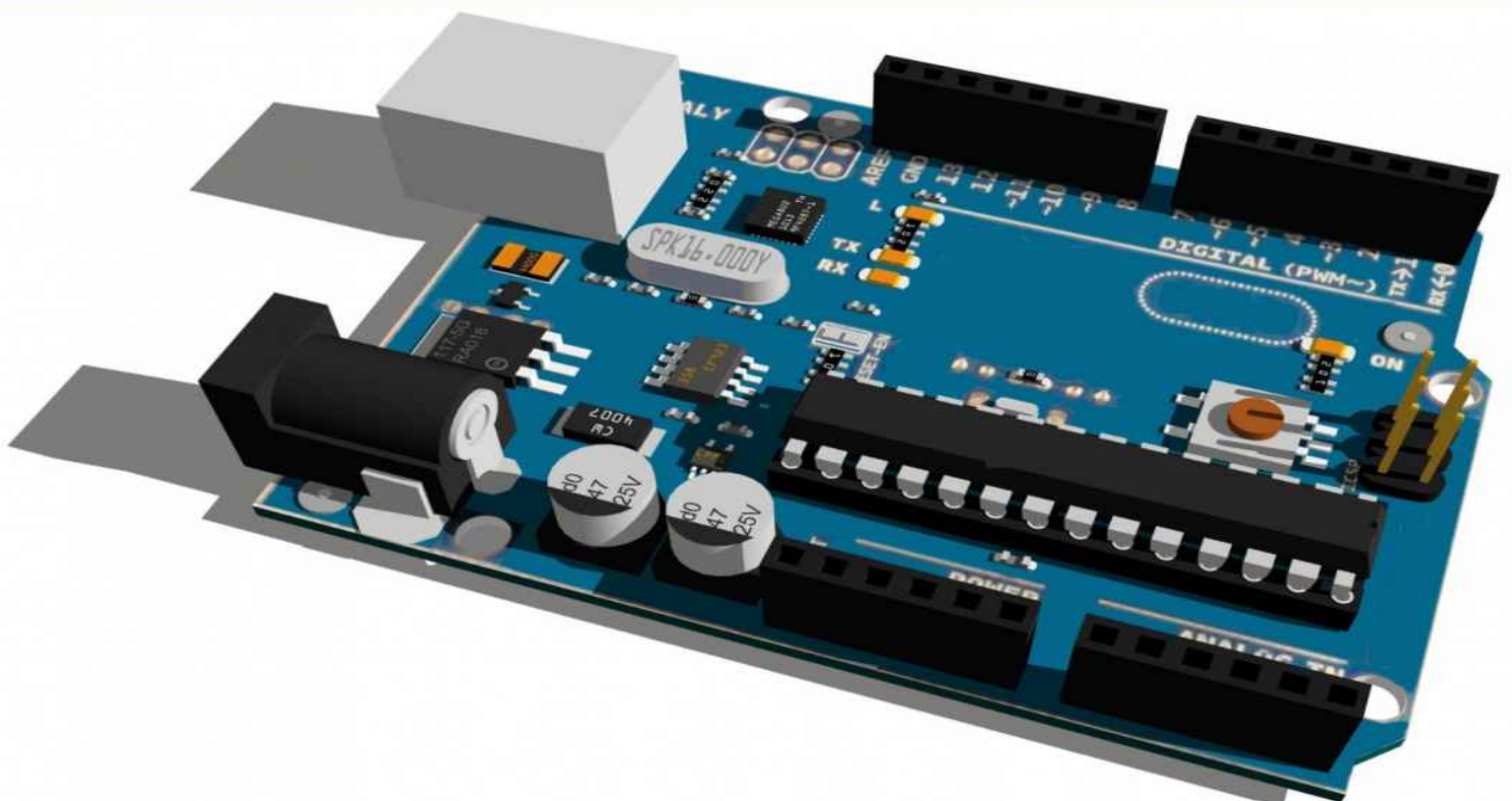


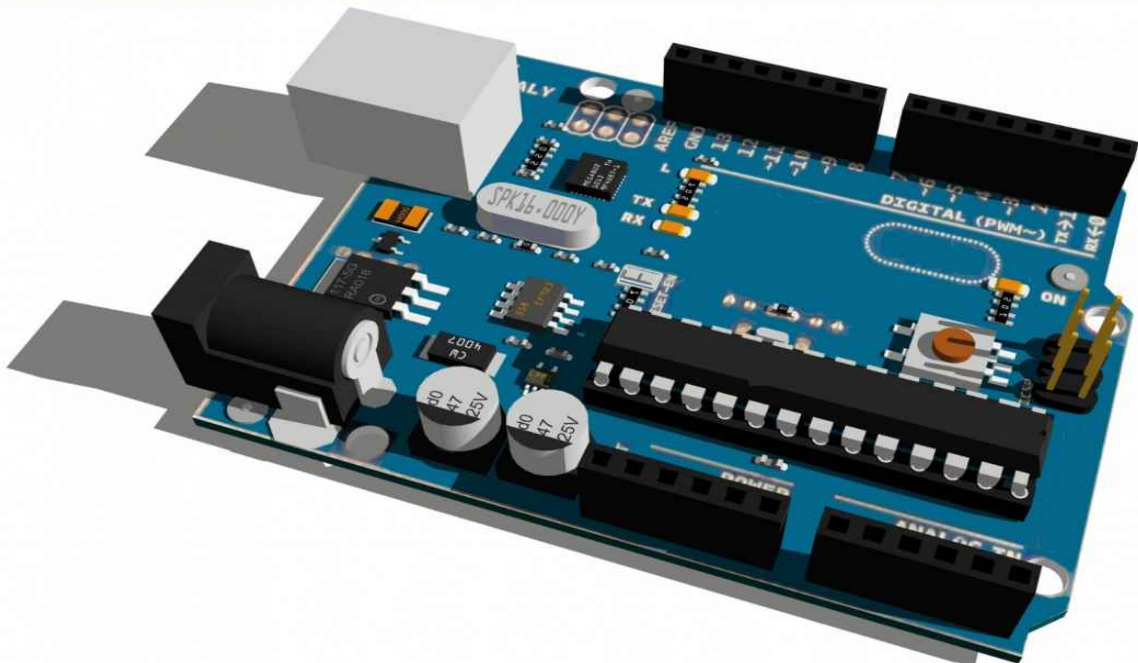
INTRODUÇÃO À PROGRAMAÇÃO C PARA ARDUINO

E À SIMULAÇÃO COM SIMULIDE
GIOVAN BARREIRA



INTRODUÇÃO À PROGRAMAÇÃO C PARA ARDUINO

E À SIMULAÇÃO COM SIMULIDE
GIOVAN BARREIRA



OceanofPDF.com

INTRODUÇÃO À PROGRAMAÇÃO C PARA ARDUINO

(E À SIMULAÇÃO COM SIMULIDE)

GIOVAN BARREIRA

<http://talkingphp.blogspot.com.br/>

<https://www.youtube.com/channel/UCrhBVHqavWMieE0wJETAUaA>

Copyright © 2018 Giovan Barreira . É proibida a reprodução, mesmo que parcial, por qualquer meio ou processo, sem a autorização por escrito do detentor dos direitos autorais ou do autor.

O autor, bem como todos os responsáveis pela publicação deste livro não se responsabilizam por danos, perdas ou acidentes de qualquer natureza que possam ocorrer ou ocorrerem referentes à manipulação dos programas, componentes e dispositivos necessários à prática das atividades e exercícios presentes neste livro.

Versão 2020.11.27

SOBRE O AUTOR

Giovan Barreira é escritor e professor de Segurança da Informação, Sistemas Operacionais Windows, Sistemas Operacionais Linux, Lógica de Programação e de programação nas linguagens ShellScript, VBA, PHP, C#, C e Java há mais de dez anos.

Graduado em Segurança da Informação, pós-graduado em Metodologia de Ensino, Giovan Barreira é fluente em Espanhol, Português, Inglês e gosta de tecnologia e de idiomas.

Quando não está trabalhando, o autor passa o tempo livre com a família, jogando Age of Empires® ou praticando Qi Gong.

Sumário

[SOBRE O AUTOR](#)

[LIVROS DO AUTOR](#)

[SOBRE AS VERSÕES ANTERIORES](#)

[INTRODUÇÃO](#)

[PRIMEIRA PARTE](#)

[ARDUINO](#)

[ONDE COMPRAR?](#)

[SIMULIDE](#)

[SEGUNDA PARTE](#)

[SOFTWARES NECESSÁRIOS](#)

[ARDUINO IDE](#)

[SIMULIDE](#)

[TERCEIRA PARTE](#)

[ELETRICIDADE BÁSICA](#)

[O ÁTOMO](#)

[TENSÃO ELÉTRICA](#)

[CORRENTE ELÉTRICA](#)

[RESISTÊNCIA ELÉTRICA](#)

[QUARTA PARTE](#)

[LINGUAGEM DE PROGRAMAÇÃO](#)

[UTILIZAÇÃO DO SIMULIDE](#)

[UTILIZAÇÃO DO ARDUINO IDE](#)

[INTRODUÇÃO À PROGRAMAÇÃO](#)

[SAÍDA \(OU ESCRITA\) EM PORTA DIGITAL](#)

[O BREADBOARD](#)

[MAIS PROGRAMAÇÃO](#)

[COMENTÁRIOS](#)

[VARIÁVEIS](#)

[OPERADORES ARITMÉTICOS](#)

[OPERADORES RELACIONAIS](#)

[OPERADORES LÓGICOS](#)

[ESTRUTURAS DE DECISÃO](#)
[ENTRADA \(OU LEITURA\) EM PORTA DIGITAL](#)
[ENTRADA \(OU LEITURA\) EM PORTA ANALÓGICA](#)
[O DISPLAY DE 7 SEGMENTOS](#)
[GERAR NÚMEROS \(PSEUDO\) ALEATÓRIOS](#)
[O DISPLAY DE LED 16x2](#)
[COMANDOS DO DISPLAY 16x2](#)
[ESCRITA E LEITURA SERIAL](#)
[CONVERSÃO DE TEXTO PARA NÚMERO INTEIRO](#)
[LEITURA COMPLEMENTAR](#)
[CRÉDITOS DAS IMAGENS](#)

LIVROS DO AUTOR

INTRODUÇÃO À PROGRAMAÇÃO C PARA ARDUINO E À
SIMULAÇÃO COM SIMULIDE

INTRODUÇÃO AO SQL COM MYSQL E MARIADB

VISUAL C#: INTRODUÇÃO À PROGRAMAÇÃO E AO PARADIGMA
“ORIENTAÇÃO A OBJETOS”

SOBRE AS VERSÕES ANTERIORES

2019.11.27

Livro completamente reescrito e revisado. Adicionado um capítulo sobre eletricidade e um sobre a cultura maker, adicionados mais de 2 novos projetos. Adicionados links para alguns vídeos do autor sobre Arduino ou SimuliDE.

2019.11.17

Adicionado o código da Atividade 02, adicionada a Figura 3B, adicionado link para vídeo introdutório do autor sobre o SimuliDE.

2019.2.21

Adicionada a seção ‘Livros do autor’ e rescrita a seção ‘Sobre o autor’.

2018.1.8

Adição das seções: ‘ESCRITA E LEITURA SERIAL’ e ‘CONVERSÃO DE TEXTO PARA NÚMERO INTEIRO’.

2017.12.1

Adição das seções: ‘O DISPLAY DE LED 16x2’ e ‘COMANDOS DO DISPLAY 16x2’.

2017.10.1

Adição das seções sobre operadores.

2017.8.2

Adição da seção ‘ONDE COMPRAR?’.

INTRODUÇÃO

Este livro é introdutório, escrito para leitores que nunca tiveram contato com ARDUINO ou que tiveram muito pouco. Eu também espero que você, meu caro leitor, esteja procurando aprendizagem e goste de ler e praticar.

Para que o leitor possa aproveitar totalmente o conteúdo deste livro é necessário saber como salvar arquivos e como compactar/descompactar arquivos.

Escrevi esta obra pensando na prática pedagógica, eu recomendo a leitura da obra em conjunto com a prática e na sequência. A linguagem aqui é simples e fácil de entender, adaptada para leigos e/ou adolescentes (boa parte dos meus alunos).

O livro está dividido em quatro partes, na primeira o leitor vai encontrar a explicação do que é um ARDUINO, onde comprá-lo, o que é um simulador (SIMULIDE) e algumas informações sobre a cultura MAKER; na segunda parte são apresentados os softwares necessários para o aprendizado bem como onde encontrá-los e como instalá-los; na terceira parte são apresentados conceitos básicos de Eletricidade e na quarta e última parte são apresentados os conceitos básicos de linguagem de programação, variáveis, laços de repetição, estruturas de decisão, escrita, leitura e a utilização dos componentes do kit básico: resistores, displays de 7 segmentos, leds e botões.

Todas as atividades deste livro (ou ao menos a esmagadora maioria) podem ser executadas apenas com o uso do SIMULIDE, sem a necessidade de aquisição de um kit ARDUINO para iniciantes, contudo a manipulação de uma placa UNO

física e de seus componentes torna a aprendizagem mais prazerosa (e desafiadora, eu comprei uma lupa logo depois que queimei alguns leds ;-)

Depois da leitura e prática das atividades desta obra, o leitor terá compreensão do que é um ARDUINO, do seu funcionamento e terá condições de programar códigos interessantes, mas o mais importante é que com esta compreensão dos fundamentos, o meu amigo leitor terá condições de aprofundar seus conhecimentos via vídeos ou pela leitura de material avançado (recomendo alguns no final deste livro).

Boa prática!

PRIMEIRA PARTE

O que é um ARDUINO?

Onde comprar?

O que é o SIMULIDE?

ARDUINO

Na verdade, a palavra ARDUINO pode se referir a uma empresa, que desenvolve hardware e software na forma de kits para utilização em ambiente educacional ou ainda como hobby ou na produção e realização de projetos eletrônicos que exijam baixo poder de processamento.

HORA DA PRÁTICA:

Eu convido o leitor a acessar este site:

<https://www.arduino.cc/>

ARDUINO também pode se referir a um compilador (compilador é um programa muito especial pois é utilizado para fazer programas que vão rodar em um computador, smartphone ou no nosso caso, em um ARDUINO UNO). Abaixo temos a imagem de um compilador ARDUINO, exibindo parte do código de um programa chamado semáforo.

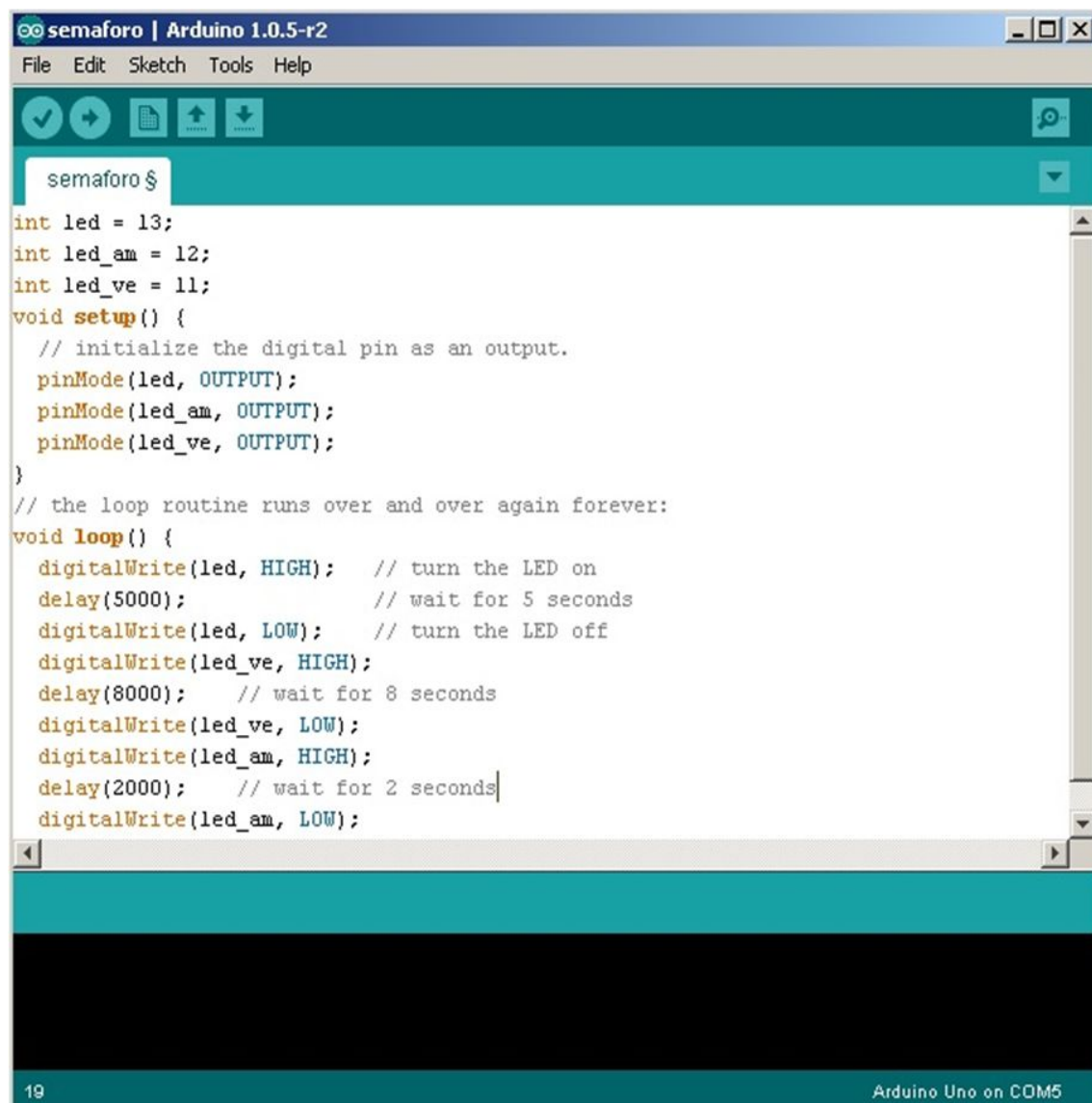


Figura 1 - O software ARDUINO IDE mostrando parte de um código para o funcionamento de um semáforo.

E por último, mas não menos importante, a palavra ARDUINO pode significar uma família de placas de circuito de I/O (entrada e saída, por exemplo: entra um sinal de luz e sai o comando para ligar uma bomba d'água para regar uma horta), na verdade eu acredito que você leitor está mais interessado neste aspecto. Abaixo tem-se a imagem de uma placa modelo UNO.

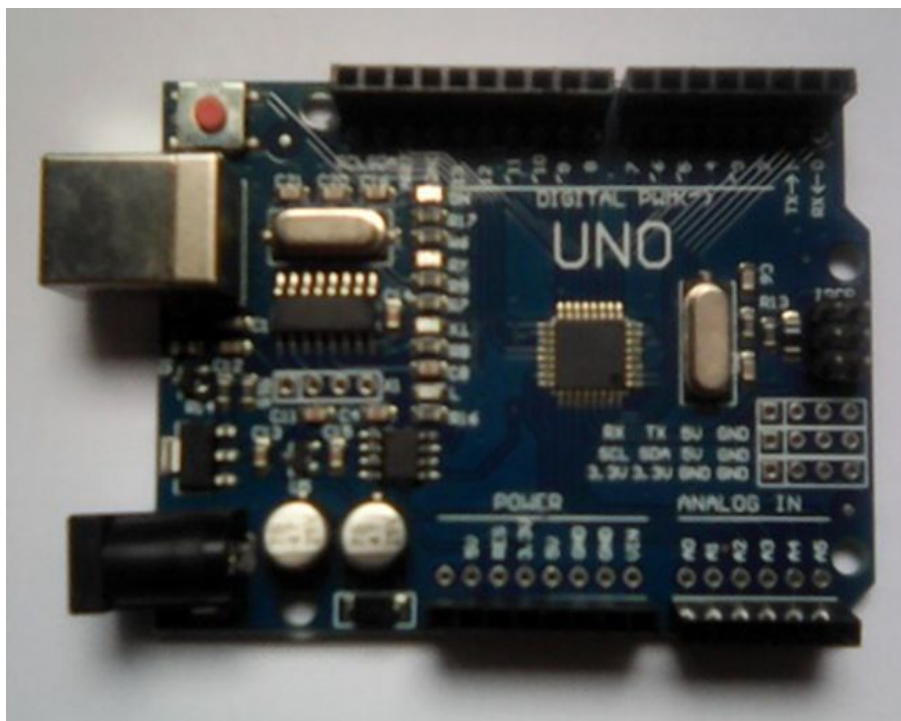


Figura 2 - Uma placa família UNO.

Uma placa ARDUINO UNO (ou uma placa genérica UNO, como a da imagem acima) possui razoável poder de processamento e é excelente para aprendizagem, além de ser muito barata por ser um modelo para iniciantes. A primeira placa surgiu na Itália, em 2005 com a intenção de ser utilizada em projetos escolares.

Atualmente existem vários modelos de placas, alguns modelos desenvolvidos com vistas ao tamanho reduzido, como o ARDUINO NANO ou ainda o MICRO, outros modelos foram desenvolvidos para maiores necessidades de processamento, como o ARDUINO MEGA ou ainda o M0 PRO e ainda existem as placas ‘vestíveis’ como o ARDUINO GEMMA e o LILYPAD.

HORA DA PRÁTICA:

Eu convido o leitor a acessar este site para saber mais sobre as placas ARDUINO:

<https://pt.wikipedia.org/wiki/Arduino>

A empresa ARDUINO produz uma grande variedade de modelos de placas e uma grande variedade de sensores e de kits, além dos genéricos que podem ser

encontrados na internet.

HORA DA PRÁTICA:

Eu convido o leitor a acessar este site e navegar pelos modelos de placas disponíveis:

<https://www.arduino.cc/en/Main/Products>

Enfim o que uma placa UNO faz? De maneira geral ela recebe a leitura de um sensor e reage de acordo com a programação que ela recebeu. Exemplo: uma placa ligada em um sensor de umidade e em uma bomba d'água pode mandar um sinal para ligar a bomba se detectar baixa umidade e depois mandar um sinal para desligar a bomba quando detectar alta umidade.

Então uma placa pode ter sua utilização resumida em entrada, reação e saída. Hoje em dia já existem vários projetos interessantes de dispositivos dos mais variados tipos utilizando ARDUINO, já existem organizações que sustentam até campeonatos de desenvolvimento de projetos por crianças.

HORA DA PRÁTICA:

Eu convido o leitor a acessar os links oficiais sobre projetos no site da ARDUINO:

<http://playground.arduino.cc/Projects/Ideas>

<https://www.arduino.cc/en/Main/Education>

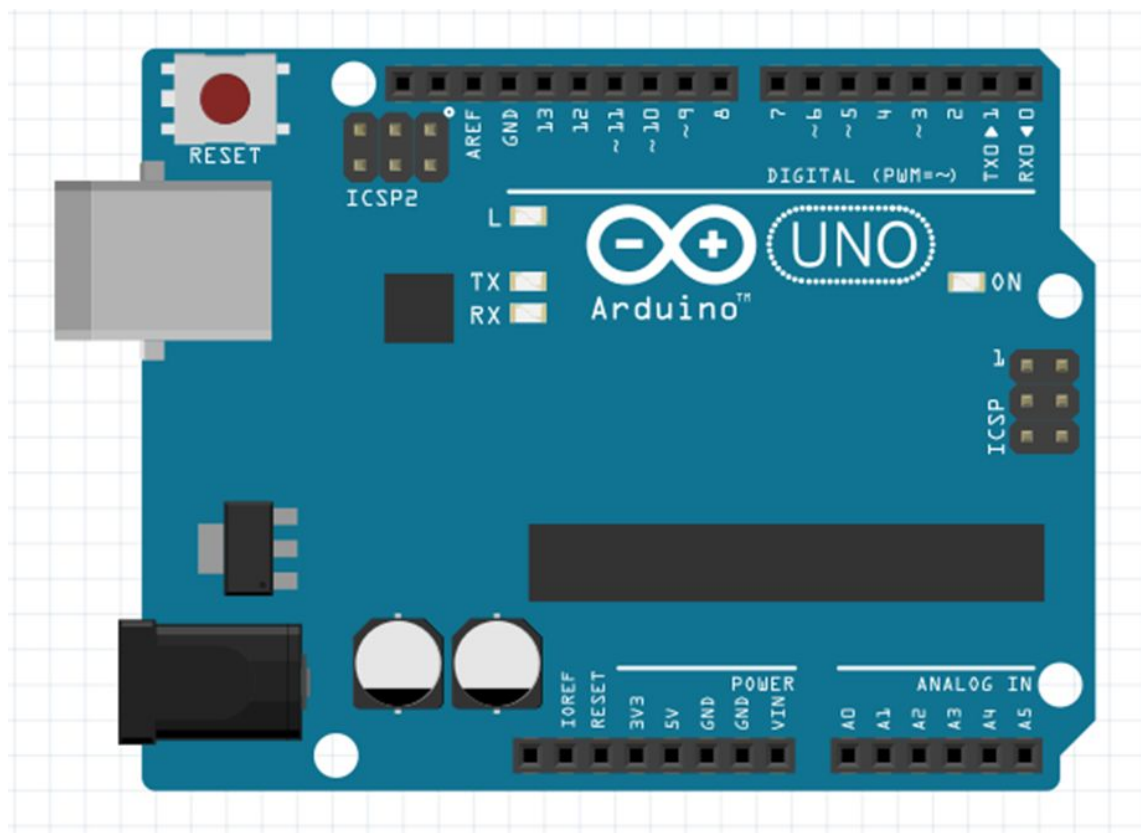


Figura 3 - Placa ARDUINO UNO

Na parte superior da placa podemos ver as portas digitais de 0 a 13. Mas as portas 0 e 1 são específicas para receber (RX) comunicação e transmitir (TX) comunicação.

Do lado esquerdo da porta 13 temos a porta GND, que significa aterramento (ou 0V).

Na parte de baixo, do lado direito temos as portas analógicas de A0 até A5. Do lado esquerdo temos duas portas GND (aterramento como citado anteriormente), uma porta que fornece 5V e uma porta para RESET da placa. A explicação das outras portas foge do escopo deste livro por terem aspectos intermediários e avançados de Eletrônica.

Ao longo deste livro utilizaremos as portas digitais de 2 a 13, além das analógicas, das GND e as portas RESET e 5V.

ONDE COMPRAR?

O leitor poderá comprar um kit ou ainda apenas a placa. Os kits para iniciantes vêm geralmente com uma placa UNO, um breadboard (uma placa para encaixar dispositivos), alguns resistores, alguns cabos e alguns leds de cores diversas. Ao longo deste livro utilizaremos o conteúdo do kit de iniciantes, um display de sete segmentos e um display 16x2.

Mas conforme o leitor pegar prática, poderá utilizar sensores de umidade, de som, saídas de som, motores de passo, sensores de pressão, de luminosidade, controle remoto, dentre outros.

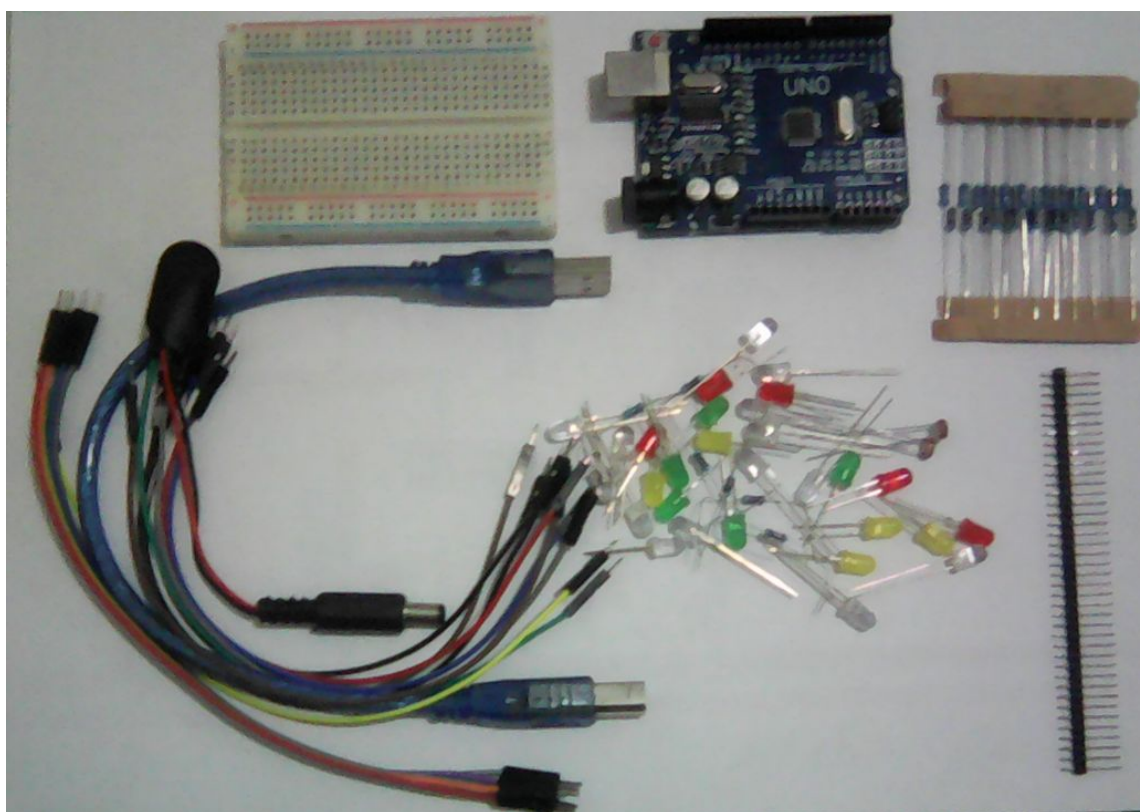


Figura 4 - Kit iniciante

Um kit pode ser comprado praticamente em qualquer loja física de componentes eletrônicos ou ainda na internet, mas se for comprar na internet, eu recomendo cautela, pois o kit comprado em uma loja virtual brasileira de grande porte tem o preço razoável e um prazo de entrega rápido (entre 2 e 7 dias) enquanto o kit comprado em uma loja virtual de importados muitas vezes tem preço muito baixo, mas prazos de entrega de até 30 dias.

HORA DA PRÁTICA:

Eu convido o leitor a acessar este site e ver os preços e kits disponíveis para ter uma noção de preços, então poderá comparar com os preços em uma loja física:

<https://www.americanas.com.br/busca?conteudo=arduino>

SIMULIDE

É um programa que simula o funcionamento de vários componentes eletrônicos e dentre estes componentes o arduino. O simulide é um programa que pode ser baixado e utilizado gratuitamente.

Na falta de uma placa UNO, podemos utilizar o SIMULIDE para executar a maioria dos códigos desenvolvidos ao longo deste livro.

Nos primeiros projetos mostrarei a execução do código no SIMULIDE e na placa UNO.

HORA DA PRÁTICA:

Eu convido o leitor a acessar o site do SIMULIDE:

<https://simulide.blogspot.com/>

Eu convido o leitor a assistir meu vídeo de introdução ao SIMULIDE.

https://www.youtube.com/watch?v=R_28BLSyfRA

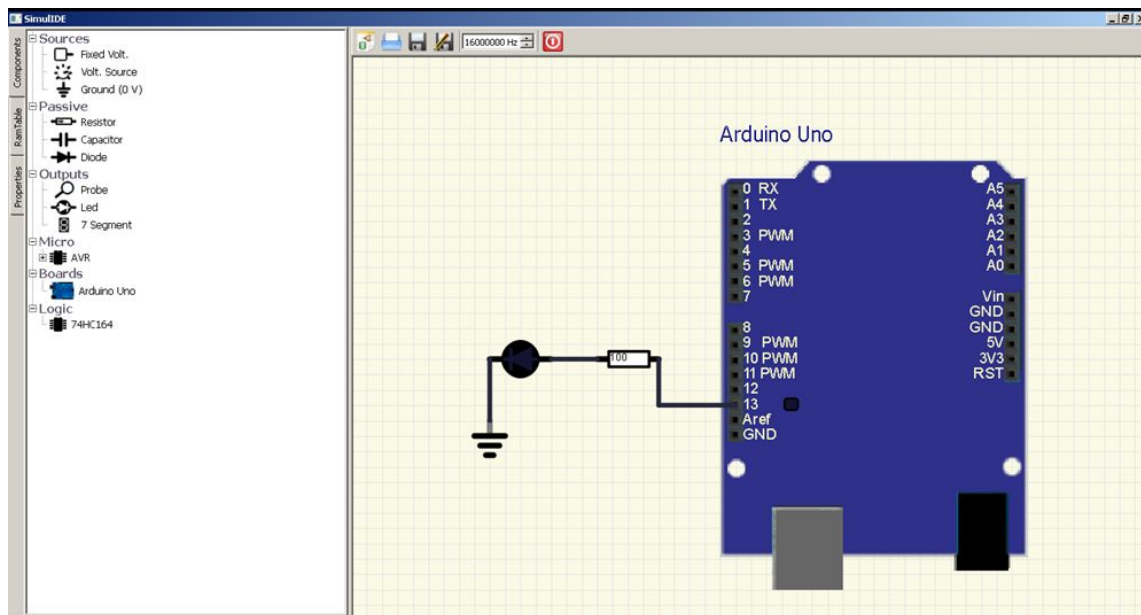


Figura 5 - SIMULIDE com uma placa ARDUINO UNO

CULTURA MAKER E STEAM

A cultura maker é moderna e é parte do movimento DIY (D.I.Y. - do it yourself ou faça você mesmo). Estes movimentos são sociais e voltados para o aprendizado humanizado além pregarem de maneira geral a sustentabilidade, trabalho em equipe, colaboração e o aprendizado ativo (estilo de aprendizado onde a pessoa aprende por conta própria lendo livros, artigos e vendo vídeos, no lugar de depender de um professor ou escola).

O universo maker atual tem relação direta com o aprendizado STEAM (S.T.E.A.M – Science, Technology, Engineering, Arts and Mathematics) que é moderno e afirma que Ciências, Tecnologia, Engenharia, Artes e Matemática devem ser aprendidos em conjunto e todas estas ciências devem se relacionar umas com as outras.

O Arduino faz parte do mundo maker pois é utilizado amplamente na construção de robôs e dispositivos domésticos e atualmente conta com grandes eventos no Brasil e no mundo, tais como OBR (Olimpíada Brasileira de Robótica), Arduino Day, Campus Party, além de vários órgãos que promovem esta cultura como o ITS Brasil e o Fab Lab Livre SP.

HORA DA PRÁTICA:

Eu convido o leitor a acessar o site oficial do Arduino Day:

<https://day.arduino.cc/>

Eu convido o leitor a acessar o site da Campus Party:

<https://brasil.campus-party.org/>

Eu convido o leitor a acessar o site da ITS Brasil:

<http://itsbrasil.org.br/>

Eu convido o leitor a acessar o site da Fab Lab Livre:

<https://fablablivresp.art.br/>

SEGUNDA PARTE

Quais softwares são necessários para o aprendizado?

Onde encontrá-los?

Como instalá-los?

SOFTWARES NECESSÁRIOS

Ao longo deste livro precisaremos do software ARDUINO IDE ([aqui](#)) e do SIMULIDE ([aqui](#)).

ARDUINO IDE

Após acessar (<https://www.arduino.cc/en/Main/Software>) e baixar o software (chamado ARDUINO IDE) será necessário instalá-lo porque para que o computador reconheça a placa é necessário um software especial chamado DRIVER que vem junto com o ARDUINO IDE.

Nota importante: se a sua placa UNO for genérica, como a da figura 2, será necessário instalar um driver chamado ch340.

Abaixo mostro o download e a instalação do ARDUINO IDE e do driver ch340.

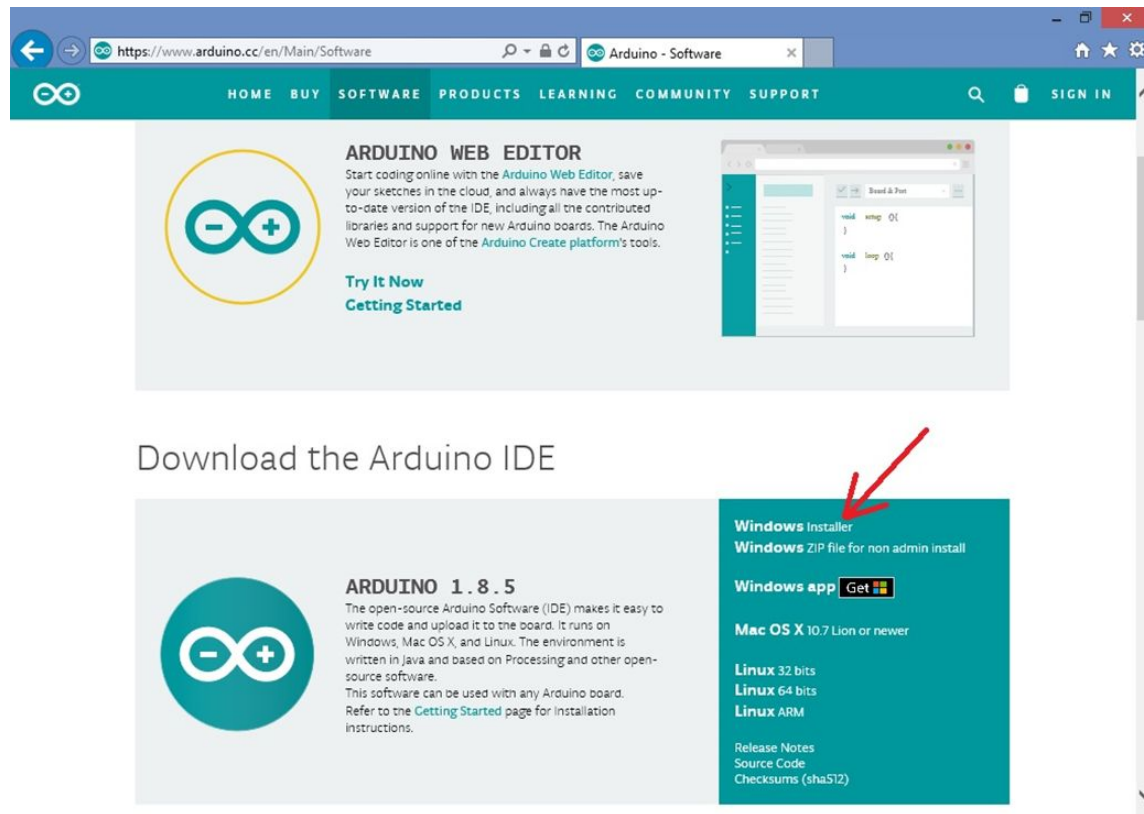


Figura 6 - <https://www.arduino.cc/en/Main/Software>

Depois de terminar o download, dar duplo clique esquerdo no arquivo.

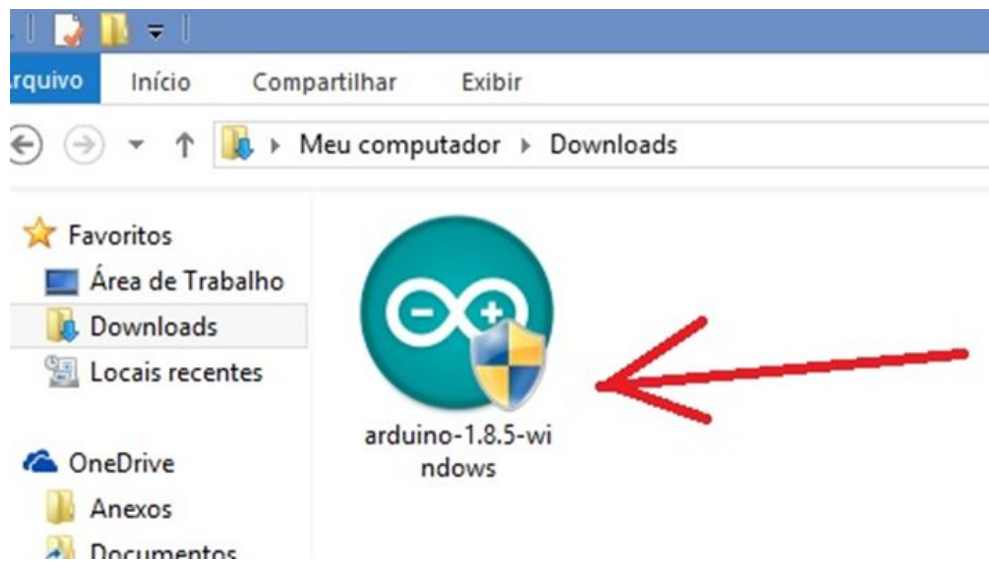


Figura 7 - Instalação do ARDUINO IDE

Iniciará a instalação do ARDUINO IDE então dar um clique esquerdo no botão “I Agree”.

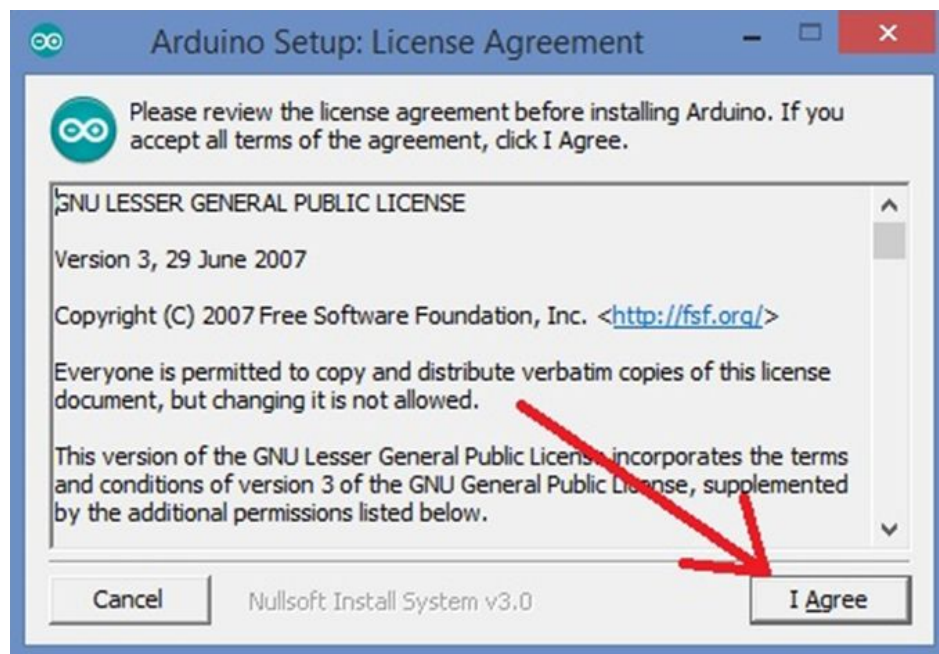


Figura 8 - Instalação do ARDUINO IDE

Depois, vai aparecer uma janela com as opções de instalação, deixar todas marcadas. Então, dar um clique esquerdo no botão “Next”.

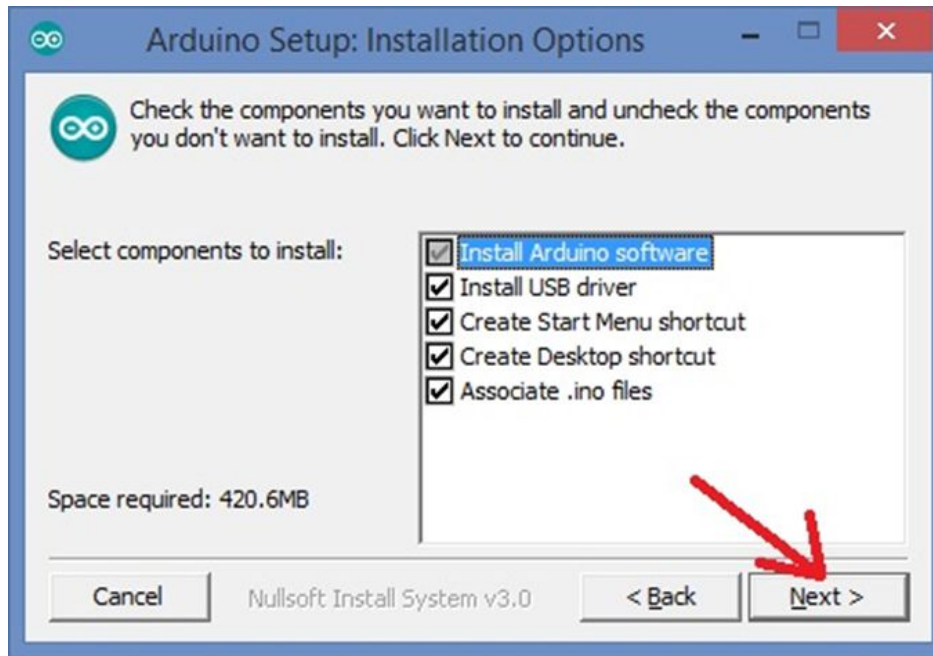


Figura 9 - Instalação do ARDUINO IDE

Depois, vai aparecer uma janela com a possibilidade de escolha da pasta de instalação, deixar como está. Então, dar um clique esquerdo no botão “Install”.

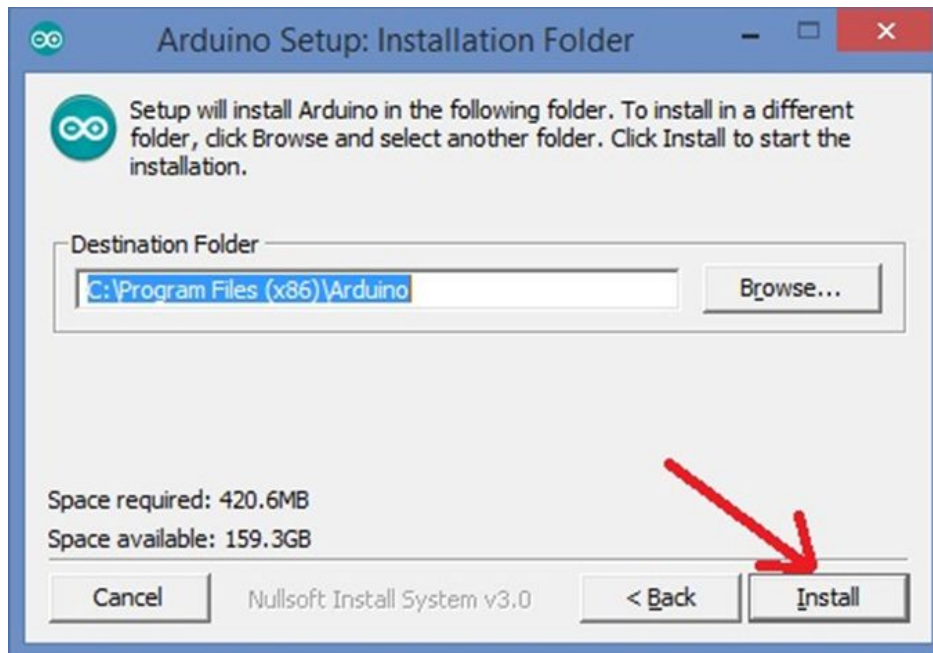


Figura 10 - Instalação do ARDUINO IDE

A instalação está ocorrendo.

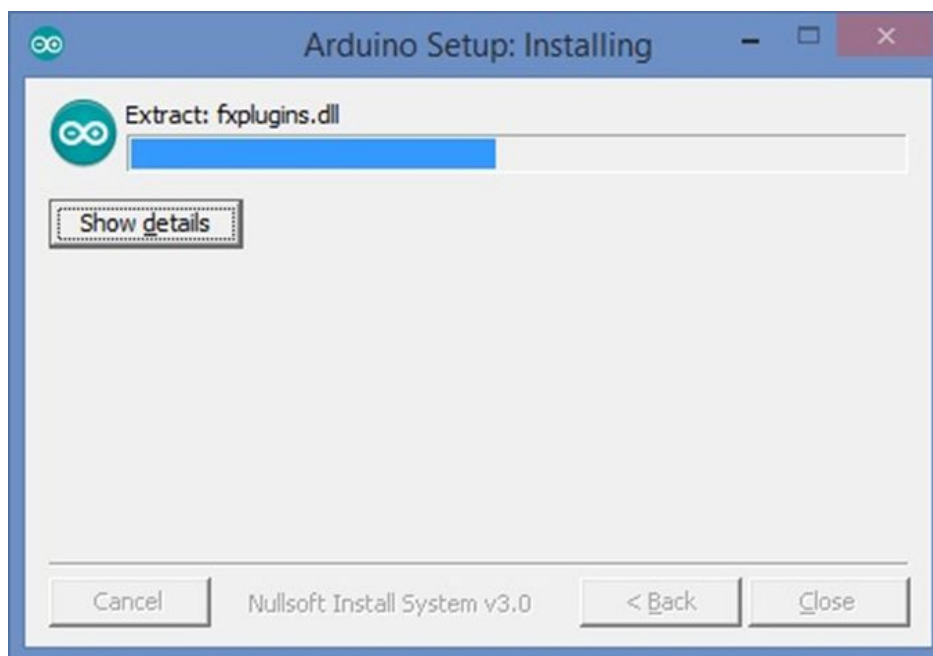


Figura 11 - Instalação do ARDUINO IDE

Depois, vai aparecer uma janela perguntando se deseja instalar um dispositivo. Então, dar um clique esquerdo no botão “Instalar”.

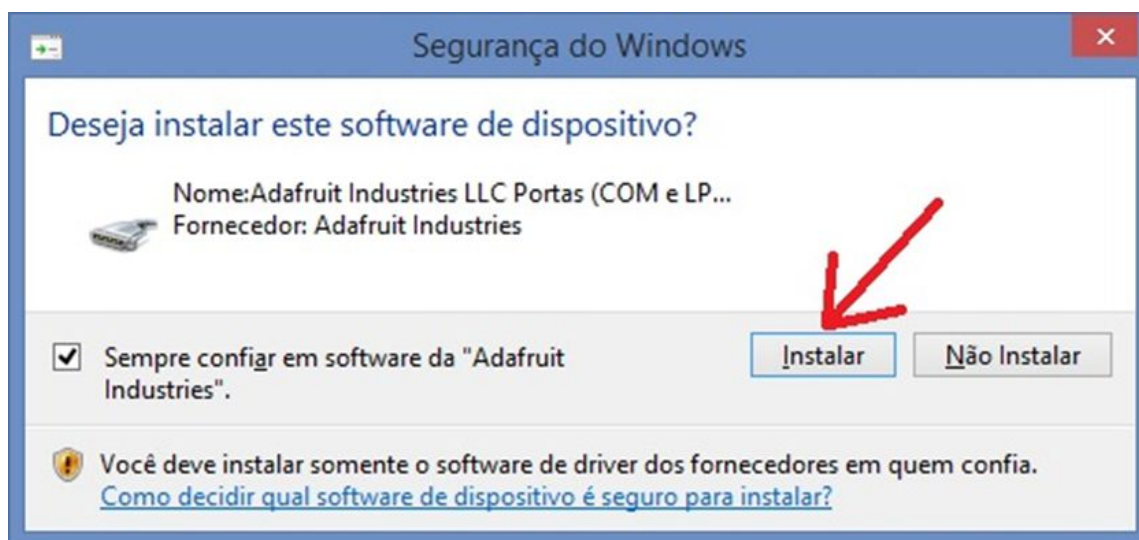


Figura 12 - Instalação do ARDUINO IDE

Depois, vai aparecer uma nova janela perguntando se deseja instalar um dispositivo. Então, dar um clique esquerdo no botão “Instalar” novamente.



Figura 13 - Instalação do ARDUINO IDE

Depois, vai aparecer uma nova janela perguntando se deseja instalar um dispositivo. Então, dar um clique esquerdo no botão “Instalar” novamente.

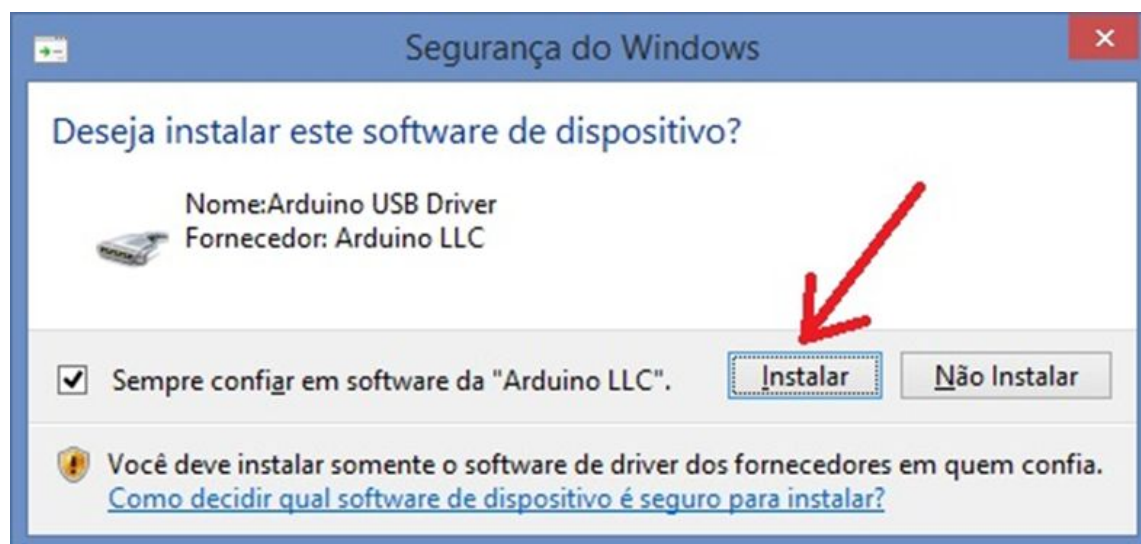


Figura 14 - Instalação do ARDUINO IDE

E para encerrar a instalação, dar um clique esquerdo no botão “Close”.

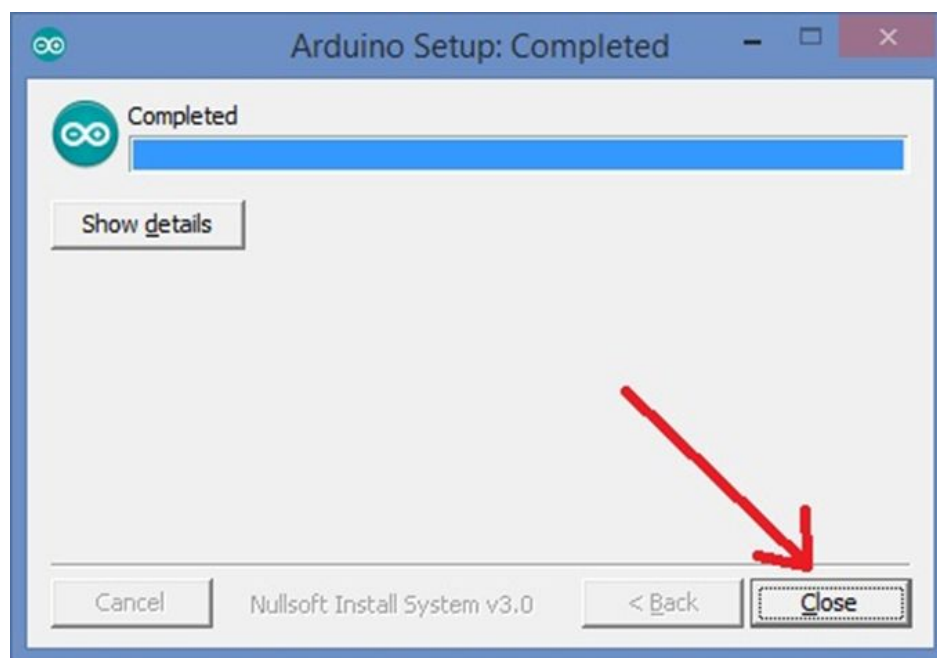


Figura 15 - Instalação do ARDUINO IDE

Para que o Windows encontre a placa UNO, caso esta não tenha sido fabricada pela ARDUINO, será necessário a instalação de um driver (programa especial que controla dispositivos) chamado ch340.

O driver ch340 pode ser baixado [aqui](https://1drv.ms/f/s!AsHT9Q2dElFsiBZFbEkUo701j8rS) (https://1drv.ms/f/s!AsHT9Q2dElFsiBZFbEkUo701j8rS) ou o leitor pode fazer uma busca na internet pesquisando “ch340 arduino windows 8” (ou qualquer outra versão do Windows).

Então o leitor deverá descompactar o arquivo e dentro da pasta ‘CH341SER’ dar duplo clique em SETUP.

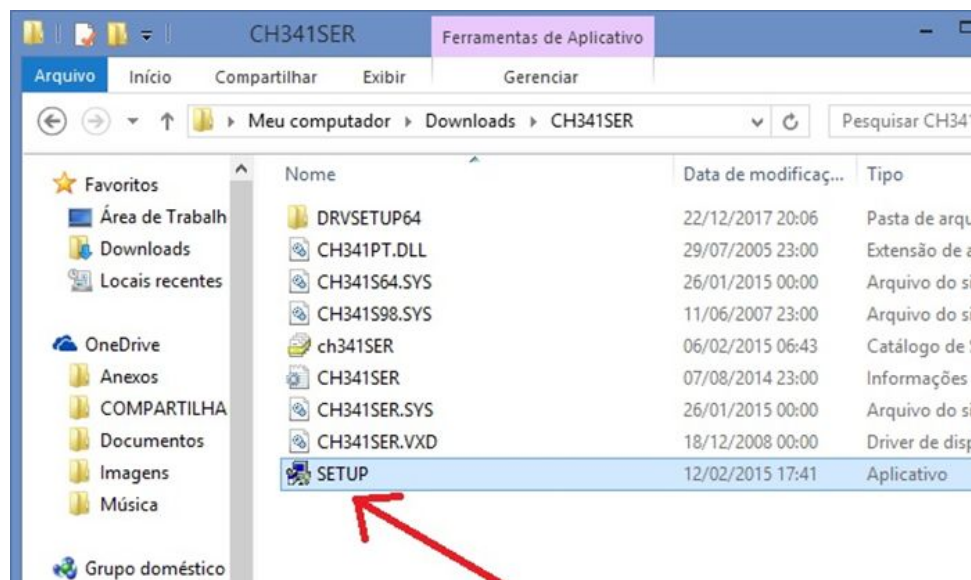


Figura 16 - Instalação do driver genérico

Quando aparecer a janela de instalação do driver, dar um clique esquerdo no botão “INSTALL”.

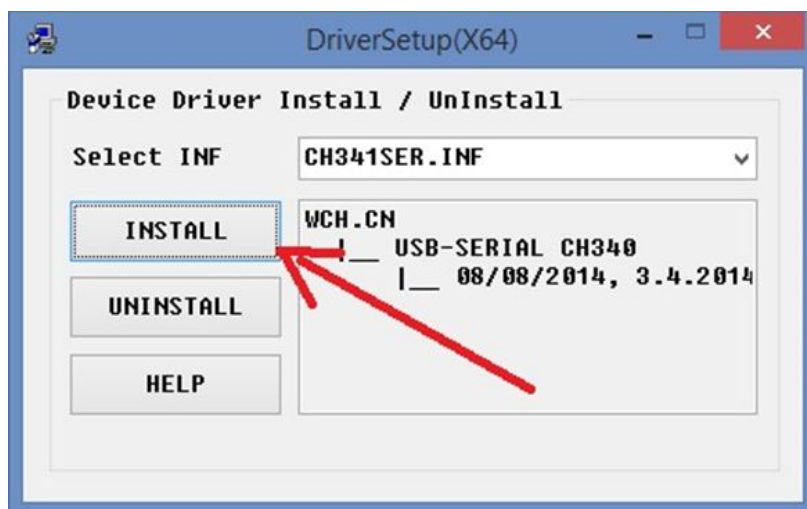


Figura 17 - Instalação do driver genérico

E para encerrar a instalação, dar um clique esquerdo no botão “OK”.

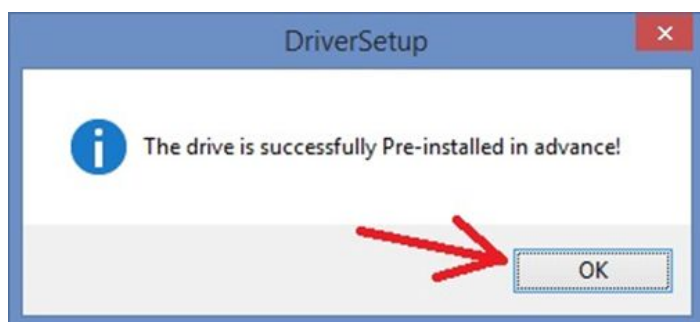


Figura 18 - Instalação do driver genérico

Caso continue aberta a janela de instalação, dar um clique esquerdo no botão “fechar”.



Figura 19 - Instalação do driver genérico

SIMULIDE

Para executar o software SIMULIDE, depois de baixado e descompactado, entrar na pasta “SimulIDE_0.1.5” e dar duplo clique no arquivo “run_simulide”.

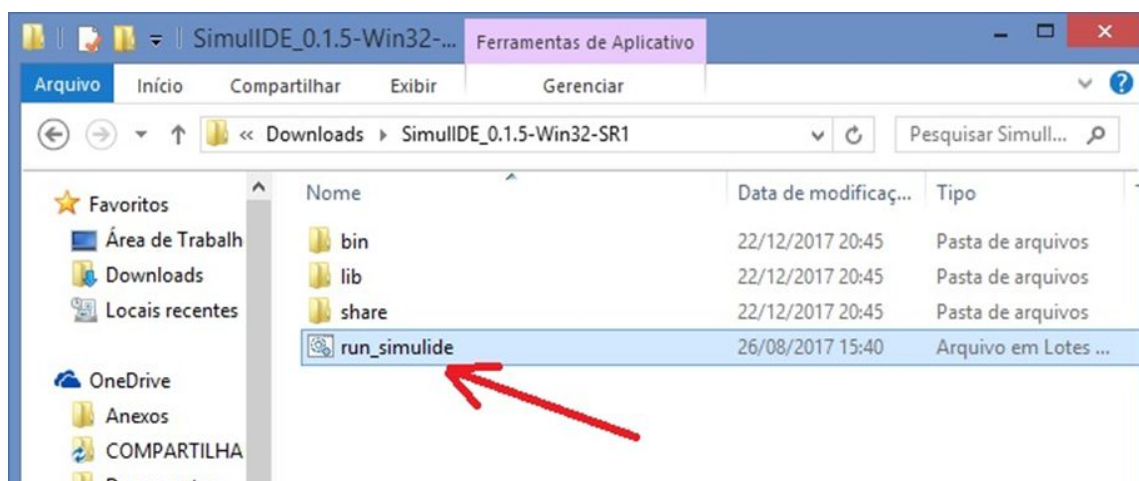


Figura 20 - Executar o SIMULIDE

Parabéns, agora o meu amigo leitor possui todo o ambiente necessário para começar a programar e executar os programas na placa UNO e no SIMULIDE.

TERCEIRA PARTE

Conceitos de Eletricidade.

Tensão elétrica.

Corrente elétrica.

Resistência elétrica.

Potência elétrica.

Características de alguns componentes do kit básico: resistores e leds.

ELETRICIDADE BÁSICA

Podemos afirmar que o mundo em que vivemos é elétrico, cada vez mais estamos dependentes da energia elétrica para alimentar nossos Arduinos, videogames, smartphones, smart tvs, lâmpadas, fornos, bicicletas, carros, etc.

Mas como funciona a eletricidade? Como funciona sua conta de energia elétrica?

O ÁTOMO

Tudo começa no átomo, ele compõe tudo o que conhecemos. O átomo é composto de três peças básicas: nêutrons, prótons e elétrons. O núcleo do átomo é composto de nêutrons e prótons enquanto os elétrons orbitam em volta do núcleo.

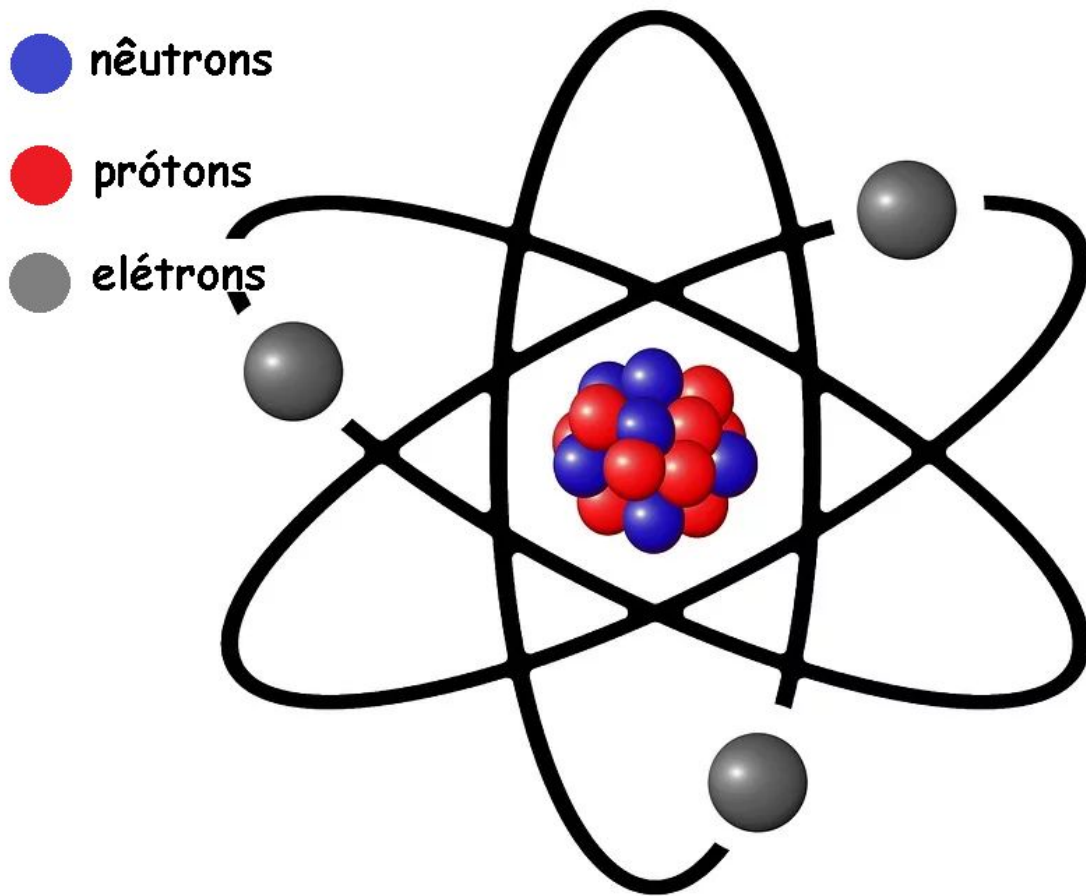


Figura 21 - O átomo com núcleo (azul e vermelho) e eletrosfera (cinza)

Em muitos materiais metálicos como ouro, prata, cobre, ferro, etc. os elétrons estão ‘soltos’ fazendo o material ser um bom condutor de eletricidade. Quando um átomo tem a mesma quantidade de elétrons e prótons dizemos que ele tem carga nula (zerada), quando ele tem diferentes quantidades de elétrons e de prótons dizemos que ele está eletrizado (possui carga diferente de zero).



Figura 22 - O escorregador doa elétrons para o corpo do garoto, a carga elétrica que o garoto acumula faz seu cabelo arrepiar

TENSÃO ELÉTRICA

Imagine um arqueiro com seu arco e flecha, quanto mais ele tensiona o arco mais rápido e forte será o disparo da flecha, o mesmo ocorre com a eletricidade. Quando dois materiais (ou objetos) têm cargas elétricas diferentes, se estes objetos entrarem em contato, haverá movimentação de elétrons de um objeto para outro (um bom exemplo é o garotinho da imagem acima, que ficou com o corpo eletrizado devido ao material do escorregador).

Quando dois pontos têm diferentes cargas elétricas dizemos que há uma diferença de potencial (ddp) entre eles, a ddp também é chamada de tensão elétrica porque faz elétrons se movimentarem de um ponto a outro, da mesma forma que a tensão do arco faz a flecha se movimentar da mão do arqueiro para o alvo.

A tensão elétrica é medida em Volts (V), usualmente o Arduino tem portas que fornecem 5 V ou ainda 3.3 V. Pilhas de controle remoto de smart tv possuem

1.5 V quando estão novas e um valor entre 1.5 V e 0 V quando estão velhas.

A tensão elétrica em uma tomada doméstica é de 127 V ou 220 V e a tensão nas velas do motor de um carro de passeio são de 14000 V ou mais (estas tensões elétricas tanto domésticas quanto nas velas de um veículo são perigosíssimas e podem resultar em morte acidental).

CORRENTE ELÉTRICA

Quando dois pontos que possuem tensão elétrica diferente de zero estão ligados, ocorre a passagem de elétrons de um ponto a outro. Tal passagem de elétrons é chamada de corrente elétrica. A corrente elétrica é medida em Ampére (A).

RESISTÊNCIA ELÉTRICA

A passagem de elétrons entre dois pontos faz com que dispositivos elétricos ou eletrônicos funcionem. Exemplo quando ligamos um led nas portas GND e 5 V (você aprenderá sobre estas portas mais adiante) no Arduino, o led acende.

Porém os elétrons precisam passar de maneira controlada pois se eles passarem de maneira livre entre dois pontos de um dispositivo eletrônico, o dispositivo será danificado. Quem controla a passagem de elétrons é o resistor, que se aquece com a passagem controlada de elétrons. A unidade de medida da corrente elétrica é o Ohm (Ω).

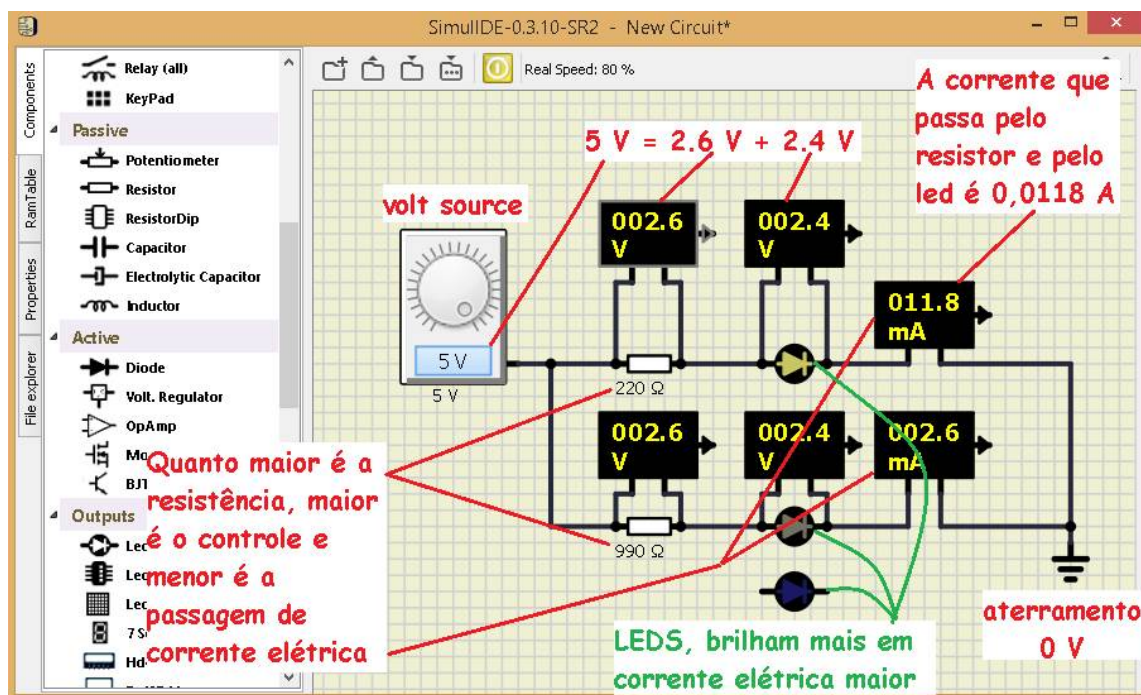


Figura 23 - Tensão, corrente e resistência elétricas relacionadas ao funcionamento de leds

HORA DA PRÁTICA:

Eu convido o leitor a assistir novamente meu vídeo de introdução ao SIMULIDE.

https://www.youtube.com/watch?v=R_28BLSyfRA

Eu convido o leitor a assistir meu vídeo sobre tensão elétrica no SIMULIDE.

<https://www.youtube.com/watch?v=ADEFXxDr5wpg>

Eu convido o leitor a acessar o site BrasilEscola e ler um artigo sobre Tensão Elétrica.

<https://brasilestola.uol.com.br/fisica/tensao-eletrica.htm>

Eu convido o leitor a acessar o site BrasilEscola e ler um artigo sobre Corrente Elétrica.

<https://brasilescola.uol.com.br/fisica/corrente-eletrica.htm>

Eu convido o leitor a acessar o site BrasilEscola e ler um artigo sobre Resistência Elétrica.

<https://brasilescola.uol.com.br/o-que-e/fisica/o-que-e-resistencia-eletrica.htm>

QUARTA PARTE

O conceito de linguagem de programação.

Variáveis.

Laços de repetição.

Estruturas de decisão.

Escrita.

Leitura.

Utilização dos componentes do kit básico: resistores, displays, leds e botões.

LINGUAGEM DE PROGRAMAÇÃO

É um conjunto de regras que possibilita a comunicação de instruções para uma máquina. Para escrever um programa que a placa UNO consiga processar, vamos precisar escrever em uma IDE (programa que cria programas) que neste caso é o ARDUINO IDE e vamos precisar de uma linguagem, no nosso caso, a lendária linguagem C (adaptada para placas ARDUINO e/ou genéricas).

HORA DA PRÁTICA:

Vamos praticar um pouco fazendo uma atividade simples no SIMULIDE.

Aqui o leitor tem duas opções: ler as instruções abaixo sobre como utilizar o simulide com o Arduino IDE ou assistir a um vídeo que fiz sobre estas

instruções:

<https://www.youtube.com/watch?v=O90Tgfxd2J4>

UTILIZAÇÃO DO SIMULIDE

Para executar o software SIMULIDE, depois de baixado e descompactado, entrar na pasta “SimulIDE_0.3.10”, depois entrar na pasta “bin” e dar duplo clique no arquivo “simulide”.

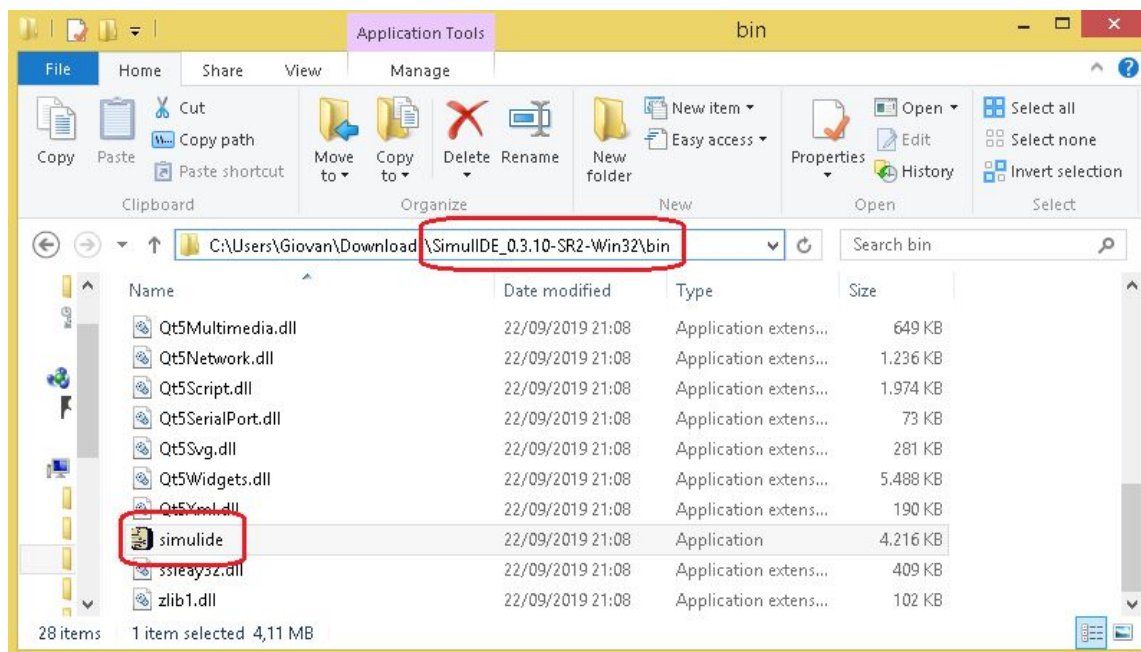


Figura 24 - Executar o arquivo simulide

Depois de aberto, ajuste a largura dos campos direito, esquerdo e do breadboard, conforme as instruções da imagem 25.

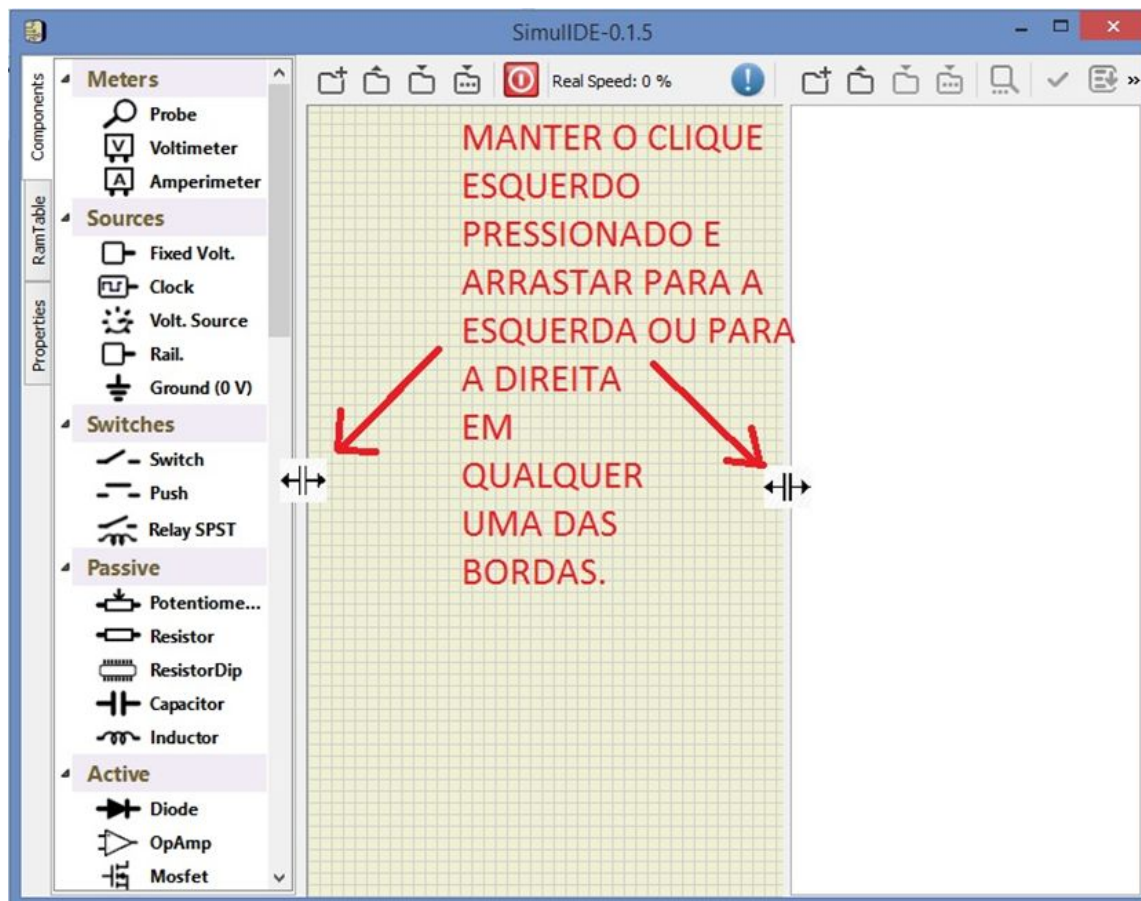


Figura 25 - Utilização do SIMULIDE

Para fechar o programa dê um clique esquerdo de mouse no 'x' na parte superior esquerda da janela do SIMULIDE. Conforme a imagem abaixo.

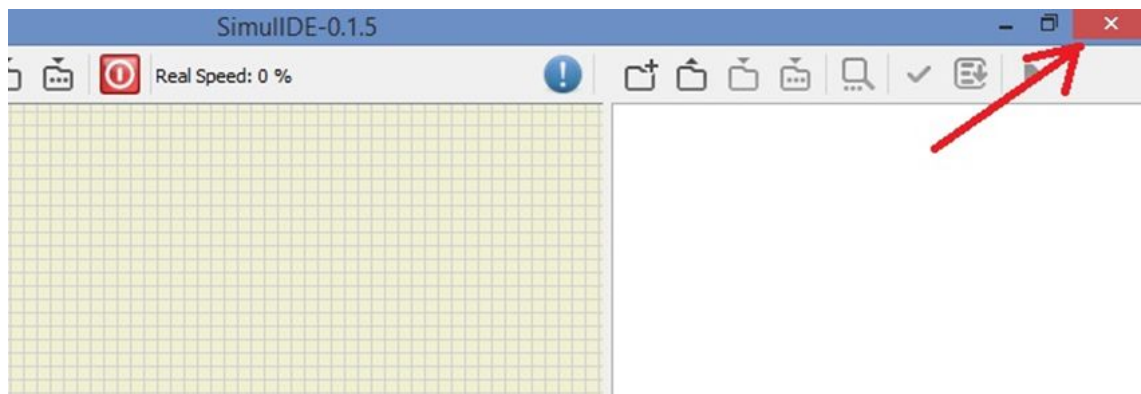


Figura 26 - Utilização do SIMULIDE

Para criar um circuito, dê um clique esquerdo no botão NEW CIRCUIT. Conforme a imagem abaixo.

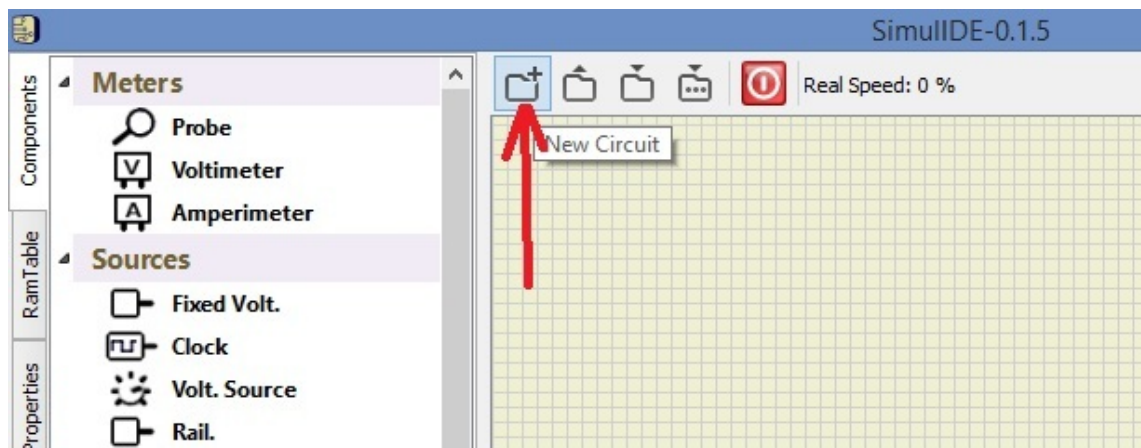


Figura 27 - Utilização do SIMULIDE

Para salvar o circuito atual, dê um clique esquerdo no botão SAVE CIRCUIT. Conforme a imagem abaixo. Se desejar, salve o circuito como “ATIVIDADE_001”.

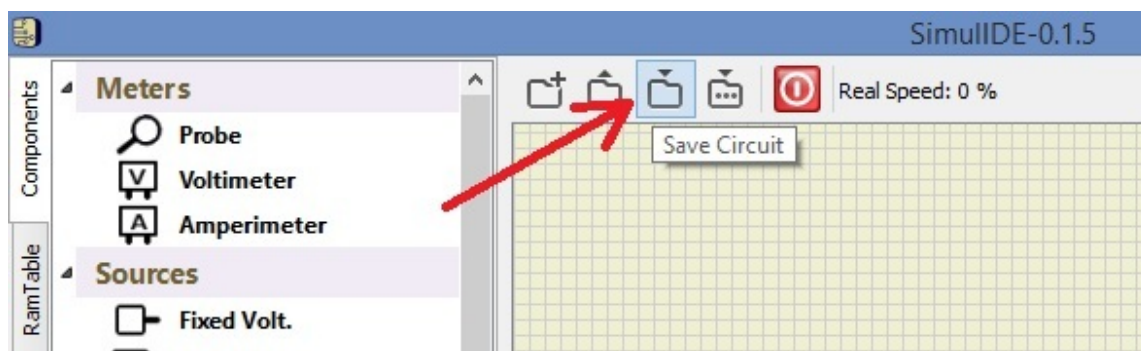


Figura 28 - Utilização do SIMULIDE

Para arrastar componentes para o protoboard, mantenha o botão esquerdo do mouse pressionado sobre um componente (parte esquerda da tela) e o arraste até o protoboard (parte direita), então solte o botão esquerdo. Conforme a imagem abaixo.

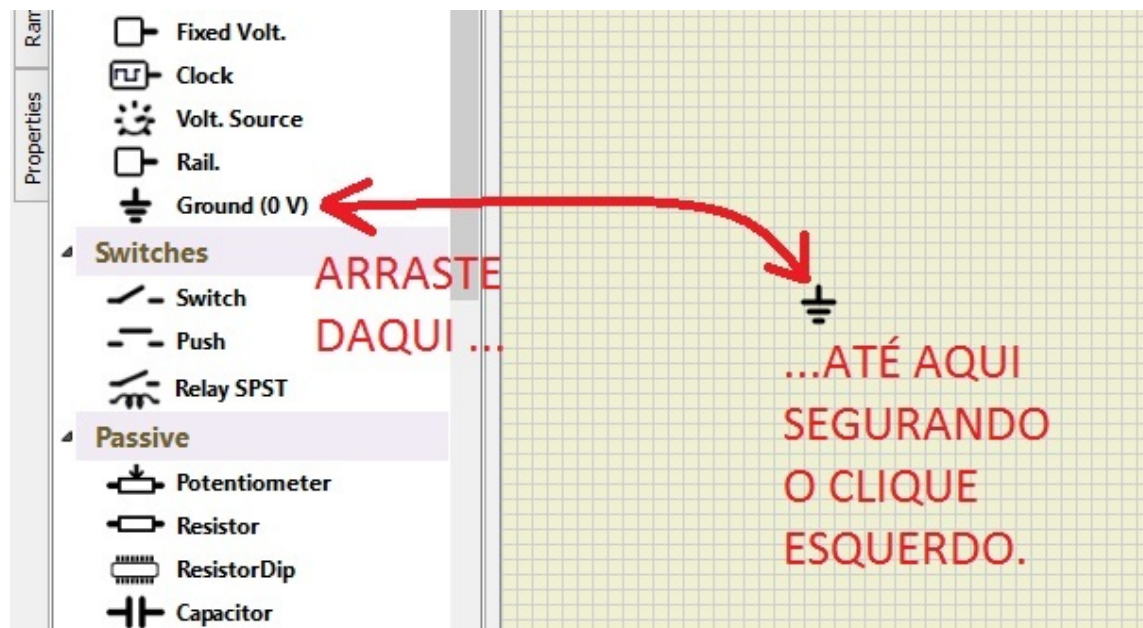


Figura 29 - Utilização do SIMULIDE

Para mover componentes pelo protoboard, mantenha o botão esquerdo do mouse pressionado sobre um componente já localizado no protoboard e o arraste, solte o botão para que o componente seja solto também. Conforme a imagem abaixo.

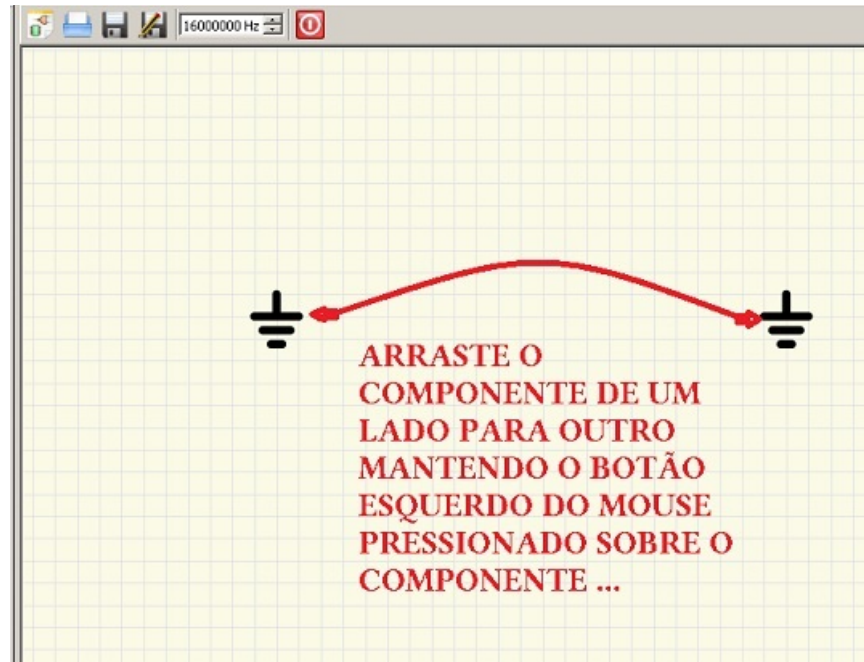


Figura 30 - Utilização do SIMULIDE

Arraste para o breadboard (área quadriculada) um “ground”, um “led” e um “volt. source”. Para conectar os componentes, dê um clique esquerdo de mouse em um terminal do componente e depois dê um clique esquerdo de mouse em outro terminal. Conforme a imagem abaixo. Obs. o led deve ser colocado exatamente como na figura abaixo porque ele tem polo, se for colocado invertido, no SIMULIDE ele não acenderá e em uma placa física ele provavelmente queimará.

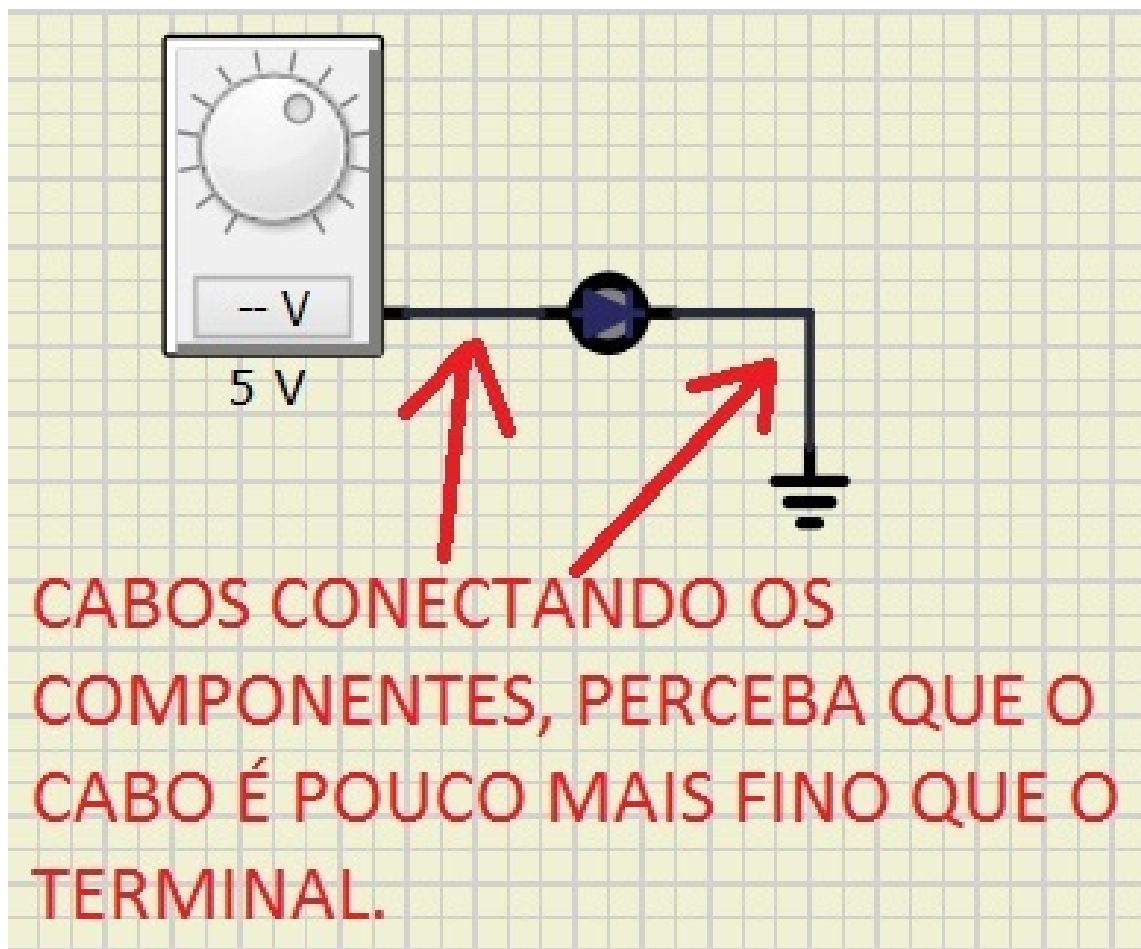


Figura 31 - Utilização do SIMULIDE

Para ligar a simulação, dê um clique esquerdo de mouse no botão POWER CIRCUIT, mas note que no exemplo abaixo foi necessário ligar o botão do VOLT SOURCE (com um clique esquerdo de mouse) e depois colocar próximo de 3 volts ou mais conforme a imagem abaixo.

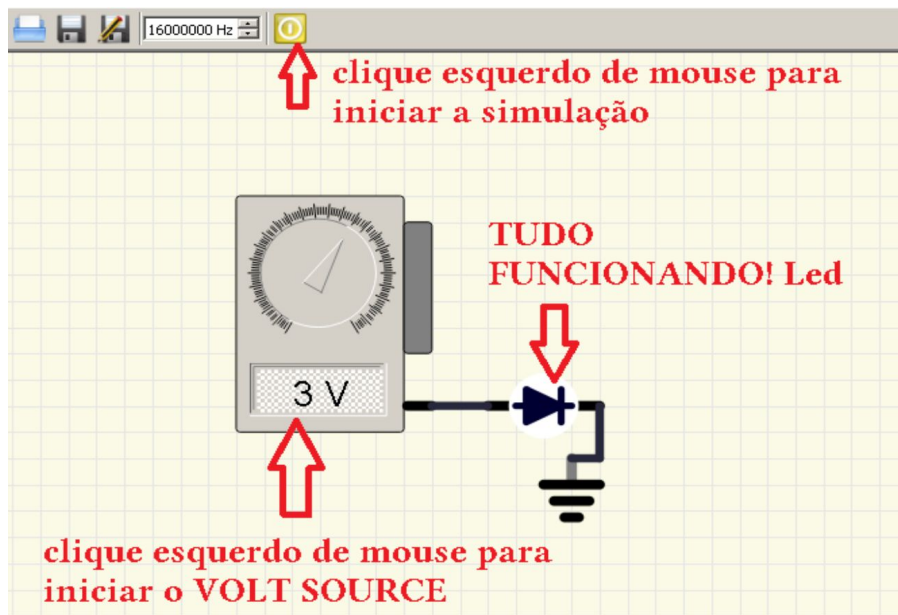


Figura 32 - Utilização do SIMULIDE

Para retirar um componente do protoboard, desligue a simulação e dê um clique direito de mouse nele, depois um clique esquerdo em "Remove".

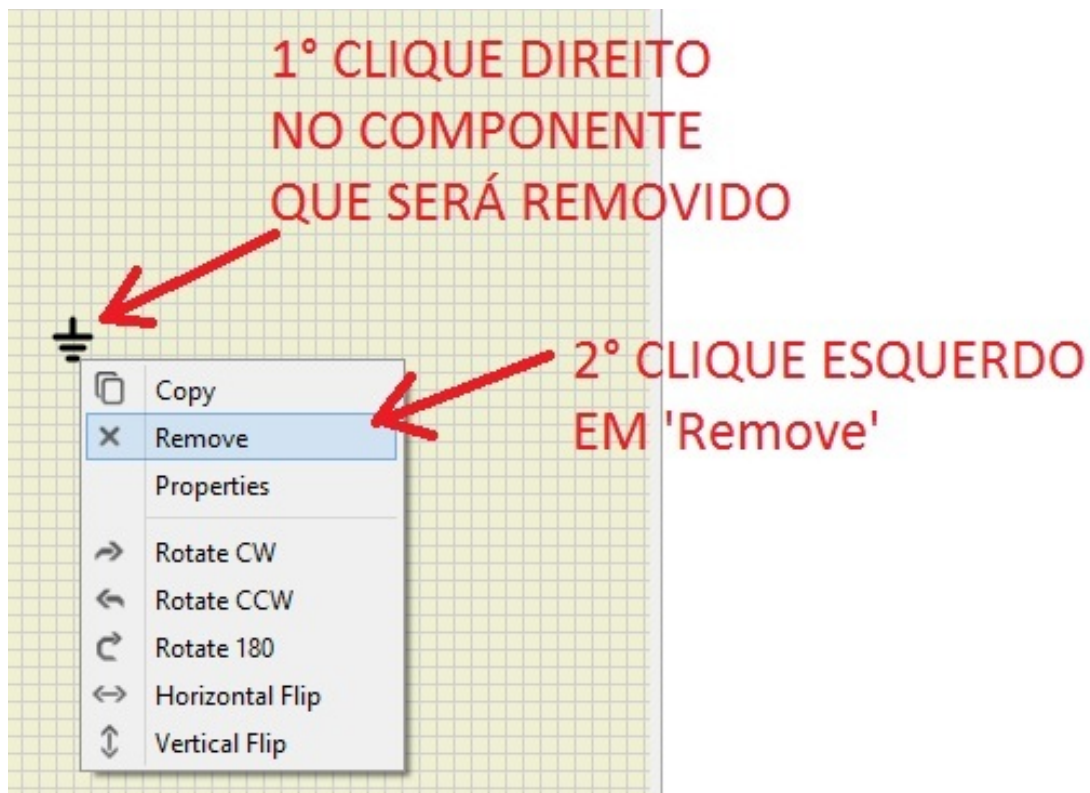


Figura 33 - Utilização do SIMULIDE

Para abrir um circuito, dê um clique esquerdo no botão OPEN CIRCUIT. Conforme a imagem abaixo.

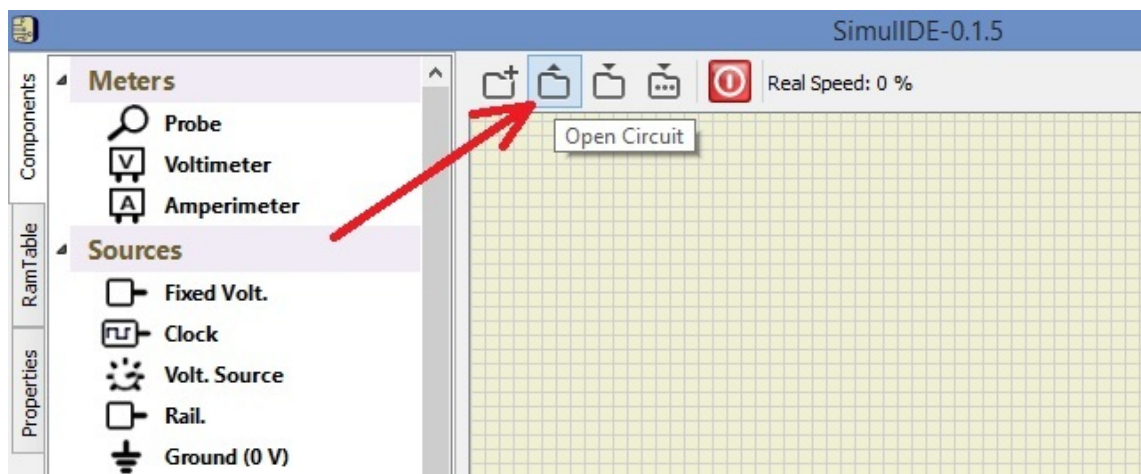


Figura 34 - Utilização do SIMULIDE

HORA DA PRÁTICA:

Vamos praticar um pouco fazendo uma atividade simples no ARDUINO IDE.

UTILIZAÇÃO DO ARDUINO IDE

Pressione o botão bandeira (2º na parte inferior esquerda) do teclado do seu computador e digite 'arduino' no teclado, o programa vai aparecer no menu iniciar do Windows, dê um clique esquerdo no nome do programa.

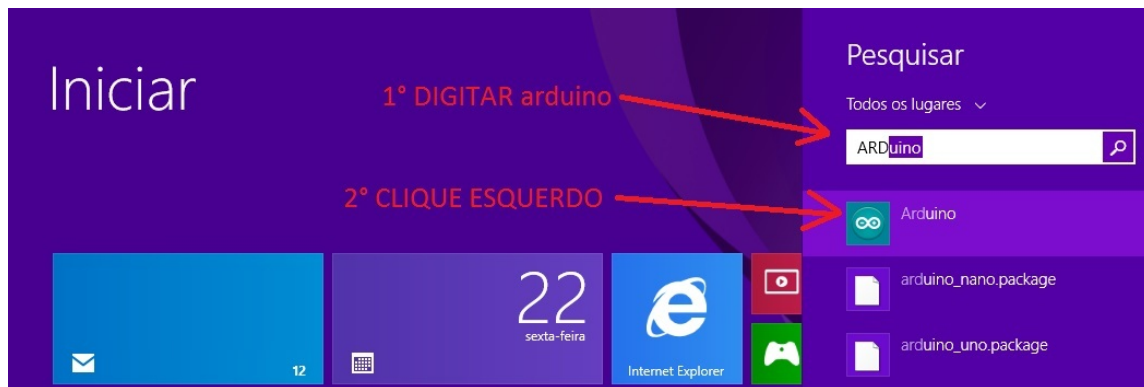


Figura 35 - Iniciando o ARDUINO IDE

O ARDUINO IDE vai abrir mostrando a estrutura básica de um código escrito na linguagem C (de ARDUINO).

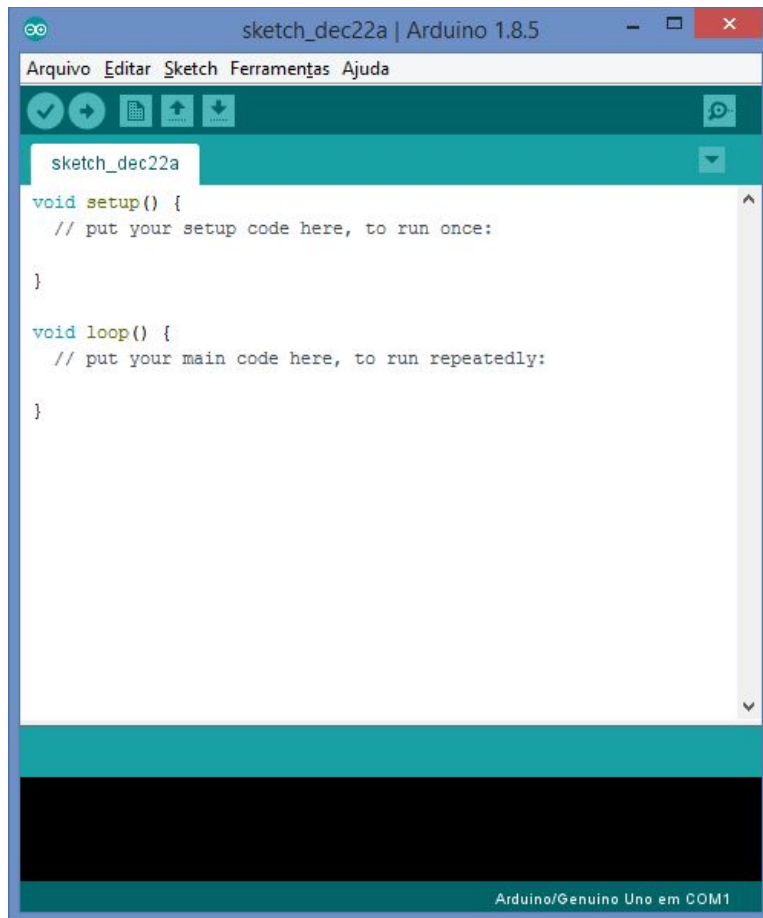


Figura 36 - Utilização do ARDUINO IDE

As frases em Inglês escritas depois de “//” são chamadas de comentários. Em linguagem de programação, o comentário é lido/escrito pelo programador (você meu caro leitor), não é lido pelo compilador (ARDUINO IDE) e nem vai para o programa gerado (arquivo “.hex”).

Vamos reescrever o código inicial da maneira mostrada na figura abaixo. Obs. Programar é uma arte! Peço que o leitor se atente e copie o código exatamente como é mostrado pois letras maiúsculas são completamente diferentes de letras minúsculas na linguagem C (dizemos que uma linguagem é Case Sensitive quando ela diferencia letras maiúsculas de minúsculas).

```
void setup() {  
    pinMode(13, OUTPUT);  
    digitalWrite(13, HIGH);  
}  
  
void loop() {  
  
}
```

Figura 37 - Utilização do ARDUINO IDE

Depois de digitar o código acima, vamos salvar o código. Para tanto, iremos em Arquivo → Salvar, daremos o nome de “ATIVIDADE_002”.

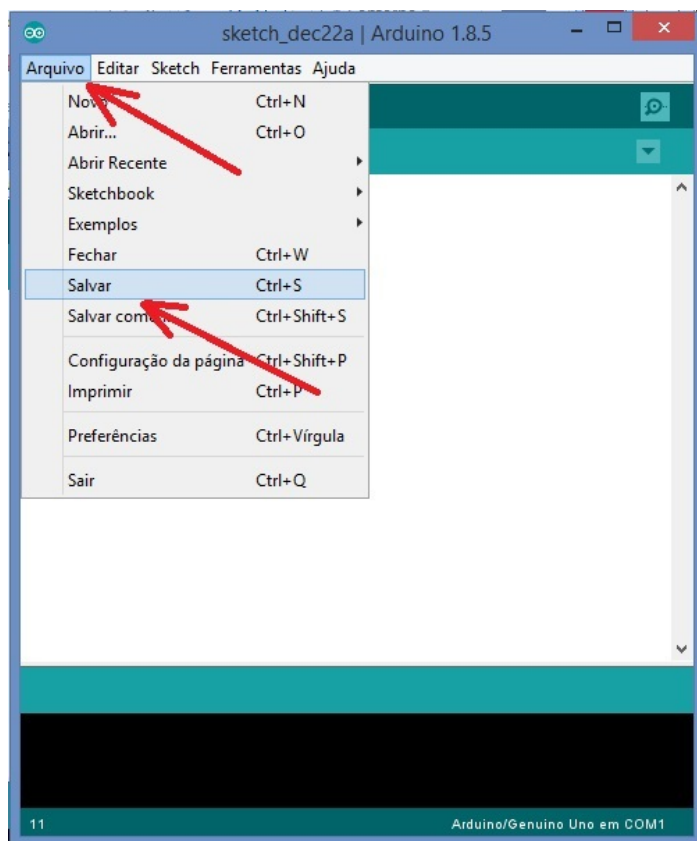


Figura 38 - Utilização do ARDUINO IDE

Após salvar o projeto com o nome “ATIVIDADE_002”, se você for até o local onde o projeto foi salvo, você perceberá que o ARDUINO IDE criou uma pasta com o nome “ATIVIDADE_002” e salvou o arquivo “ATIVIDADE_002” dentro.

O leitor perceberá também que a guia onde está escrito o código agora tem como nome “ATIVIDADE_002”, além da janela do ARDUINO IDE, conforme a imagem abaixo.



Figura 39 - Utilização do ARDUINO IDE

Agora que o projeto foi salvo, vamos compilar (transformar um código escrito por um programador em um arquivo que uma máquina entende, no nosso caso, um arquivo “.hex”).

Para compilar, dar um clique esquerdo em Sketch → Exportar Binário compilado. Conforme a figura abaixo.

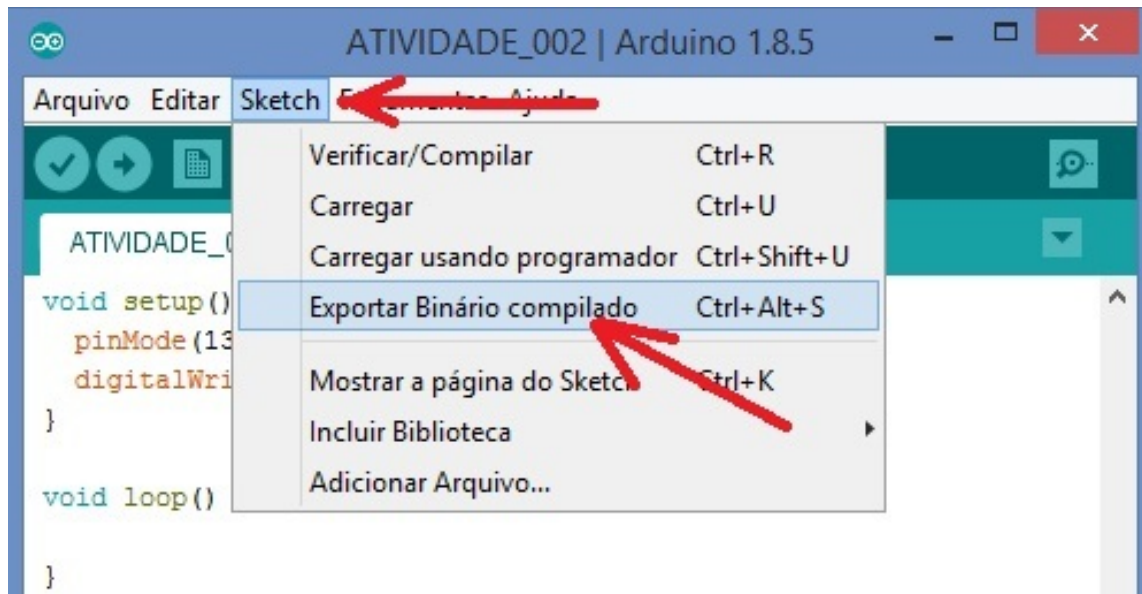


Figura 40 - Utilização do ARDUINO IDE

O arquivo “.hex” (também chamado de firmware) foi gerado com o nome “ATIVIDADE_002.ino.standard.hex” dentro da pasta chamada “ATIVIDADE_002”.

Agora vamos abrir o SIMULIDE e vamos arrastar uma placa UNO para o breadboard. Conforme a figura abaixo.

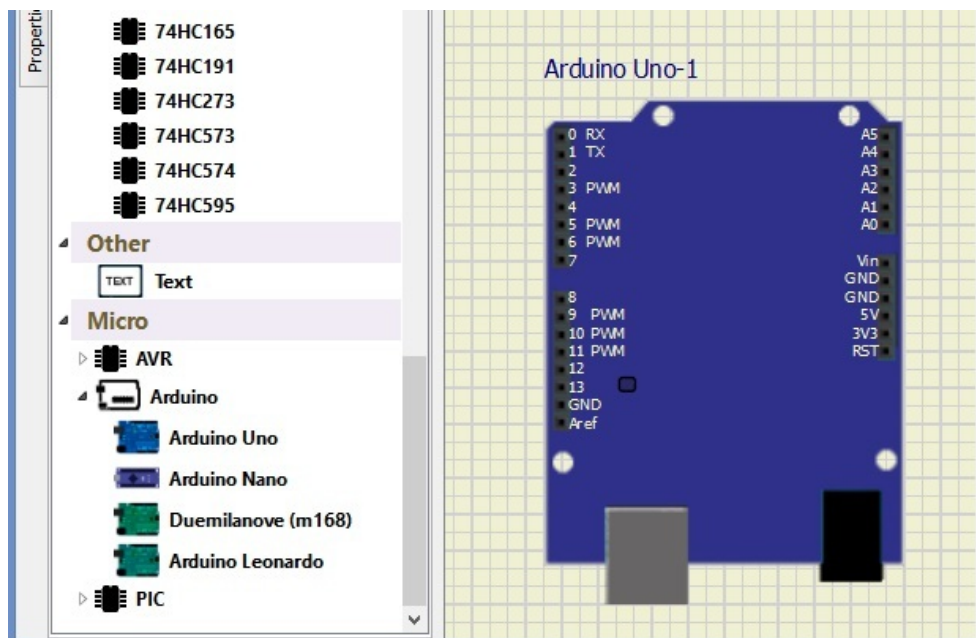


Figura 41 - Utilização do ARDUINO IDE

Obs. depois que a placa UNO for arrastada para o breadboard, pode ser dado zoom mantendo a tecla CTRL pressionada no teclado e girando o SCROLL do mouse.

Agora, para enviar o arquivo “.hex” (também chamado de firmware) para o simulador, teremos que encontrá-lo na pasta chamada “ATIVIDADE_002” e indicá-lo para a placa virtual UNO (clique direito sobre o ARDUINO UNO virtual e depois escolha a opção LOAD FIRMWARE, então vá para a pasta chamada “ATIVIDADE_002” e escolha o arquivo “ATIVIDADE_002.ino.standard.hex”).

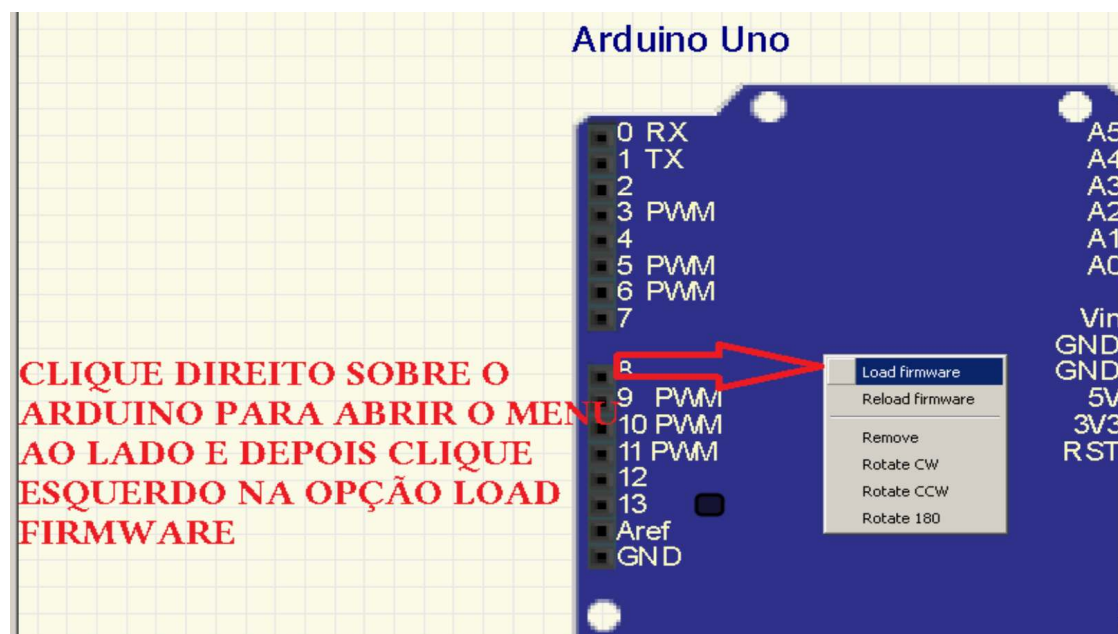


Figura 42 - Carregando firmware no SIMULIDE

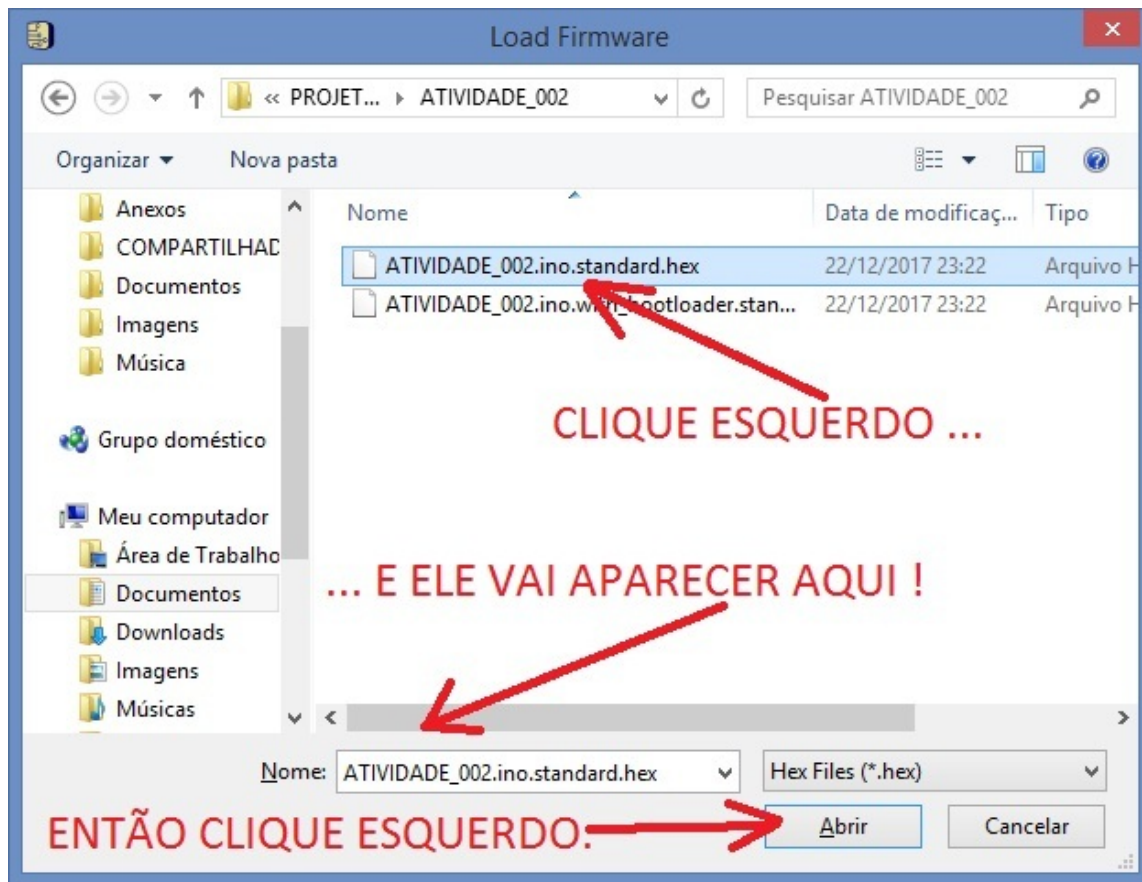


Figura 43 - Carregando firmware no SIMULIDE

Depois é só iniciar a simulação e o led interno localizado ao lado da porta 13 vai acender. Conforme a figura abaixo

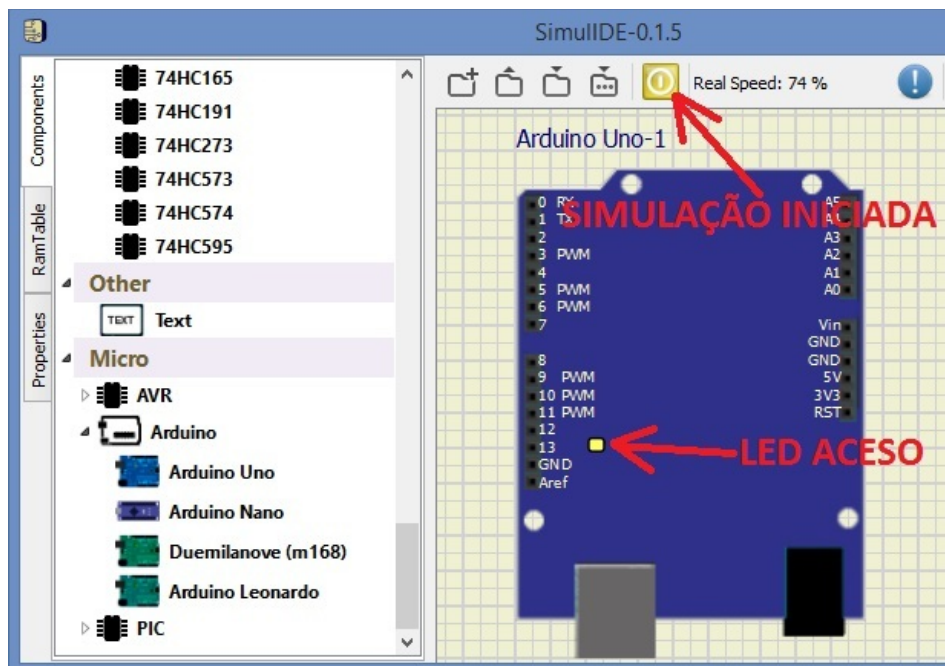


Figura 44 - Iniciando a simulação no SIMULIDE

Se você chegou até aqui e sua placa acendeu o led, parabéns, este é o caminho. Agora vamos sair da simulação e acender o led interno da placa real.

Para tanto, conecte a sua placa real no seu computador utilizando o cabo USB que acompanha o ARDUINO UNO. Neste momento, provavelmente aparecerá uma tela associando o driver ch340 à placa UNO. Conforme figura abaixo.

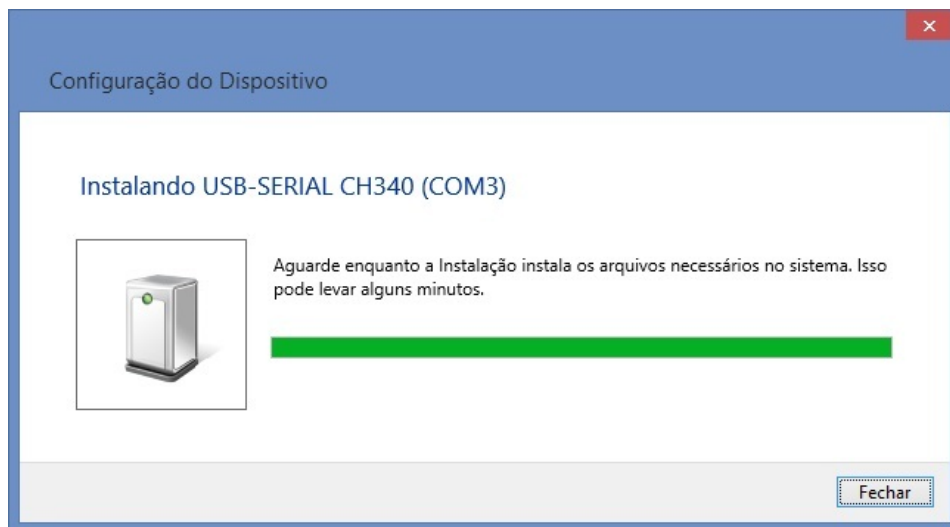


Figura 45 - Windows 8.1 configurando o driver da placa genérica UNO

Jogar o firmware (arquivo “.hex”) em um ARDUINO UNO físico é extremamente fácil, basta dar um clique esquerdo no botão UPLOAD (ou CARREGAR) depois que o código estiver digitado.

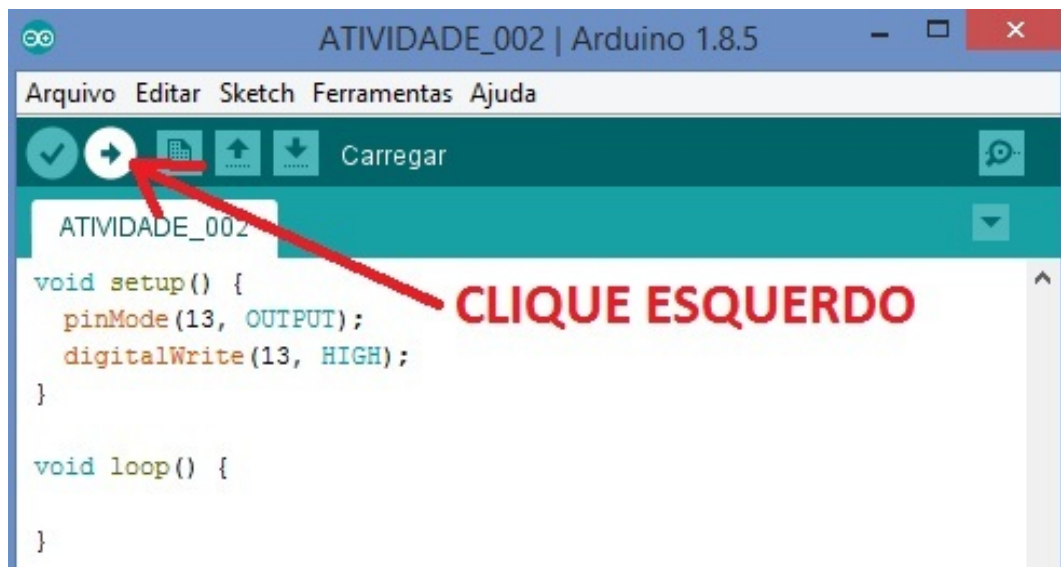


Figura 46 - Exportar o firmware para a placa UNO física

Obs. Por padrão o compilador espera o ARDUINO UNO na porta “COM1”, mas se o seu ARDUINO estiver em uma porta USB reconhecida pelo compilador como diferente de “COM1”, basta dar um clique esquerdo em Ferramentas → Porta: → COM2 (ou COM3, ou COM4, etc.) conforme figura abaixo (no meu caso, a porta encontrada foi a COM3).

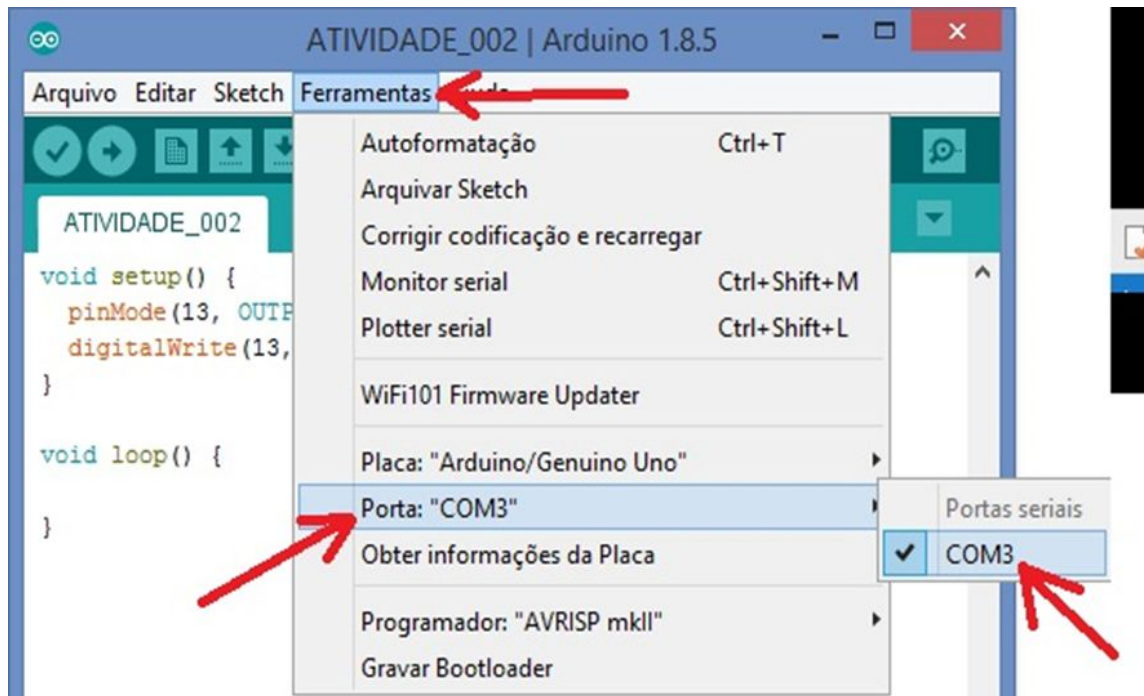


Figura 47 - Configurar a porta correta, se necessário

INTRODUÇÃO À PROGRAMAÇÃO

Agora vamos estudar em detalhes o código utilizado na ATIVIDADE_002.

No nosso código, 'void setup() { }' é uma função onde é colocado tudo o que vai acontecer uma vez só, enquanto 'void loop() { }' é uma função onde é colocado tudo o que vai acontecer repetidamente.

SAÍDA (OU ESCRITA) EM PORTA DIGITAL

O comando 'pinMode' habilita uma porta (13 no nosso caso) para ser utilizada para saída (OUTPUT).

O comando 'digitalWrite' configura a tensão em uma porta (13 no nosso caso) para saída com 5v (HIGH) ou 0v (LOW).

OBS. o led interno do ARDUINO UNO pode ser acessado pela porta 13 ou pelo comando LED_BUILTIN.

Agora vamos pensar na seguinte situação: Como fazer o led interno piscar duas vezes?

Devemos escolher uma porta (13 no nosso caso) e fazer o led ligar, depois fazer o led esperar um tempo, depois desligar, depois esperar um tempo e depois repetir o ciclo mais uma vez.

O comando para fazer um componente esperar um tempo é o 'delay(1000)' onde o 1000 vale 1 segundo, 1500 vale 1,5 segundos, 20000 vale 20 segundos e assim por diante.

Um led acende se a tensão for HIGH e apaga se a tensão for LOW.

HORA DA PRÁTICA:

ATIVIDADE_003. Fazer o led interno acender e apagar duas vezes.

CÓDIGO

```
void setup() {  
    pinMode(13, OUTPUT);//habilitou a porta 13 para saída  
    digitalWrite(13, HIGH);//acendeu  
    delay(1000);//esperou 1 segundo  
    digitalWrite(13, LOW);//apagou  
    delay(1000);//esperou 1 segundo  
    digitalWrite(13, HIGH);//acendeu  
    delay(1000);//esperou um segundo  
    digitalWrite(13, LOW);//apagou  
    //e vai permanecer apagado  
}  
  
void loop() {  
}
```

EXERCÍCIOS

1) Escrever um código para acender um LED na porta digital 2. Simular e depois executar na placa física.

2) Escrever um código para acender e apagar três vezes um LED na porta digital 2. Simular e depois executar na placa física.

3) Escrever um código para piscar infinitamente um LED na porta digital 3. Simular e depois executar na placa física.

4) Escrever um código para acender e depois apagar um LED na porta digital 4. Simular e depois executar na placa física.

5) Desafio: escrever um código para fazer funcionar um semáforo. Simular e depois executar na placa física.

SOBRE A RESOLUÇÃO

Para resolver os exercícios do 1 ao 4, será necessário 1 led, 1 resistor de 220 ohm, 2 cabos e o breadboard (além da placa UNO).

O BREADBOARD

A figura abaixo ilustra o funcionamento de um breadboard.



Figura 48 - Funcionamento do breadboard

A figura abaixo ilustra como deve ser colocado um componente no breadboard.

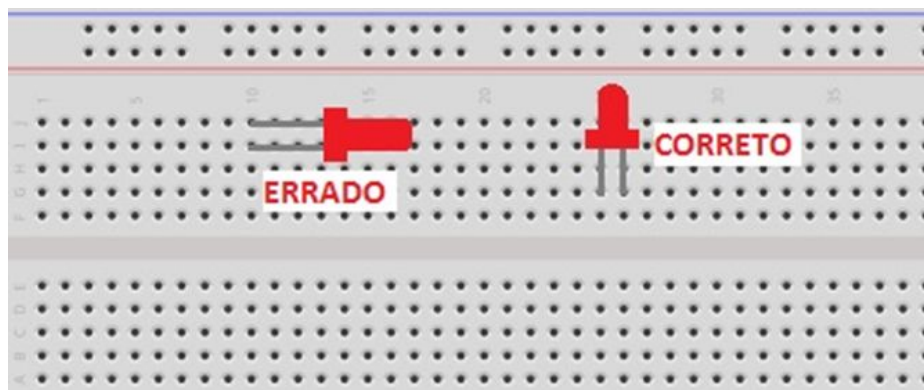


Figura 49 - Posicionamento correto de um componente no breadboard

O led é um componente eletrônico que tem polaridade (ou seja, tem lado certo) então o terminal mais curto deve sempre ser ligado no GND da placa UNO, se o terminal mais longo for ligado ao GND, o led pode queimar.

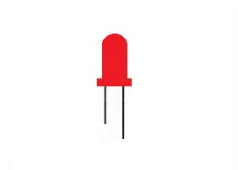


Figura 50 - LED

O resistor não tem polaridade.



Figura 51 - Resistor de 220 Ω

Obs. O resistor da figura 46 possui 4 faixas nas cores: vermelho, vermelho, preto, preto, marrom.

Um ponto importante a ser notado é que para piscar um led com a placa UNO física, sempre deve ser ligado o led em série com um resistor de 220 Ω (ambos acompanham a maioria dos kits para iniciantes), já no SIMULIDE, podemos utilizar o led em série com um resistor de 100 Ω , embora seja simples mudar o valor da resistência do resistor no simulide.

Para este livro não ficar demasiadamente longo, eu deixarei a resolução dos exercícios (código, diagramas e projetos simulide) [aqui](#).

Dentro da pasta de cada exercício há uma imagem, que é o diagrama de montagem dos componentes na placa UNO, há um arquivo Arduino, que contém o código em C, um arquivo “ino.standard.hex” que é o firmware que pode ser carregado no ARDUINO virtual dentro do SIMULIDE e um arquivo “.simu” que é o projeto que pode ser aberto no SIMULIDE.

MAIS PROGRAMAÇÃO

COMENTÁRIOS

Um programador experiente sabe que quanto mais organizado é um código, mais fácil é para dar manutenção e para aprimorar tal código. Uma ferramenta útil na organização de um código é o comentário. Ele pode ser representado por // para comentários de apenas uma linha ou por /* no começo e */ no final de comentários de mais que uma linha.

VARIÁVEIS

Em programação, uma variável se refere a um espaço na memória do dispositivo. O programador pode manipular variáveis declarando-as (criando variáveis) e atribuindo valores a elas.

As variáveis podem armazenar números inteiros (dizemos que a variável tem tipo int), números que possuem casas decimais (tipo float) dentre outros tipos de variáveis, que não serão exploradas neste momento.

Vamos imaginar a seguinte situação: temos que desenvolver um programa para fazer nossa placa UNO acender e apagar um led, depois outro, depois outro. Vamos resolver primeiramente sem variáveis.

HORA DA PRÁTICA:

ATIVIDADE_004 – Resolução do problema acima sem a utilização de variáveis.

CÓDIGO

```
void setup() {  
  pinMode(2, OUTPUT);  
  pinMode(3, OUTPUT);  
  pinMode(4, OUTPUT);  
  digitalWrite(2, HIGH);  
  delay(1000);  
  digitalWrite(2, LOW);  
  digitalWrite(3, HIGH);  
  delay(1000);  
  digitalWrite(3, LOW);  
  digitalWrite(4, HIGH);  
  delay(1000);  
  digitalWrite(4, LOW);  
}  
  
void loop() {  
}
```

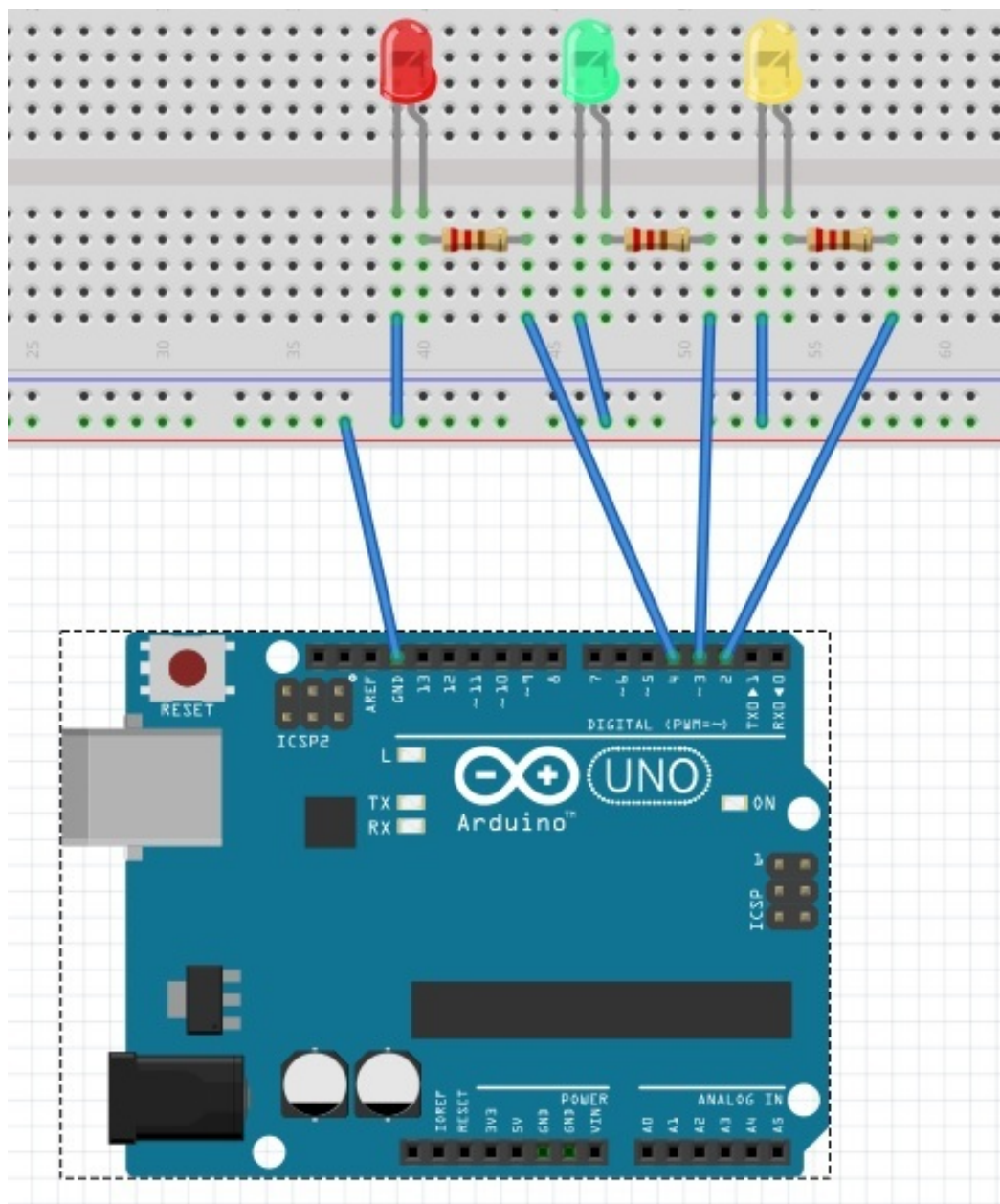


Figura 52 - Atividade 4 – Esquema da placa física

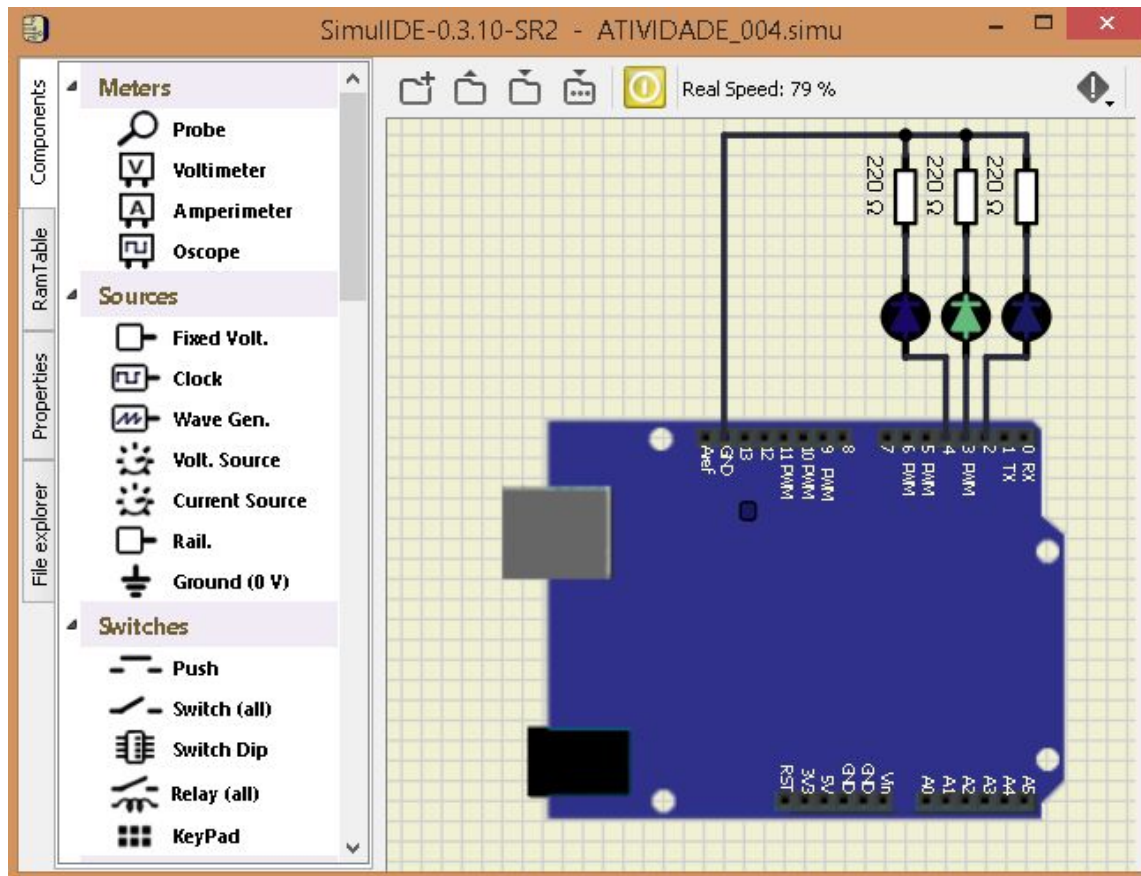


Figura 53 - Atividade 4 – Execução no SIMULIDE

ATIVIDADE_005 – Resolução do problema acima com a utilização de variáveis.

Obs. os esquemas da placa física e do SIMULIDE são idênticos aos da atividade 4.

CÓDIGO

```
int porta_a=2;//a variável porta_a vale 2
int porta_b=3;//a variável porta_b vale 3
int porta_c=4;//a variável porta_c vale 4
void setup() {
  /*abaixo, onde estiver a porta_a, na verdade é 2
  *onde estiver a porta_b é 3 */
```

```
pinMode(porta_a, OUTPUT);  
pinMode(porta_b, OUTPUT);  
pinMode(porta_c, OUTPUT);  
digitalWrite(porta_a, HIGH);  
delay(1000);  
digitalWrite(porta_a, LOW);  
digitalWrite(porta_b, HIGH);  
delay(1000);  
digitalWrite(porta_b, LOW);  
digitalWrite(porta_c, HIGH);  
delay(1000);  
digitalWrite(porta_c, LOW);  
}  
void loop() {  
}
```

Ao compararmos os códigos das atividades 4 e 5, se precisássemos mudar da porta 2 para a 5 na atividade 4, teríamos que alterar o número 2 para 5 em três linhas, enquanto que a mesma situação na atividade 5 exigiria a alteração apenas de uma linha (a 1º linha, que contém o 2 atribuído para a variável porta_a).

OPERADORES ARITMÉTICOS

São operadores utilizados em cálculos, alguns são muito comuns, como estes: '+' (adição), '-' (subtração), '*' (multiplicação) e '/' (divisão), enquanto este nem tanto: '%' (operador módulo, calcula o resto da divisão entre dois números inteiros).

Exemplo 1:

$$3 + 2 = 5$$

Exemplo 2:

$11 \% 2 = 1$ (o resto da divisão de 11 por 2 é 1)

Exemplo 3:

$6 * 2 = 12$

OPERADORES RELACIONAIS

Agora vamos estudar os operadores relacionais, eles são necessários para a realização de testes lógicos pela placa UNO e depois, pela decisão de executar ou não um determinado trecho de código em função do resultado dos testes.

Um teste lógico pode resultar em ‘verdadeiro’ ou ‘falso’. Para comparar se dois valores são iguais ou não, utiliza-se ‘==’ (dois sinais de igual juntos) porque na linguagem C, a utilização de apenas um sinal de igual representa o armazenamento de um valor por uma variável.

Exemplo 1:

`n1 = 2; //a variável n1 armazenou o valor 2`

`n1 == 5; //está comparando n1 com 5, o resultado é falso porque n1 vale 2`

Outros operadores lógicos são o > (maior), o < (menor), o >= (maior ou igual) o <= (menor ou igual) e o != (diferente).

Exemplo 2:

`n1 = 7;`

`n1 > 8; //testa se n1 é maior que 8, o resultado é falso`

EXERCÍCIOS

Classificar como verdadeiro ou falso cada expressão lógica abaixo.

a) `n1 = 3;` b) `abc = 0;` c) `x = 2;` d) `y = 11;`

`n1 >= 3;` `abc <= 0;` `x != 7;` `y > 100;`

RESPOSTAS

a) v b) v c) v d) f

OPERADORES LÓGICOS

Os operadores lógicos são utilizados quando um teste lógico é composto de mais de um teste. Aqui apresento os operadores ‘&’ (e) e ‘||’ (ou).

Exemplo 1:

```
n1 = 12; //n1 vale 12
```

```
n1 > 10 & n1 < 20; //afirmar que n1 é maior que 10 e menor que 20 é verdadeiro
```

Exemplo 2:

```
n1 = 20; //n1 vale 20
```

```
n1 == 20 || n1 > 100; //afirmar que n1 é 20 ou é maior que 100 é verdadeiro
```

A tabela abaixo ilustra os resultados finais quando temos um teste composto pelos testes A e B combinado com ‘&’.

A	B	A & B
V	V	V
V	F	F
F	V	F
F	F	F

A tabela abaixo ilustra os resultados finais quando temos um teste composto pelos testes A e B combinado com ‘||’.

A	B	A B
V	V	V
V	F	V
F	V	V
F	F	F

ESTRUTURAS DE DECISÃO

Uma estrutura de decisão é um trecho de código onde ocorre um teste lógico (um teste que pode dar verdadeiro ou falso apenas) e a partir do resultado a execução do código segue um de dois caminhos possíveis.

Aqui veremos dois tipos de estrutura de decisão: if/else e switch/case.

A estrutura do tipo if/else pode dividir o código em duas partes, uma roda se o resultado do teste lógico for verdadeiro e a outra roda se o resultado for falso. Resumo da sintaxe:

```
if (teste)
{
  Código que executa se o resultado do teste for verdadeiro.
  ...
}
else
{
  Código que executa se o resultado do teste for falso.
  ...
}
```

A estrutura do tipo if/else pode dividir o código em mais de duas partes, a primeira roda se o resultado do primeiro teste lógico for verdadeiro, a segunda roda se o resultado do segundo teste for verdadeiro (mas for falso do ponto de vista do primeiro teste) e a última parte roda quando todos os testes anteriores forem falsos. Resumo da sintaxe:

```
if (teste 1)
{
  Código que executa se o resultado do teste 1 for verdadeiro.
  ...
}
else if (teste 2)
{
  Código que executa se o resultado do teste 2 for verdadeiro e o resultado
do teste 1 for falso.
  ...
}
```

```
}  
else  
{  
Código que executa se os resultados de todos os testes anteriores forem  
falsos.  
...  
}
```

A estrutura do tipo switch/case pode dividir o código em várias partes, cada uma roda se o resultado do teste lógico específico for verdadeiro e a última roda se todos os resultados anteriores forem falsos. Resumo da sintaxe:

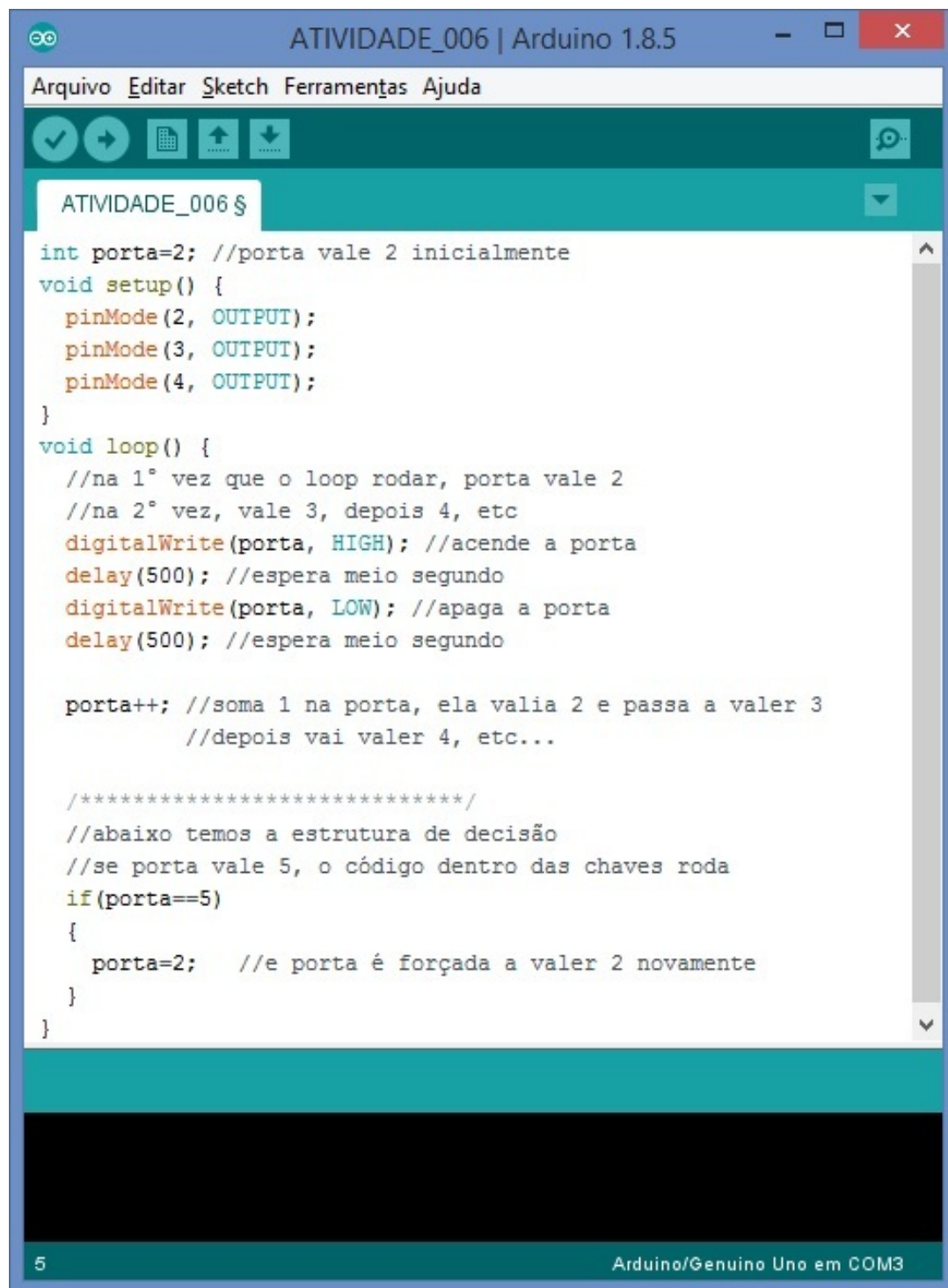
```
switch (nome_da_variável)  
{  
case valor_1 :  
Código para executar quando ocorrer o valor_1;  
break; //termina a execução  
case valor_2 :  
Código para executar quando ocorrer o valor_2;  
break; //termina a execução  
  
case valor_3 :  
Código para executar quando ocorrer o valor_3;  
break; //termina a execução  
  
default :  
Código para executar quando todos os anteriores forem falsos;  
}
```

Mas é muito importante notar que podem ser executadas qualquer quantia de testes, mas o último (default) ocorre quando todos os anteriores forem falsos, ou seja, a mesma coisa que o else no caso do if.

Se o leitor achar muito pesado ter que aprender as duas estruturas, poderá praticar apenas o if/else e resolver qualquer problema com ele.

HORA DA PRÁTICA:

ATIVIDADE_006 – Fazer três leds acenderem e apagarem na sequência indefinidamente, com intervalo de 0,5 s.



ATIVIDADE_006 | Arduino 1.8.5

Arquivo Editar Sketch Ferramentas Ajuda

ATIVIDADE_006 \$

```
int porta=2; //porta vale 2 inicialmente
void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}
void loop() {
  //na 1ª vez que o loop rodar, porta vale 2
  //na 2ª vez, vale 3, depois 4, etc
  digitalWrite(porta, HIGH); //acende a porta
  delay(500); //espera meio segundo
  digitalWrite(porta, LOW); //apaga a porta
  delay(500); //espera meio segundo

  porta++; //soma 1 na porta, ela valia 2 e passa a valer 3
           //depois vai valer 4, etc...

  /*****/
  //abaixo temos a estrutura de decisão
  //se porta vale 5, o código dentro das chaves roda
  if(porta==5)
  {
    porta=2; //e porta é forçada a valer 2 novamente
  }
}
```

5 Arduino/Genuino Uno em COM3

Figura 54 - Atividade 6 – Código utilizando if

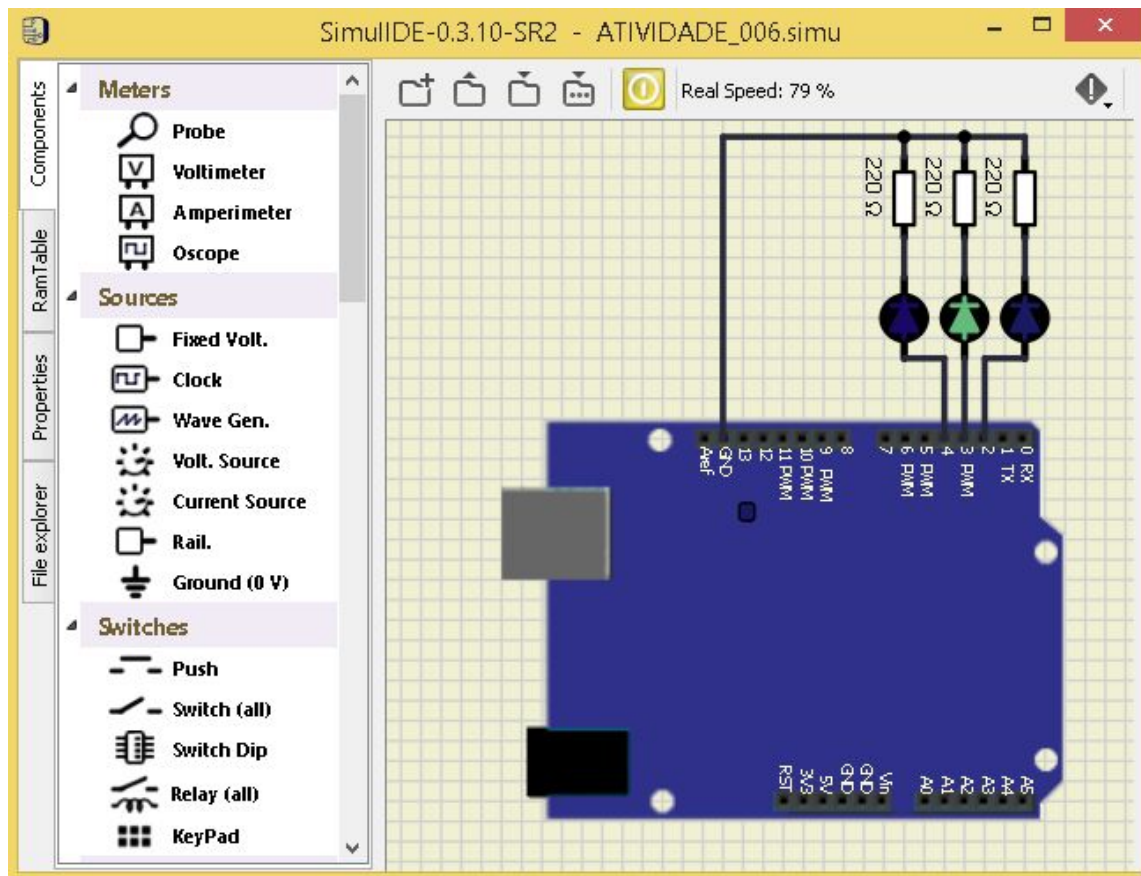


Figura 55 - Atividade 6 – Execução no SIMULIDE

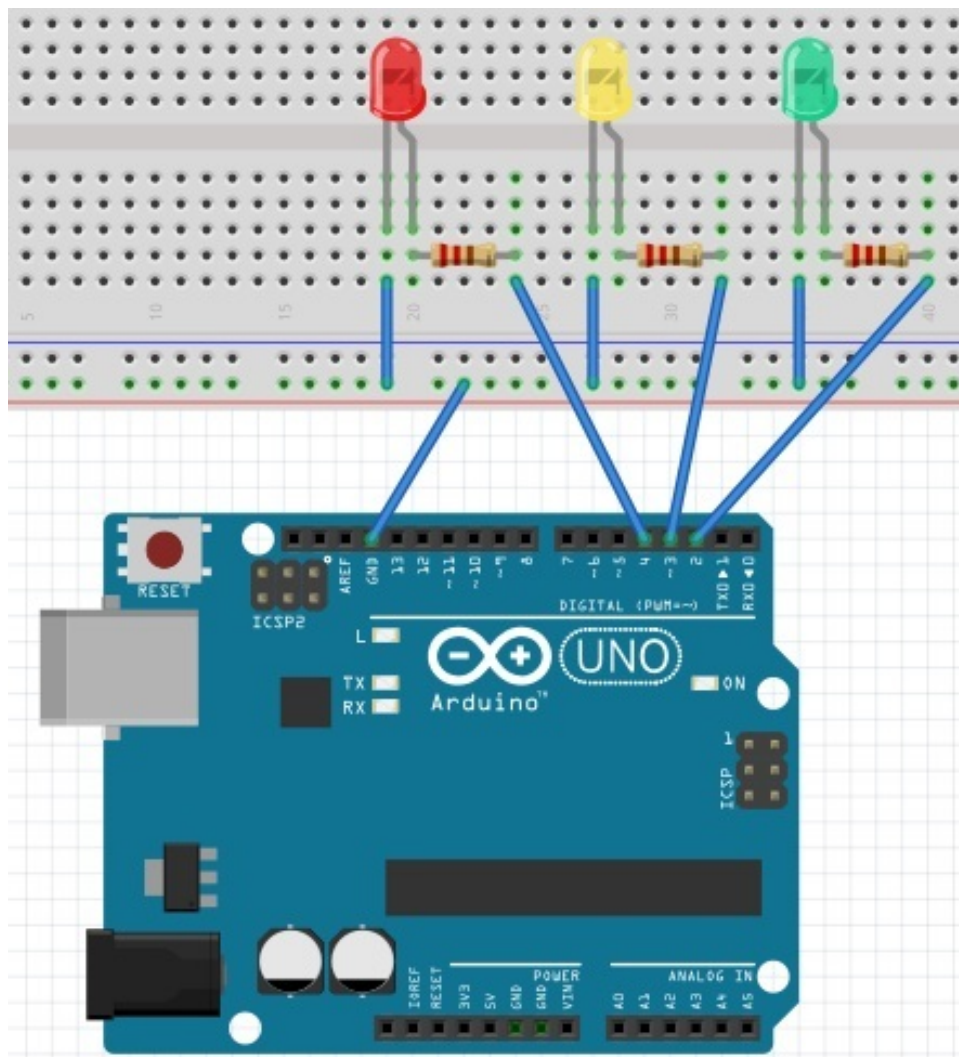


Figura 56 - Atividade 6 – Esquema da placa física

ATIVIDADE_006_1 – Utilizando três leds, escolher qual vai acender utilizando os números 2,3 ou 4 em uma variável e a estrutura de decisão if/else. Para esta atividade, os esquemas da placa física e do SIMULIDE são os mesmos da atividade 006.

CÓDIGO

```
//depois de rodar o código uma vez, mudar o valor abaixo de 2 para 3,  
//depois para 4, depois para qualquer outro número  
int porta=2;  
void setup() {
```

```
pinMode(2, OUTPUT);  
pinMode(3, OUTPUT);  
pinMode(4, OUTPUT);
```

```
digitalWrite(2, LOW);  
digitalWrite(3, LOW);  
digitalWrite(4, LOW);
```

```
if(porta == 2)  
{  
    digitalWrite(2, HIGH);  
}  
else if(porta == 3)  
{  
    digitalWrite(3, HIGH);  
}  
else if(porta == 4)  
{  
    digitalWrite(4, HIGH);  
}  
else  
{  
    digitalWrite(2, HIGH);  
    digitalWrite(3, HIGH);  
    digitalWrite(4, HIGH);  
}  
}  
  
void loop() {  
}
```

ATIVIDADE_006_2 – Utilizando três leds, escolher qual vai acender utilizando os números 2, 3 ou 4 em uma variável e a estrutura de decisão switch/case. Para esta atividade, os esquemas da placa física e do SIMULIDE são os mesmos da atividade 006.

CÓDIGO

//depois de rodar o código uma vez, mudar o valor abaixo de 2 para 3,

//depois para 4, depois para qualquer outro número

int porta=2;

void setup() {

pinMode(2, OUTPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

digitalWrite(2, LOW);

digitalWrite(3, LOW);

digitalWrite(4, LOW);

switch(porta)

{

case 2:

digitalWrite(2, HIGH);

break;

case 3:

digitalWrite(3, HIGH);

break;

case 4:

digitalWrite(4, HIGH);

break;

default:

```
    digitalWrite(2, HIGH);  
    digitalWrite(3, HIGH);  
    digitalWrite(4, HIGH);  
}  
  
void loop() {  
}
```

ENTRADA (OU LEITURA) EM PORTA DIGITAL

As portas digitais da placa UNO podem ser utilizadas para saída (OUTPUT) ou para entrada (INPUT) de corrente elétrica. O comando para a leitura digital é o **digitalRead(número_da_porta)**.

HORA DA PRÁTICA:

ATIVIDADE_007 – Fazer um teste utilizando a porta digital 2 como entrada e as portas digitais 3 e 4 como saída. Se a porta 2 receber 5 V (ou seja HIGH), um led verde acenderá na porta 3, se a porta 2 receber 0 V (ou seja LOW) um led vermelho acenderá na porta 4.

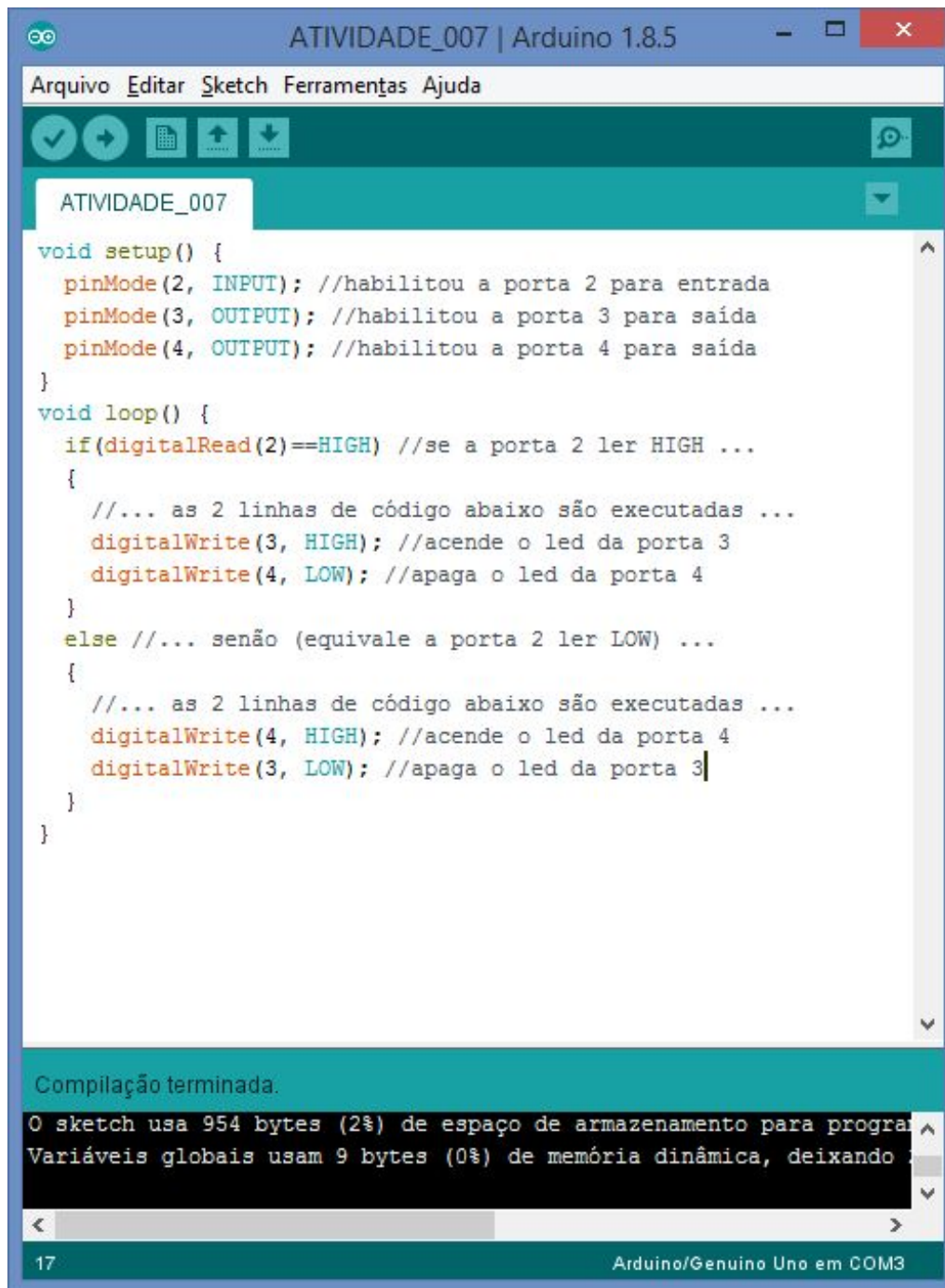


Figura 57 - Atividade 7 – Código utilizando if e entrada de uma porta digital

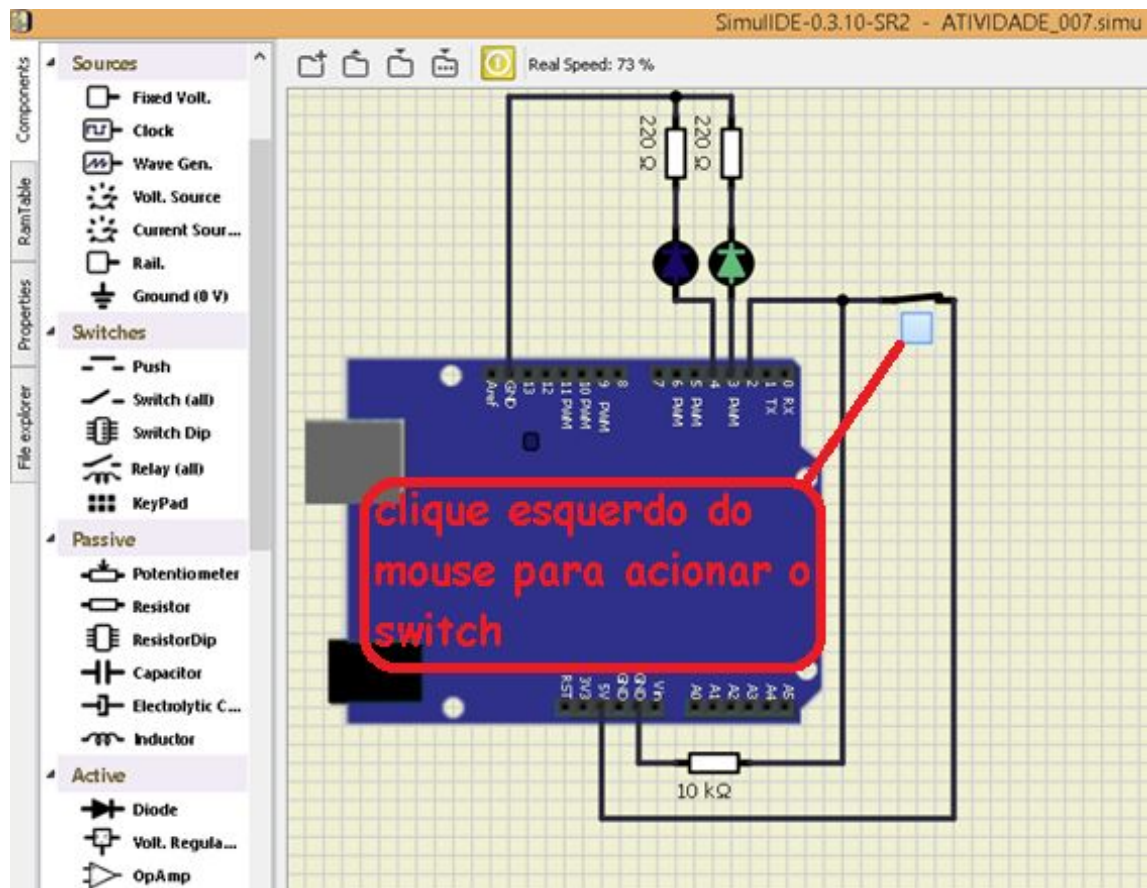


Figura 58 - Atividade 7 – Execução no SIMULIDE

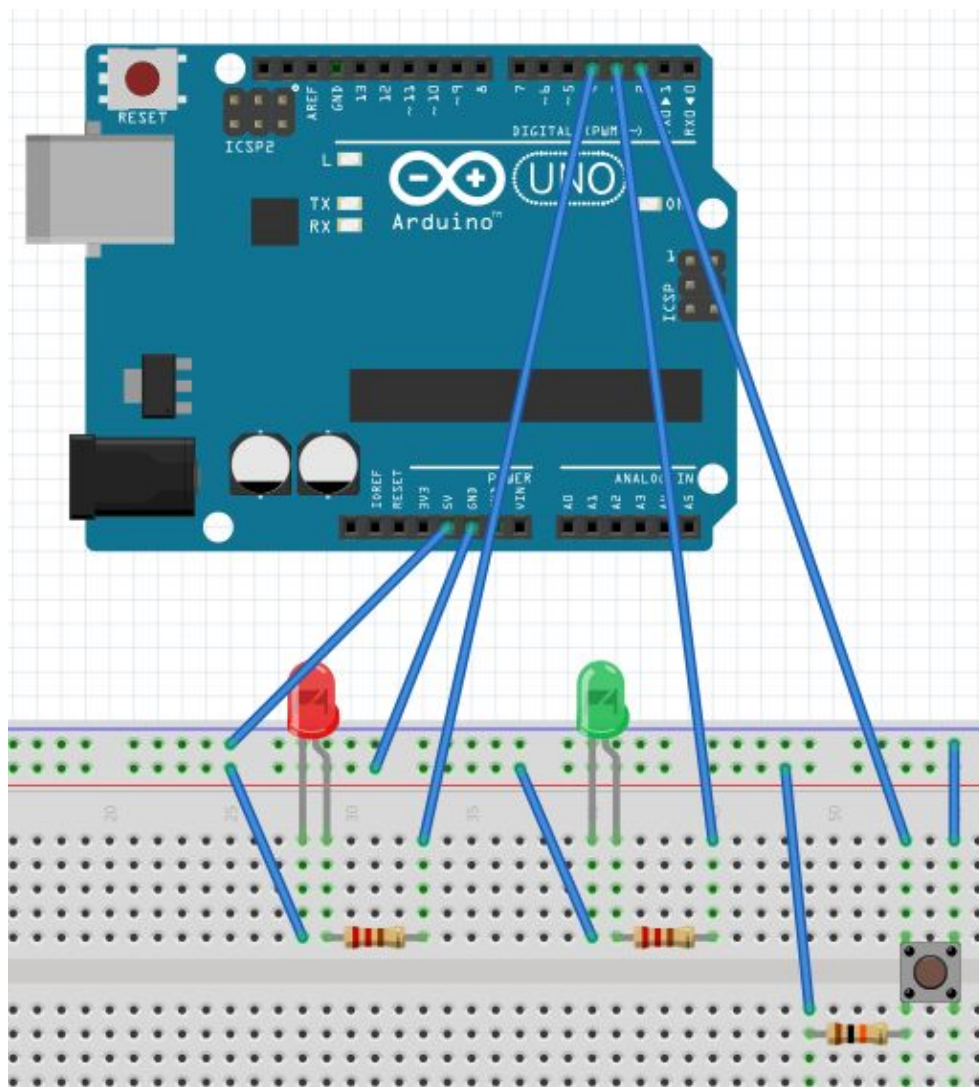


Figura 59 - Atividade 7 – Esquema da placa física

Como alguns kits de iniciantes não possuem botões, a figura abaixo ilustra as ligações sem botão. O fio vermelho se ligado conforme a imagem faz a luz verde acender, se desligado, faz a luz vermelha acender.

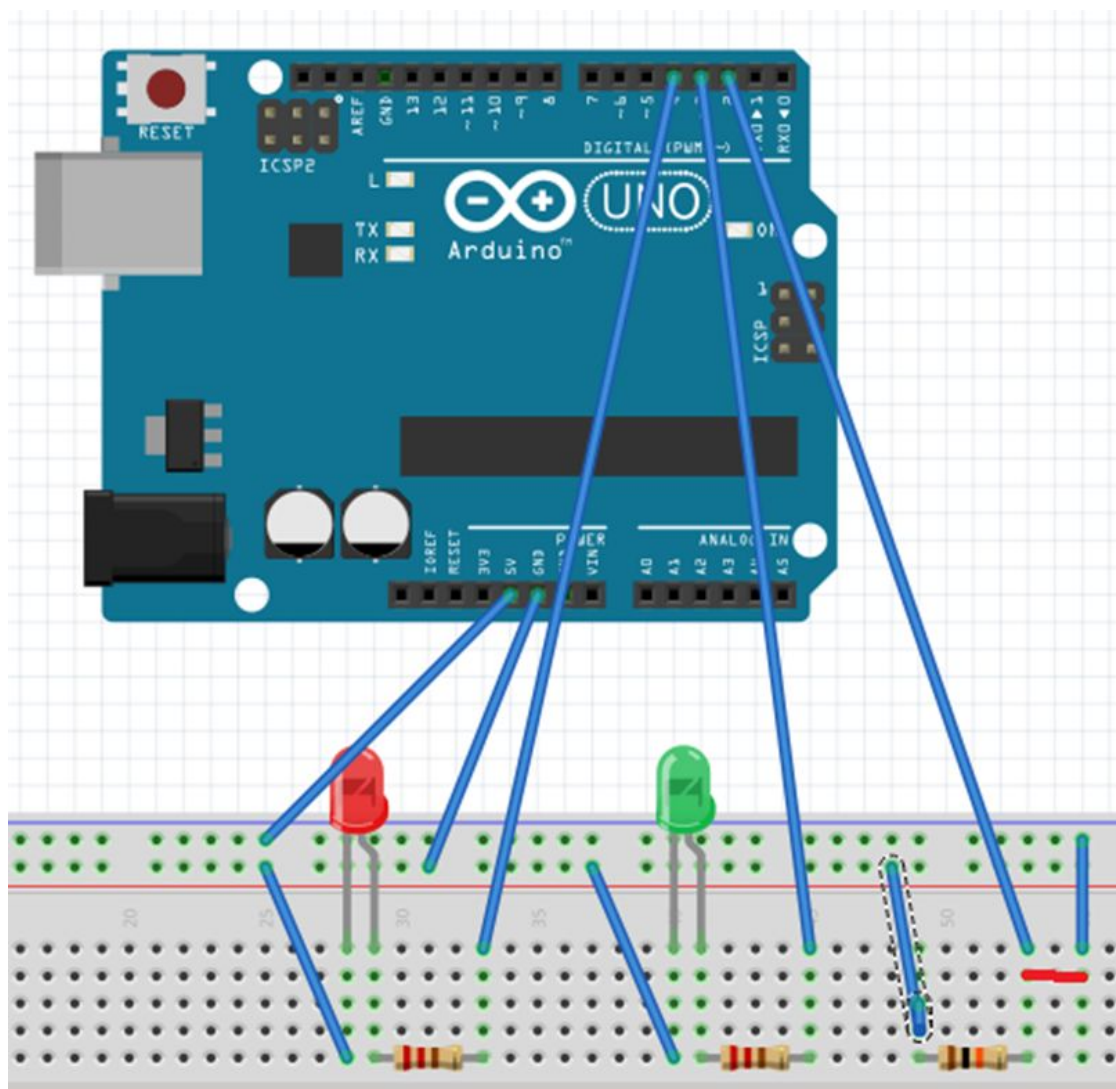


Figura 60 - Atividade 7 – Esquema alternativo da placa física. (Fio vermelho no lugar de um botão)

ATIVIDADE_008 – Faremos uma buzina utilizando a porta digital 2 como entrada e a porta digital 3 como saída. Se a porta 2 receber 5 V (ou seja, HIGH), um buzzer ativo tocará ligado na porta 3.

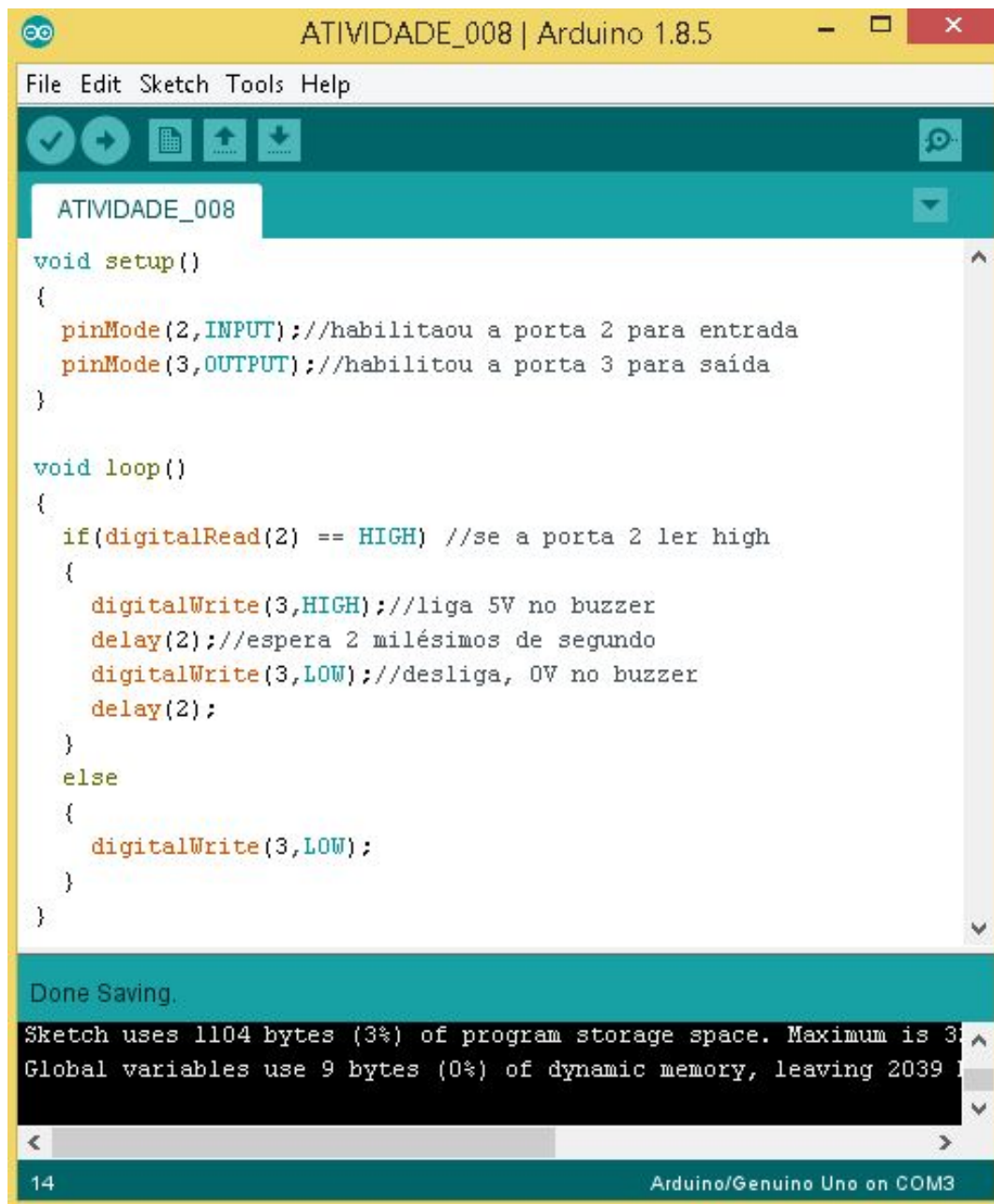


Figura 61 - ATIVIDADE 8 - Código da buzina

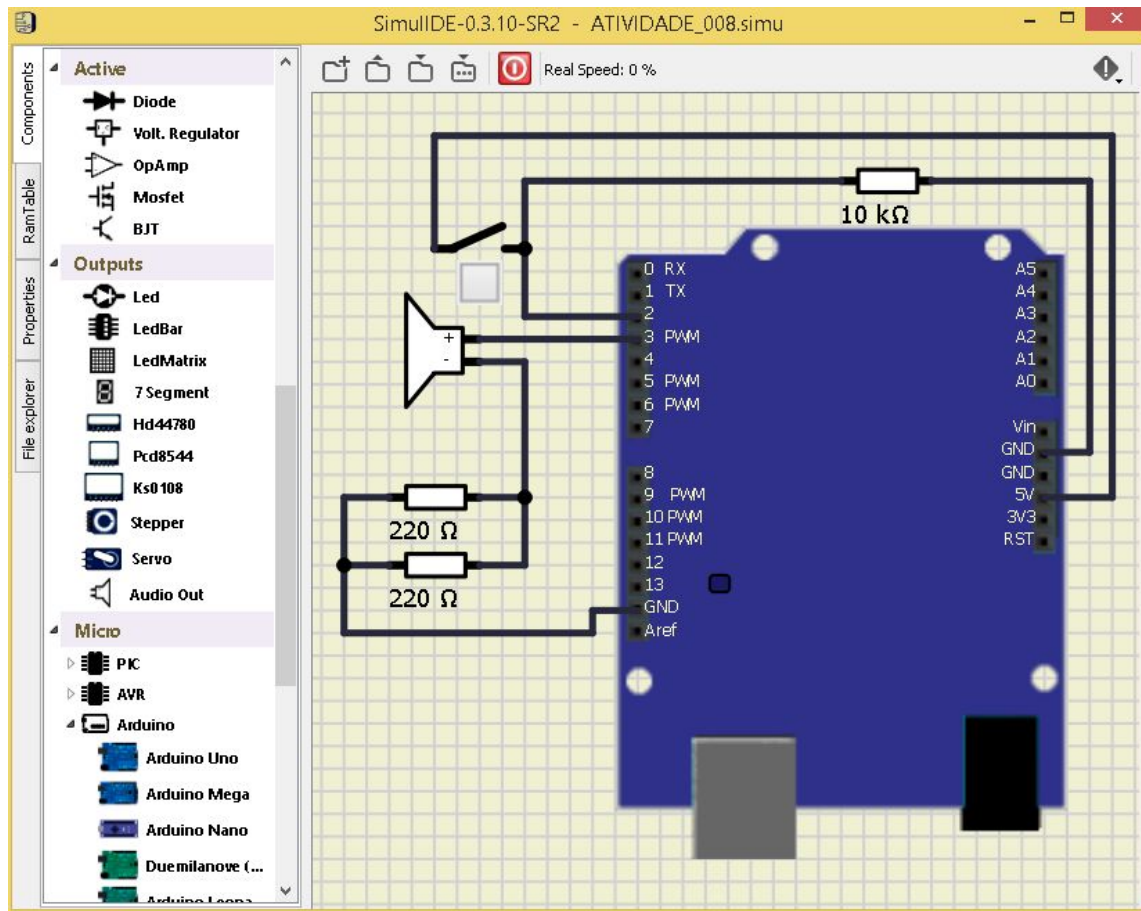


Figura 62 - Atividade 8 – Execução no SIMULIDE

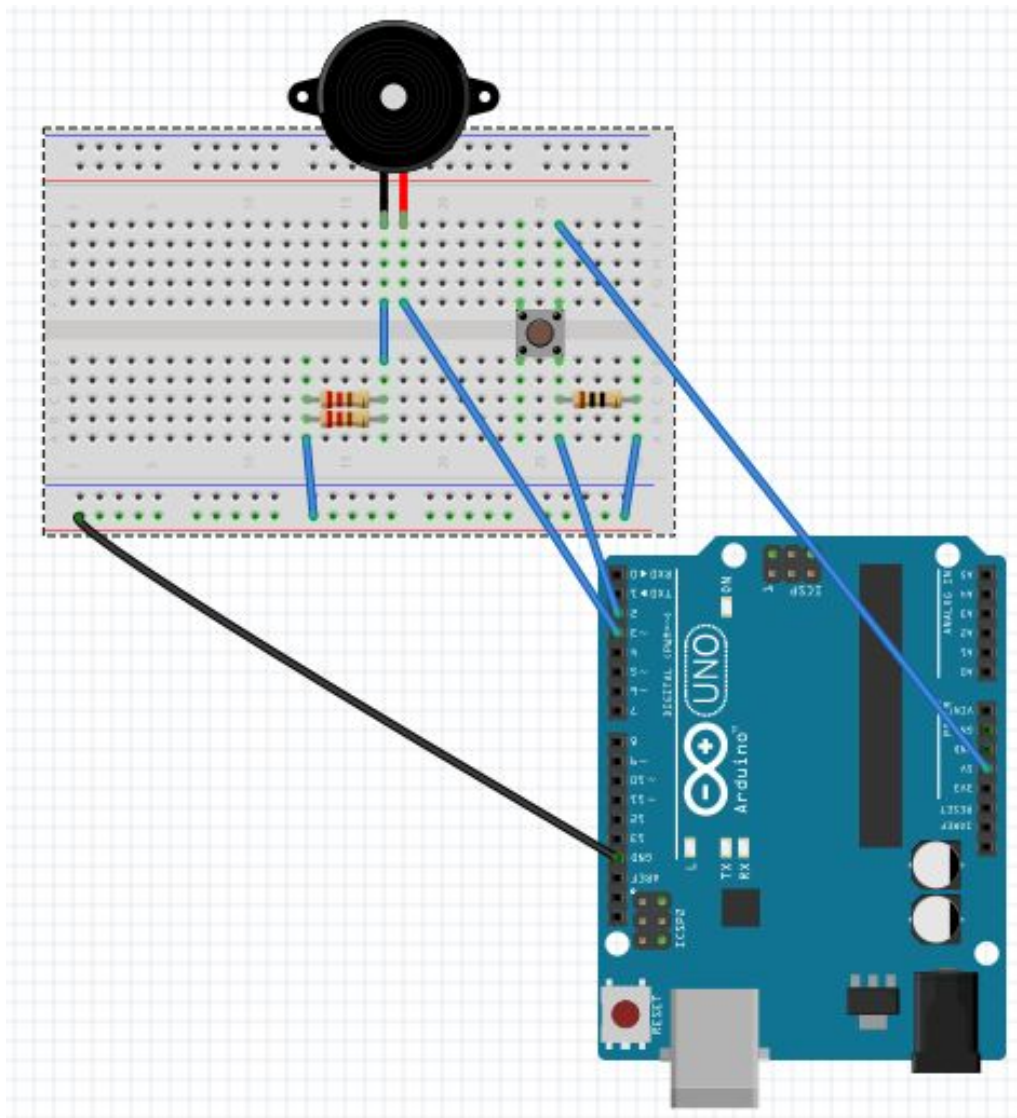


Figura 63 - Atividade 8 – Esquema da placa física

ENTRADA (OU LEITURA) EM PORTA ANALÓGICA

A placa UNO consegue ler valores entre 0 e 1023 em uma porta analógica. O comando para a leitura analógica é o **analogRead(número_da_porta)**.

Mas note que a tensão na porta é um valor entre 0 v e 5 v, que então é convertido por Regra de Três para um valor inteiro entre 0 e 1023.

HORA DA PRÁTICA:

ATIVIDADE_009 – Fazer um teste utilizando a porta analógica 0 como entrada e as portas digitais 11, 12 e 13 como saída. Se a porta analógica 0 retornar um valor entre 0 e 340, o led da porta 11 acenderá, se a porta analógica 0 retornar um valor entre 340 e 680, os leds das portas 11 e 12 acenderão, para valores maiores, os três leds acenderão. Para tanto utilizaremos um potenciômetro de 1k ohm.

Obs. como potenciômetros não costumam vir no kit básico, realizaremos este projeto apenas no SIMULIDE.

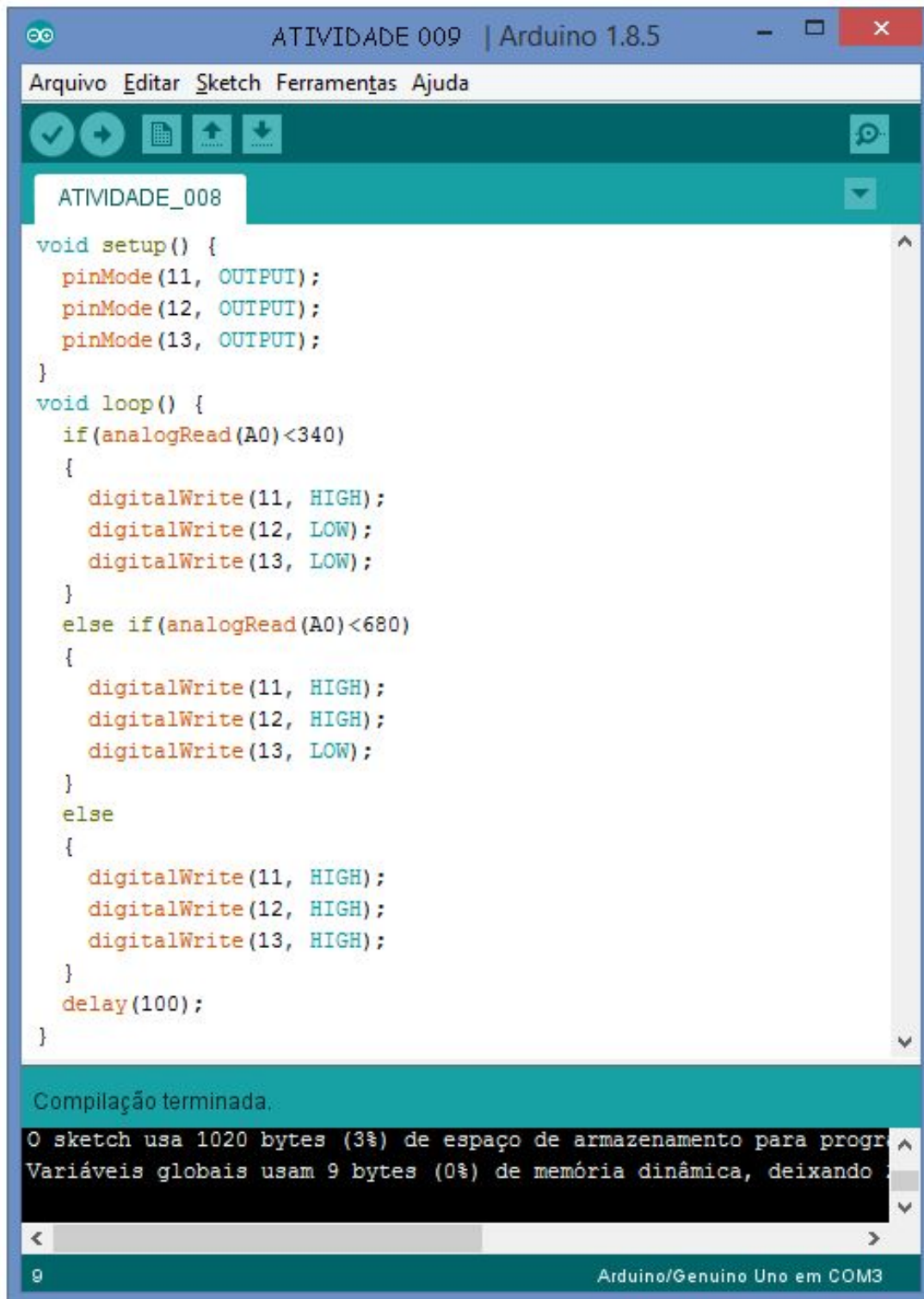


Figura 64 - Atividade 9 – Código utilizando if e entrada de uma porta analógica

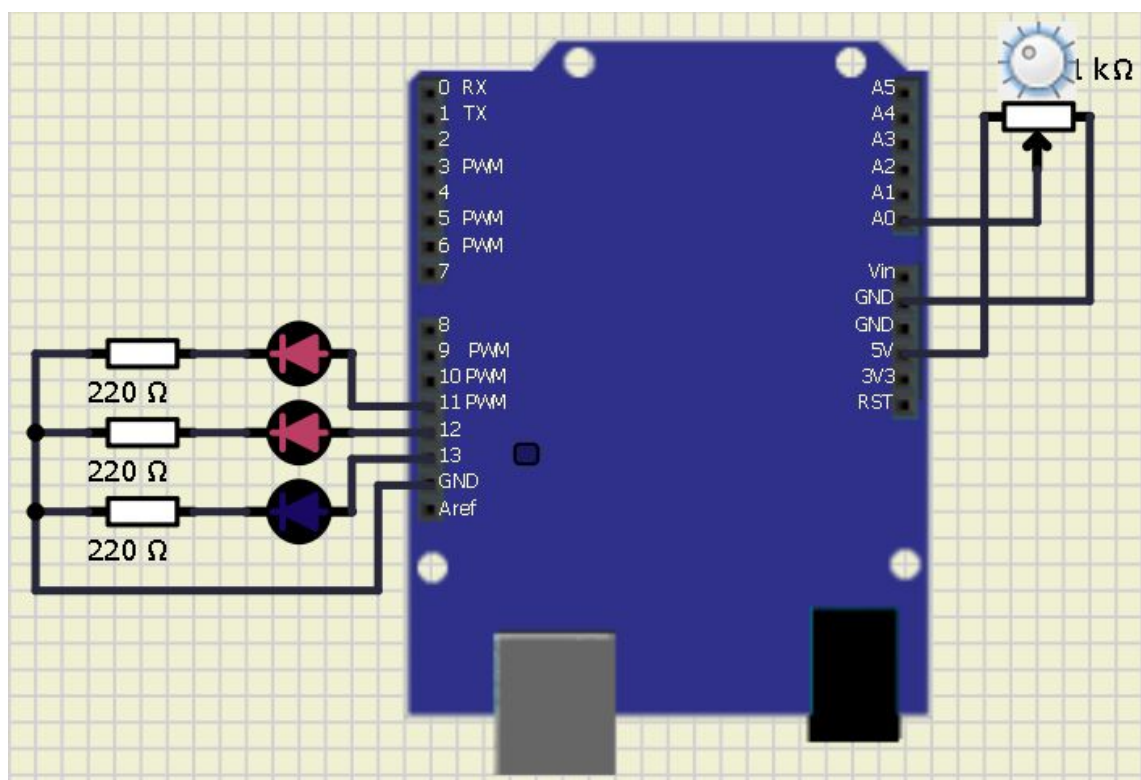


Figura 65 - Atividade 9 – Execução no SIMULIDE

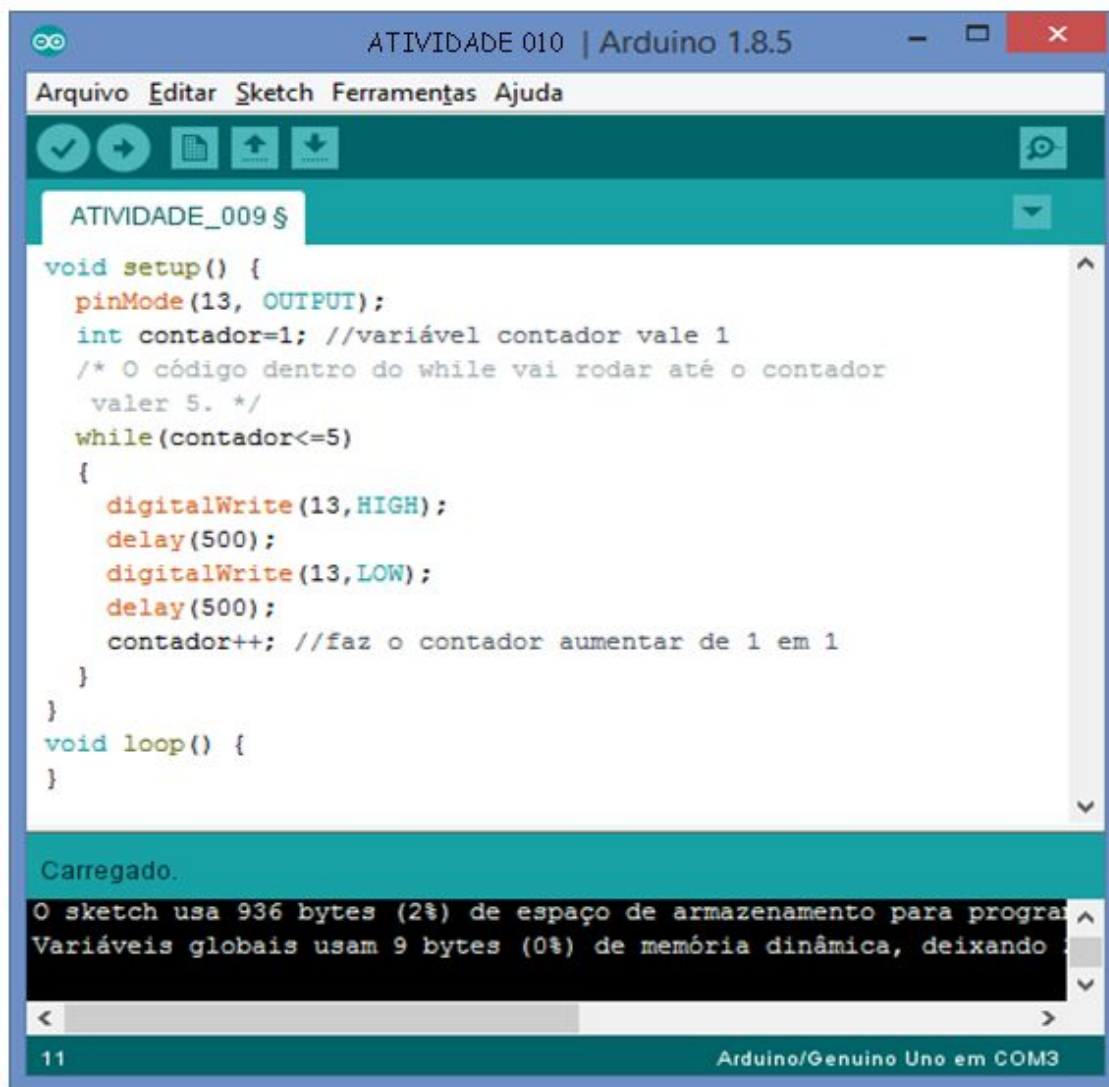
ESTRUTURAS DE REPETIÇÃO

As estruturas de repetição fazem um teste lógico e executam um trecho de código, depois efetuam novamente o teste lógico e executam o trecho de código e assim sucessivamente até que o teste lógico falhe, então a execução da estrutura de repetição termina.

Aqui neste livro apresentarei a estrutura de repetição chamada ‘while’. Para utilizar o ‘while’ é necessário declarar uma variável inteira antes (variável de contagem do número de vezes que o código dentro do while será executado) e colocar a variável seguida de ‘++’ na última linha dentro do ‘while’.

HORA DA PRÁTICA:

ATIVIDADE_010 – Fazer o led interno piscar 5 vezes com intervalo de meio segundo.



The screenshot shows the Arduino IDE interface with the title bar 'ATIVIDADE 010 | Arduino 1.8.5'. The menu bar includes 'Arquivo', 'Editar', 'Sketch', 'Ferramentas', and 'Ajuda'. The toolbar contains icons for saving, running, uploading, and downloading. The file name is 'ATIVIDADE_009 \$'. The code in the editor is as follows:

```
void setup() {  
  pinMode(13, OUTPUT);  
  int contador=1; //variável contador vale 1  
  /* O código dentro do while vai rodar até o contador  
   valer 5. */  
  while(contador<=5)  
  {  
    digitalWrite(13,HIGH);  
    delay(500);  
    digitalWrite(13,LOW);  
    delay(500);  
    contador++; //faz o contador aumentar de 1 em 1  
  }  
}  
void loop() {  
}
```

Below the code editor, the status bar shows 'Carregado.' and memory usage information: 'O sketch usa 936 bytes (2%) de espaço de armazenamento para programa' and 'Variáveis globais usam 9 bytes (0%) de memória dinâmica, deixando...'. The bottom status bar indicates '11' and 'Arduino/Genuino Uno em COM3'.

Figura 66 - ATIVIDADE 10 - Código utilizando a estrutura de repetição 'while'

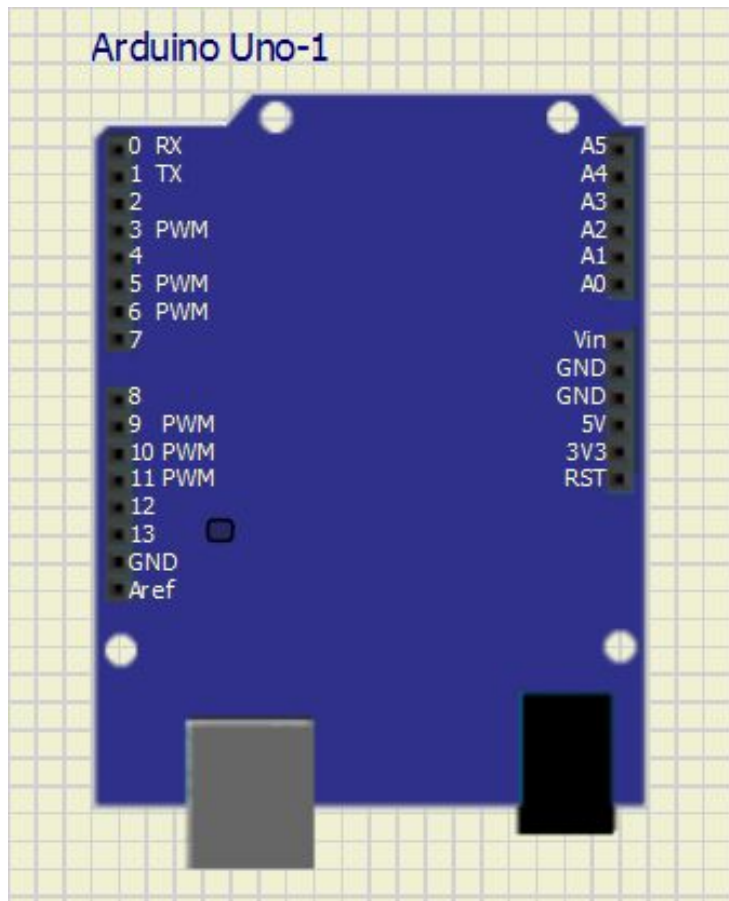


Figura 67 - Atividade 10 – Execução no SIMULIDE

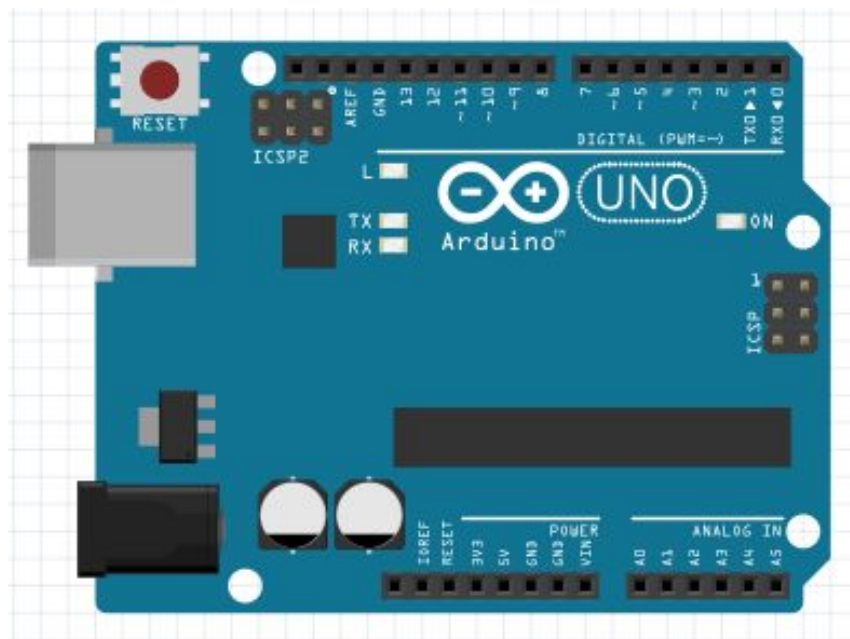


Figura 68 - Atividade 10 – Esquema da placa física

O DISPLAY DE 7 SEGMENTOS

HORA DA PRÁTICA:

ATIVIDADE_011 – Agora vamos escrever um código para piscar cada segmento de um display de 7 segmentos. Para tanto faremos inicialmente de uma forma longa (sem while) e depois com uma forma curta (com while).

Obs. como displays de 7 segmentos não costumam vir no kit básico, realizaremos este projeto apenas no SIMULIDE.

Obs2: os displays de 7 segmentos podem ser do tipo positivo ou do tipo negativo. No SIMULIDE é conectado o terminal genérico do display no GND da placa UNO e 'digitalWrite(x,HIGH)' faz com que os segmentos se acendam. Mas no meu kit (avançado), o terminal genérico do display é ligado no 5V e 'digitalWrite(x,LOW)' faz o segmento acender.

CÓDIGO LONGO

```
void setup() {  
  pinMode(0, OUTPUT);  
  pinMode(1, OUTPUT);  
  pinMode(2, OUTPUT);  
  pinMode(3, OUTPUT);  
  pinMode(4, OUTPUT);  
  pinMode(5, OUTPUT);  
  pinMode(6, OUTPUT);  
  pinMode(7, OUTPUT);  
}
```

```
void loop() {  
  digitalWrite(0,HIGH);  
  delay(500);
```

```
digitalWrite(0,LOW);  
delay(500);  
digitalWrite(1,HIGH);  
delay(500);  
digitalWrite(1,LOW);  
delay(500);digitalWrite(2,HIGH);  
delay(500);  
digitalWrite(2,LOW);  
delay(500);digitalWrite(3,HIGH);  
delay(500);  
digitalWrite(3,LOW);  
delay(500);digitalWrite(4,HIGH);  
delay(500);  
digitalWrite(4,LOW);  
delay(500);digitalWrite(5,HIGH);  
delay(500);  
digitalWrite(5,LOW);  
delay(500);digitalWrite(6,HIGH);  
delay(500);  
digitalWrite(6,LOW);  
delay(500);digitalWrite(7,HIGH);  
delay(500);  
digitalWrite(7,LOW);  
delay(500);  
}
```

CÓDIGO CURTO

```
int porta = 2;  
void setup() {
```

```
while(porta <=9)  
{  
    pinMode(porta, OUTPUT);  
    porta++;  
}  
}  
void loop() {  
    if(porta==10)  
        porta=2;  
    digitalWrite(porta, HIGH);  
    delay(500);  
    digitalWrite(porta, LOW);  
    delay(500);  
    porta++;  
}
```

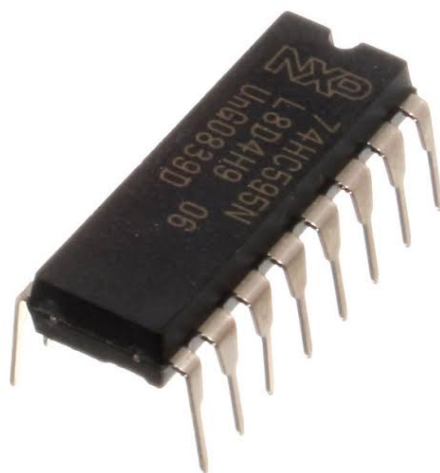



Figura 70 - Circuito integrado 74HC595

CÓDIGO PARA A PLACA FÍSICA REAL

//meu display liga no 5v para display que liga no gnd

//trocar os 0 por 1 e os 1 por 0 na função abaixo

byte seven_seg_digits[16] = { B00000011, // = 0

B10011111, // = 1

B00100101, // = 2

B00001101, // = 3

B10011001, // = 4

B01001001, // = 5

B01000001, // = 6

B00011111, // = 7

B00000001, // = 8

B00011001, // = 9

B00010001, // = a

B11000001, // = b

B01100011, // = c

```

        B10000101, // = d
        B01100001, // = e
        B01110001  // = f
    };

    // conectar a porta 8 do arduino no ST_CP do 74HC595 (pin 12,latch
pin)
    int latchPin = 8;
    // conectar a porta 12 do arduino no SH_CP of 74HC595 (pin 11, clock
pin)
    int clockPin = 12;
    // conectar a porta 11 do arduino no DS of 74HC595 (pin 14)
    int dataPin = 11;

    void setup() {
        // marca as 3 portas do Arduino como saída
        pinMode(latchPin, OUTPUT);
        pinMode(clockPin, OUTPUT);
        pinMode(dataPin, OUTPUT);
    }

    // função para mostrar 1 dígito no display
    void sevenSegWrite(byte digit) {
        // configurações iniciais antes do display mostrar 1 núm.
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST, seven_seg_digits[digit]);
        digitalWrite(latchPin, HIGH);
    }

    //escreve de f até a depois conta de 9 p/ 0

```

```

void loop() {
  for (byte digit = 16; digit > 0; --digit) {
    delay(1000);
    sevenSegWrite(digit - 1);
  }

  delay(4000); //espera 4s
}

```

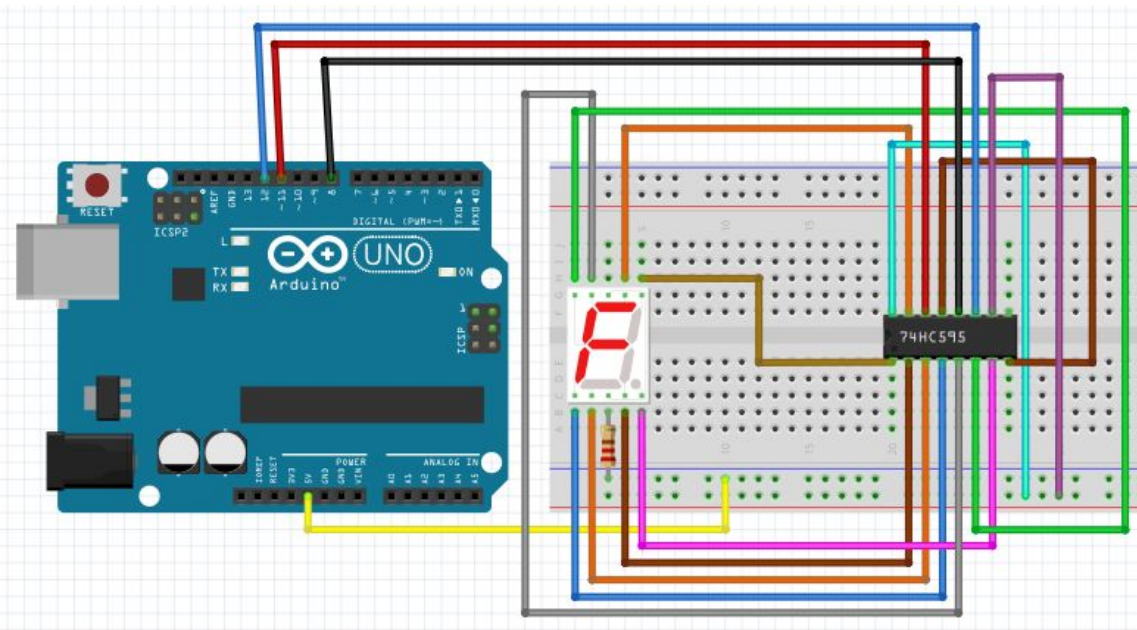


Figura 71 - Esquema da placa física

ATIVIDADE_011_C – Agora vamos escrever um código para contar de 0 a 9 utilizando o display de 7 segmentos.

O código é relativamente longo e para ficar mais organizado, eu decidi programar cada número de 0 a 9 dentro de uma função (função é um trecho de código que pode ficar localizado fora das funções ‘void setup()’ e ‘void loop()’, funções são muito utilizadas por programadores experientes para organizar e tornar o código mais legível e mais fácil de dar manutenção).

CÓDIGO PARA O SIMULIDE.

```
void setup() {  
    pinMode(2, OUTPUT);  
    pinMode(3, OUTPUT);  
    pinMode(4, OUTPUT);  
    pinMode(5, OUTPUT);  
    pinMode(6, OUTPUT);  
    pinMode(7, OUTPUT);  
    pinMode(8, OUTPUT);  
    pinMode(9, OUTPUT);  
}
```

```
void loop() {  
    int tempo = 1000; //TEMPO DE 1 SEGUNDO PARA TODAS AS  
FUNÇÕES  
    todos_segmentos(tempo, LOW); //APAGA TODOS OS  
SEGMENTOS  
    um(tempo, HIGH); //FUNÇÃO 1, ACENDE OS SEGMENTOS DO 1  
    um(tempo, LOW); //FUNÇÃO 1, APAGA OS SEGMENTOS DO 1  
    dois(tempo, HIGH);  
    dois(tempo, LOW);  
    três(tempo, HIGH);  
    três(tempo, LOW);  
    quatro(tempo, HIGH);  
    quatro(tempo, LOW);  
    cinco(tempo, HIGH);  
    cinco(tempo, LOW);  
    seis(tempo, HIGH);  
    seis(tempo, LOW);  
    sete(tempo, HIGH);  
    sete(tempo, LOW);  
}
```

```
oito(tempo, HIGH);  
oito(tempo, LOW);  
nove(tempo, HIGH);  
nove(tempo, LOW);  
zero(tempo, HIGH);  
zero(tempo, LOW);  
}
```

```
void um(int tempo, int estado)//FUNÇÃO PARA ESCREVER 1  
{  
    digitalWrite(2,estado);  
    digitalWrite(8,estado);  
    delay(tempo);  
}
```

```
void dois(int tempo, int estado)//FUNÇÃO PARA ESCREVER 2  
{  
    digitalWrite(3,estado);  
    digitalWrite(4,estado);  
    digitalWrite(6,estado);  
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void três(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(6,estado);
```

```
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void quatro(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(5,estado);  
    digitalWrite(6,estado);  
    digitalWrite(8,estado);  
    delay(tempo);  
}
```

```
void cinco(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(5,estado);  
    digitalWrite(6,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void seis(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(4,estado);  
    digitalWrite(5,estado);  
}
```

```
    digitalWrite(6,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void sete(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void oito(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(4,estado);  
    digitalWrite(5,estado);  
    digitalWrite(6,estado);  
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void nove(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(5,estado);
```

```
    digitalWrite(6,estado);  
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void zero(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(4,estado);  
    digitalWrite(5,estado);  
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

```
void todos_segmentos(int tempo, int estado)  
{  
    digitalWrite(2,estado);  
    digitalWrite(3,estado);  
    digitalWrite(4,estado);  
    digitalWrite(5,estado);  
    digitalWrite(6,estado);  
    digitalWrite(7,estado);  
    digitalWrite(8,estado);  
    digitalWrite(9,estado);  
    delay(tempo);  
}
```

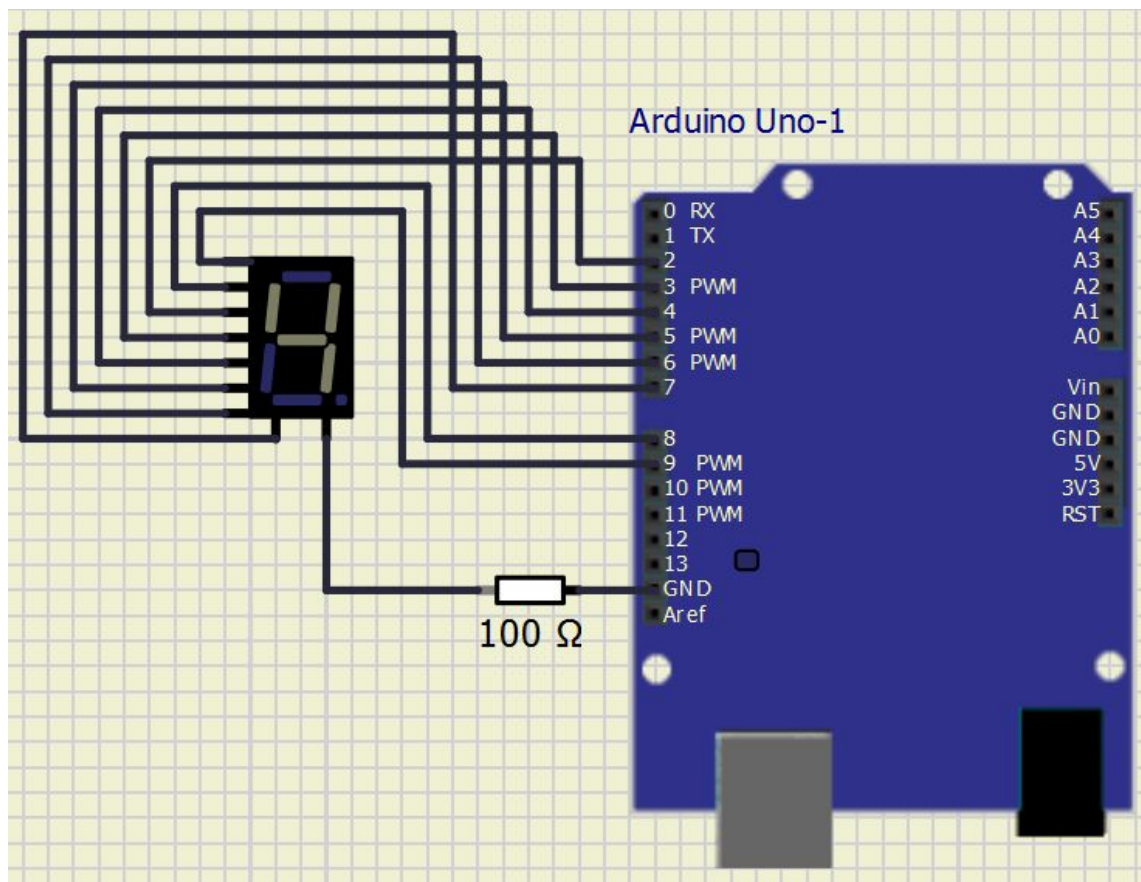


Figura 72 - Atividade 11 C – Execução no SIMULIDE

No caso do meu display de 7 segmentos, ele tem o polo comum ligado em 5V e cada terminal ligado em uma porta digital do ARDUINO em estado LOW (contrário do display de 7 segmentos padrão do SIMULIDE).

Abaixo apresento o esquema de ligação para um display (anodo) com polo comum ligado em 5 V e cada terminal ligado em uma porta digital da placa UNO em estado LOW. Obs. os números que aparecem abaixo, de 2 a 9 correspondem às portas digitais de 2 a 9.

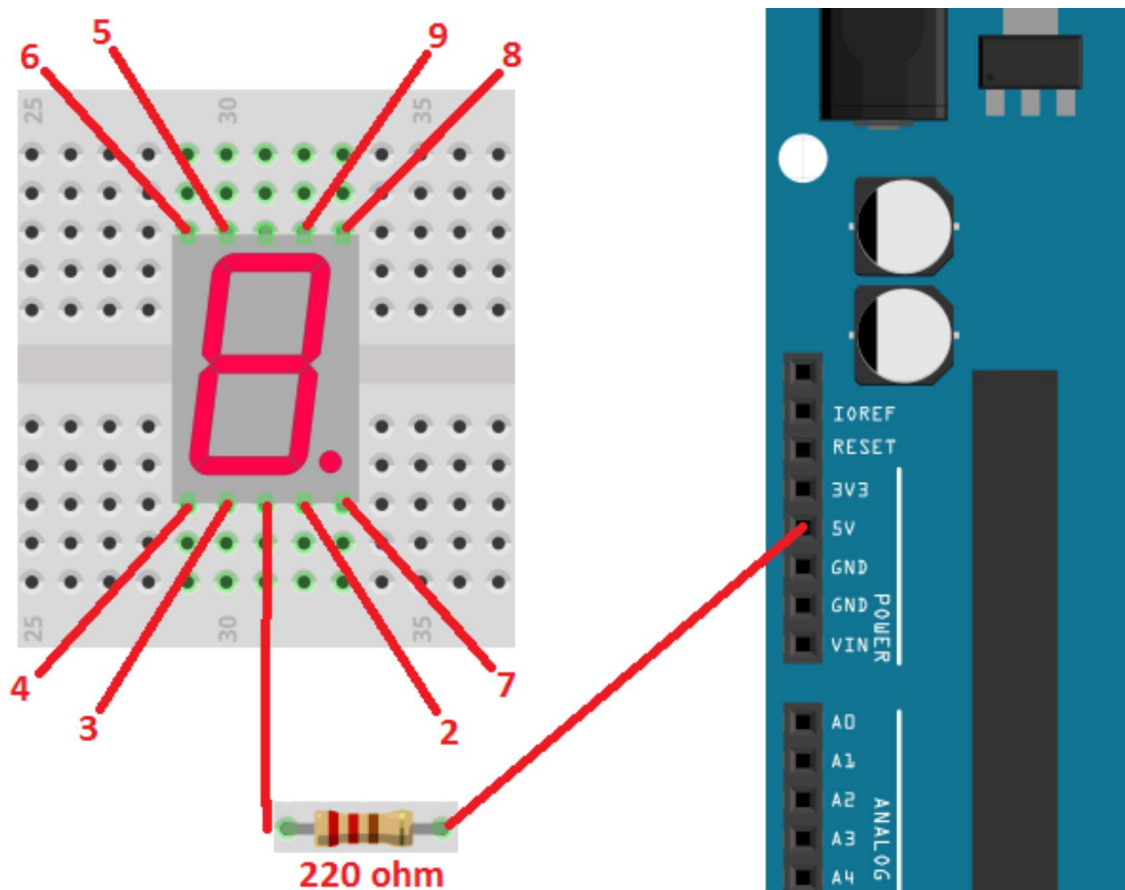


Figura 73 - Atividade 11 C – Esquema de ligação na placa física

Obs. o código apresentado anteriormente não funciona no esquema acima por conta da polaridade. Veja o código da placa física com o nome 'ATIVIDADE_011_FISICO'.

GERAR NÚMEROS (PSEUDO) ALEATÓRIOS

HORA DA PRÁTICA:

ATIVIDADE_012 – Agora vamos escrever um código para simular um dado utilizando o display de 7 segmentos.

Nesta atividade, tanto o esquema do SIMULIDE quanto o esquema físico são idênticos aos das atividades 11. A única diferença é o código, que agora utiliza

o comando ‘ **randomSeed()** ’ para gerar uma sequência de números pseudoaleatórios, o comando **random(x,y)**; para limitar o intervalo gerado que vai do x até 1 a menos que o y.

Exemplo: para sortear números entre 1 e 6 podemos utilizar ‘random(1,7)’ e para que cada sequência gerada seja diferente parecendo aleatório, utilizamos **analogRead(0)**; dentro do comando randomSeed(); porque a porta analógica 0 tem flutuações de tensão, então fica muito bom utilizar as flutuações de tensão para iniciar a geração da sequência.

Obs. no SIMULIDE, dependo da versão, a sequência gerada é sempre a mesma (por se tratar de uma simulação ainda em desenvolvimento).

Obs2: os números gerados em sistemas eletrônicos não costumam ser considerados verdadeiramente aleatórios por matemáticos, então o nome mais correto é pseudoaleatório.

Ver os arquivos ATIVIDADE_012 para o SIMULIDE e ATIVIDADE_012_FISICO para a placa UNO física.

O DISPLAY DE LED 16x2

O display de led 16x2 pode ser um Hitachi HD44780 ou ainda um compatível, como por exemplo o JHD 162A. O 16x2 refere-se a 2 linhas e 16 colunas. A figura abaixo exibe um display com a numeração dos pinos.

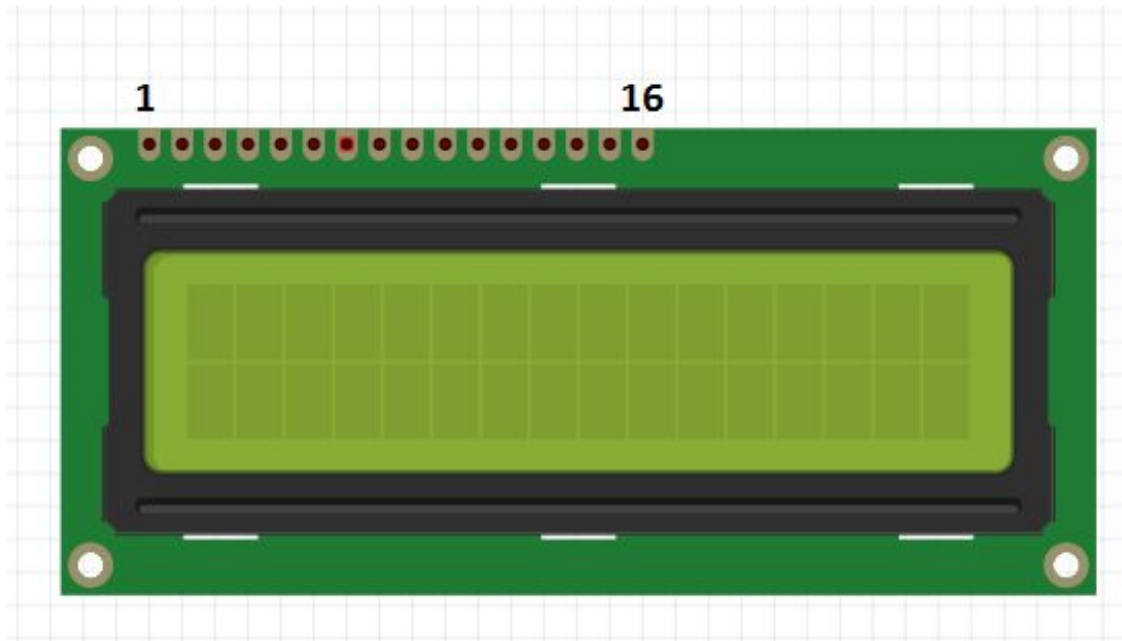


Figura 74 - Display 16x2

Pino 1: VSS, ligado no GND da placa UNO.

Pino 2: VCC (alguns chamam de VDD), ligado no 5V da placa UNO.

Pino 3: VEE (alguns chamam de V0), ligado ao pino central (saída) do potenciômetro de 10K ohm (dos outros dois terminais do potenciômetro, um deve ser ligado no 5V e o outro no GND). Este pino é responsável pela tonalidade do lcd.

Pino 4: RS, ligado na porta digital 12 da placa UNO.

Pino 5: R/W, ligado no GND da placa UNO. É responsável por habilitar a função de escrita ou de leitura do display (GND é escrita).

Pino 6: E, ligado na porta digital 11 da placa UNO.

Pino 7: DB0

Pino 8: DB1

Pino 9: DB2

Pino 10: DB3

Pino 11: DB4, ligado na porta digital 5 da placa UNO.

Pino 12: DB5, ligado na porta digital 4 da placa UNO.

Pino 13: DB6, ligado na porta digital 3 da placa UNO.

Pino 14: DB7, ligado na porta digital 2 da placa UNO.

Pino 15: LED+, responsável por acender o led do fundo do display. Se for ligado, deve ser ligado a um resistor de 220 ohm antes de ser ligado ao 5V ou ainda ser ligado ao terminal do meio de um potenciômetro de 1K ohm.

Pino 16: LED-, responsável por acender o led do fundo do display. Se for ligado, deve ser ligado no GND da placa UNO.

Nos projetos deste livro, para simplificar as ligações do display, sempre utilizo apenas os pinos de 1 a 6 e de 11 a 14, totalizando dez pinos ligados.

Obs. este modo de ligação dos pinos onde sobram quatro pinos DB sem ligação é chamado de modo 4 bits, é muito utilizado no ARDUINO para economizar portas. Em outros circuitos, pode-se utilizar o modo 8 bits, que utiliza todos os pinos do display.

COMANDOS DO DISPLAY 16x2

Para utilizar o display, primeiramente devemos incluir no código a linha **#include <LiquidCrystal.h>** que habilita o uso da biblioteca (conjunto de comandos feito por outro programador) chamada LiquidCrystal.h, que possui os comandos necessários para a manipulação do display 16x2.

O comando **LiquidCrystal lcd(rs, e, db4, db5, db6, db7);** associa as portas com cada pino do display.

O comando **lcd.begin(16, 2);** define o tipo de display.

O comando **lcd.print("mensagem");** passa para o display a mensagem que vai ser exibida.

O comando **lcd.setCursor(coluna, linha);** passa para o display onde a mensagem que vai ser exibida deve começar a ser escrita.

HORA DA PRÁTICA:

ATIVIDADE_013 – Agora vamos desenvolver um código para escrever ‘LIVRO DE ARDUINO’ utilizando o display 16x2.

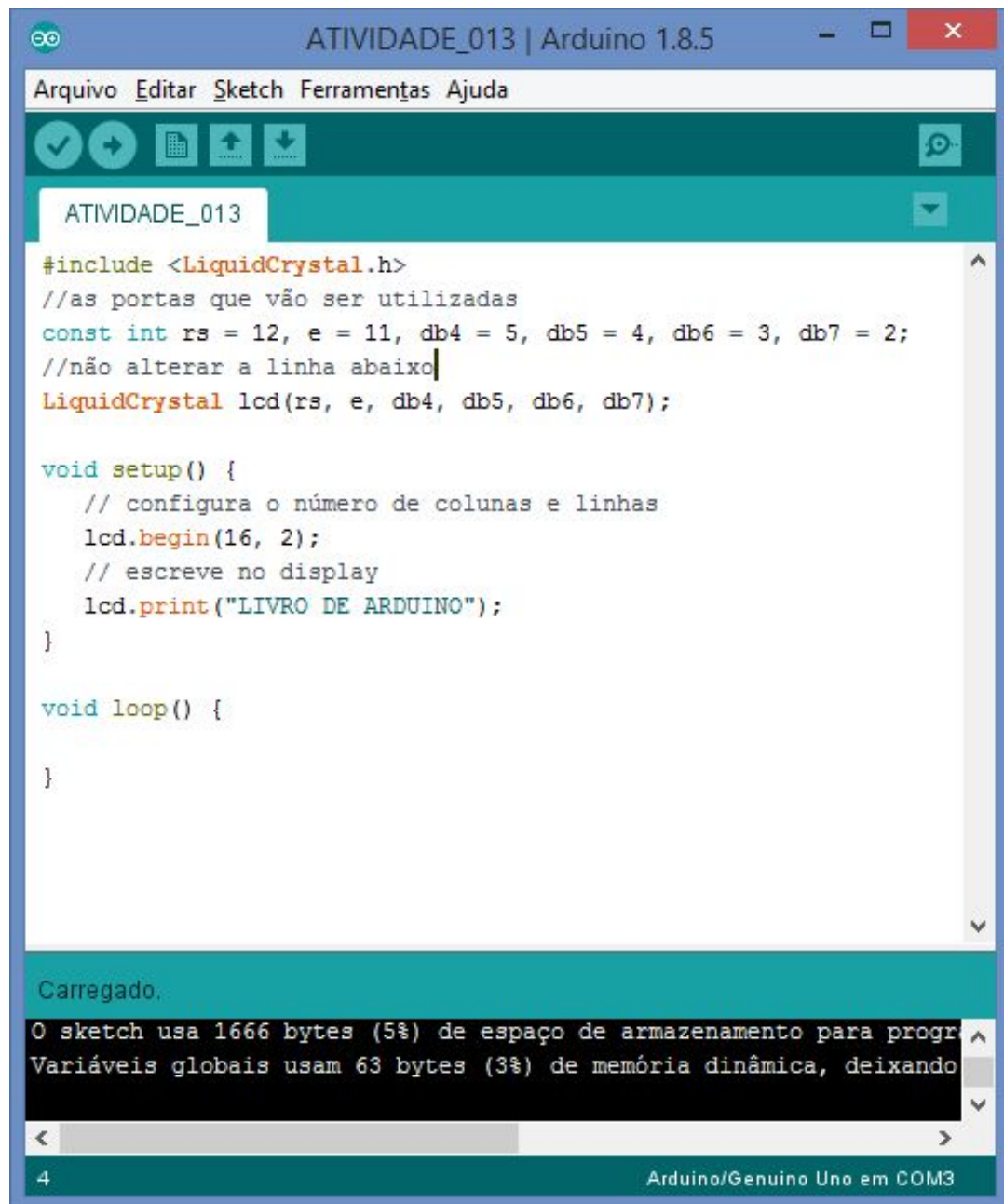


Figura 75 - ATIVIDADE 13 - Escrita no display 16x2

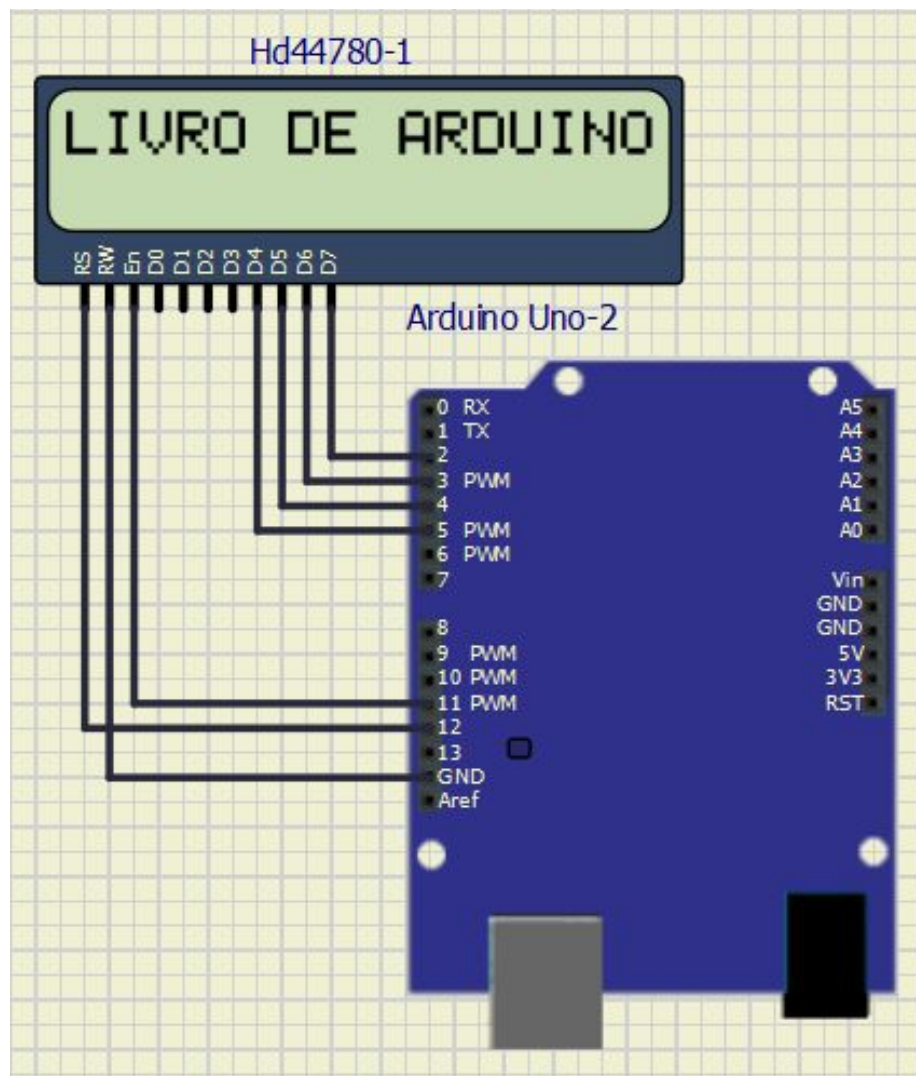


Figura 76 - Atividade 13 – Execução no SIMULIDE

Obs. Perceba que o display 16x2 no SIMULIDE é simplificado e não possui os pinos VSS, VCC e VEE.

ESCRITA E LEITURA SERIAL

A placa UNO pode comunicar-se com o computador via porta serial (ligada pelo cabo USB que conecta a placa no computador).

Para utilizar a comunicação serial, usamos o comando **Serial.begin(9600);** que abre a porta da placa para a comunicação com o computador na taxa de 9600 bps (só para ter ideia, uma conexão de internet provê uma taxa de mais de 8000000 bps).

Depois de aberta a porta utilizamos o comando **Serial.print("mensagem");** para escrever uma linha de mensagem ou ainda **Serial.println("mensagem");** para escrever uma linha e depois ir para a linha de baixo.

Quando a intenção for escrever números lidos pela placa, então utiliza-se **Serial.write(numero);** já a leitura se dá pelo comando **Serial.read();** .

HORA DA PRÁTICA:

ATIVIDADE_014 – Agora vamos desenvolver um código para escrever ‘LIVRO DE ARDUINO’ utilizando a porta serial.

Obs. esta atividade será desenvolvida apenas na placa UNO pois o SIMULIDE ainda não simula a comunicação serial.

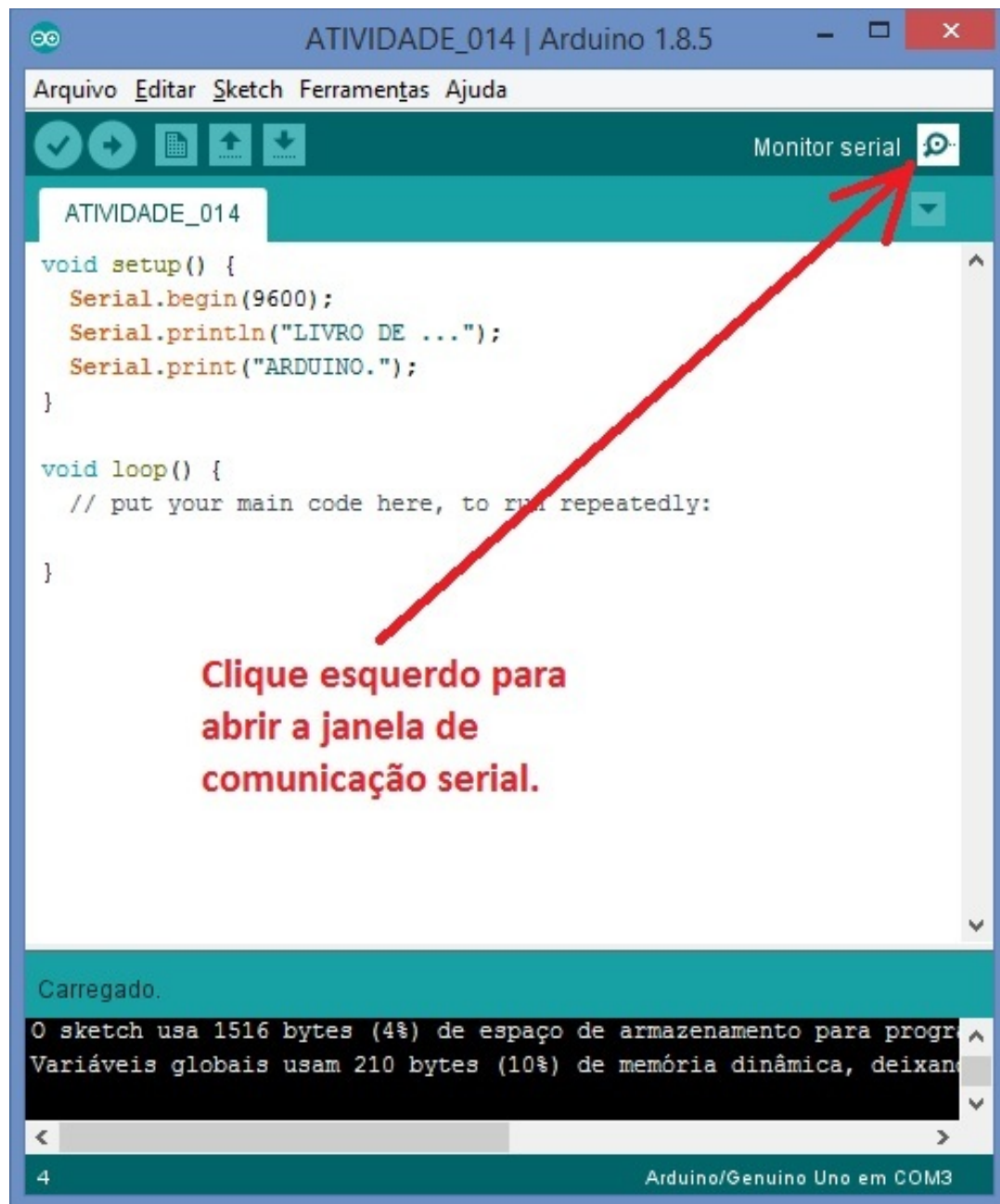


Figura 77 - Atividade 14 – Escrita utilizando a porta serial

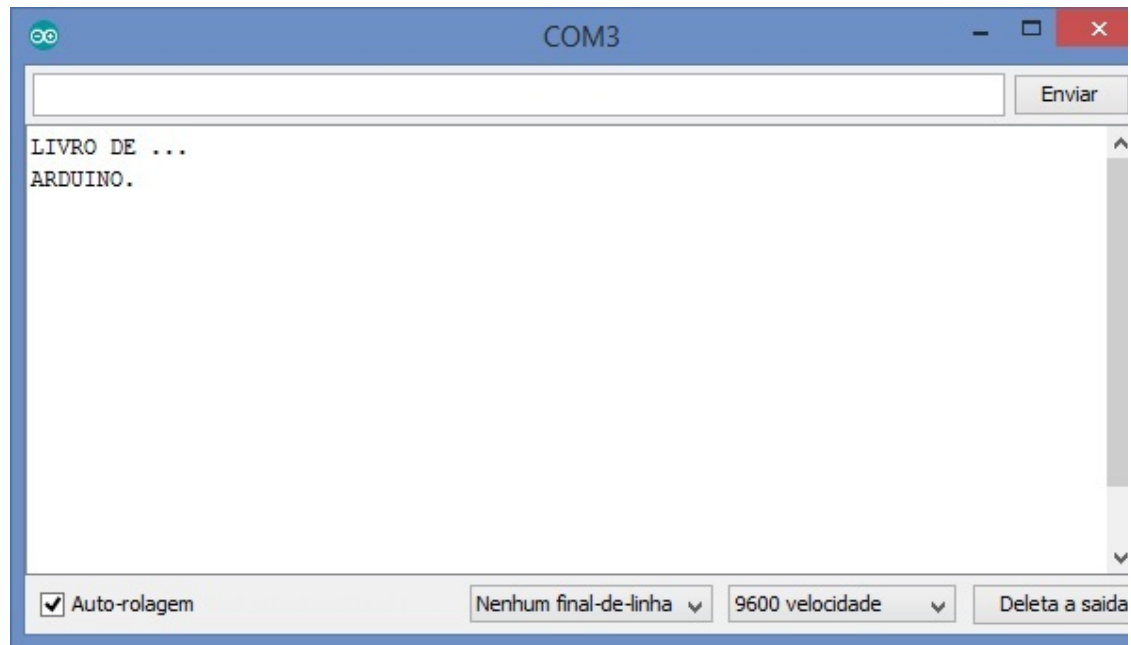


Figura 78 - Atividade 14 – Janela de comunicação serial

ATIVIDADE_015 – Agora vamos desenvolver um código onde vamos passar um número para a placa e ela vai escrever de volta o número utilizando a porta serial.

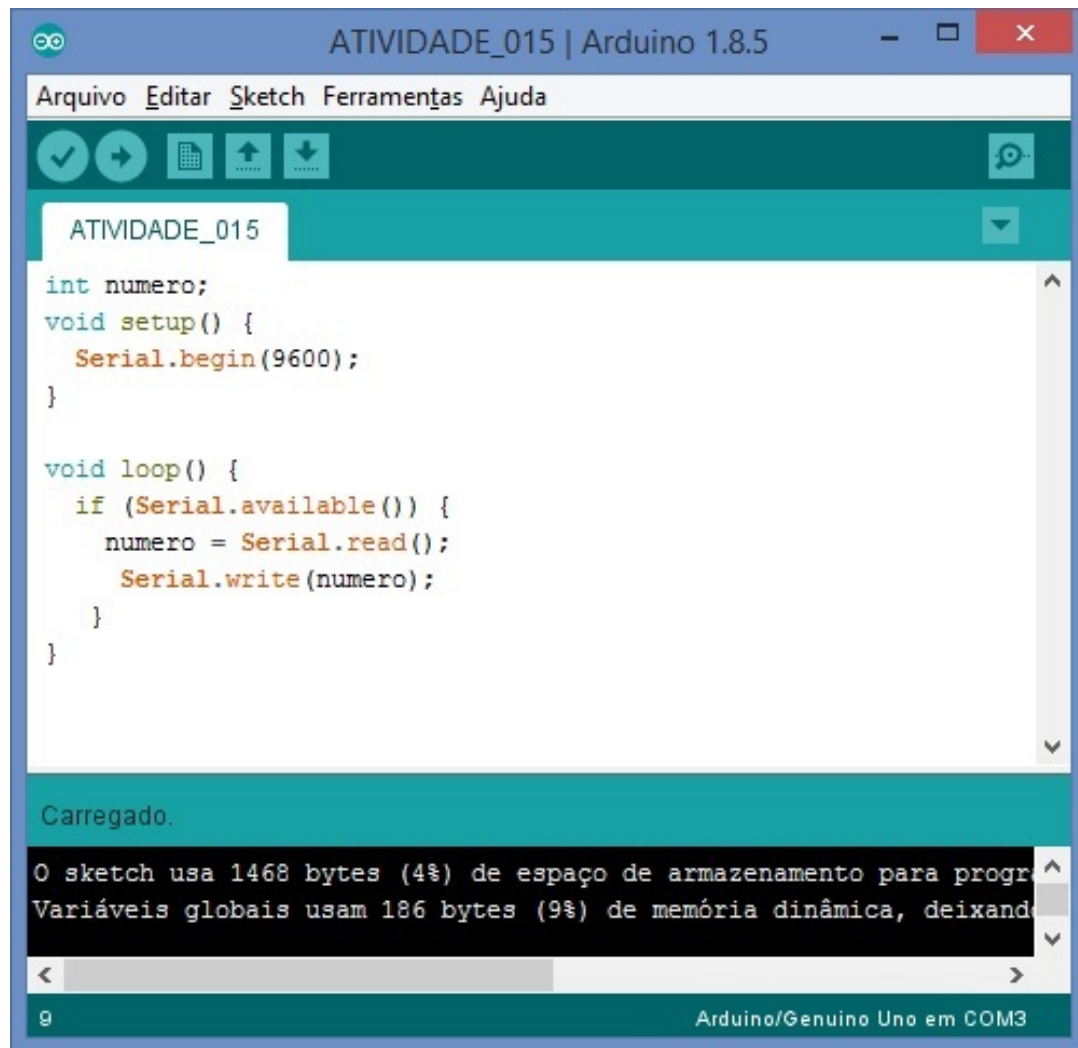


Figura 79 - Atividade 15 – Escrita e leitura utilizando a porta serial

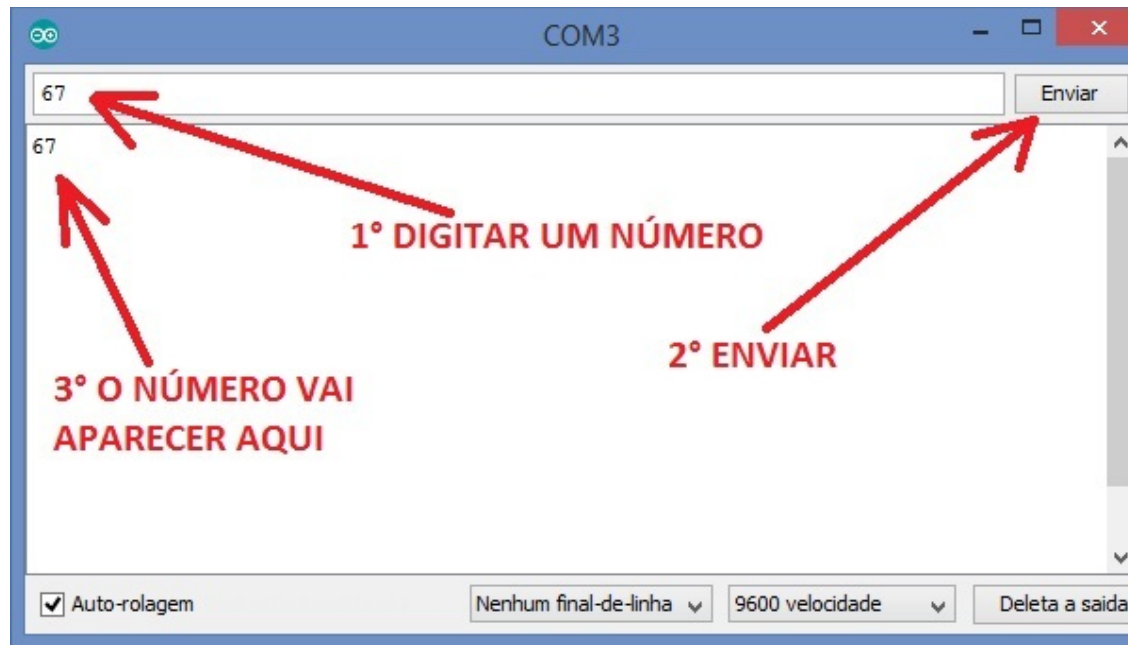


Figura 80 - Atividade 15 – Escrita e leitura utilizando a porta serial

ATIVIDADE_016 – Agora vamos desenvolver um código onde a placa vai gerar um número entre 1 e 5 e vai escrever de volta o número utilizando a porta serial.

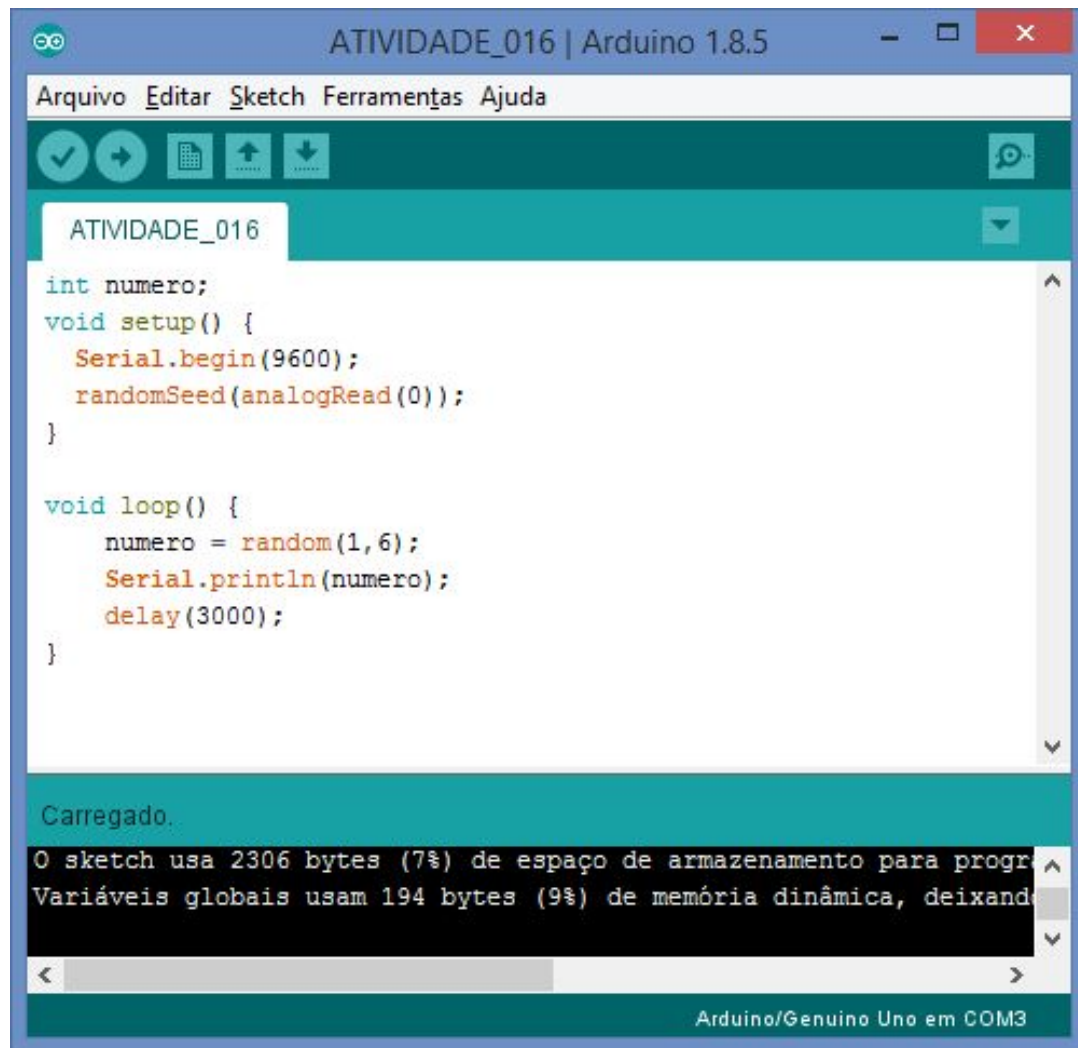


Figura 81 - Atividade 16 – Gerar um número aleatório e escrever via porta serial

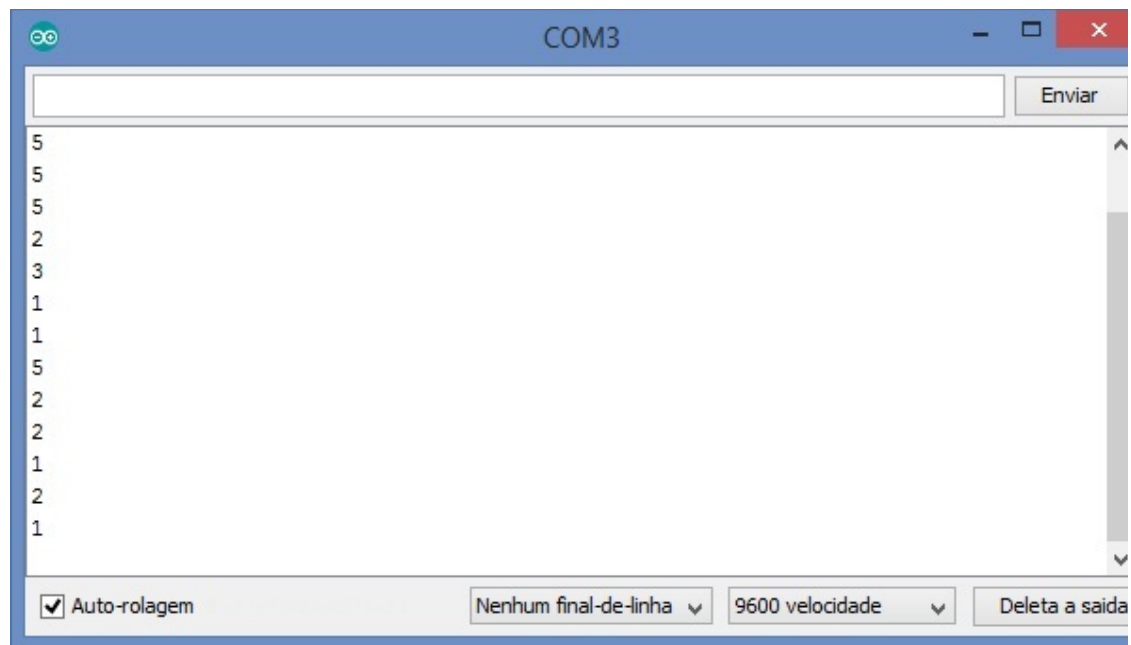


Figura 82 - Atividade 16 – Números aleatórios gerados

CONVERSÃO DE TEXTO PARA NÚMERO INTEIRO

Obs. os números lidos pela placa são interpretados como texto, então devemos converter os números lidos de texto para número inteiro pela função **atoi(valor_lido);** .

ATIVIDADE_017 – Agora vamos desenvolver um código onde a placa vai gerar um número entre 1 e 15, vai ler um número pela porta serial, vai escrever de volta o número gerado e o lido utilizando a porta serial e ainda vai acender um led amarelo quando o número sorteado for maior que o lido, vai acender um led vermelho quando o número sorteado for menor que o lido e por último vai acender um led verde se os números forem iguais. Sem dúvida, esta é a atividade mais complexa deste livro, tanto do ponto de vista da lógica.

CÓDIGO

//VARIÁVEL QUE ARMAZENA O VALOR LIDO COM ATÉ 4 CARACTERES, MAS NÃO É NÚMERO, É TEXTO

char valor_lido[4]="";

int numero_lido=0; //VARIÁVEL QUE ARMAZENA O TEXTO LIDO CONVERTIDO PARA NÚMERO

int numero_sorteado=0; //VARIÁVEL QUE ARMAZENA O VALOR SORTEADO

int i = 0; //VARIÁVEL CONTADORA, UTILIZADA NO WHILE

void setup() {

//HABILITA AS PORTAS DIGITAIS 2,3 E 4 PARA OUTPUT

pinMode(2,OUTPUT);

pinMode(3,OUTPUT);

pinMode(4,OUTPUT);

Serial.begin(9600); //HABILITA A PORTA SERIAL PARA COMUNICAR NA TAXA 9600

randomSeed(analogRead(0)); //SEMENTE PARA AJUDAR A GERAR NÚMEROS PSEUDOALEATÓRIOS

}

void loop() {

//PORTAS DIGITAIS 2,3 E 4 EM ESTADO LOW (LED APAGADO)

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

delay(500); //ESPERA MEIO SEGUNDO COM OS LEDS APAGADOS

//ROTINA DE LEITURA

while(Serial.available()){

valor_lido[i] = Serial.read();

```
    i++;  
}
```

**i = 0; //REINICIA A VARIÁVEL CONTADORA PARA QUANDO
HOVER OUTRA LEITURA**

**numero_lido=atoi(valor_lido); //CONVERTE O TEXTO valor_lido
EM NÚMERO INTEIRO numero_lido**

**numero_sorteado=random(1,16); //SORTEIA UM NÚMERO
ENTRE 1 E 15**

**//ROTINA QUE TESTA SE O NÚMERO SORTEADO É MAIOR,
IGUAL OU MENOR QUE O NÚMERO LIDO**

```
if(numero_sorteado>numero_lido)  
    digitalWrite(2,HIGH);  
else if(numero_sorteado == numero_lido)  
    digitalWrite(3,HIGH);  
else if(numero_sorteado < numero_lido)  
    digitalWrite(4,HIGH);  
else  
    digitalWrite(3,LOW);
```

**//ROTINA QUE ESCRIBE OS NÚMEROS LIDO E SORTEADO
NA PORTA SERIAL**

```
Serial.print(numero_sorteado);  
Serial.print(" - ");  
Serial.println(numero_lido);
```

```
delay(2000); //ESPERA 2 SEGUNDOS COM O LED ACESO  
}
```

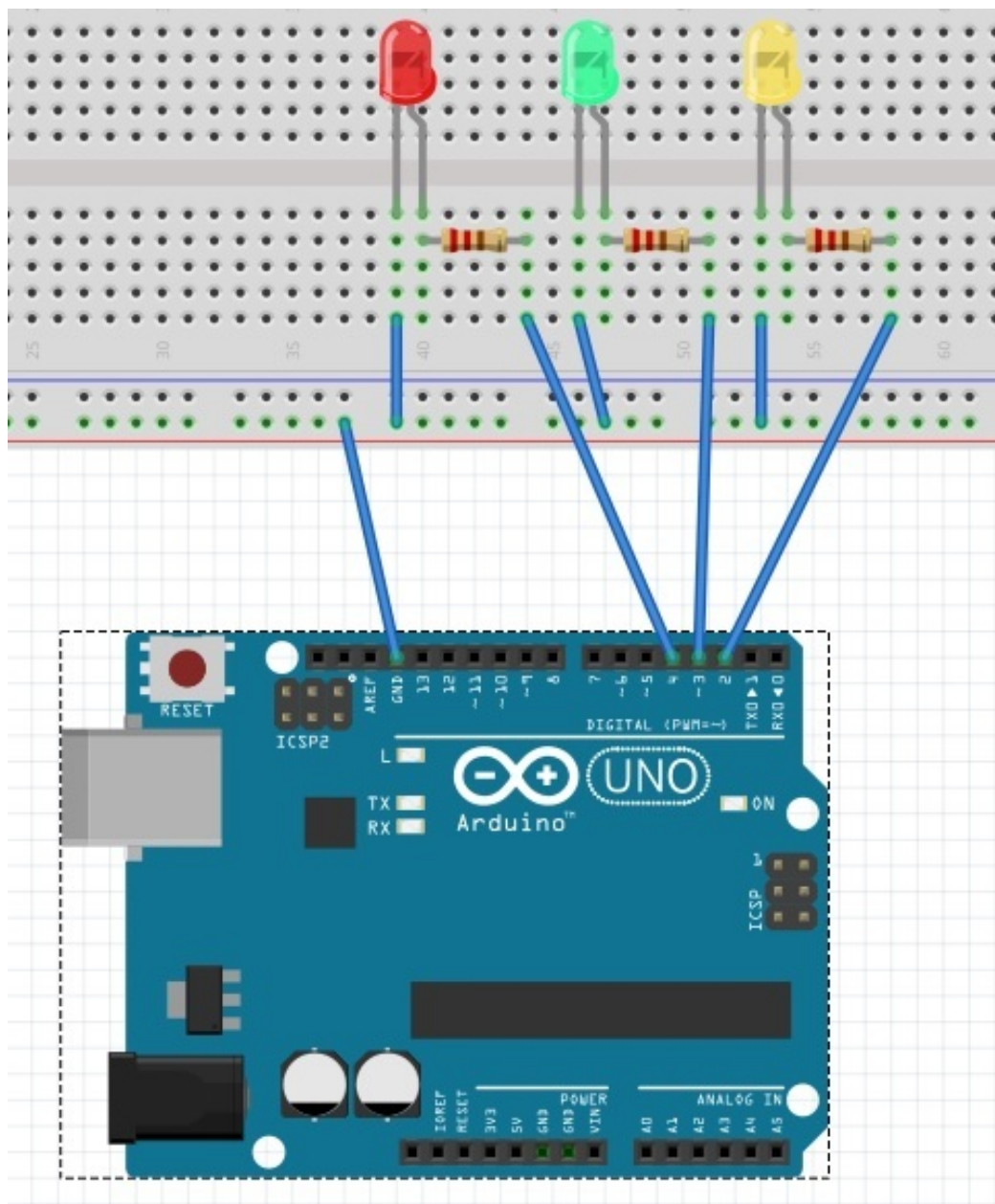


Figura 83 - Atividade 17 – Esquema de ligação na placa física

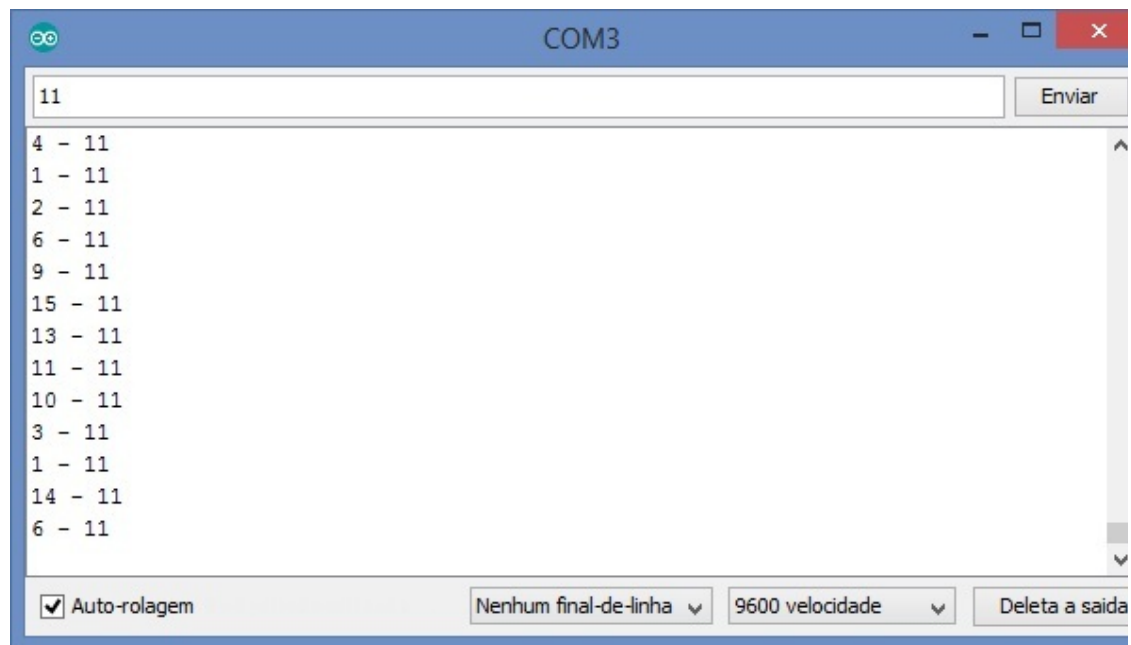


Figura 84 - Atividade 17 - Números sorteado e lido exibidos na porta serial

LEITURA COMPLEMENTAR

Parabéns por chegar até aqui. O ARDUINO é uma ferramenta muito especial por ser barata e relativamente fácil de usar e de aprender. Dentre todos os projetos que já executei com o ARDUINO, o mais interessante é a utilização pedagógica visando a aprendizagem de Lógica de Programação e conceitos básicos de Física, como tensão, corrente, resistência e luz.

Contudo as possibilidades de uso são enormes, algumas pessoas desenvolvem máquinas interessantes, brinquedos controlados por smartphone e já existe a utilização de uma placa (ARDUINO LILYPAD) em roupas.

Para lidar com a placa UNO precisamos ter 3 conhecimentos: noções de lógica de programação, de programação em C e de Eletrônica.

Para aprofundar seu conhecimento em lógica eu recomendo o livro ‘Lógica de Programação’, editora Senac, autor Gley F. C. Xavier.

Para aprofundar seu conhecimento em programação C eu recomendo o livro ‘Começando a Programar em C para leigos’, autor Dan Gookin.

Para aprofundar seu conhecimento em Eletrônica eu recomendo o livro ‘Eletrônica para leigos’, autores Cathleen Shamieh e Gordon McComb.

E obviamente, agora o leitor tem condições de compreender os projetos (básicos e com um pouco de persistência os intermediários e até os avançados) da página oficial da empresa ARDUINO (<https://www.arduino.cc/en/Tutorial/BuiltInExamples>).

CRÉDITOS DAS IMAGENS

Figura - O escorregador doa elétrons para o corpo do garoto, a carga elétrica que o garoto acumula faz seu cabelo arrepiar

[Imagem](#) de [Ken Bosma](#) por [flickr](#)

Figura - O átomo com núcleo (azul e vermelho) e eletrosfera (cinza)

[Imagem](#) de [geralt](#) por [pixabay](#).

Figura - Circuito integrado 74HC595

[Imagem](#) de [oomlout](#) por [wikimedia](#)

OceanofPDF.com