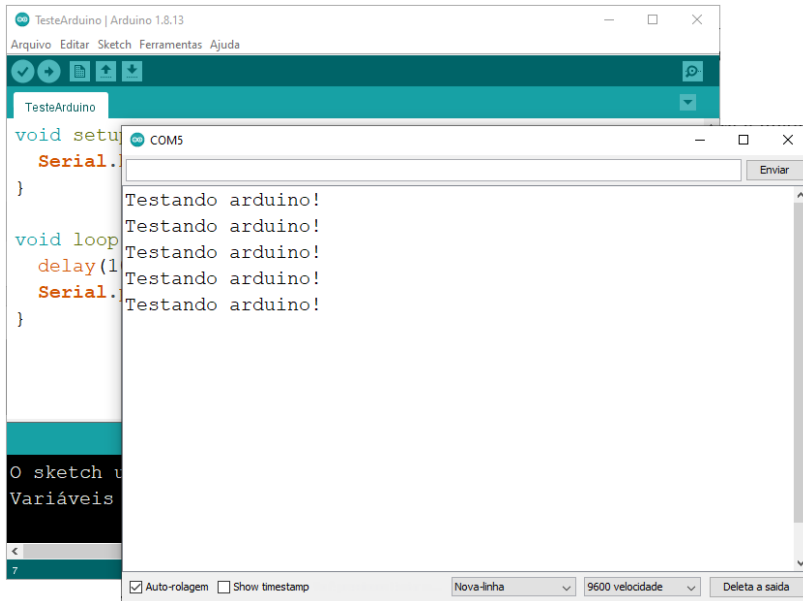
Vamos testar então se esta execução está ocorrendo com sucesso. O comando `Serial.println` utilizado no exemplo, faz com que o Arduino envie via Serial (para quem estiver conectado através da USB a ele, neste caso, nosso PC), a mensagem “Testando arduino”, em um loop que se repete a cada 1 segundo. A 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.

Configurando o ambiente de trabalho

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.

Configurando o ambiente de trabalho

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.



Configurando o ambiente de trabalho

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.

