NATHAN RABELO MARTINS

UM APLICATIVO EM PYTHON PARA EXPERIMENTOS DE FÍSICA AUTOMATIZADOS COM ARDUINO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ensino de Física, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2018

NATHAN RABELO MARTINS

UM APLICATIVO EM PYTHON PARA EXPERIMENTOS DE FÍSICA AUTOMATIZADOS COM ARDUINO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ensino de Física, para obtenção do título de *Magister Scientiae*.

APROVADA: 29 de junho de 2018.

Eduardo Nery Duarte de Araújo

Orlando Pinheiro da F. Rodrigues

Ricardo dos Santos Ferreira

Ricardo Reis Cordeiro
(Orientador)

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
M386a
2018

Martins, Nathan Rabelo, 1987-
Um aplicativo em Python para experimentos de física automatizados com o Arduino / Nathan Rabelo Martins. – Viçosa, MG, 2018.
vi, 101 f. : il. (algumas color.) ; 29 cm.

Inclui apêndice.
Orientador: Ricardo Reis Cordeiro.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 24-26.

1. Física - Experimentos. 2. Python (Linguagem de programação de computador). 3. Arduino (Controlador programável). 4. Física - Estudo e ensino. I. Universidade Federal de Viçosa. Departamento de Física. Programa de Pós-Graduação em Ensino de Física. II. Título.

CDD 22. ed. 530.0724

AGRADECIMENTOS

Uma das maiores virtudes humanas é a gratidão. Para conseguir o término desta dissertação, e consequente conclusão deste mestrado, foi necessário muito apoio. Portanto o que não me falta são motivos e pessoas para agradecer.

Gostaria de agradecer a minha esposa Tânia por estar sempre ao meu lado me apoiando incessantemente e confiando no meu potencial. Agradeço também a minha pequena Anne Marie por todo seu amor, carinho, mas sobretudo pela resignação que abriu mão de minha presença me fazendo valorizar os instantes distante dela.

Ao Marcos Paulo, por todas as dicas, empréstimos, discussões e apoio. Acredito que não teria entrado nessa se não fosse por sua ajuda.

Agradeço ao meu orientador Ricardo Cordeiro por ter aceito esta empreitada junto comigo sendo, sem dúvidas, um exemplo de profissional ético e dedicado. Meu muito obrigado também a todos demais professores do programa: Alexandre, Álvaro, Orlando, Regina e Daniel, que me fizeram entender mais sobre a física, história da ciência, teorias da educação e técnicas de ensino.

Aos meus pais Anselmo e Regina por seu amor incondicional e por moldarem o meu caráter. Também a minha irmã Nayara por dividir sua vida comigo.

Agradeço a todo o campus UFV-CRP, em especial aos meus colegas de setor: Alysson, por ser o pior *personal coaching* imaginável, José Antônio, por todo o companheirismo e empatia, Helber, por seu interesse inesgotável por gambiarras e Marco Antônio, por toda ajuda dada neste período.

A todos os meus colegas de curso, sobretudo aos meus amigos Cláudio, Diego e Maurício, os quais acabamos passando mais tempo juntos. Vocês tornaram todas estas viagens mais agradáveis e fizeram menos difícil o tempo longe de casa.

Enfim, agradeço a todos que torceram por mim e que de maneira direta ou indireta contribuíram para a realização deste trabalho! Obrigado pessoal, vocês são esplendorosos!

LISTA DE ILUSTRAÇÕES

RESUMO

MARTINS, Nathan Rabelo, M.Sc., Universidade Federal de Viçosa, junho de 2018. **Um aplicativo em Python para experimentos de Física automatizados com Arduino.** Orientador: Ricardo Reis Cordeiro.

Este trabalho objetiva possibilitar a utilização da plataforma Arduino em aulas experimentais de Física, com coleta de dados automatizada, por professores de todo o Brasil, inclusive aqueles que não possuem conhecimento de programação. Para tanto, criamos um programa que facilita a relação entre usuário e Arduino, possibilitando que o mesmo seja utilizado de uma maneira prática, dinâmica e simples. Este programa, desenvolvido na linguagem Python, traz uma interface amigável possibilitando que os experimentos com Arduino, após sua montagem, sejam realizados apenas com alguns cliques e a inserção de alguns valores. Como mencionado, utilizamos para a criação do programa a linguagem Python e fizemos a conversação necessária entre ela e a plataforma do Arduino, baseada em C/C++, por meio do protocolo firmata. Englobamos inicialmente no programa três experimentos distintos afim de abranger um grande número de áreas da Física. Os experimentos escolhidos foram sobre o pêndulo simples, carga e descarga de um capacitor e absorção de radiação eletromagnética. Através do programa desenvolvido é possível monitorar os valores lidos pelos sensores concomitante a realização do experimento. E ao término das leituras foi também possível visualizar os resultados em forma gráfica sem a necessidade de utilizar outros *softwares* ou planilhas eletrônicas. Desta forma, o *software* criado torna os fenômenos físicos inerentes aos experimentos mais evidentes, deixando os resultados mais claros e palpáveis aos usuários.

ABSTRACT

MARTINS, Nathan Rabelo, M.Sc., Universidade Federal de Viçosa, June, 2018**. A appliction in Python for Physics experiments autmated with Arduino.** Adviser: Ricardo Reis Cordeiro.

This work aims to enable the use of the Arduino platform in experimental Physics classes, with automated data collection, by teachers from all over Brazil, including those who do not have programming knowledge. To do so, we created an program that facilitates the relationship between user and Arduino, enabling it to be used in a practical, dynamic and simple way. This program, developed in the Python language, brings a user-friendly interface allowing the experiments with Arduino, after assembly, to be performed with just a few clicks and the insertion of some values. As mentioned, we used the Python language for the creation of the program and made the necessary conversation between it and the Arduino platform, based on C / C ++, through the signature protocol. We initially enclose three separate experiments in order to cover a large number of areas of Physics. The experiments chosen were on the simple pendulum, charge and discharge of a capacitor and absorption of electromagnetic radiation. Through the developed program it is possible to monitor the values read by the sensors concomitant to the realization of the experiment. And at the end of the readings it was also possible to visualize the results graphically without the need to use other software or spreadsheets. This way, the software created makes the physical phenomena inherent to the experiments more evident, leaving the results more transparent and palpable to the users.

SUMÁRIO

1 INTRODUÇÃO

Aprender física é um processo complicado para grande parte dos alunos. Isso acontece devido à dificuldade de abstração dos fenômenos e ao excesso de matematização desta disciplina por parte de alguns professores. Como resultado se tem que muitos alunos apresentam fraco desempenho em temas como a Mecânica, Ótica, Termodinâmica e Eletromagnetismo. Isso ocorre particularmente com aqueles estudantes que possuem dificuldade com a matemática e também com aqueles que apresentam um preconceito em relação à Física já no início do Ensino Médio. Uma maneira de torná-la mais receptiva e palpável é a realização de aulas experimentais já que nesse tipo de metodologia, em geral, os fenômenos físicos ficam mais nítidos e se pode estabelecer relações numéricas de forma mais intuitiva. Essa proposta vai de encontro aos Parâmetros Curriculares Nacionais do Ensino Médio (PCN) do Ministério da Educação onde se pode ler que

> "…o ensino de Física tem-se realizado frequentemente mediante a apresentação de conceitos, leis e fórmulas, de forma desarticulada, distanciados do mundo vivido pelos alunos e professores e não só, mas também por isso, vazios de significado. Privilegia a teoria e a abstração, desde o primeiro momento, em detrimento de um desenvolvimento gradual da abstração que, pelo menos, parta da prática e de exemplos concretos. Enfatiza a utilização de fórmulas, em situações artificiais, desvinculando a linguagem matemática que essas fórmulas representam de seu significado físico efetivo. Insiste na solução de exercícios repetitivos, pretendendo que o aprendizado ocorra pela automatização ou memorização e não pela construção do conhecimento através das competências adquiridas. Apresenta o conhecimento como um produto acabado, fruto da genialidade de mentes como a de Galileu, Newton ou Einstein, contribuindo para que os alunos concluam que não resta mais nenhum problema significativo a resolver. Além disso, envolve uma lista de conteúdos demasiadamente extensa, que impede o aprofundamento necessário e a instauração de um diálogo construtivo. "

Outra maneira de estimular os alunos e tornar as aulas mais interessantes é incorporando tecnologias atuais no cotidiano das aulas. Assim, acreditamos que a realização de experimentos assistidos pelo computador pode despertar ainda mais o interesse dos alunos. Cavalcante, *et al* (2011) concordam com este pensamento, quando afirmam que:

> "Além da melhoria da precisão dos resultados, a redução no tempo de coleta de dados e a rápida representação dos mesmos em forma de gráficos, permitem criar no laboratório de física um ambiente de construção do conhecimento físico. O estudante pode observar o fenômeno, predizer o resultado, isto é, formular hipóteses, rapidamente comparar os resultados obtidos com os previstos pelo modelo teórico, explicar possíveis diferenças entre o previsto e o observado e ainda, reformular suas hipóteses, fazer ajustes experimentais e testá-las novamente. O dinamismo desse processo provoca a curiosidade e maior interesse dos estudantes já que a aula de laboratório torna-se desafiadora."

Em muitas escolas públicas há uma escassez de equipamentos de qualidade para a realização de aulas experimentais. Esse déficit se deve, entre outros motivos, ao alto valor dos equipamentos utilizados. Na expectativa de contornar este

problema muitos professores trabalham com experimentos feitos com materiais de baixo custo, muitas vezes improvisados. Isso é, sem dúvida, uma maneira de mostrar o fenômeno de modo criativo. Porém, a parte do estudo quantitativo do problema fica muito limitada, por vezes com margens de erro muito grandes, fazendo com que a prática fuja da teoria, o que acreditamos não ser o desejado.

É nesse ponto que entendemos a utilização do Arduino como um importante auxílio neste processo já que é possível montar um laboratório com diversos experimentos automatizados por um baixo custo se comparado a um laboratório convencional (equipado com instrumentos fabricados e comercializados por empresas especializadas no ramo). Moraes, *et al* (2017) desenvolveram três protótipos em seu trabalho, que são carga e descarga de um capacitor, pêndulo simples e absorção de radiação luminosa, por corpos de cores diferentes. E ao comparar os custos dos seus protótipos, com a média dos preços dos equipamentos similares adquiridos pela Universidade Federal de Viçosa – Campus Rio Paranaíba, concluíram:

> "Apesar de não se tratarem dos mesmos experimentos, de forma geral é possível comparar os equipamentos oferecidos pelas empresas com os protótipos desenvolvidos neste trabalho. A relação custo é claramente discrepante. Os protótipos que desenvolvemos tem um custo de cerca de 1% em relação aos kits."

Os resultados obtidos por Moraes, *et al* (2017) também mostram que os protótipos criados com o Arduino geram resultados que não apenas condizem com as predições teóricas, mas que possuem um erro relativo muito baixo, principalmente se comparado com experimentos sem interface computacional.

A plataforma Arduino vem ganhando espaço na automação e robótica, tanto do ponto de vista de programação, já que utiliza uma linguagem muito semelhante a C/C++[1], quanto financeiro, uma vez que, por se tratar de um *open source*[2], pode ser adquirido por um custo bem pequeno.

Apesar das múltiplas possibilidades de uso para o Arduino, este trabalho foca na sua utilização como uma ferramenta no ensino/aprendizagem da Física. Para tanto faremos uma breve revisão bibliográfica, apresentando as funcionalidades principais do Arduino e os periféricos utilizados, além de abordar os resultados obtidos anteriormente em trabalhos com objetivos próximos ao nosso.

---

[1] Documentação da linguagem C++ disponível em: <http://en.cppreference.com/w/>.

[2] Open source ou *software* livre são aplicativos ou programas desenvolvidos e disponibilizados a comunidade de forma gratuita. As pessoas além de poderem utilizar o *software* sem custos, podem também consultá-lo e editá-lo, criando assim novas versões. São exemplos de *open source* o Linux e a linguagem de programação Python.

Em várias destas literaturas o Arduino é destacado como sendo bastante dinâmico uma vez que o usuário (experimentador) tem grande liberdade para modificar os experimentos existentes, bem como para criar novos. Porém, apesar deste fato ser estimulante para seus usuários mais experientes ou para quem está imerso nas linguagens de programação, ele também é um obstáculo para muitos dos professores que não são familiarizados com a lógica de programação. Esta realidade foi objeto de observação por parte de Martinazzo, *et al* (2014):

> "A importância e a presença, quase onipresente, da tecnologia e da informação no cotidiano da maioria das pessoas são inquestionáveis. A questão é: o quanto estão os professores preparados para enfrentar uma realidade mutante do ponto de vista tecnológico e comportamental? Os alunos já não se satisfazem apenas com aulas expositivas de Física, e anseiam por mais e os professores estão angustiados diante da evolução tecnológica e da mudança comportamental de seus alunos que estão irrequietos com as aulas tradicionais. "

É com foco nesse problema que pensamos este trabalho, cujo objetivo foi criar um *software* de fácil interação, utilizando a linguagem Python, e que "converse" com o Arduino. Desse modo se pretende que o usuário consiga realizar os experimentos englobados pelo programa sem a necessidade de digitar e alterar códigos de programação, mas somente utilizando botões e inserindo informações solicitadas em campos pré-determinados. Para exemplificar seguem as figuras 1 e 2.

Figura 1: Menu principal do programa.
Fonte: Autor.

Figura 2: Menu de configuração do experimento carga e descarga de um capacitor.
Fonte: Autor.

A interação entre o *software* criado e a placa microcontroladora, através da porta serial, será feita através do pyFirmata[3]*, que* é uma biblioteca do Python. Com essa ferramenta poderemos receber as leituras feitas, salvá-las e manipulá-las.

O programa desenvolvido engloba a programação do Arduino, os esquemas de montagem, além de textos de apoio referentes aos experimentos incorporados. O usuário irá interagir com ele apenas selecionando opções e inserindo informações, referentes aos componentes utilizados, quando solicitado. Este programa será por nós denominado *PyDuino*.

Durante a execução do experimento, os dados coletados serão tratados no programa desenvolvido em Python e os resultados serão mostrados simultaneamente à execução da prática experimental, podendo também ser apresentados de modo gráfico, por meio do Matplotlib[4], uma outra biblioteca do Python utilizada para elaborações de gráficos matemáticos.

O programa objetiva fazer com que os professores possam utilizar as vantagens dos experimentos montados com a plataforma Arduino sem ter necessidade de conhecer uma linguagem de programação.

[3] Documentação da biblioteca pyFirmata disponível em: <https://pyfirmata.readthedocs.io/en/latest/#>.
[4] Documentação da biblioteca Matplotlib disponível em: <https://matplotlib.org/contents.html>.

O *software* desenvolvido terá código aberto, uma vez que acreditamos que o produto criado tenha potencial para crescimento, e com a ajuda de colaboradores ele poderá abordar muitos outros experimentos nos mais diversos ramos da Física.

Pensando nesta colaboração, precisávamos escolher uma linguagem, para escrever o programa, que fosse bastante difundida e que os códigos fossem de fácil entendimento. Optamos pelo Python por se tratar de uma linguagem de programação de alto nível, por ser um programa livre de fácil portabilidade e com grande apoio da comunidade de usuários, tendo vários módulos e bibliotecas com as mais diversas funcionalidades. Outro motivo que nos fez optar pelo Python foi a sua estrutura de programação que faz com que os códigos dos programas sejam facilmente assimilados por outros usuários.

2 OBJETIVOS

2.1 OBJETIVO GERAL

O objetivo principal deste trabalho é desenvolver um programa que possibilite aos professores a execução de experimentos de Física com qualidade em suas aulas de Ensino Médio. Deseja-se que estes professores tenham a possibilidade de utilizar uma biblioteca de experimentos com Arduino voltados para a Física sem a necessidade de conhecerem programação.

2.2 OBJETIVOS ESPECÍFICOS

Como objetivo secundário queremos despertar a curiosidade de professores e alunos sobre o Arduino e suas funcionalidades, motivando-os a buscar entender seu funcionamento e a criarem seus próprios experimentos além de poderem modificar e melhorar os já englobados pelo programa desenvolvido neste trabalho.

3 PYTHON

Python é uma linguagem de programação de alto nível e foi criada em 1991, por Guido van Rossum. Desde seu surgimento vem ganhando cada vez mais usuários e colaboradores devido, primeiramente, à sua simplicidade, tanto na construção dos códigos quanto no entendimento dos mesmos, mas também pela variedade de aplicações que possui e pela sua robustez.

Josué Labaki (s.d.), define uma linguagem de programação de alto nível como sendo aquela que tem um elevado grau de abstração. Portando para a utilização deste tipo de linguagem não é necessário que o usuário conheça programação em código de máquina, uma vez que ela é abstraída pelo Python, e desta forma o programador se comunica com o computador através de uma linguagem próxima à humana. Podemos citar como exemplo de linguagens de alto nível C++, Fortran, PHP e Java.

Python é uma linguagem que tem suporte tanto para programação estruturada, quanto para programação orientada a objetos. Isso dá a ela versatilidade e aumenta a sua aplicabilidade nas mais diversas necessidades, podendo ser utilizado em programas simples, mas também nos mais complexos.

A linguagem Python se diferencia das outras linguagens de alto nível por ser estruturada, de modo obrigatório, em um sistema de indentação. Isso facilita que os usuários entendam os códigos escritos por outras pessoas.

> “A filosofia por trás do desenvolvimento do Python era criar códigos flexíveis, legíveis e claros para expressar facilmente seus conceitos. A ênfase em usar a indentação de uma maneira única diferencia Python de outras linguagens populares de alto nível.” (DESAI, 2015)

Lutz (2013) cita outras vantagens do Python sobre as demais linguagens de alto nível. Uma delas é que os códigos escritos em Python são executados imediatamente, sem a necessidade de serem compilados, o que faz com que os programadores consigam ser mais ágeis. Outra vantagem citada por ele é a portabilidade dos códigos, que fazem com que eles sejam executados sem problemas na maioria dos sistemas operacionais. Mais do que isso, os *scripts* em Python, graças a sua grande diversidade de módulos e bibliotecas, podem se comunicar e se integrar facilmente com aplicativos e ferramentas feitos em outras linguagens de programação.

> “Programas em Python podem se comunicar facilmente com outras partes de um aplicativo, usando uma variedade de mecanismos de integração. Essas integrações permitem que o Python seja usado como uma ferramenta de personalização e extensão de produtos. Hoje, o código Python pode invocar bibliotecas C e C ++, pode

> ser chamado por programas C e C ++, pode se integrar com componentes Java e .NET ... ” LUTZ (2013)

Lutz (2007) afirma que a linguagem Python também aumenta a produtividade do programador primeiro, porque como já citado, não são necessários compilamentos e em segundo pelo fato dos códigos escritos em Python serem mais enxutos tendo cerca de 1/3 a 1/5 do tamanho dos equivalentes em C++ ou Java. Isso implica em menos digitação, menos depuração e menos manutenção após o desenvolvimento.

Para Pratik Desai (2015) o principal diferencial de Python é exatamente seu vasto número de bibliotecas e módulos gratuitos, o que faz com que ele tenha uma infinidade de aplicações, podendo trabalhar com programação funcional, imperativa, orientada a objetos e com gerenciamento automático de memória. Sua robustez se deve, principalmente, à contribuição da sua comunidade de usuários uma vez que, por se tratar de um *software* aberto, eles podem criar novas bibliotecas e módulos com as mais diversas funções.

Entre estas bibliotecas destacamos aqui a pyFirmata, que trabalha com o protocolo Firmata e serve para fazer com que os códigos criados em Python possam ser implementados em placas de prototipagem através da porta serial do computador e que foi de fundamental importância na execução deste trabalho.

Outras bibliotecas que merecem destaque por terem sido utilizadas programa que desenvolvemos, são a Tkinter[5] e o Matplotlib. A primeira foi utilizada na criação das interfaces gráficas do *software* e a segunda nas apresentações de valores numéricos em modo gráfico. Utilizamos também dois módulos, o Os[6] e o Time[7], que foram utilizados respectivamente para abertura dos arquivos vinculados ao programa e para contagem de tempo.

O único ponto negativo comentado pelos autores acima é quanto a velocidade de processamento dos programas escritos em Python já que, em alguns tipos de implementação, ela é menor que o de escritos em outras linguagens, como a C++ por exemplo. Está redução na velocidade se deve exatamente ao fato do código não ser compilado.

---

[5] Documentação da biblioteca Tkinter disponível em: <https://docs.python.org/2/library/tkinter.html#module-Tkinter>

[6] Documentação do módulo Os disponível em: <https://docs.python.org/3/library/os.html>

[7] Documentação do módulo Os disponível em: <https://docs.python.org/3/library/time.html?highlight=time#module-time>

Apesar de também reconhecer este ponto desfavorável à utilização da linguagem Python, Lutz (2007) afirma que boa parte dos códigos em Python é executada rápida o bastante para seus domínios de aplicação. Outro ponto apresentado por eles é que muitas das bibliotecas do Python, como as de manipulação de arquivos ou de interfaceamento gráfico, como a Tkinter, têm suas tarefas executadas em C e estas são compiladas dentro do interpretador Python. Assim o programa é otimizado e não apresenta diferença de performance significativa em sua execução. Essa diferença se torna ainda menor quando tomamos por perspectiva o poder de processamento dos computadores atuais.

A conclusão final que Lutz (2007) chega é que, apesar de existir uma pequena limitação de velocidade de processamento em alguns casos, ela se faz pequena diante ao ganho de tempo obtido no desenvolvimento dos programas.

4 ARDUINO

Seria possível desenvolver este trabalho utilizando a maioria das placas de prototipagem existente no mercado, uma vez que o pyFirmata é capaz de se comunicar com os mais diversos tipos destas placas. Porém, optamos pelos experimentos realizados com a placa Arduino. A escolha se deu, em primeiro momento, pelo fato dela ter tanto *hardware* quanto *software* abertos. Consequentemente, ela possui um menor custo, quando comparada a outras placas *similares* tais como *Raspeberry Pi*[8] ou *Intel Galileo*[9].

O Arduino surgiu na Itália no ano de 2005 objetivando a redução nos custos na prototipagem que eram muito elevados na época e acabou ganhando ainda mais destaque devido a sua linguagem simples e padronização na programação. De acordo com Moraes, *et al* (2017):

> "A placa (Arduino) se baseia num microcontrolador ATMEL passível de programação. O microcontrolador é um microprocessador que pode ser programado para funções específicas. Eles possuem memórias de programa, de dados e RAM, temporizadores e circuitos de clock embutidos."

Existem vários tipos de placas Arduino no mercado. Moraes, *et al* (2017) cita como exemplo os seguintes tipos: Arduino UNO, Arduino MEGA, Arduino Due, Arduino Nano, Arduino Leonardo, ArduinoPro Mini, Arduino ADK e o Arduino Esplora. Ele explica que a diferença entre elas está no número de portas, quantidade de memória, tensão de operação, tamanho, etc. Apesar das diferenças enunciadas, todas as placas apresentadas possuem linguagem compatível entre si, segue abaixo a figura 3 com a finalidade de listar as diferenças de cada uma delas.

---

[8] Página principal dos desenvolvedores da Raspeberry Pi disponível em <https://www.raspberrypi.org/>.

[9] Página com informações da Intel Galileo disponível em <https://www.intel.com.br/content/www/br/pt/support/articles/000005912/boards-and-kits/intel-galileo-boards.html>.

	UNO	MEGA 2560	LEONARDO	DUE	ADK	NANO	PRO MINI	ESPLORA
Microcontrolador	ATmega328	ATmega2560	ATmega32u4	AT91SAM3X8E	ATmega2560	ATmega168 (versão 2.x) ou ATmega328 (versão 3.x)	ATmega168	ATmega32u4
Portas digitais	14	54	20	54	54	14	14	-
Portas PWM	6	15	7	12	15	6	6	-
Portas analógicas	6	16	12	12	16	8	8	-
Memória	32K (0,5K usado pelo bootloader)	256K (8K usado pelo bootloader)	32K (4K usado pelo bootloader)	512K disponível para aplicações	256K (8K usado pelo bootloader)	16K (ATmega168) ou 32K (ATmega328) (bootloader: 2K)	16K (2K usado pelo bootloader)	32K (4K usado pelo bootloader)
Clock	16Mhz	16Mhz	16Mhz	84Mhz	16Mhz	16Mhz	8Mhz (modelo 3.3v) ou 16Mhz (modelo 5v)	16Mhz
Conexão	USB	USB	Micro USB	Micro USB	USB	USB Mini-B	Serial/Módulo USB externo	Micro USB
Conector para alimentação externa	Sim	Sim	Sim	Sim	Sim	Não	Não	Não
Tensão de operação	5V	5V	5V	3.3V	5V	5V	3.3 ou 5V, dependendo do modelo	5V
Corrente máxima portas E/S	40mA	40mA	40mA	130mA	40mA	40mA	40mA	-
Alimentação	7-12Vdc	7-12Vdc	7-12Vdc	7-12Vdc	7-12Vdc	7-12Vdc	3.35-12V (modelo 3.3v) ou 5-12V (modelo 5v)	5V

Figura 3: Tabela dos modelos de Arduino disponíveis e suas especificidades.
Fonte: https://www.filipeflop.com/blog/tipos-de-arduino-qual-comprar/

Optamos pela placa Arduino UNO, por ser a placa mais difundida no mercado atualmente. Ela tem 14 pinos de *input/output digitais*, com mais 6 de entrada analógica, trabalha com tensão de 5 V e um processador ATmega328. Outro motivo para a escolha do Arduino é o fato dela ser compatível com uma infinidade de módulos/sensores. Esse fato proporciona os mais diversos tipos de aplicação para esta placa. Martinazzo, *et al* (2014) citam que suas aplicações se estendem inclusive ao ensino de Física:

> "No que diz respeito ao Ensino de Física, tem grande aplicabilidade, pois é possível ler dados de qualquer fenômeno físico detectável por sensores, ou seja, basicamente é um sistema que lê sinais elétricos em sensores expostos ao ambiente a partir de suas portas digitais e analógicas. "

Acreditamos que, mesmo em experimentos para fins didáticos, é necessário que haja um mínimo de precisão nos resultados, para que os fenômenos fiquem claros, conciliando assim a teoria e a prática e facilitando o estabelecimento das relações quantitativas, acreditamos que experimentos realizados por estudantes cujo resultado tenha grande margem de erro faz com que eles criem uma descrença com relação a teoria. Os resultados obtidos com o Arduino têm sido promissores: Cavalcante, et al (2011) conseguiram resultados semelhantes aos feitos por equipamentos tradicionais nos experimentos de carga e descarga de um capacitor.

Martinazzo, *et al* (2014) e Moraes, *et al* (2017) por sua vez, citam experimentos de aceleração, movimento uniformemente variado, oscilações, resfriamento, evaporação e queda dos corpos. Já Fernandes, *et al* (2014) utilizaram o Arduino nos estudos da eletrodinâmica.

Ainda segundo Moraes, *et al* (2017), outra vantagem dos protótipos usando Arduino sobre os kits feitos por empresas é que estes últimos são muitas vezes engessados, ficando restritos a um único objetivo. Os protótipos com Arduínos possuem mais liberdade de modificação, podendo ser alterados conforme a curiosidade do usuário.

Uma limitação da placa Arduino foi citada por Martinazzo, *et al* (2014), sendo relacionada ao fato dela sozinha não permitir o processamento de dados para apresentação gráfica. Eles contornam o problema tratando os dados em planilhas eletrônicas, assim como fizeram Moraes, *et al* (2017). Cavalcante, *et al* (2011) utilizaram um outro *software*, o Processing[10], para tratar os dados coletados e construir os respectivos gráficos, tudo em tempo real.

[10] Página principal do Processing disponível em <https://processing.org/>.

Com a preocupação de resolver a limitação descrita acima, durante a execução deste trabalho fizemos o processamento dos dados, com a ajuda da pyFirmata e plotamos os resultados gráficos em nosso programa utilizando a Matplotlib, que é muito eficaz neste tipo de tarefa.

5 METODOLOGIA

Buscar ferramentas que auxiliam na execução de aulas experimentais no ensino de Física tem sido a preocupação de muitos pesquisadores, uma vez que é indiscutível a contribuição deste tipo de prática no processo de ensino aprendizagem.

Sharma, *et al* (2010), desenvolveram dois projetos com foco nas IDLs (Interactive Lecture Demonstration), ambos aplicados na Universidade de Sidney. O primeiro constava de uma análise projetos anteriores feitos entre os anos de 1999 a 2009. O segundo acompanhado foi a reprodução desta metodologia entre os anos de 2007 e 2009. Eles tinham por objetivo verificar a eficiência das IDLs em um contexto australiano e buscar entender o porquê deste tipo de metodologia não ser amplamente utilizada nos cursos de Física.

Eles relatam resultados de ganhos de compreensão conceitual menores do que de outros trabalhos envolvendo este tipo de metodologia, porém superiores aos obtidos utilizando metodologias tradicionais (Giz, fala e powerpoint). Eles concluem que a utilização das IDLs são uma contribuição valiosa para a aprendizagem dos alunos e desenvolvimento profissional do docente, atribuindo as diferenças nos resultados à diferença de contexto entre os trabalhos.

Sharma, *et al* (2010) também afirmam que existem dificuldades na implementação, como a necessidade de suporte técnico e o tempo adicional necessário no início das aulas. Eles relatam que existem casos em que a metodologia é implementada por um único professor dedicado, porém a necessidade de equipamentos específicos e a grande escassez de tempo dos professores de grandes departamentos de ensino faz com que as IDLs não sejam utilizadas.

Almejando mais precisão e automação na coleta de dados, muitos pesquisadores se atentaram a utilização do Arduino nesse tipo de metodologia, conseguindo experimentos que não só tiveram resultados muito satisfatórios, como também possuíam um preço muito acessível para execução.

Porém, percebemos que a incorporação deste tipo de experimento, na rotina didática da grande maioria dos professores, ainda esbarra em algumas barreiras tais como a pouca familiaridade com a linguagem de programação ou um conhecimento de eletrônica muito restrito.

Buscando auxiliar esse público, pensamos na criação de um programa de computador. Este que traria a possibilidade do usuário realizar o experimento sem a

preocupação de manipular as linhas de programação e com a possibilidade de análise dos resultados sem a utilização de outras ferramentas. Por fim, incorporamos também ao produto desta dissertação, lista de materiais, esquemas de montagens e textos de apoio. Abaixo segue a figura 3 afim de ilustrar os esquemas de montagem.

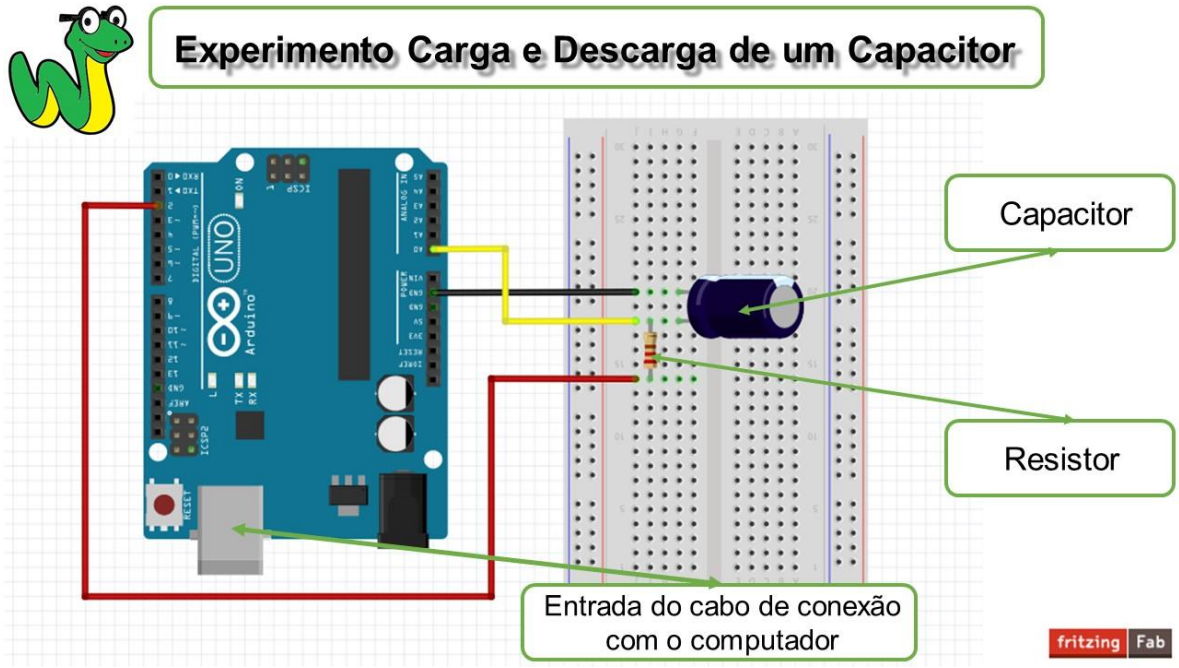


Figura 4: Esquema de montagem do experimento carga e descarga de um capacitor.
Fonte: Autor.

Para a criação da interface gráfica do programa, optamos por uma biblioteca do Python chamada Tkinter. Contribuiu para esta escolha o fato da Tkinter já vir incluso na maioria das distribuições do Python, além de ela ser bem robusta, suprindo todas as necessidades inerentes ao programa.

Ao pensarmos em quais experimentos colocaríamos inicialmente no produto desta dissertação, optamos por três experimentos bem distintos para que assim pudéssemos envolver um grande número de áreas da física.

Os experimentos escolhidos foram o de pêndulo simples, a carga e descarga de um capacitor e o de emissão e absorção de radiação eletromagnética por corpos de diferentes cores. Desta forma abordamos fenômenos eletromagnéticos, térmicos e o movimento harmônico simples.

Inicialmente imaginamos que incorporaríamos os experimentos escolhidos em nosso produto fazendo apenas a conversão entre linguagens de programação do Arduino (C/C++) para Python. Porém, neste trabalho optamos por fazer pequenas modificações na estrutura de alguns dos experimentos buscando adaptá-los ou até mesmo simplificá-los, tornando-os mais acessíveis às pessoas com pouca

familiaridade com eletrônica, informática e prototipagem. Outra adaptação que se fez necessária, para dar mais acessibilidade à análise dos dados coletados durante o experimento, foi possibilitar por meio do próprio *software* a elaboração de gráficos dos resultados.

Para gerarmos os gráficos utilizamos a biblioteca do Python chamada Matplotlib. Este trabalha em conjunto com a interface criada em Tkinter, abrindo uma janela auxiliar com os gráficos toda vez que o botão gráfico for selecionado pelo usuário. Além de gerar o gráfico, este procedimento traz a possibilidade de redimensionar a janela, dar *zoom*, movimentar os gráficos, além de possibilitar a inserção de vários gráficos em uma única janela, como mostrado na figura 4.

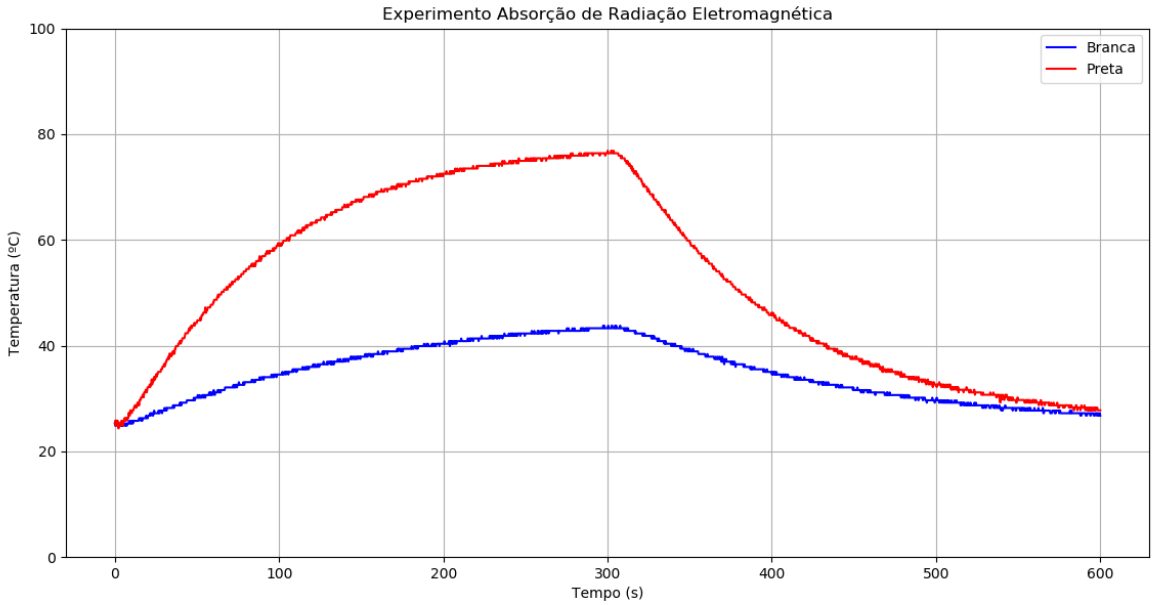


Figura 5: Gráfico do experimento de absorção de radiação eletromagnética, com placas preta e branca e tempo de coleta de 10 minutos.
Fonte: Autor.

O programa escrito foi dividido em várias funções, que são basicamente blocos de programação com um objetivo específico. Estas funções serão executadas apenas se determinado *input* for dado por parte do usuário. Na figura 5 ilustramos o fluxograma do programa afim de mostrar como as funções criadas se relacionam.

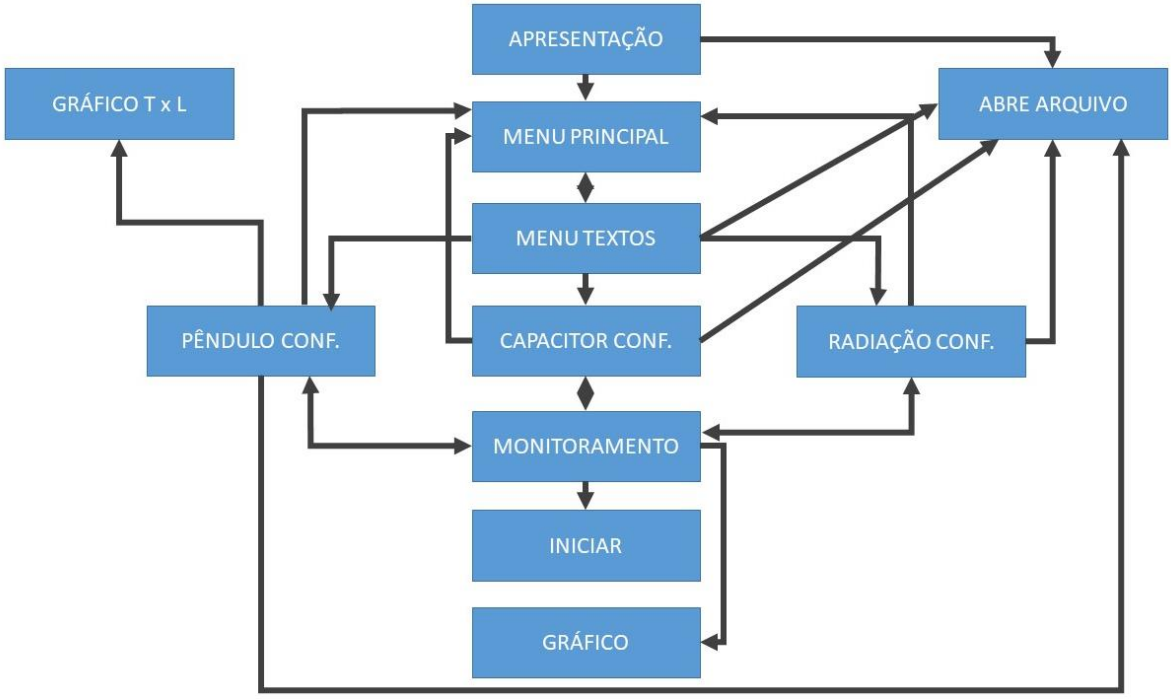


Figura 6: Fluxograma do programa PyDuino.
Fonte: Autor.

Nos próximos tópicos discutiremos a metodologia utilizada em cada um dos experimentos abordados por este trabalho.

5.1 PÊNDULO SIMPLES

O experimento de pêndulo simples pode ser trabalhado tanto dentro da mecânica, quanto como exemplo do movimento harmônico simples. Contudo focamos apenas no segundo caso na realização deste trabalho.

Dentro desse experimento gostaríamos de evidenciar duas coisas: primeiro, que o período de oscilação do pêndulo não depende da sua massa e, segundo, que este mesmo período tem proporção direta com a raiz quadrada do comprimento do fio de sustentação.

Moraes, *et al* (2017) traz este experimento em seu trabalho. Porém, eles focam apenas em automatizar o registro do período de oscilação do corpo, sendo possível monitorar em tempo real os dados obtidos. No entanto, para uma análise gráfica ou para comparar os resultados de medidas diferentes, seria necessário a utilização de outros recursos tais como *softwares* de análise de dados.

Pensando em dar maior visibilidade aos resultados, trouxemos duas possibilidades de análise gráfica dentro do próprio programa, sendo a primeira o valor

do período medido em cada oscilação e a segunda a relação entre o período e o comprimento do fio de sustentação. A primeira possibilidade foi incluída na tela de monitoramento do experimento, ficando a geração do gráfico disponível assim que findadas as medições dos períodos.

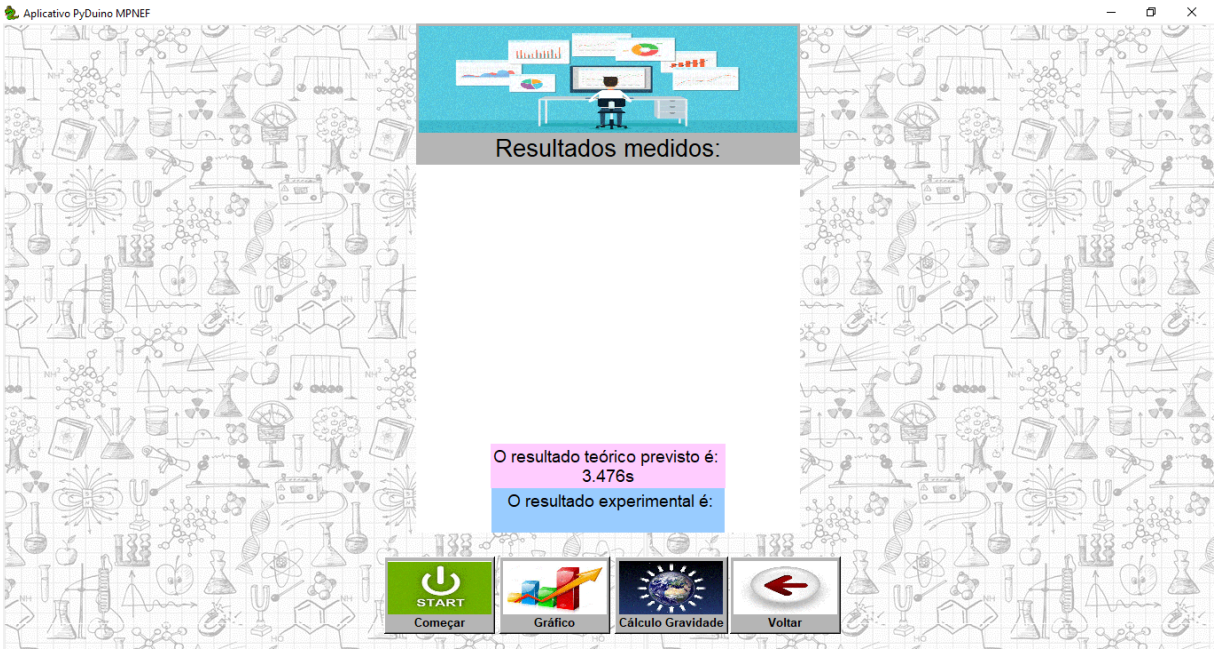


Figura 7: Tela monitoramento de monitoramento do experimento pêndulo simples.
Fonte: Autor.

Nesta etapa encontramos uma pequena dificuldade pois, durante as medidas do sensor de infravermelho com o processamento usando a linguagem Python, eventualmente os períodos medidos apresentavam flutuações razoáveis. Estas variações mostravam sempre valores da ordem de múltiplos de meio período, denotando que algumas leituras não eram registradas durante a execução do experimento. Este problema não foi diagnosticado quando o experimento foi realizado na linguagem nativa do Arduino. Assim, acreditamos que este fato seja fruto do processamento em Python. Para contornarmos este obstáculo optamos por descartar as medidas que tivessem desvios superiores a 25% do período teórico, que é o período calculado a partir das informações fornecidas pelo usuário. Desta forma conseguimos deixar o experimento válido e viável uma vez que não estamos considerando apenas medidas corretas, mas sim descartando valores que são frutos de erros de medição ou de processamento.

Para analisar o problema do ponto de vista da relação entre o período médio e o comprimento do pêndulo, primeiro criamos um arquivo de texto que armazena o histórico dos experimentos já realizados, salvando o comprimento do fio

utilizado e o período experimental obtido cada vez que o experimento é feito. Esses dados são mostrados na tela do programa e o usuário possui liberdade para gerar o gráfico relacionado clicando no botão correspondente. É também possível apagar alguns dados caso isso seja necessário. Segue uma imagem extraída do *software*, onde o histórico do experimento está mostrado no lado direito da tela.

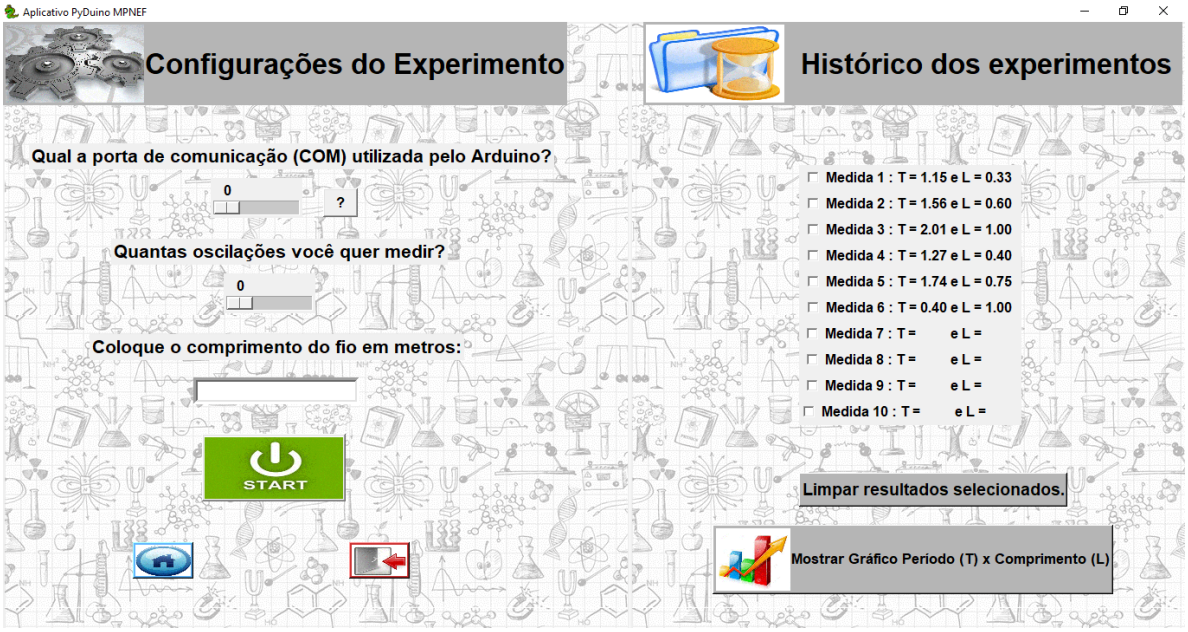


Figura 8: Tela de configuração e histórico do experimento pêndulo simples.
Fonte: Autor.

5.2 CARGA E DESCARGA DE UM CAPACITOR

Capacitores são trabalhados na física do Ensino Médio geralmente no terceiro ano. Apesar de estarem presentes na maioria dos dispositivos eletroeletrônicos no cotidiano das pessoas, poucos sabem identificá-los ou até mesmo entender superficialmente seu funcionamento.

Cavalcante, *et al* (2011) traz em seu trabalho uma prática para abordar este conteúdo. Esta faz a automação das leituras da tensão elétrica em um capacitor através da utilização do Arduino. Nesta prática eles fizeram a análise, em tempo real, dos dados em forma gráfica utilizando o Processing. Este que é uma linguagem/ambiente que possibilita programar imagens, animações ou interações. Este mesmo experimento também foi abordado por Moraes, *et al* (2017) em seu trabalho. Porém eles utilizaram *softwares* de análises de dados para o estudo dos resultados em forma gráfica.

Tanto Cavalcante, *et al* (2011), quanto Moraes, *et al* (2017) conseguiram resultados muito satisfatórios com seus experimentos, motivo pelo qual optamos por trazê-lo também ao nosso produto.

Em nosso produto optamos por analisar outras variáveis em conjunto com a diferença de potencial no capacitor, que são a corrente elétrica no circuito e a carga total armazena no capacitor. Essas outras duas variáveis não são medidas diretamente pelo Arduino, mas sim calculadas pelo *software* supondo que os resistores utilizados são ôhmicos. Resistores ôhmicos possuem resistência elétrica constante independente da tensão que são submetidos. Assim a corrente elétrica será sempre diretamente proporcional a tensão nos seus polos.

Para o cálculo da corrente elétrica no circuito utilizamos a segunda lei de Ohm. O valor da resistência é informado pelo usuário e para a tensão do circuito existe duas possibilidades: durante a carga do capacitor a tensão é dada pela diferença entre a tensão fornecida pelo Arduino, que é de 5V, e a tensão acumulada no capacitor. Já na descarga a tensão do circuito é a própria DDP no capacitor.

Já o cálculo da carga armazenada no capacitor é feito através da capacitância do mesmo, cujo valor também é fornecido pelo usuário no programa. Assim temos que a carga armazenada no capacitor é o produto entre a capacitância deste componente e a sua tensão elétrica.

O *software* recebe do usuário os valores da capacitância, da resistência do resistor e o número de ciclos de carga e descarga que o usuário deseja medir. Esta informação é importante pois o programa só será encerrado após o término de todos os ciclos previamente estabelecidos. Durante a realização do experimento será mostrado na tela do programa, a cada intervalo de 0,5s, o tempo total decorrido desde o início do experimento, o valor atual da tensão e da carga do capacitor, além da corrente elétrica no circuito.

Neste experimento, bem como no de absorção de radiação eletromagnética, nos quais é contabilizada a passagem do tempo durante o experimento, é possível perceber um pequeno tempo de processamento do código. Ele fica evidente uma vez que as medições eventualmente não são feitas nos intervalos exatos de tempo de espera previamente definidos, que são 0,3 e 0,5 segundos respectivamente. Esse fato, porém, não gera nenhum tipo de prejuízo aos resultados do experimento.

No final da medição dos ciclos pré-definidos o botão gráfico ficará liberado, e ao ser acionado mostrará simultaneamente três gráficos: da tensão no capacitor versus tempo, da carga armazenada no capacitor versus tempo e da corrente elétrica no circuito versus tempo. Abaixo a figura 9 para ilustrar a disposição dos gráficos.

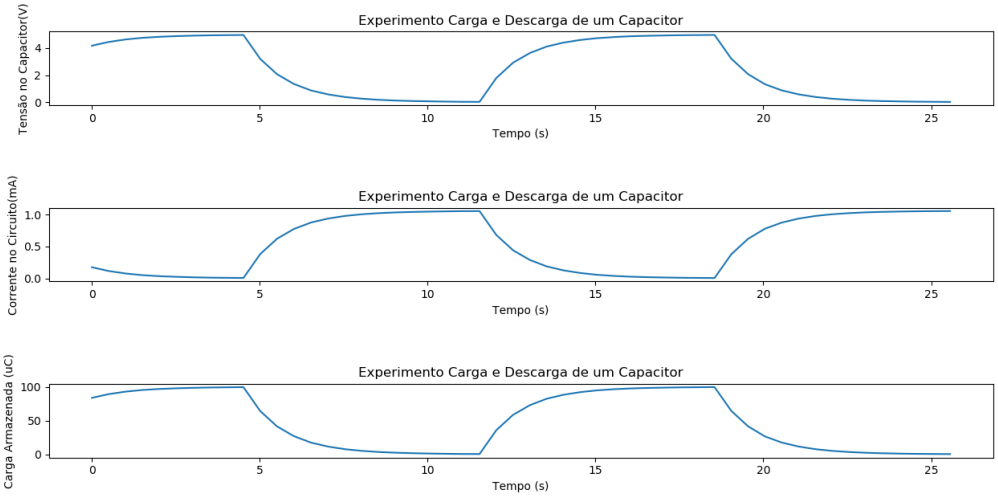


Figura 9: Gráfico do experimento carga e descarga de um capacitor com 2 ciclos de carga e descarga e utilizando um resistor de 4.7 KΩ e um capacitor de 20μF.
Fonte: Autor.

5.3 ABSORÇÃO DE RADIAÇÃO ELETROMAGNÉTICA

Os conceitos que envolvem a Termometria são trabalhados geralmente no primeiro bimestre do segundo ano do Ensino Médio. Porém, focaremos aqui apenas no tocante a trocas de calor, sobretudo as trocas de calor por radiação.

Moraes, *et al* (2017), trazem em seu trabalho um experimento que aborda este conteúdo. Eles utilizam uma lâmpada incandescente como fonte de calor e quatro placas pintadas de cores diferentes. O objetivo é medir a curva de aquecimento e de resfriamento de cada uma das placas através de sensores de temperatura. As leituras foram automatizadas por meio do Arduino. Eles utilizaram o sensor digital de temperatura DS18B20 e duas bibliotecas específicas: uma para converter o sinal digital do sensor para valores analógicos e outra para discriminar mais de um sensor conectado a mesma porta digital do Arduino. As bibliotecas são a OneWire[11] e a DallasTemperature[12]. Estas bibliotecas não são padrão da IDLE do Arduino, sendo necessário baixá-las e instalá-las externamente.

[11] Informações da biblioteca OneWire disponível em <https://playground.arduino.cc/Learning/OneWire>.
[12] Documentação da biblioteca DallasTempeture disponível em <https://www.arduinolibraries.info/libraries/dallas-temperature>.

Escolhemos este experimento para colocar no nosso produto. Porém optamos por utilizar apenas duas placas coloridas por vez afim de simplificar as conexões eletrônicas.

Tentamos utilizar o mesmo sensor, mas não obtivemos êxito ao utilizá-lo, pois não encontramos bibliotecas que fossem similares as utilizadas em Python. Assim, optamos por utilizar outro sensor, o LM35. Este é analógico e não faz uso de nenhuma biblioteca específica. Ele consegue leituras entre -55ºC até 150ºC sendo, inclusive, um pouco mais sensível a perturbações externas do que o DS18B20.

Possibilitamos, em nosso produto, que o usuário acompanhe as medidas das temperaturas das placas simultaneamente a execução do experimento além de poder visualizar os resultados obtidos para ambos os corpos em forma gráfica. Para facilitar a comparação dos resultados optamos por acoplar os dois gráficos no mesmo plano cartesiano. Como já mostrado na figura 4.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento do presente trabalho possibilitou a criação de um programa que facilitasse a utilização da plataforma Arduino, dando acesso a esses experimentos a qualquer tipo de público, sobretudo para aqueles que não possuem conhecimento sobre linguagem de programação.

Além da automatização dos experimentos, fazem parte do produto vários textos como guias para configurações do computador e do Arduino, listas dos materiais utilizados em cada experimento, ilustrações das montagens eletrônicas dos circuitos construídos, além de textos de apoios com explicações detalhadas sobre os experimentos. Todos os textos são acessados pelo próprio programa e estão anexos a esta dissertação em forma de apêndices.

Para que o produto funcione é necessário que o sistema operacional do computador seja Windows e que sejam instalados o Python e as bibliotecas utilizadas no programa. Para ajudar os usuários fizemos um tutorial com o passo a passo dessas instalações e a disponibilizamos junto ao produto desenvolvido nesta dissertação.

Os resultados das medidas dos sensores são visualizados simultaneamente a realização do experimento, como nos experimentos realizados pela própria plataforma Arduino. Porém, o nosso produto não exige do usuário qualquer conhecimento de programação. Outra possibilidade oferecida pelo nosso produto é a de visualizar os resultados obtidos em forma gráfica sem a utilização de outros *softwares* ou de planilhas eletrônicas.

O desempenho do programa desenvolvido por nós na linguagem Python tem desempenho similar a IDLE do Arduino (*software* nativo), ilustrando com qualidade os fenômenos físicos trabalhados nos experimentos englobados.

O produto criado se trata de um *open source* e o código do *software* será disponibilizado em forma de apêndice nesta dissertação para que possa ser modificado, ampliado e melhorado pelos usuários.

REFERÊNCIAS BIBLIOGRÁFICAS

ARDUINO and Python. Disponível em: <http://playground.arduino.cc/interfacing/python>. Acesso em: 25 jul. 2017.

BRUIJN, Tino de. **pyFirmata 1.0.3:** A Python interface for the Firmata procotol. Disponível em: <https://pypi.python.org/pypi/pyFirmata>. Acesso em: 25 mai. 2017.

CAVALCANTE, Marisa Almeida; TAVOLARO, Cristiane Rodrigues Caetano; MOLISANI, Elio. **Física com Arduino para iniciantes.** Revista Brasileira de Ensino de Física, v. 33, n.4, 4503, 2011.

DESAI, Pratik. **Python Programming for Arduino:** Develop practical Internet of Things prototypes and application program s with Arduino and Python. Birmingham: Packt Publishing, 2015. 400 p.

FERNANDES, Moacir Borges; HARTMANN, Ângela Maria; DORNELES, Pedro Fernando Teixeira. **A Física no Ensino Médio integrado:** uma sequência didática sobre eletricidade com aplicação do Arduino. Revista Brasileira de Física Tecnológica Aplicada, v. 1, n. 2, p.19-34, dez. 2014.

Sharma, Manjula D. et al. **Use of interactive lecture demonstrations:** A ten year study. School of Physics, Universtity of Sydney, New South Wales 2006, Australia.

LUTZ, Mark. **Learning Python.** 5. ed. Sebastopol: O'reilly Media, 2013. 1540 p.

MARTINAZZO, Claodomir Antonio et al. **Arduino:** uma tecnologia no ensino de física. Perspectiva, Erechim, v. 38, n. 143, p.21-30, set. 2014.

MORAES, Thales Fernandes; LOUREIRO, Marcos Paulo de Oliveira. **Análise de viabilidade para criação de um laboratório de física aplicada utilizando tecnologia de baixo custo.** 2017. 1 v. TCC (Graduação) - Curso de Engenharia de Produção, Universidade Federal de Viçosa, Rio Paranaíba, 2017.

LABAKI, Josué. **Introdução a Python:** Módulo A. Disponível em: <http://www.dcc.ufrj.br/~fabiom/python/pythonbasico.pdf>. Acesso em: 10 out. 2018.

LABAKI, Josué. **Introdução a Python:** Módulo C - Tkinter. Disponível em: <https://www.dcc.ufrj.br/~fabiom/mab225/tutorialtkinter.pdf>. Acesso em: 22 out. 2018.

Parâmetros Curriculares Nacionais. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/ciencian.pdf>. Acesso em: 16 dez. 2017.

MOTA, Alan; **Sensor de temperatura LM35 – Medindo temperatura com Arduino.** Disponível em: <https://portal.vidadesilicio.com.br/lm35-medindo-temperatura-com-arduino/>. Acesso em: 05 fev. 2018.

NOVA ELETRONICA; **LM35 – O sensor de Temperatura mais Popular.** Disponível em: <http://blog.novaeletronica.com.br/lm35-o-sensor-de-temperatura-mais-popular/>. Acesso em: 05 fev. 2018.

FERG, Steven; **Pensando em Tkinter.** Disponível em: <http://www.dcc.ufrj.br/~fabiom/mab225/PensandoTkinter.pdf>. Acesso em: 12 fev. 2018.

Faria, Wellington; **Programando o Arduino em Python [PyFirmata].** Disponível em: <https://pt.linkedin.com/pulse/programando-arduino-em-python-pyfirmata-wellington-c-faria>. Acesso em: 18 fev. 2018.

Otávio, João; **TKinter:** Interfaces Gráficas em Python. Disponível em: <https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956>. Acesso em: 18 fev. 2018.

HUNTER, John et al. **Matplotlib:** Release 2.2.2. Disponível em: <https://matplotlib.org/Matplotlib.pdf>. Acesso em: 15 abr. 2018.

PINTO, Alice Regina et al. **Manual de normalização de trabalhos acadêmicos**. Viçosa, MG, 2010. 88 p.

Matplotlib: Docs. Disponível em: <https://matplotlib.org/contents.html>. Acesso em: 16 mai. 2018.

Tkinter: Documentation. Disponível em: <https://docs.python.org/2/library/tkinter.html#module-Tkinter>. Acesso em: 16 mai. 2018.

pyFirmata: Docs**.** Disponível em: <https://pyfirmata.readthedocs.io/en/latest/#>. Acesso em: 16 mai. 2018.

DallasTemperature. Disponível em: <https://www.arduinolibraries.info/libraries/dallas-temperature>. Acesso em: 16 mai. 2018.

Dallas Semiconductor's 1: Wire Protocol. Disponível em: <https://playground.arduino.cc/Learning/OneWire>. Acesso em: 16 mai. 2018.

C++ Reference. Disponível em: <http://en.cppreference.com/w/>. Acesso em: 16 mai. 2018.

Thomsen, Adilson. **Qual Arduino Comprar? Conheça os Tipos de Arduino.** Disponível em: < https://www.filipeflop.com/blog/tipos-de-arduino-qual-comprar/>. Acesso em: 16 mai. 2018.

Raspeberry Pi. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 16 mai. 2018.

Introdução à Placas Intel® Galileo. Disponível em: <https://www.intel.com.br/content/www/br/pt/support/articles/000005912/boards-and-kits/intel-galileo-boards.html>. Acesso em: 16 mai. 2018.

Processing. Disponível em: <https://processing.org/>. Acesso em: 16 mai. 2018.

Módulos e Pacotes em Python: Artigo, com exemplo, ilustrando o funcionamento de módulos e pacotes em Python. Disponível em: <http://www.devfuria.com.br/python/modulos-pacotes/> Acessado em 25 mai. 2018.

Time: Time access and conversions. Disponível em: <https://docs.python.org/3/library/time.html?highlight=time#module-time>. Acesso em: 27 mai. 2018.

Os: Miscellaneous operating system interfaces. Disponível em: <https://docs.python.org/3/library/os.html>. Acesso em: 27 mai. 2018.

Meriat, Vitor. **Programando seu Arduino com Visual Studio.** 2015. Disponível em: <http://www.vitormeriat.com.br/2015/07/12/programando-seu-arduino-com-visual-studio/>. Acesso em: 27 mai. 2018.

NATHAN RABELO MARTINS

PYDUINO: LABORATÓRIO PARA O ENSINO DE FÍSICA.

VIÇOSA – MINAS GERAIS
2018
NATHAN RABELO MARTINS

RESUMO

Este produto é uma compilação de todo o material inerente ao *software* criado durante o desenvolvimento da dissertação: um programa feito em Python para experimentos de Física automatizados com o Arduino. Inicialmente faremos uma apresentação do produto criado, falando passo a passo dos seus menus iniciais e identificando onde e qual dos textos aqui presentes serão acessados pelo usuário por meio do programa.

Depois serão apresentados cada texto utilizado pelo programa na integra, desde tutoriais de instalação, configuração e atualização dos *softwares* e hardwares utilizados até textos de apoio aos usuários referentes aos experimentos englobados pelo *software*, fornecendo informações dos fenômenos envolvidos, dos periféricos utilizados, das funcionalidades do programa, além de dicas de utilização.

A linguagem apresentada será informal, no geral, em formato de um diálogo, visando torná-lo amigável e de fácil assimilação pelo usuário.

Na terceira parte deste trabalho traremos as listas dos materiais utilizados em cada experimento. Na seção seguinte serão apresentados os esquemas das montagens eletrônicas de cada uma das práticas experimentais. Por fim, na última seção disponibilizaremos o código do programa criado.

SUMÁRIO

1 APRESENTAÇÃO

O desenvolvido será disponibilizado em uma pasta e os arquivos inclusos corresponderão, além do executável do programa em Python, aos textos e imagens utilizados pelo *software*. A pasta principal do produto terá dois arquivos e uma pasta secundária. Um dos arquivos é o próprio executável Python do programa, nomeado por nós de PyDuino. O outro é um arquivo PDF chamado Leia me. Este trará o texto presente no tópico 2.1. (LEIA ME ANTES DE UTILIZAR O PROGRAMA PYDUINO), que tem como função dar as primeiras orientações aos usuários e ajuda-los na instalação do Python e das bibliotecas externas utilizadas. Na pasta secundária, nomeada como Componentes, estarão todos os demais arquivos utilizados pelo *software*, entre eles as imagens e os textos.

Ao executar o programa a seguinte tela inicial será mostrada:

Este texto, de visualização obrigatória, informa ao usuário que, caso seja a primeira utilização de uma placa Arduino, a mesma deve ser configurada. Para saber como proceder, o usuário deverá clicar sobre o botão firmata. Ao clicar sobre este botão será aberto um arquivo PDF que contém o texto presente no tópico 2.2. (DICAS PARA A PRIMEIRA UTILIZAÇÃO DO PROGRAMA PYDUINO). Caso a placa já esteja configurada basta selecionar o botão start que o menu abaixo será apresentado.

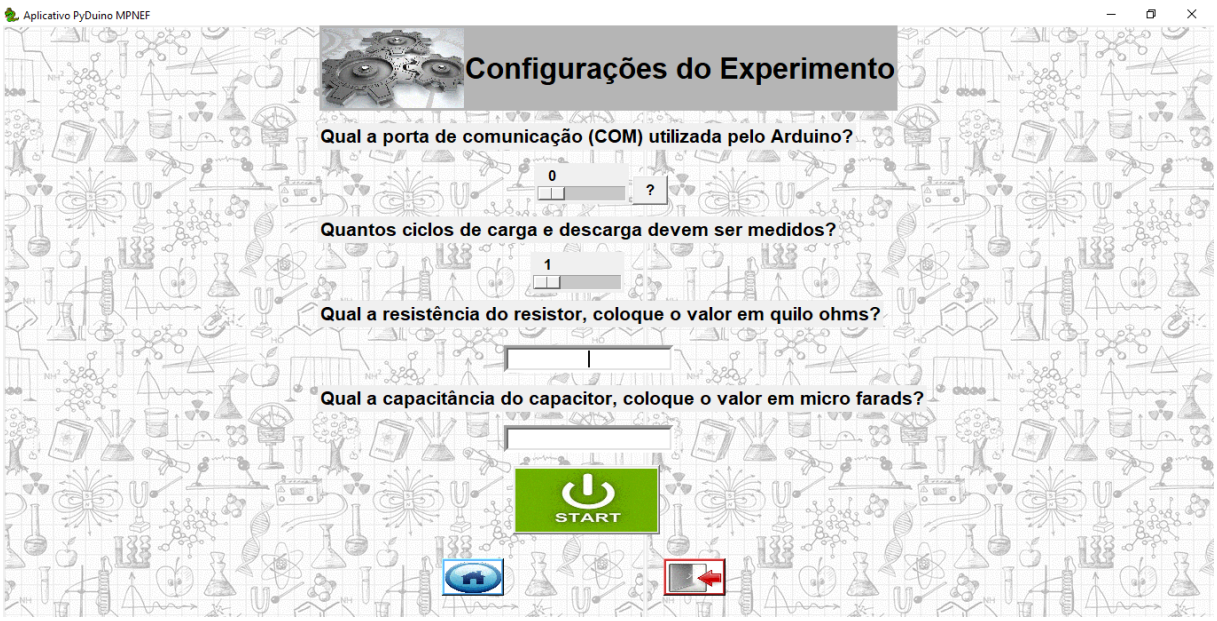
Este é o menu principal e é nele que o usuário escolherá entre os experimentos englobados por nosso *software*. Inicialmente trouxemos três experimentos: pêndulo simples, carga e descarga de um capacitor e absorção de radiação eletromagnética. O botão Experimento 4 serve apenas para mostrar que este é um produto contínuo e que poderá receber o acréscimo de novos experimentos com o passar do tempo. Para escolher entre um dos experimentos, basta clicar sobre ele que o menu correspondente será exibido. Utilizaremos como ilustração o menu do experimento de pêndulo simples, que segue abaixo.

Caso o usuário escolha a opção Material necessário, será aberto um arquivo PDF com a lista de materiais utilizados no respectivo experimento. Estas

listas, em sua integridade, estão na seção 3 deste trabalho. Ao clicar sobre o botão Esquema de montagem, será aberto um arquivo PDF mostrando como deverão ser feitas as ligações na placa Arduino. Estes esquemas compõe a seção 4. Se o usuário escolher a opção Texto de apoio, então será aberto um arquivo PDF com um texto detalhado sobre o experimento escolhido com explicações sobre os fenômenos e dicas de utilização. Estes textos fazem parte da seção 2, sendo o tópico 2.4 para o experimento de pêndulo simples, o tópico 2.5 para a prática de carga e descarga de um capacitor e o tópico 2.6 para o experimento de absorção de radiação eletromagnética.

Todos estes textos são de leitura opcional. Caso o usuário já tenha conhecimento de como funciona a prática e já esteja com toda a montagem pronta ele poderá acessar diretamente o menu de configuração do experimento clicando sobre o botão com este nome.

Para todos os experimentos será necessário informar qual a porta de comunicação utilizada, uma vez que o *software* se comunicará com o Arduino através dela. Essa informação deverá ser fornecida em todos os casos no menu de configuração de experimento. Segue abaixo o menu do experimento carga e descarga de um capacitor para servir de ilustração.



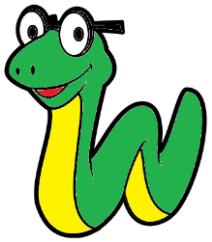
Para auxiliar o usuário a identificá-la fizemos um pequeno tutorial. Este texto está presente no tópico 2.3 (DESCOBRINDO QUAL A PORTA DE COMUNICAÇÃO GERADA PARA O SEU ARDUINO) e será acessado caso o usuário

clique sobre o botão com um ponto de interrogação, posicionado ao lado do campo para informar a porta de comunicação utilizada.

2 TEXTOS DE APOIO PARA OS USUÁRIOS.

A seguir serão apresentados os textos de apoio ao usuário utilizados ao longo do programa.

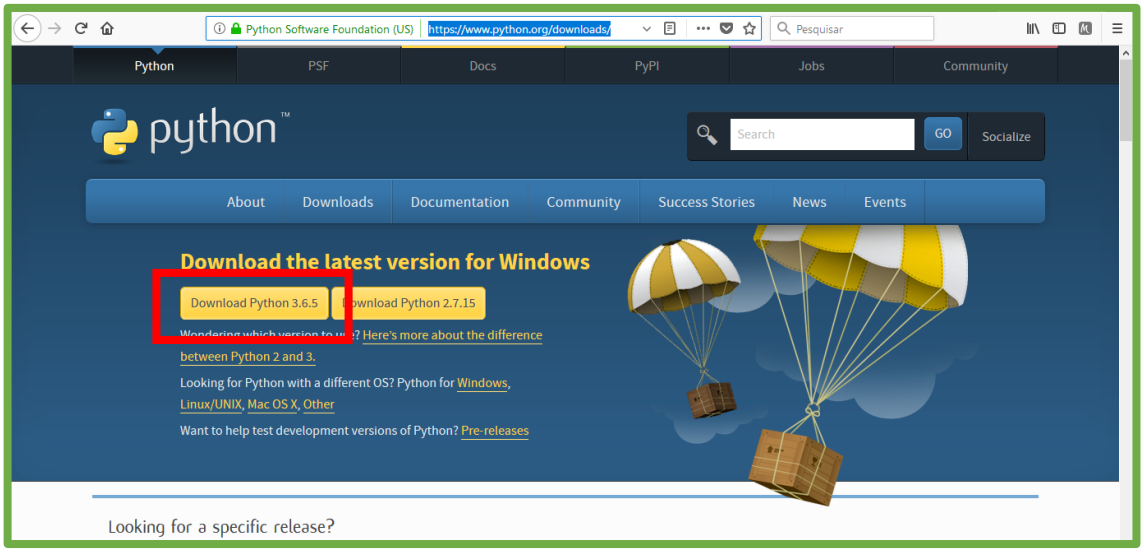
2.1 LEIA-ME ANTES DE UTILIZAR O PROGRAMA PYDUINO



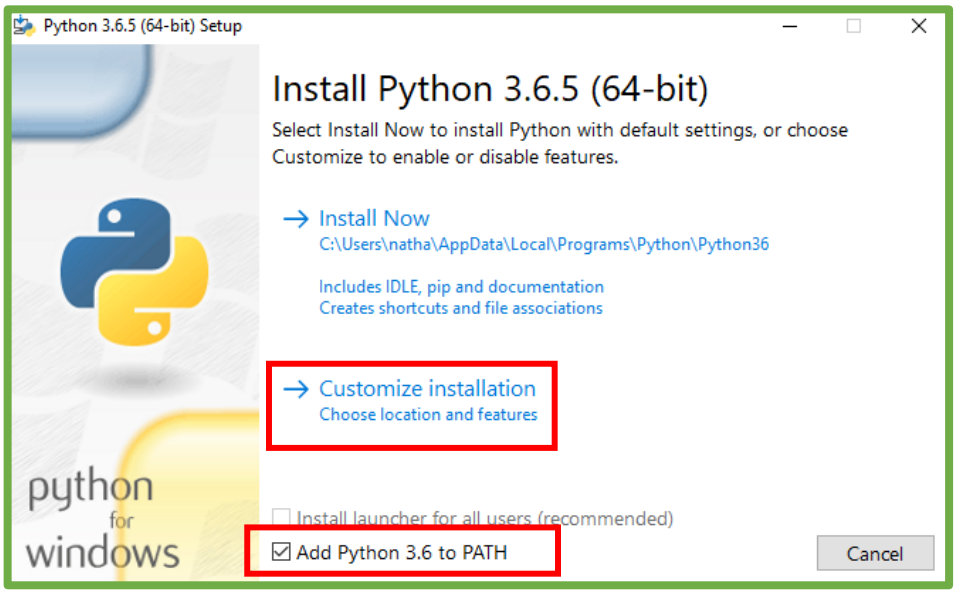
Leia me antes de utilizar
o programa PyDuino.

Para que o nosso programa funcione em seu computador é necessário que o seu computador utilize o sistema operacional Windows e que tenha o Python 3.4 ou superior instalado, além de algumas bibliotecas. Não se preocupe quanto a isso pois apresentaremos, logo a seguir, um tutorial sobre como instalar o Python e seus módulos.

Para instalar o Python, primeiro acesse o endereço a seguir <https://www.python.org/downloads/>. Lá click em Download Python 3.6.5 ou outra versão mais atual que estiver disponível.

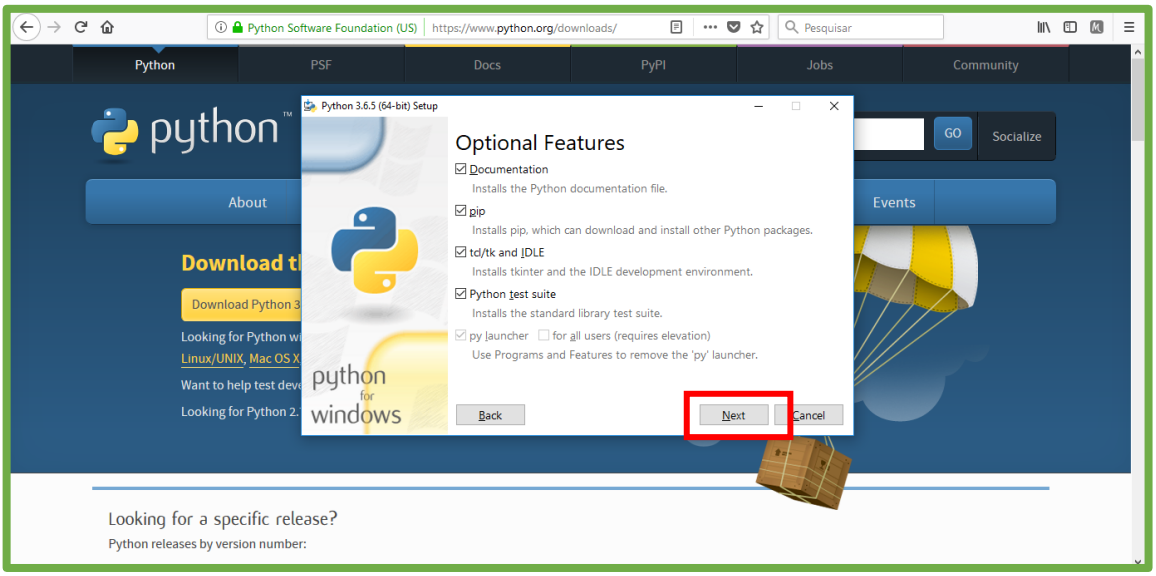


Após o término do download, execute o instalador do Python. A janela a seguir então será aberta.

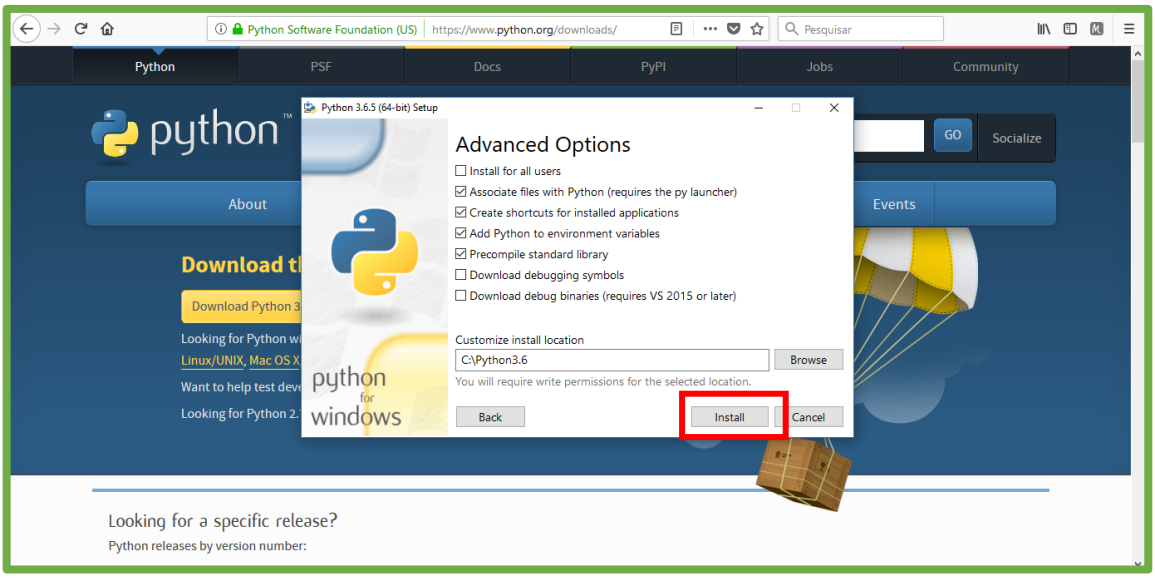


IMPORTANTE: Deixe a opção add python3.6 to PATH marcada e click sobre a opção Customize installation.

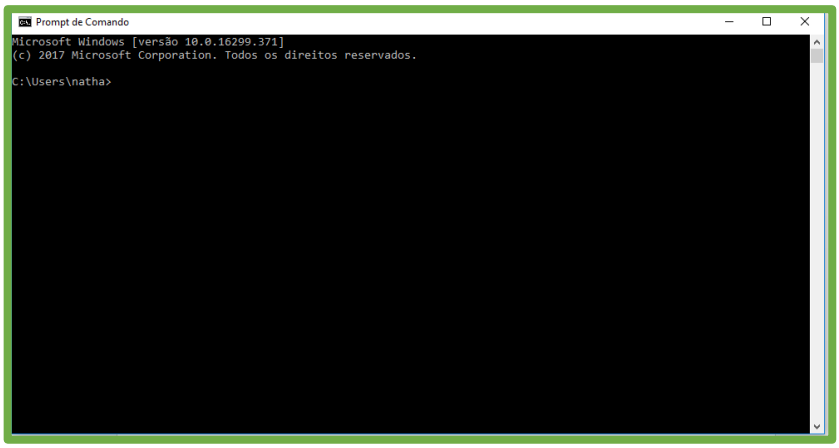
A tela seguinte será mostrada. Nela não será preciso alterar nenhuma opção. Apenas click em next.



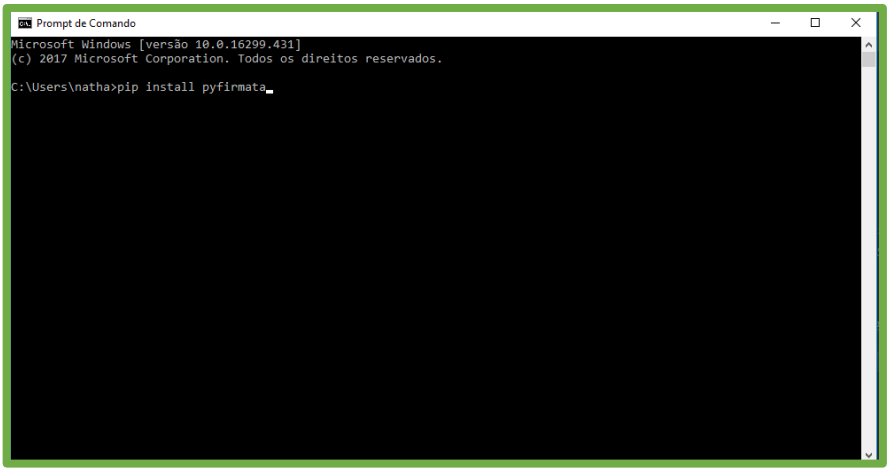
Selecione as opções mostradas na tela abaixo e selecione a pasta em que deseja instalar o Python e click em Install.



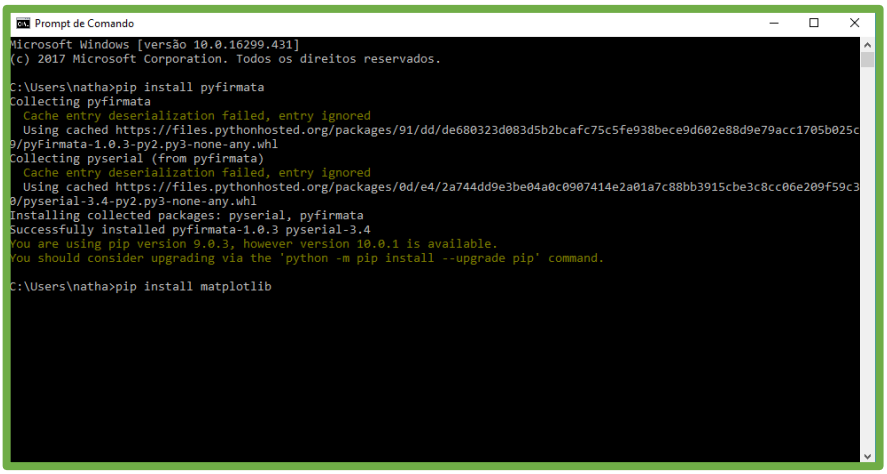
Após o término da instalação, abra o Prompt de comando do Windows para instalar as bibliotecas necessárias. Para isso click no logo do Windows e digite cmd. Ao executar o Prompt a seguinte tela será exibida.



Digite o seguinte comando: pip install pyfirmada e dê um Enter.

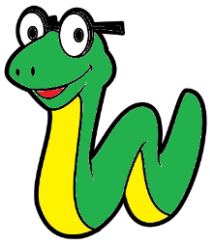


Digite o seguinte comando: pip install matplotlib e dê um Enter.



Após o término das instalações feche o Prompt de comando e o programa já estará funcionando em seu computador.

2.2 DICAS PARA A PRIMEIRA UTILIZAÇÃO DO PROGRAMA PYDUINO



Dicas para a primeira utilização do programa PyDuino.

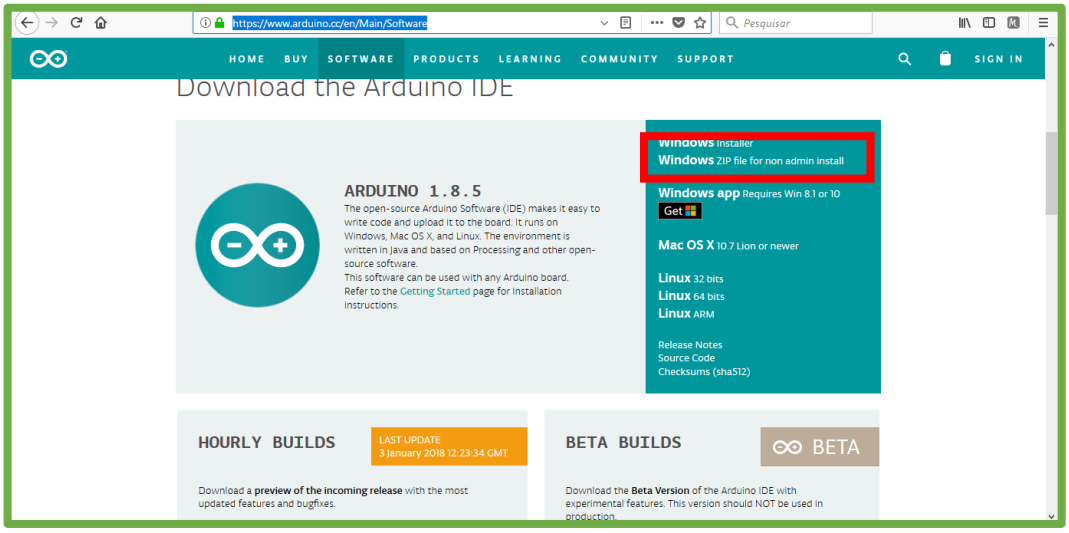
Bem vindo ao PyDuino e obrigado por utilizar está ferramenta em suas práticas educacionais.

A primeira utilização de uma placa Arduino em nosso programa exige a implementação do protocolo Firmata. Mas não se assuste, este procedimento será feito uma única vez. Ele se faz necessário porque permite a placa Arduino interagir com diferentes *softwares* do computador por meio da porta serial.

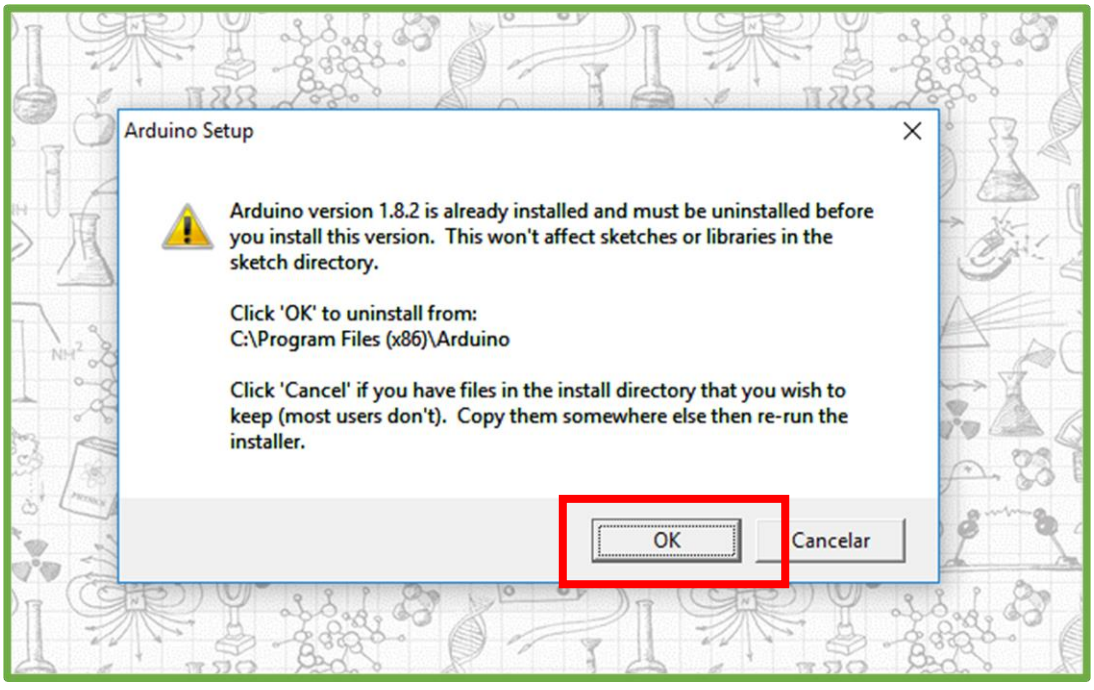
Reforço que esse procedimento precisará ser feito uma única vez e após sua realização, você poderá realizar qualquer experimento englobado por este programa, quantas vezes quiser.

Dito isso, vamos ao passo a passo:

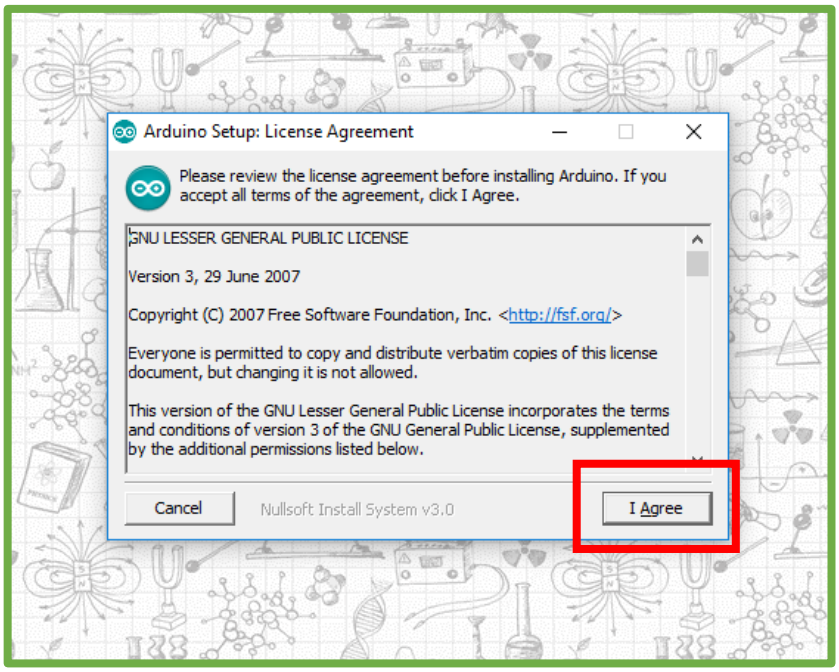
Primeiro devemos instalar a IDE do Arduino. Portanto, faça o download no endereço: <https://www.arduino.cc/en/Main/Software>. Nesta existem versões para vários sistemas operacionais. Porém o PyDuino é compatível apenas com o Windows e, portanto, você deve fazer a seleção deste sistema operacional.



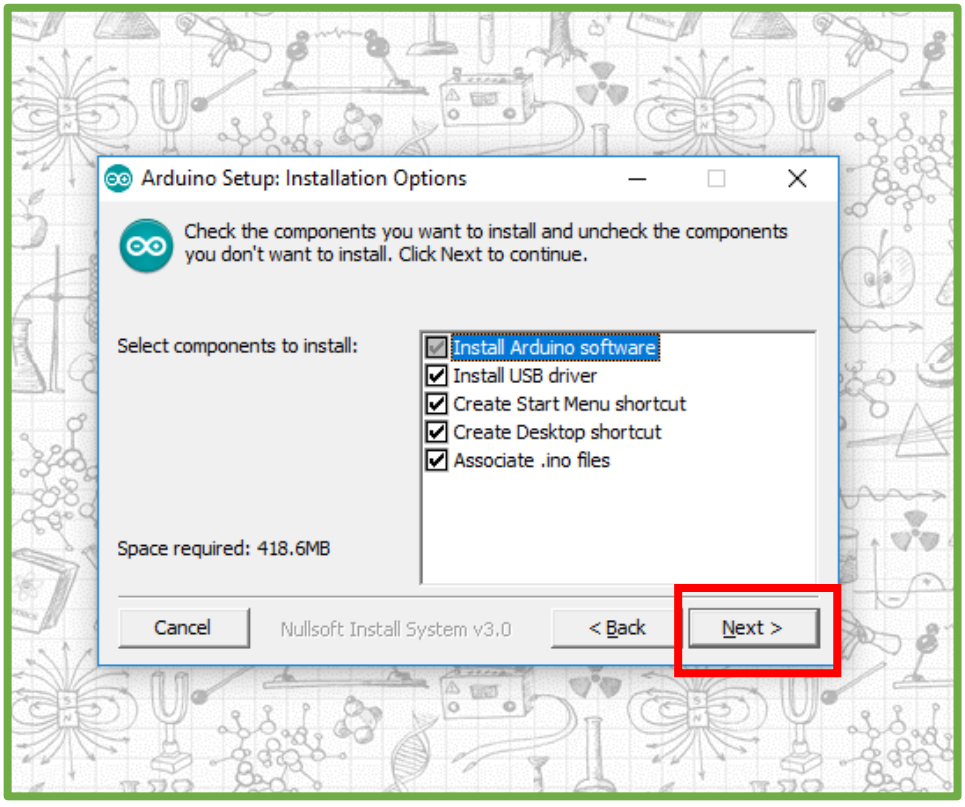
Após o término do download abra o arquivo baixado e siga os passos abaixo.



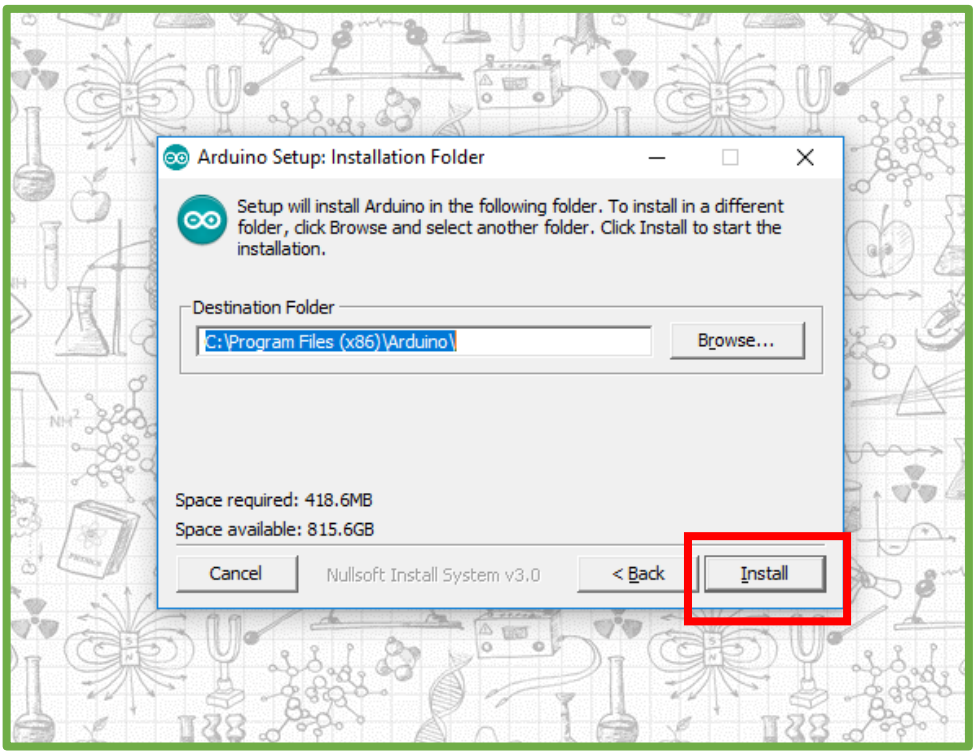
Na próxima tela, clique em “I Agree”, caso concorde com os termos de licença apresentados.



Na próxima tela clique em Next.



Agora escolha qual pasta deseja salvar os arquivos e clique em Install.

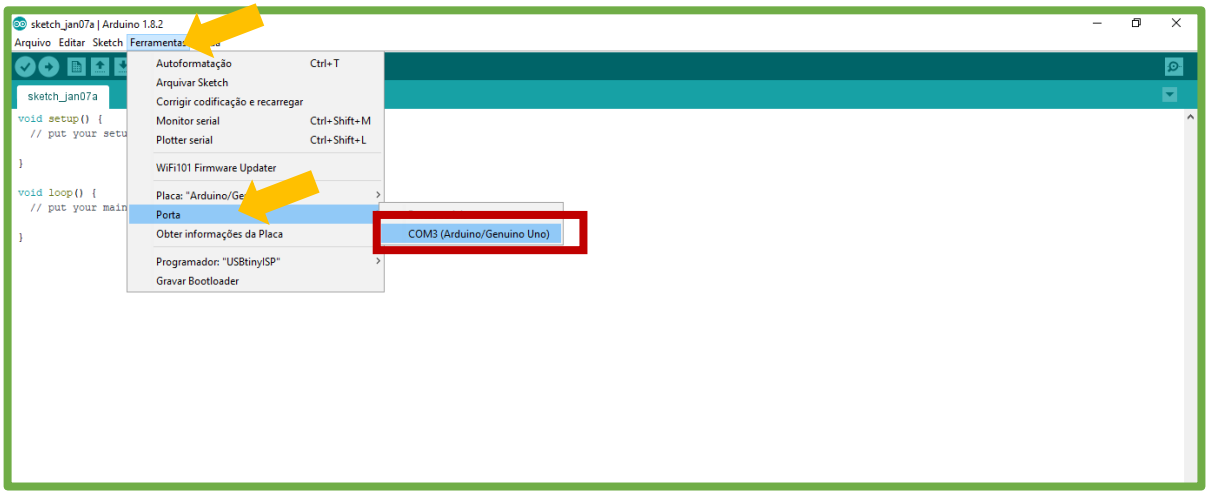


Após o termino da instalação, acople a placa Arduino em alguma entrada USB do seu computador e execute o programa.

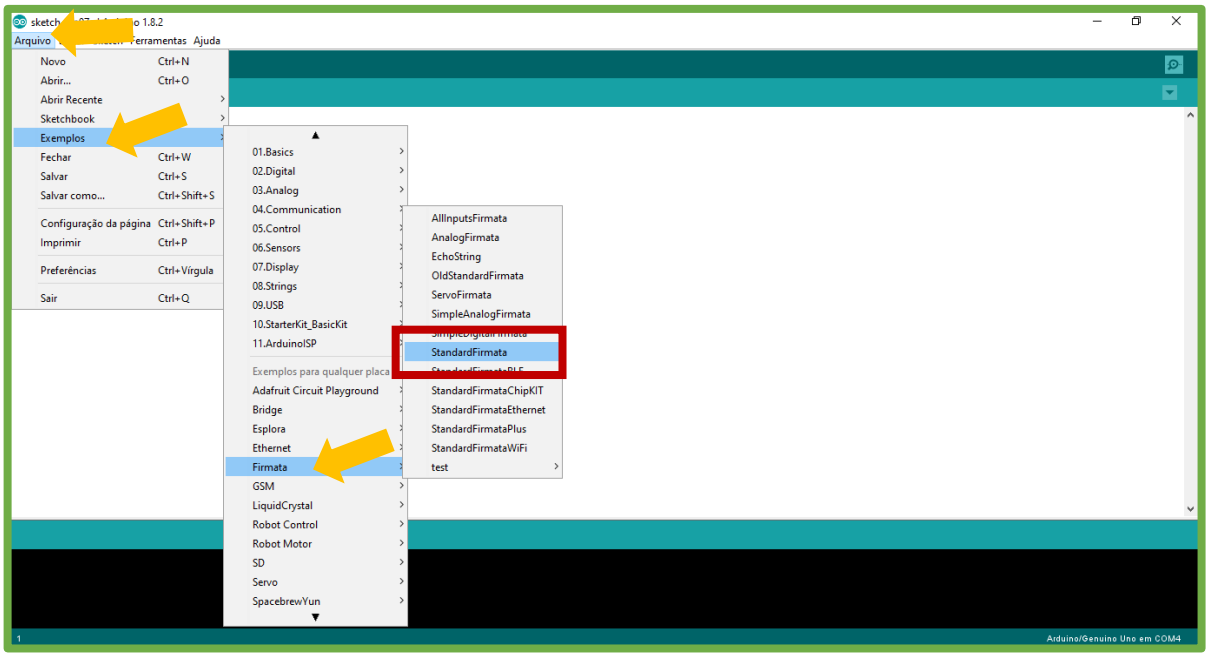
Lá selecione o menu ferramentas e pare o mouse sobre o campo Porta: o próprio *software* mostrará qual a porta está sendo utilizada. Então selecione a porta mostrada.

Fique atento a este número, pois em momentos futuros você precisará dele no PyDuino.

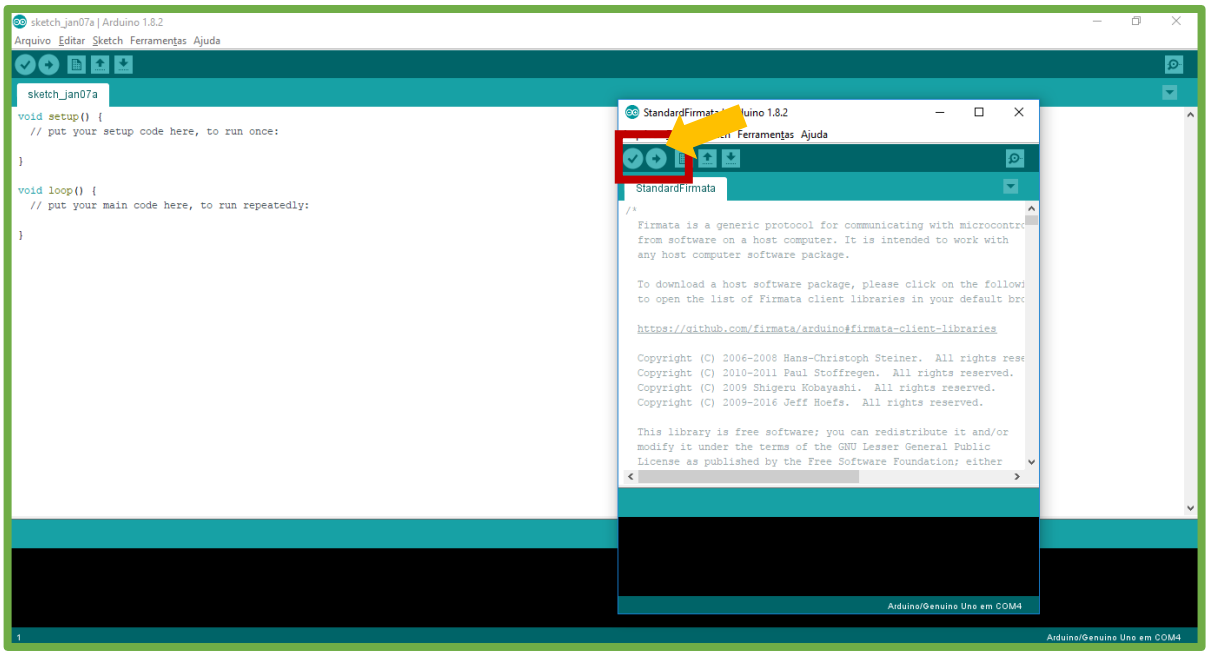
Obs: Caso a porta não apareça tente reconectar o Arduino à porta USB.



Agora, selecione o menu Arquivos. Depois pare o mouse sobre Exemplos, e então passe o mouse sobre Firmata. Por fim, selecione a opção StandardFirmata.

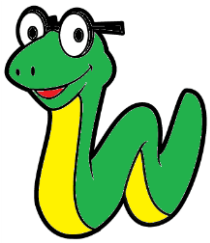


Será aberta uma nova janela. Nela clique na setinha mostrada abaixo e espere até que o código seja compilado no Arduino.



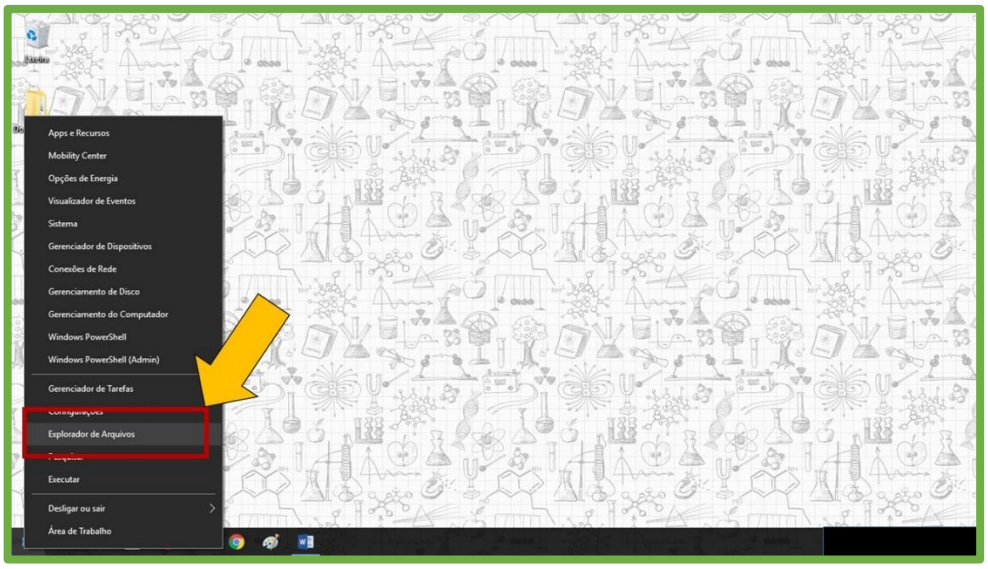
Pronto! Agora é sua placa de Arduino está pronta para ser utilizada em qualquer prática do PyDuino.

2.3 DESCOBRINDO QUAL A PORTA DE COMUNICAÇÃO GERADA PARA O SEU ARDUINO

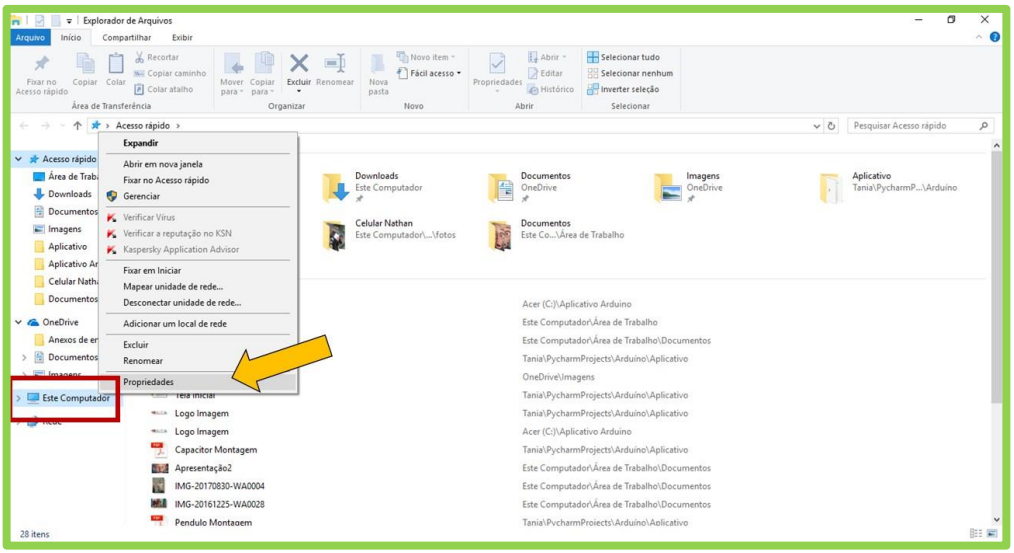


Descobrindo qual a porta de comunicação gerada para o seu Arduino

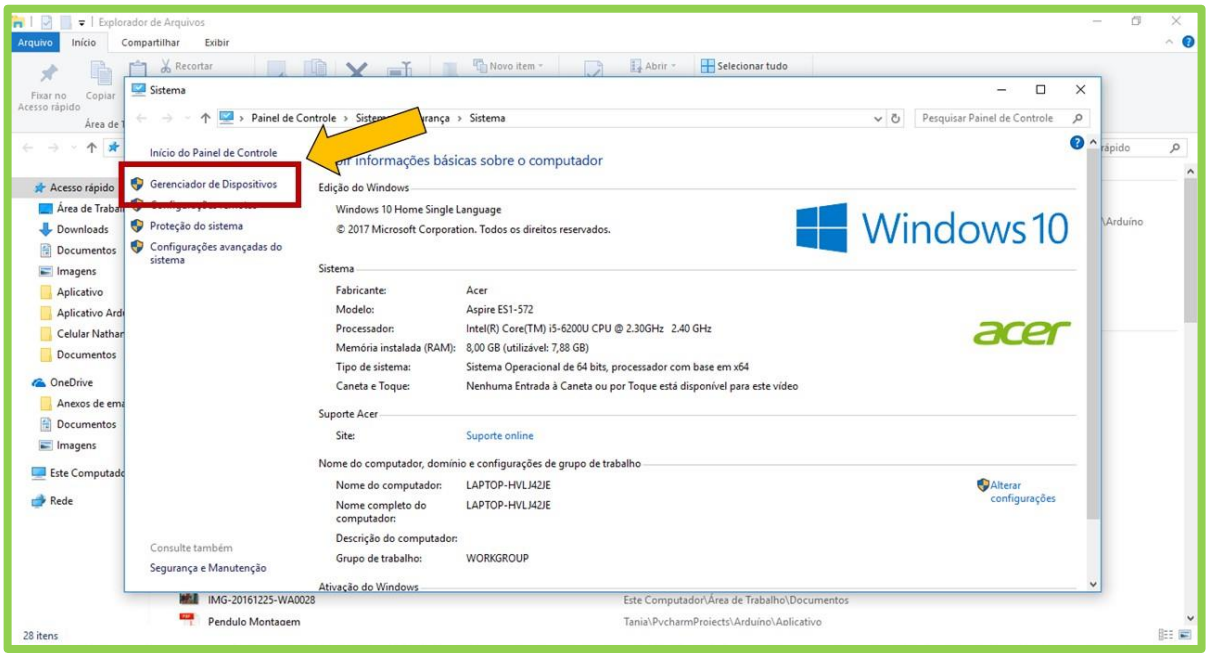
O primeiro passo é abrir a pasta *Windows Explorer ou Explorador de Arquivos*. Para isso basta dar um clique com o botão direito do mouse sobre o ícone do Windows e escolher a opção *Windows Explorer* ou *Explorador de Arquivos*.



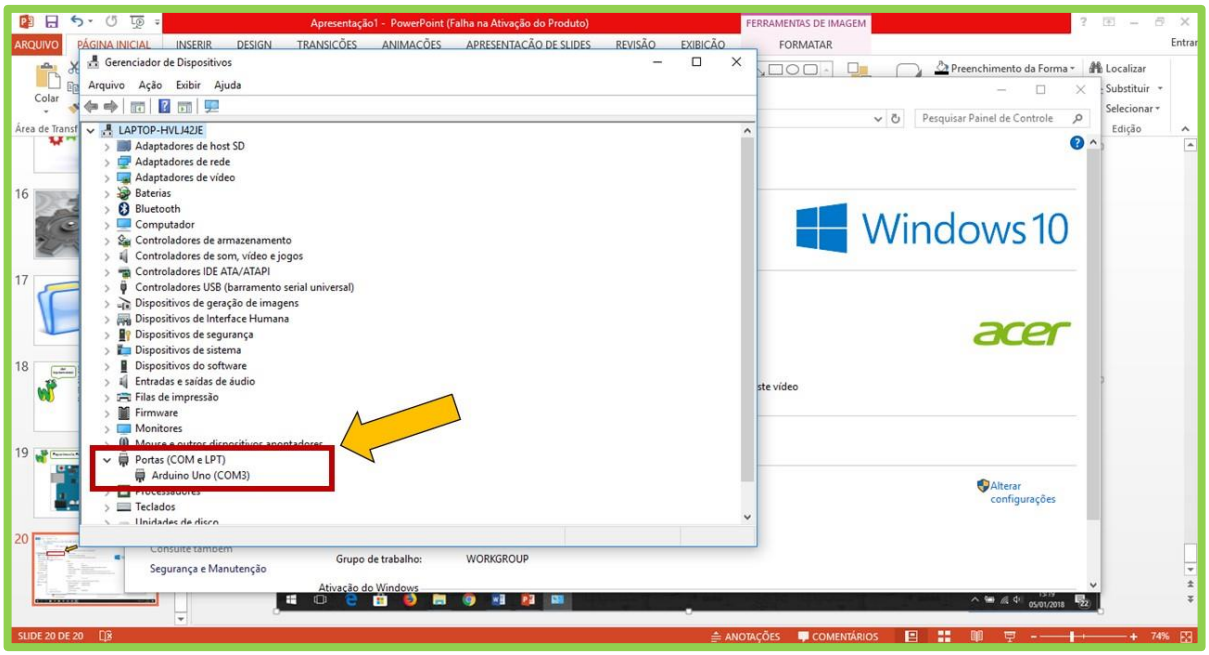
Depois clique com o botão direito sobre a pasta *Este Computador* e selecione a opção propriedades.



Agora selecione a opção *Gerenciador de Dispositivos*.



Por fim, basta olhar, no campo *Portas,* qual a porta de comunicação utilizada.



Neste exemplo a porta de comunicação criada é a COM3.

2.4 TEXTO DE APOIO PARA O EXPERIMENTO PÊNDULO SIMPLES

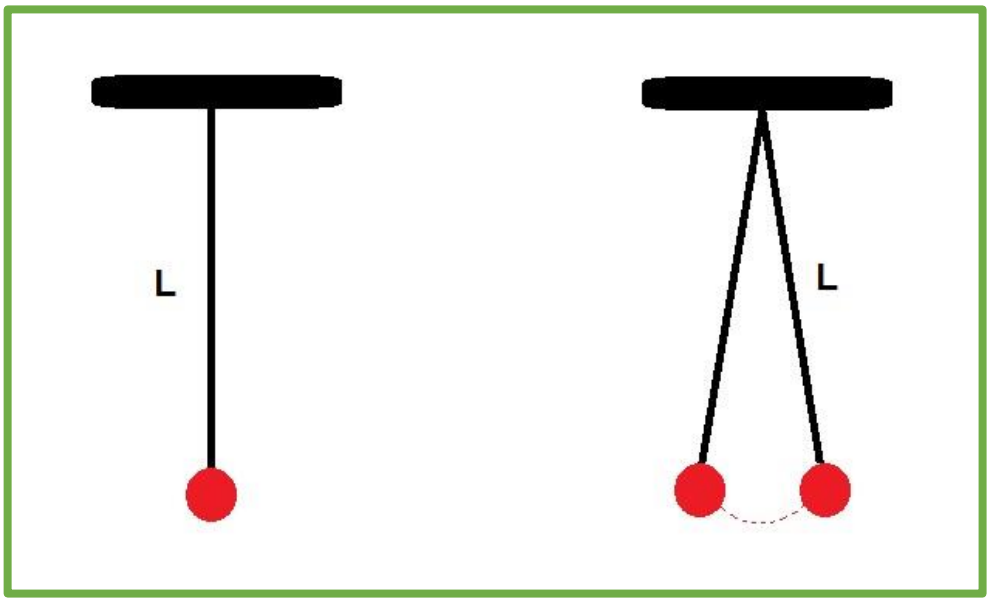
Experimento Pêndulo Simples

O Movimento Harmônico Simples:

Movimentos harmônicos simples são geralmente empregados na Física no segundo ano do Ensino Médio. Este tipo de movimento é definido como sendo simultaneamente oscilatório e periódico. Oscilatório porque o corpo vai e volta em

relação a um ponto de referência, e periódico porque se repete em intervalos de tempos iguais.

O pêndulo simples é um dos exemplos mais clássicos desse tipo de movimento, sendo uma das maneiras mais fáceis de exemplificá-lo. Ele é basicamente um corpo que oscila por meio de um fio de sustentação tendo uma de suas extremidades fixa e o movimento ocorre com um pequeno ângulo de abertura, menor que 15º, como ilustrado pela figura abaixo.



Ao abordar este conteúdo, percebíamos que os alunos traziam internalizações errôneas sobre quais são as grandezas que influenciam no tempo de oscilação do corpo. E nossa experiência mostra que apenas aulas expositivas sobre o assunto não trazem uma aprendizagem significativa.

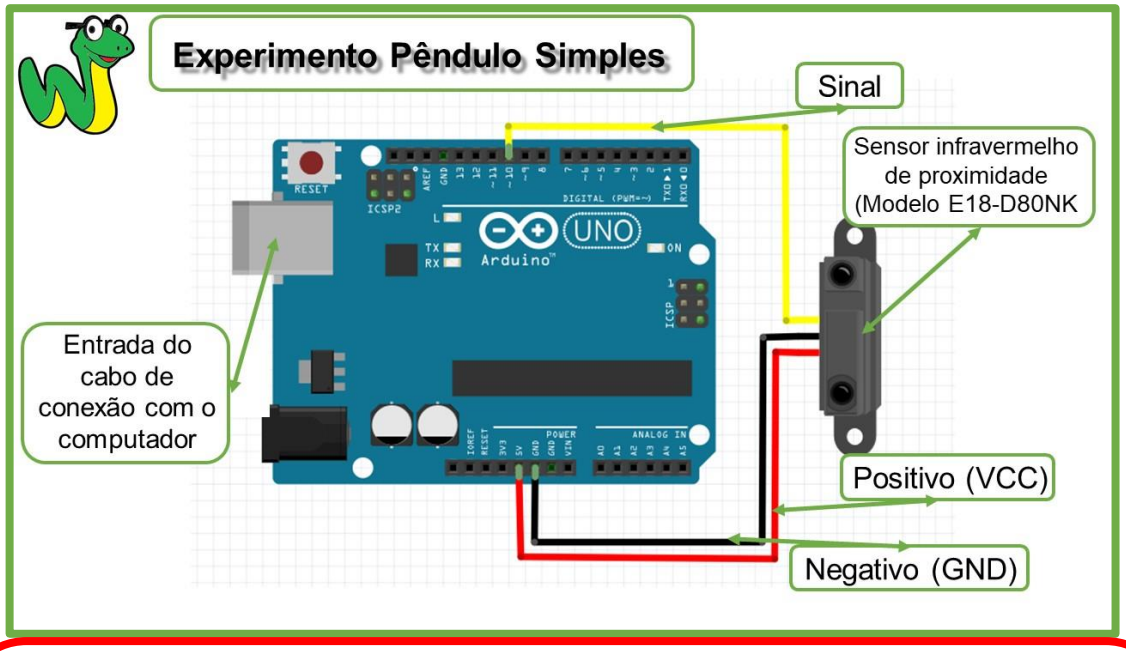
Trazer aulas experimentais sobre este tema agrega muito à aprendizagem dos alunos. Porém, práticas feitas com coleta de tempo manual muitas vezes trazem resultados imprecisos fazendo com que a prática não demonstre a teoria e, assim, não contribuindo de forma efetiva com o aprendizado dos alunos.

O PyDuino, nome que demos ao nosso programa, nos dá a possibilidade de realizar a coleta de dados deste experimento de forma automatizada por meio de um sensor que trabalha com emissão e recepção de ondas com frequência na faixa do infravermelho. Ele consegue perceber a passagem do fio em sua frente e, automaticamente, o *software* registra o instante em que isso aconteceu.

Com o PyDuino, acreditamos que o experimento possa ser feito com uma boa precisão, sendo uma ferramenta muito útil no ensino deste tópico da Física básica.

A Montagem:

Neste momento você já deve estar com a montagem do circuito pronta. Confirme se as conexões estão como as da figura abaixo. É importante dar muita atenção a esta etapa do processo pois qualquer fio mal conectado pode afetar na realização do experimento.



IMPORTANTE: Alguns desses sensores de infravermelho vêm com os fios de cores diferentes, utilizando se o azul, marrom e preto no lugar das cores indicadas no esquema. Se este for o seu caso, basta utilizar a seguinte relação. O marrom é o positivo (VCC) e no nosso esquema está representado pela cor vermelha; o azul é o negativo (GND) e está representado pelo fio preto no esquema; o preto é o fio de sinal (SINAL) e no nosso esquema está representado pela cor amarela.

Para a montagem do experimento em si vamos precisar de uma base com uma haste de sustentação. Para a realização do experimento não é necessário um aparato exatamente como o mostrado. Você pode usar o material que tiver em mãos para a construção do mesmo. O principal é que a haste fique estável, livre de vibrações durante a realização do experimento, e que o sensor fique posicionado em frente à linha, o mais baixo possível, quando o pêndulo estiver em equilíbrio.

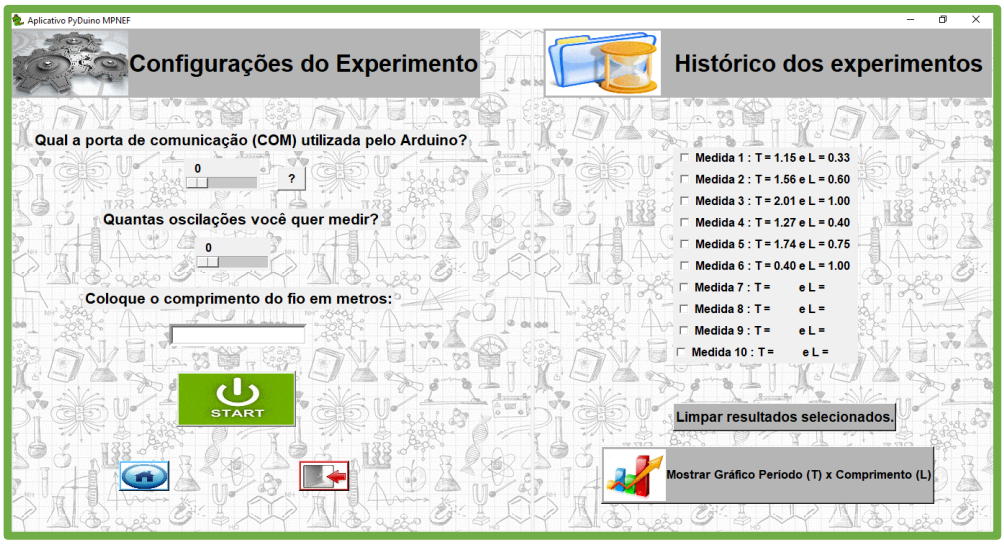
É interessante também que o fio possa ter diferentes dimensões. Neste caso aconselhamos que você não utilize fios muito curtos pois é difícil gerar oscilações com ângulos pequenos de aberturas. O ideal é que você trabalhe com comprimentos entre 60cm e 150cm.

Com a base pronta e o sensor posicionado, basta prender uma extremidade do fio no suporte a outra extremidade na chumbada (corpo de massa muito superior a massa do fio fixado utilizado no pêndulo).

Configurando o Experimento:

Após a montagem da estrutura e do circuito do experimento é hora de configurá-lo no programa. Abaixo vamos mostrar quais são as funcionalidades para isso.

A tela que virá, quando você clicar sobre o botão “Experimento”, será esta:



Na coluna do lado esquerdo estão os campos para a configuração do experimento. No primeiro campo você deve fornecer qual o número da porta de comunicação gerada para o Arduino utilizado. Caso não saiba, é só clicar no botão com o ponto de interrogação que você terá acesso ao passo a passo para a obtenção do número da porta. No segundo campo será informado quantas oscilações completas serão medidas e, por fim, qual o comprimento do fio em metros.

IMPORTANTE: O comprimento do fio deve ser medido do ponto em que ele foi fixado até o centro de massa do corpo preso.

Após a inserção dos dados, clique no botão “START” para ir à tela do experimento. A abertura da próxima tela não será imediata, pois é neste momento que toda a programação será passada para o Arduino. Isto leva apenas leva apenas alguns segundos.

Na coluna do lado direito aparece o histórico dos dez últimos experimentos realizados, mostrando para cada um o valor do período médio experimental e do comprimento do pêndulo. Caso você queira apagar algum dos valores listados, basta selecioná-lo e clicar no botão “Limpar resultados selecionados”. Optando pelo botão abaixo será mostrado o gráfico da variação do período de oscilação do pêndulo pelo seu comprimento.

O gráfico esperado tem um caráter crescente, mostrando que, para pequenas amplitudes de oscilação, o período de oscilação cresce com o comprimento do pêndulo. É importante tomar cuidado pois, apesar de parecer que estas grandezas são diretamente proporcionais, na verdade o período é proporcional à raiz quadrada do comprimento do pêndulo.

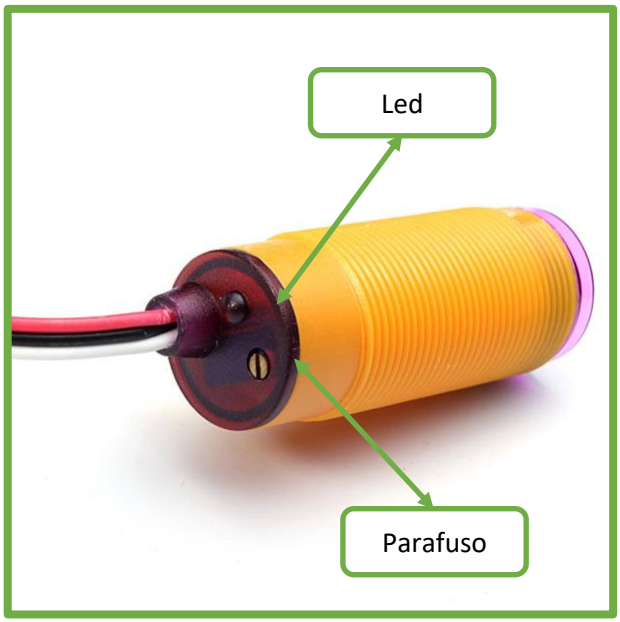
É importante evidenciar que a função T α $L^{1/2}$ não fica totalmente evidenciada uma vez que construímos o gráfico a partir de poucos valores, no máximo dez, analisando desta forma apenas uma pequena parte de todo o universo amostral.

A Realização do Experimento:

A tela de monitoramento, mostrada abaixo, exibe o resultado teórico calculado a partir da equação $T = 2\pi\sqrt{\frac{L}{g}}$, onde T é o período do movimento, L é o comprimento do fio e g a aceleração da gravidade local. Cabe frisar que esta equação é válida apenas para pequenos ângulos de oscilação. Assim, é importante que, ao realizar o experimento, se utilize sempre pequenos ângulos de aberturas.

Existe também um campo destinado a mostrar o resultado experimental onde o valor de “T” será a média dos períodos obtidos. Este valor será exibido apenas após o término das medições previamente estabelecidas.

Acreditamos que este seja o melhor momento para fazer a regulagem do sensor infravermelho. Ela se faz necessária porque o sensor pode ser ajustado para perceber objetos a uma distância de 3 a 80 cm. Entretanto, esta tarefa é bem simples de ser executada: Deixe a linha na frente do sensor e com uma ajuda de uma chave de fenda aperte ou afrouxe o parafuso localizado atrás do sensor, sem forçar, até que um led, também atrás do sensor, se acenda. Quando ele acender significa que o sensor já está captando a linha.



IMPORTANTE: Indicamos que os ajustes no sensor sejam feitos neste momento pois ele só começa a receber a alimentação do Arduino, e consequentemente só começa a funcionar, após a compilação do código no mesmo. Assim, fica mais fácil saber qual a posição certa do parafuso para o suporte feito por você.

Para iniciar o experimento, basta tirar a linha da frente do sensor, clicar no botão “Começar” e fazer o lançamento do pêndulo. Os resultados medidos serão exibidos na tela simultaneamente a realização do experimento. Apesar de alguma flutuação, o período se manterá praticamente constante durante toda a execução.

É possível que, em alguns momentos, você perceba que o pêndulo realizou uma oscilação completa, mas que a tela de resultados não foi atualizada. Isso pode acontecer caso o sensor não capte a linha por qualquer motivo que seja. Neste caso o resultado é descartado pelo programa. A tela de resultados será atualizada somente quando a leitura for feita de maneira correta. Este detalhe foi introduzido no programa para evitar erros de leitura do sensor.

Após o final das leituras previamente estabelecidas, o botão “Gráfico” ficará disponível. Nele será possível estabelecer como o período variou com o tempo. O valor esperado é que ele tenha se mantido praticamente constante durante todo o experimento.

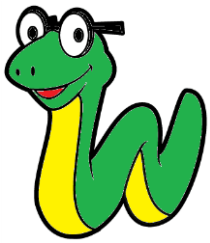
IMPORTANTE: Aconselhamos que o experimento seja repetido outra vez com o fio tendo o mesmo comprimento, porém com a massa do corpo sendo diferente. Antes, questione os alunos sobre o que eles esperam que aconteça e

depois realize novamente o experimento. O resultado obtido deve ser muito próximo ao anterior mostrando que a massa não influenciará no período do pêndulo.

Realize o experimento quantas vezes quiser com fios de tamanhos diferentes e corpos com massas diferentes. Entretanto, consideramos importante enfatizar que, além do aumento do comprimento do fio resulta em um aumento do período de oscilação do pêndulo, a aceleração da gravidade também é capaz de modificar o período de oscilação, sendo o período inversamente proporcional à raiz quadrada da aceleração da gravidade.

Sugerimos que os testes iniciais não sejam feitos juntos aos alunos, e que o professor se ambiente com o programa antes da utilização em sala de aula para evitar qualquer imprevisto e poder corrigir alguma falha que por algum motivo possa aparecer.

2.5 TEXTO DE APOIO PARA O EXPERIMENTO CARGA E DESCARGA DE UM CAPACITOR

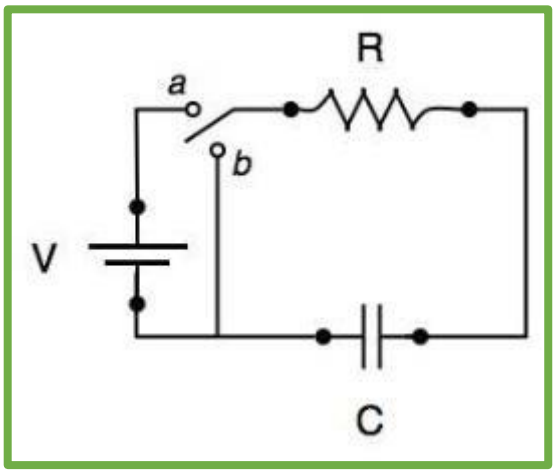


Experimento Carga e Descarga de um Capacitor

O Circuito RC (Resistor e Capacitor):

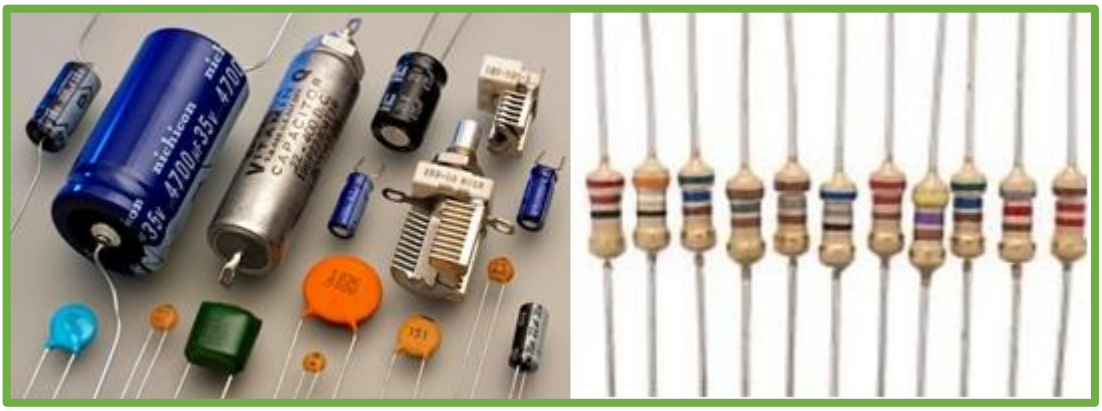
O experimento sobre capacitores que vamos trabalhar corresponde ao circuito do tipo RC. Este circuito eletrônico é constituído, além da fonte de tensão “V”, de um resistor “R” e um capacitor “C”. Esse tipo de montagem compõe a maioria dos aparelhos eletroeletrônicos presentes no nosso dia a dia. Segue abaixo montagem esquemática deste tipo de circuito. Observe que, quando a chave está ligada em a, o

capacitor está sendo carregado pela fonte V. Ao passar a chave para a posição b, a fonte é desligada e o capacitor passa a alimentar o circuito sendo assim descarregado.



Um resistor é um dispositivo eletrônico que, quando atravessado por uma corrente elétrica, é capaz de transformar energia elétrica em energia térmica e, consequentemente, limitar o valor da corrente que passa pelo circuito. Ele é feito de material condutor possuindo uma resistência elétrica que, no nosso caso, será conhecida. Já o capacitor é um dispositivo capaz de armazenar energia elétrica pelo acumulo de cargas, podendo liberá-las muito mais rapidamente se comparado a uma fonte de tensão tradicional de mesma voltagem.

Segue abaixo exemplos de capacitores e resistores de vários tipos.



Tipos de capacitores. Tipos de Resistores

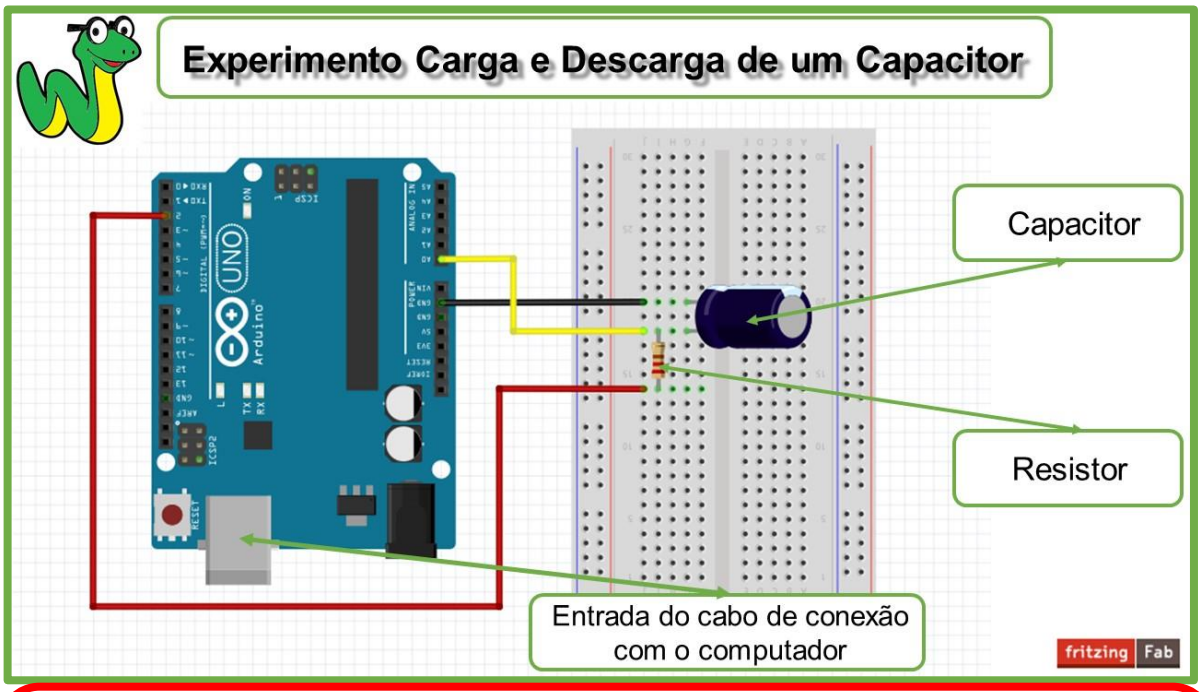
Ao contrário do que a maioria dos alunos acaba deduzindo, o ciclo de carga e descarga de um capacitor não acontece de forma linear, mas sim de forma exponencial. Por isso, é importante trabalhar este conteúdo de forma experimental, sobre tudo tratando os resultados de forma gráfica. Consideramos que esta abordagem ajuda em muito no processo de ensino aprendizagem.

O PyDuino auxiliará você nesta tarefa pois, além dele fazer a coleta de dados de forma autônoma, os resultados serão mostrados simultaneamente a

realização do experimento. Outra vantagem é que após o término dos ciclos de carga e descarga previamente estabelecidos por você, será possível visualizar os resultados de forma gráfica. Os gráficos exibidos serão da tensão, da carga nas placas do capacitor e da corrente no circuito, sendo estes dois últimos frutos de cálculos baseados em resistores ôhmicos.

A Montagem:

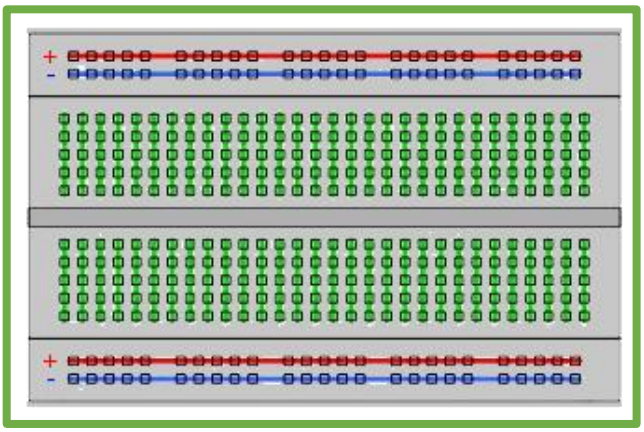
Neste momento você já deve estar com a montagem do circuito pronta. Confirme se as conexões estão como as da figura abaixo. É importante dar muita atenção a esta etapa do processo pois qualquer fio mal conectado pode afetar na realização do experimento.



IMPORTANTE: Cuidado ao ligar o capacitor: no esquema acima, é preciso prestar atenção a sua polaridade. O lado em destaque do capacitor, com detalhe em branco, deve ser ligado no GND do Arduino enquanto a outra na porta analógica A0.

Caso você não esteja familiarizado com o uso da protoboard, cabe aqui uma breve explicação. As duas primeiras e duas últimas fileiras da placa têm os furos conectados horizontalmente. Estas estão representadas no esquema abaixo pelas cores azul e vermelho. Elas geralmente são utilizadas para distribuir o polo positivo e negativo. Porém, estas fileiras não foram utilizadas neste experimento. Já os demais

furos são conectados verticalmente de 5 em 5, representadas no esquema abaixo pela cor verde.



O que faz com que o circuito esteja hora carregando o capacitor e hora o descarregando é a porta digital A2 do Arduino. É ela que faz o papel da chave do circuito RC.

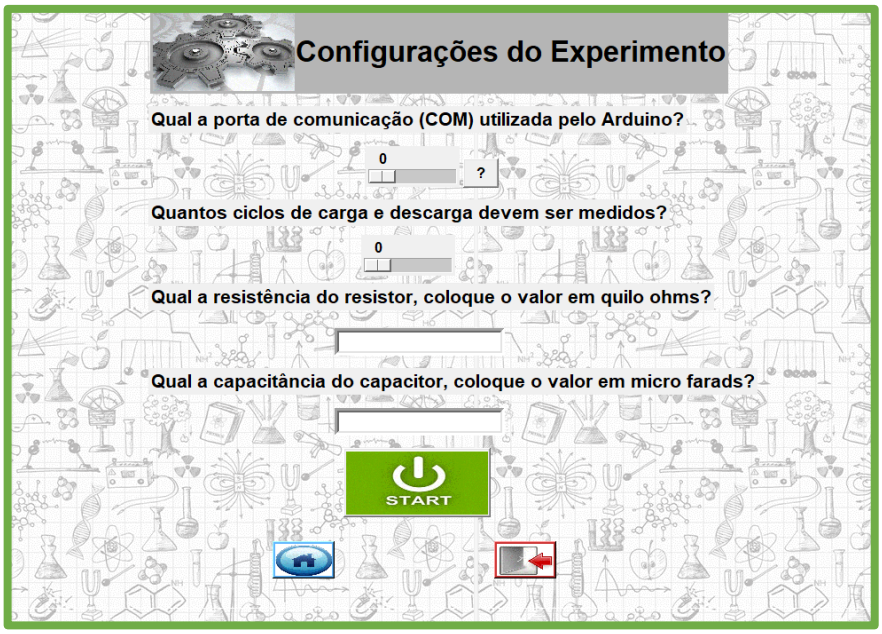
Uma porta digital só trabalha com dois tipos de valores: 1 (HIGH) ou 0 (LOW). O sinal 1 (HIGH), para este tipo de porta, é análogo à posição ligada da chave do circuito RC, ou seja, a porta fornece tensão ao circuito carregando o capacitor. Quando o capacitor está próximo de sua carga máxima, o *software* é programado para mudar o sinal da porta para 0 (LOW) automaticamente. Assim a porta não fornece mais diferença de potencial ao circuito, funcionando como a posição desligada da chave do circuito RC. Nessa posição o capacitor será descarregado.

O *software*, ao reconhecer que o capacitor está com tensão em seus polos próximo a zero, faz a mudança automaticamente do sinal da porta para 1 (HIGH). Neste momento é completado um ciclo de carga e descarga do capacitor. Este procedimento será repetido até que o número de ciclos preestabelecidos pelo usuário sejam atingidos.

Configurando o Experimento:

Após a montagem do circuito do experimento é hora de configurá-lo no *software*. Abaixo vamos mostrar quais são as funcionalidades para isso.

A tela que virá, quando você clicar sobre o botão “Experimento” será está:



No primeiro campo você deve fornecer qual o número da porta de comunicação gerada para o Arduino utilizado. Caso não saiba, é só clicar no botão com o ponto de interrogação que você terá acesso ao passo a passo para a obtenção do número da porta.

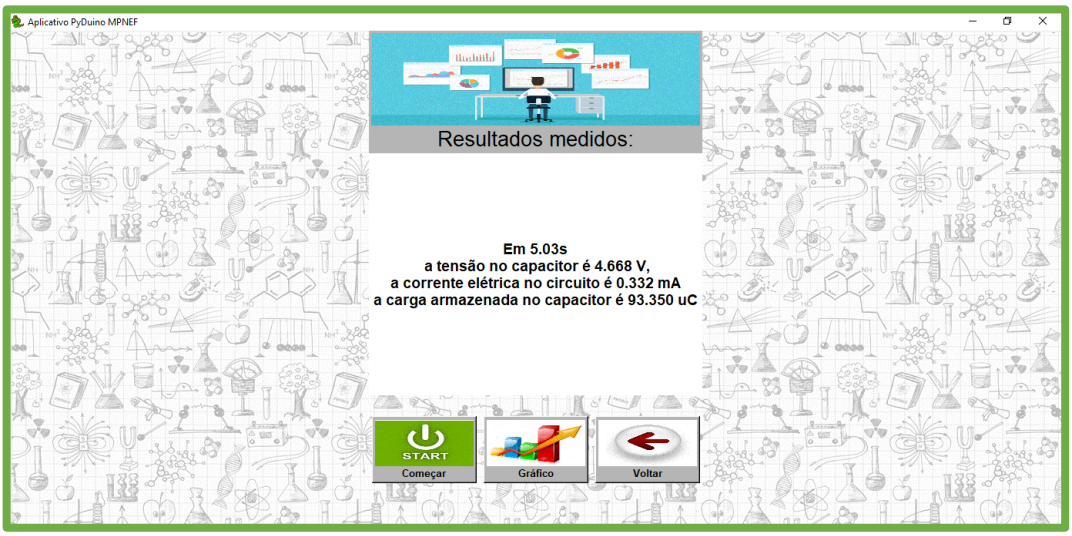
No segundo campo será informado quantas ciclos completos de carga e descarga serão medidos até o fim do experimento. No terceiro é necessário informar qual a resistência elétrica do resistor em quilohm e, por fim, qual a capacitância do capacitor utilizado em microfarads. Utilizamos para a captura de tela um resistor de 1 quilohm e um capacitor eletrolítico de 20 microfarads.

IMPORTANTE: O tempo gasto no ciclo depende de qual resistor e capacitor foram escolhidos. Apesar de esta ser uma escolha aberta ao usuário, nossos testes mostram que é melhor escolher resistores com resistência elétrica maiores que 1 quilohm e capacitores com capacitâncias pequenas na casa dos microfarads. A opção por essa faixa de valores garante que o tempo dos ciclos não seja muito pequeno e que o fenômeno possa ser plenamente estudado.

Após a inserção dos dados, clique no botão “START” para ir para a tela do experimento. A abertura da próxima tela não será imediata, pois é neste momento que toda a programação será passada para o Arduino. Mas não se preocupe, isto leva apenas alguns segundos.

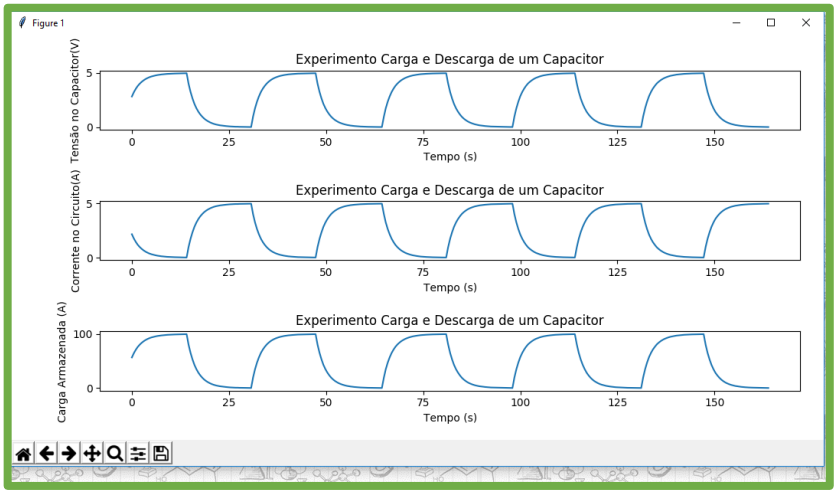
A Realização do Experimento:

Já na próxima tela basta clicar no botão “Começar” para que comece efetivamente a realização do experimento. Assim que o experimento for iniciado, será mostrado na tela o tempo decorrido, o valor da tensão do capacitor medida pelo Arduino e o resultado dos cálculos da carga armazenada no capacitor e da corrente no circuito.



Assim que tiverem terminado os ciclos de carga e descarga preestabelecidos, a coleta de dados parará automaticamente e será exibida uma mensagem de aviso. Agora basta clicar sobre o botão Gráfico para visualizar os resultados na forma de um gráfico ilustrando a variação exponencial das variáveis trabalhadas.

O programa exibirá os gráficos das três grandezas simultaneamente, sendo possível redimensionar a janela além de dar zoom em alguma parte específica. O gráfico abaixo é resultado da contabilização de 5 oscilações.



Realize o experimento quantas vezes quiser com capacitores e resistores diferentes. Observe que, apesar das grandezas continuarem com a mesma curva de variação, o tempo de carga, a carga máxima atingida pelo capacitor e a corrente máxima no circuito variarão para cada par de capacitor e resistor escolhidos.

2.6 TEXTO DE APOIO PARA O EXPERIMENTO ABSORÇÃO DE RADIAÇÃO ELETROMAGNÉTICA

Experimento Absorção
de Radiação Eletromagnética

Trocas de Calor:

Quando dois corpos, com diferentes temperaturas, estão termicamente em contato entre si e isolados em relação ao exterior, eles trocam energia térmica em forma de calor. Este flui do corpo mais quente, diminuindo assim sua temperatura, e indo para o corpo mais frio que consequentemente tem a sua temperatura aumentada. Este fluxo de calor se dará pelo tempo necessário até que os corpos atinjam a mesma temperatura, ou seja, até que eles atinjam o equilíbrio térmico.

O calor pode ser transferido por três vias: condução, convecção e radiação, sendo que as duas primeiras maneiras necessitam de um meio material para acontecerem. Na condução o calor é transferido entre as partículas dos corpos, predominando em corpos que estejam no estado sólido. Ao colocarmos uma das extremidades de um corpo em contato com uma fonte com maior temperatura, há uma elevação na agitação das partículas desta extremidade do corpo. Estas partículas mais agitadas começam a colidir com as demais partículas do corpo fazendo com que elas também aumentem sua agitação.

A convecção acontece apenas em meio fluídos, ou seja, em gases e em líquidos. O calor se propaga via a movimentação do próprio meio. Esta ocorre devido a diferença de densidade entre a parte de maior temperatura e a de menor temperatura do fluído. A parte do fluxo próxima a fonte quente é aquecida, elevando assim sua temperatura e dilatando-se. Desta forma a respectiva densidade diminui. A parte menos densa tende a subir, levando consigo o calor anteriormente presente na fonte quente. Já a parte com menor temperatura, e consequentemente mais densa, tende a descer e ficar próxima a fonte quente, completando um ciclo. E neste movimento ocorre a transferência de calor da parte quente fluído para os corpos mais frios em contato.

Já a transferência de calor por radiação ocorre via propagação de ondas eletromagnéticas. Como estas são capazes de se propagar pelo vácuo, então não há necessidade de um meio material para que este tipo de transferência de calor possa correr. O exemplo mais clássico que podemos dar é o Sol emitindo calor para o nosso planeta e este último absorvendo parte desta radiação. Entretanto, se sabe que qualquer corpo com uma temperatura superior ao zero absoluto emite calor por radiação eletromagnética. Além disso, essa emissão é tão maior quanto maior for a temperatura do corpo. Uma outra propriedade importante dos corpos está relacionada ao fato deles, além de emitirem radiação eletromagnética, também poderem absorvê-las.

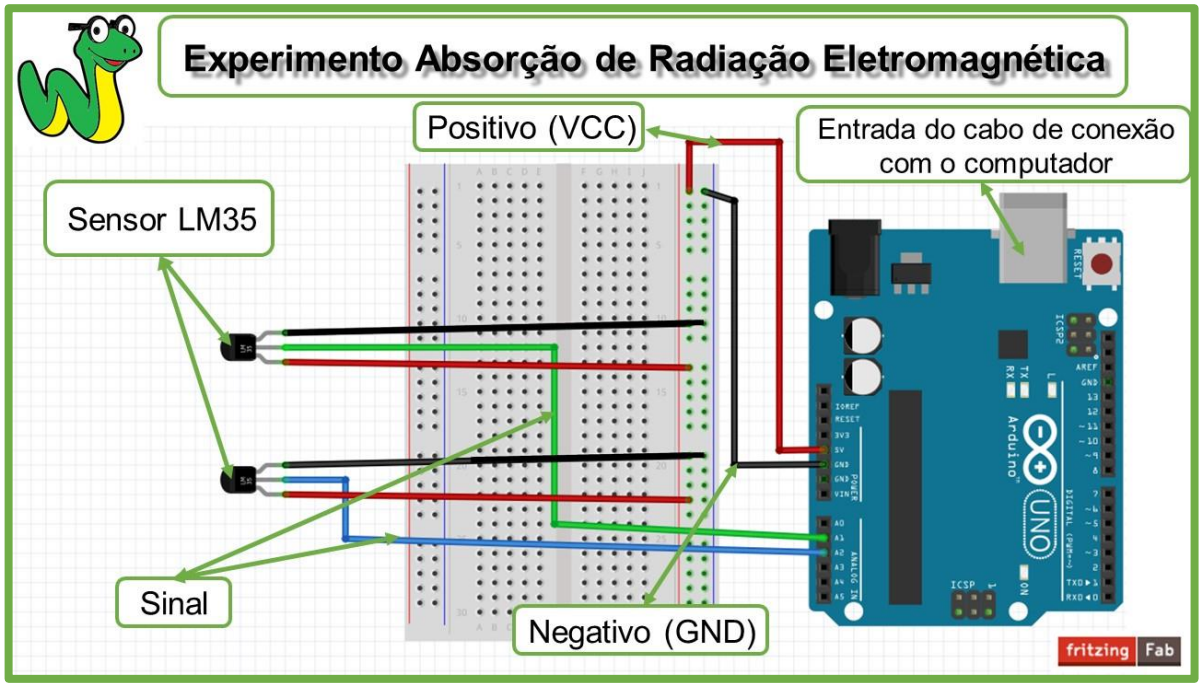
No nosso experimento usaremos uma lâmpada incandescente que, de forma sucinta, emite luz por aquecer um filamento de tungstênio até torná-lo incandescente.

Como é bem conhecido, nem toda radiação eletromagnética incidente sobre um corpo é absorvida pelo mesmo. De fato, se sabe que uma parte da radiação incidente é refletida. É este fato que nos permite enxergar os corpos com as cores que lhe são características. Cada cor de luz corresponde a uma frequência ou um comprimento de onda específico. Este comprimento de onda para o espectro da luz na faixa do visível vai do vermelho até o violeta.

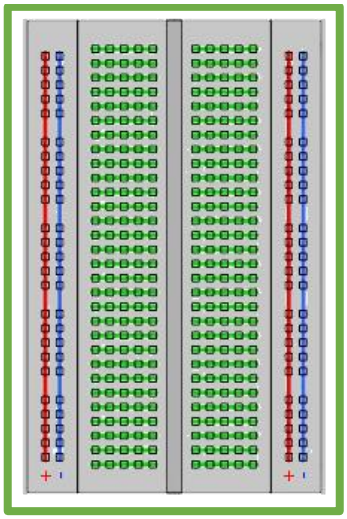
Objetos brancos refletem praticamente todos os comprimentos de ondas visíveis. Portanto, quase não absorvem radiação eletromagnética na faixa do visível podendo, entretanto, absorver radiações em outras faixas de frequência. Objetos pretos refletem quase nenhuma radiação na faixa do visível, absorvendo-as quase que completamente.

A Montagem:

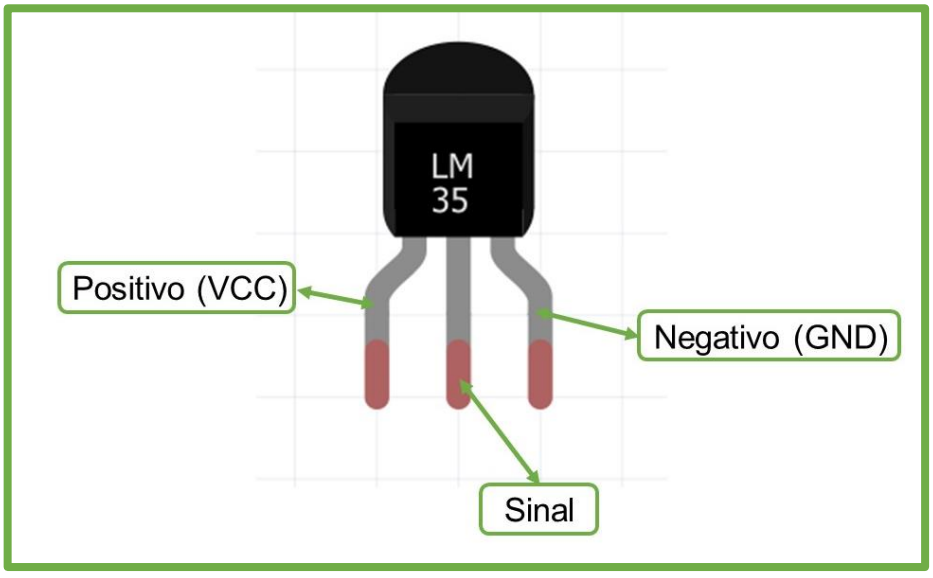
Neste momento você já deve estar com a montagem do circuito pronta. Confirme se as conexões estão como as da figura abaixo. É importante dar muita atenção a esta etapa do processo pois qualquer fio mal conectado pode afetar na realização do experimento.



Caso você não esteja familiarizado com o uso de protoboard, cabe aqui uma breve explicação. As duas primeiras e duas últimas fileiras da placa tem os furos conectados verticalmente. Estas estão representadas no esquema abaixo pelas cores azul e vermelho. Elas geralmente são utilizadas para distribuir o polo positivo e negativo. Neste caso utilizamos a protoboard apenas para fazer esta distribuição, usamos o fio vermelho para distribuir o polo positivo (5V) e o fio preto para distribuir o negativo (GND). Já os demais furos são conectados horizontalmente de 5 em 5, representadas no esquema abaixo pela cor verde, neste experimento não utilizamos nenhum destes campos.



IMPORTANTE: O sensor de temperatura utilizado, o LM35, possui polaridade. Portanto é preciso ficar atento ao conectá-lo. Abaixo segue uma imagem esquemática do sensor identificando cada um dos seus terminais. O posicionamento do sensor no esquema se refere a parte frontal do mesmo, que é plana.



Os sensores serão acoplados as placas metálicas, para aferirem a temperatura delas. Portanto é interessante usar fios um pouco maiores nos mesmos, de pelo menos 50 cm.

Prenda o sensor atrás das placas metálicas, utilizamos fita adesiva neste processo. O experimento fica mais preciso caso você utilize pasta térmica entre a placa e o sensor. Esta pasta facilita o fluxo de calor entre a placa e o sensor fazendo

com que ele consiga leituras mais próximas às temperaturas das placas. Depois de prendê-lo, coloque um pedaço de isopor atrás do sensor, afim de isolá-lo termicamente.

Pinte cada placa de uma cor diferente. Inicialmente sugerimos que utilize uma branca e outra preta já que o fenômeno será mais evidente. E agora basta colocar a lâmpada a mesma distância das duas placas metálicas para realizar o experimento.

Configurando o Experimento:

Após a montagem do circuito do experimento é hora de configurá-lo no *software*. Abaixo vamos mostrar quais são as funcionalidades para isso.

A tela que virá, quando você clicar sobre o botão “Experimento”, será esta:

No primeiro campo você deve fornecer qual o número da porta de comunicação gerada para o Arduino utilizado. Caso não saiba é só clicar no botão com o ponto de interrogação que você terá acesso ao passo a passo para a obtenção do número da porta.

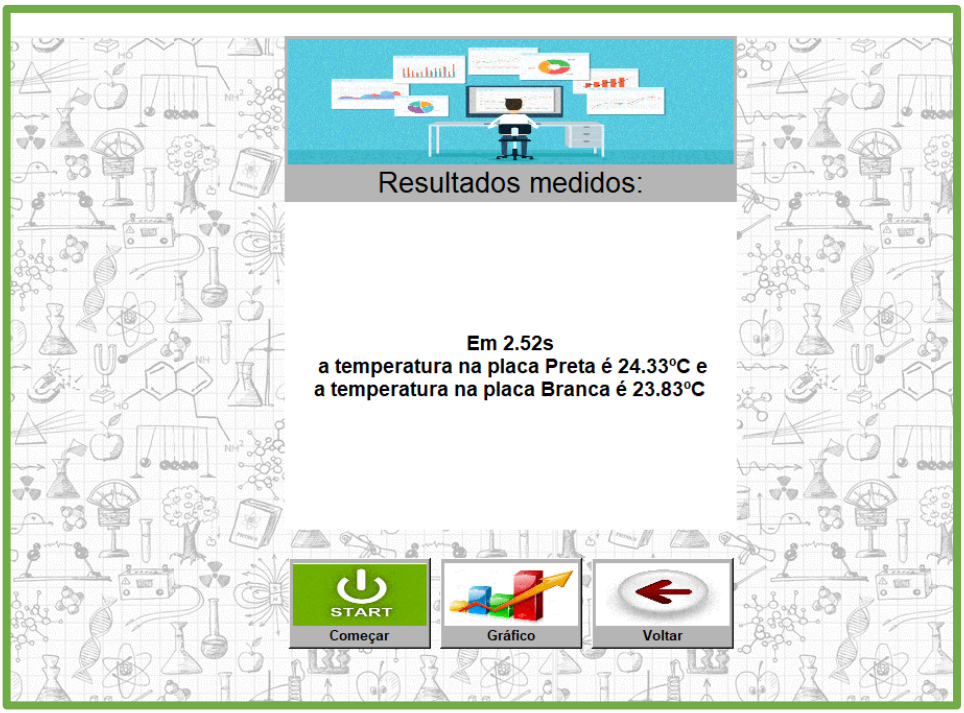
O segundo campo é destinado a informar por quanto tempo deseja aferir a temperatura dos corpos. Aqui é preciso informar o tempo em minutos.

O terceiro e quarto campos servem para informar quais as cores que as placas foram pintadas. Esse campo serve apenas para personalizar os textos e legendas durante a execução do experimento.

Após a inserção dos dados, clique no botão “Começar” para ir para a tela do experimento. A abertura da próxima tela não será imediata, pois é neste momento que toda a programação será passada para o Arduino. Mas não se preocupe, isto leva apenas alguns segundos.

A Realização do Experimento:

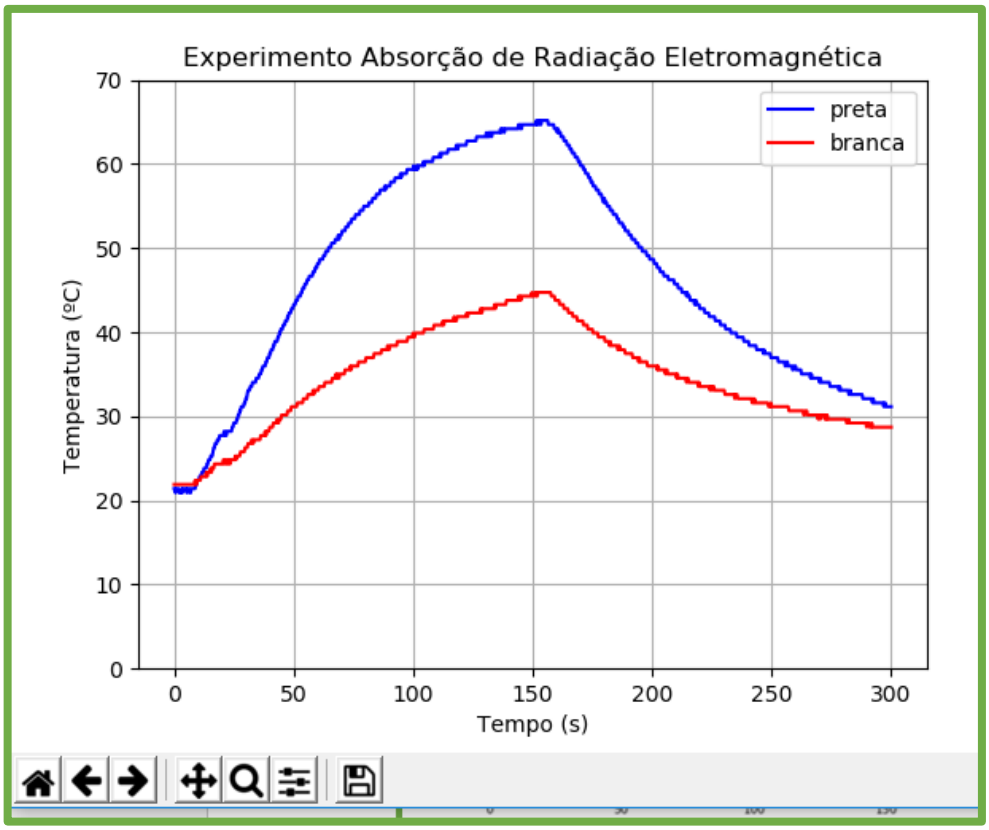
Já na próxima tela basta clicar no botão START para que comece efetivamente a realização do experimento. Assim que o experimento for iniciado será mostrado na tela o tempo decorrido e o valor da temperatura medida em cada placa.



O sensor que utilizamos apresenta uma precisão 0,5 ~~ºC~~ °C para mais ou para menos. Esta imprecisão faz com que as leituras apresentem uma pequena oscilação. Pode ser que até mesmo no início do experimento, quando as placas provavelmente estarão em equilíbrio térmico com o ambiente, as leituras apresentem uma pequena diferença. É importante entender que uma diferença de até 1 °C ainda está dentro da margem de erro dos sensores.

Assim que tiver decorrido o tempo preestabelecido a coleta de dados parará automaticamente e será exibida uma mensagem de aviso. Pode ser que o experimento dure alguns décimos de segundo a mais do que o tempo estipulado, pois a aferição da temperatura é feita a cada 0,3 segundos. Agora basta clicar sobre o botão Gráfico para visualizar os resultados na forma gráfica.

O programa exibirá as duas curvas de temperatura das duas placas no mesmo gráfico para ficar mais fácil comparar a variação da temperatura das duas placas, seja quando a lâmpada estiver ligada ou desligada. O gráfico abaixo ilustra como os resultados serão mostrados.



Observe que os gráficos apresentam uma pequena flutuação praticamente durante todo o tempo devida a margem de erro dos sensores.

Para abordar melhor o assunto, realize o experimento quantas vezes quiser com placas de cores diferentes.

3. TEXTOS MATERIAIS NECESSÁRIOS

A seguir serão apresentados os textos utilizados no programa referentes aos materiais utilizados em cada experimento.

3.1. PÊNDULO SIMPLES

Material Necessário

Experimento

Pêndulo Simples

Seguem abaixo os materiais necessários para a realização do experimento de pêndulo simples:

- 01 Arduino Uno;

- 01 Cabo USB de conexão (vem junto ao Arduino);

- 01 base com haste. Segue exemplo;

- Chumbadas de massas diferentes;

- Jumpers para ligação (macho/fêmea);

- 02 metros de linha (de preferência linha 10 de pipa);

- 01 sensor infravermelho de proximidade (modelo E-D80NK 18);

3.2. CARGA E DESCARGA DE UM CAPACITOR

Material Necessário

Experimento

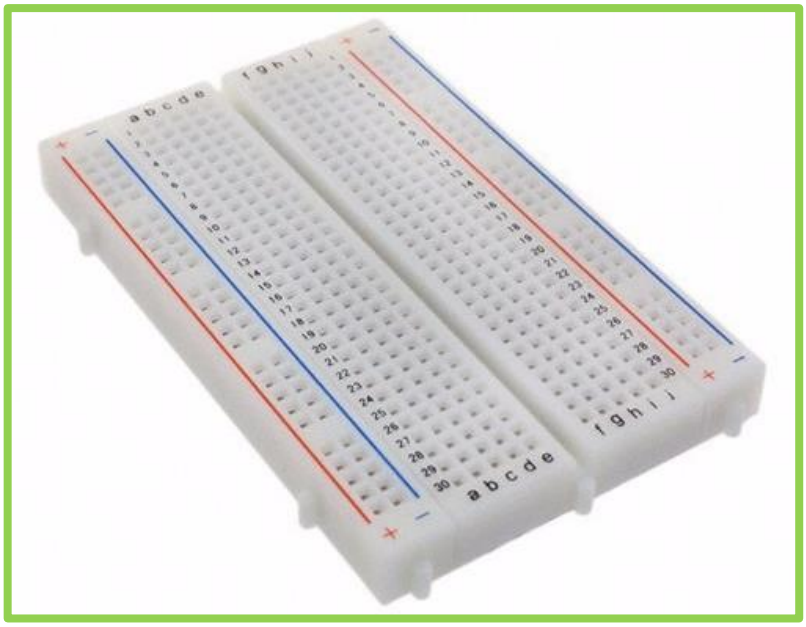
Carga e Descarga de um Capacitor

Seguem abaixo os materiais necessários para a realização do experimento de carga e descarga de um capacitor:

- 01 Arduino Uno;

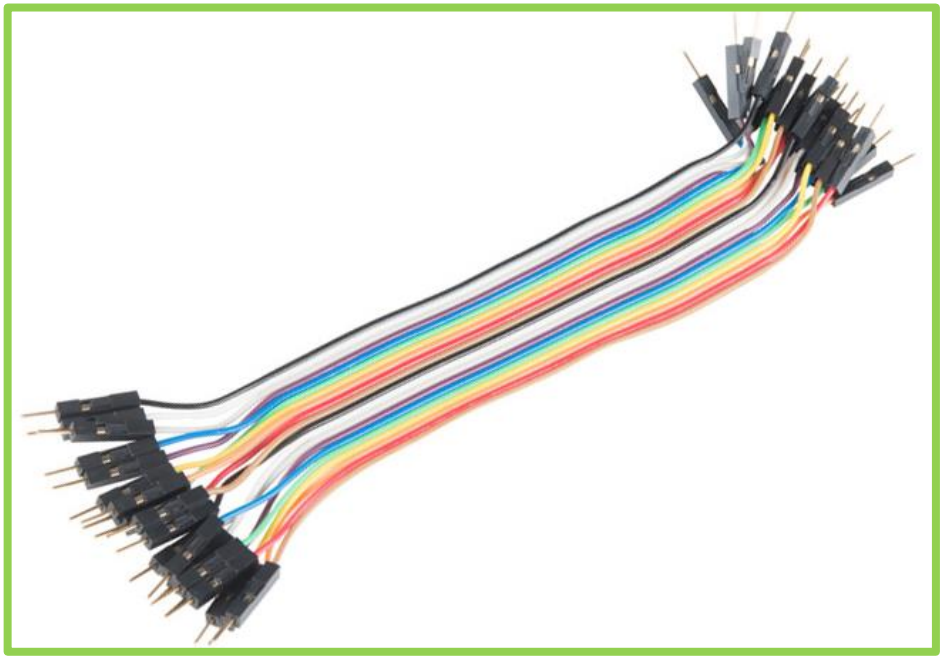
- 01 Cabo USB de conexão (vem junto ao Arduino);

- 01 Protoboard;

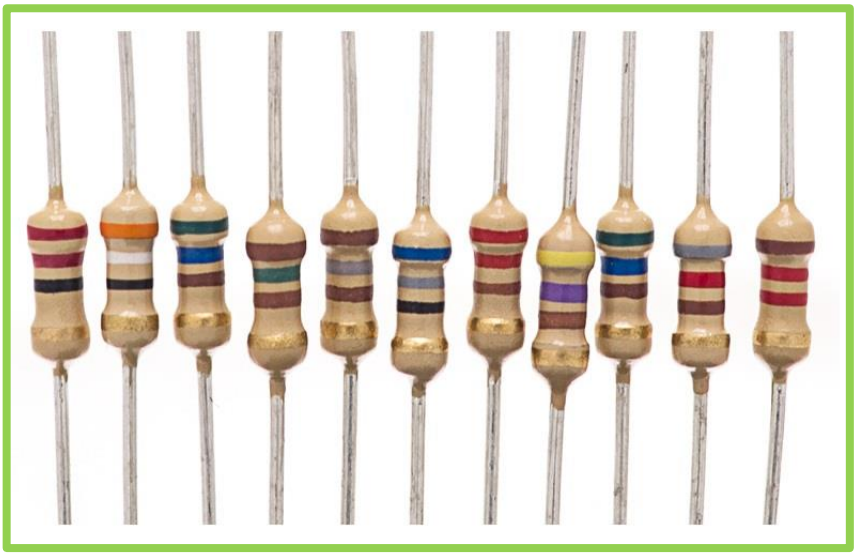


- 01 Capacitor (sugerimos que não utilize modelos com mais de 100 micro farads);

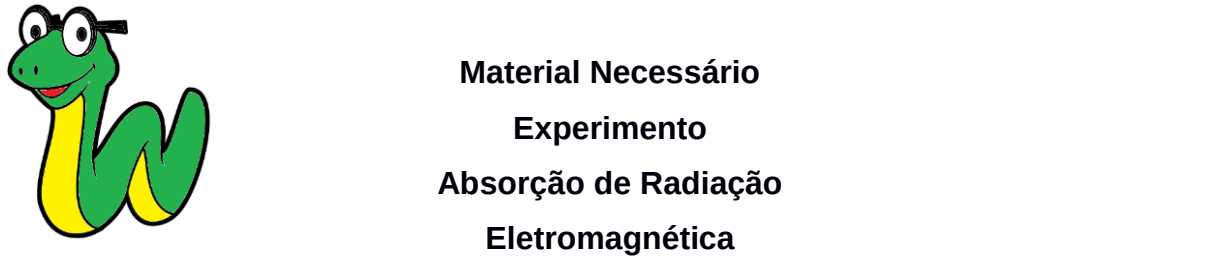
- Jumpers para ligação (macho/macho);



- Resistores (sugerimos que utilize modelos com mais de 1000 ohms);



3.3. EMISSÃO E ABSORÇÃO DE RADIAÇÃO ELETROMAGNÉTICA



Material Necessário

Experimento

Absorção de Radiação

Eletromagnética

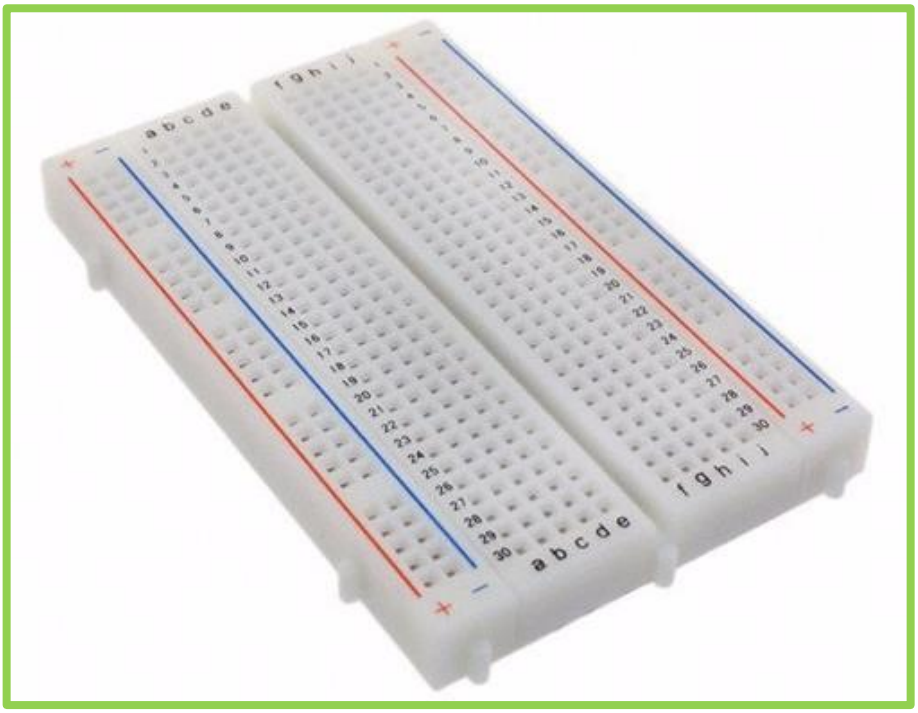
Seguem abaixo os materiais necessários para a realização do experimento de absorção de radiação eletromagnética:

- 01 Arduino Uno;

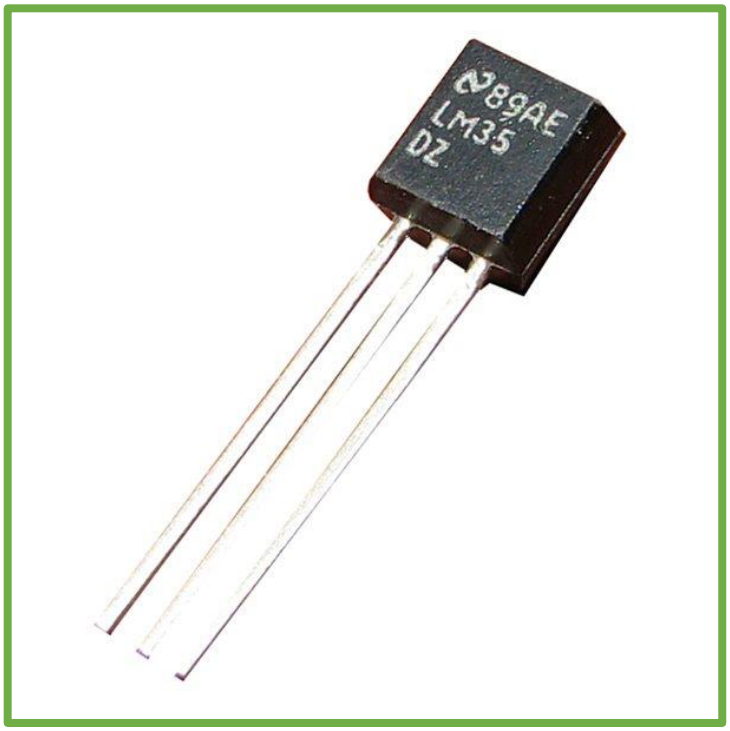
- 01 Cabo USB de conexão (vem junto ao Arduino);

- Jumpers para ligação (macho/macho e macho/fêmea);

- 01 Protoboard;



- 2 sensores de temperatura (modelo LM35);

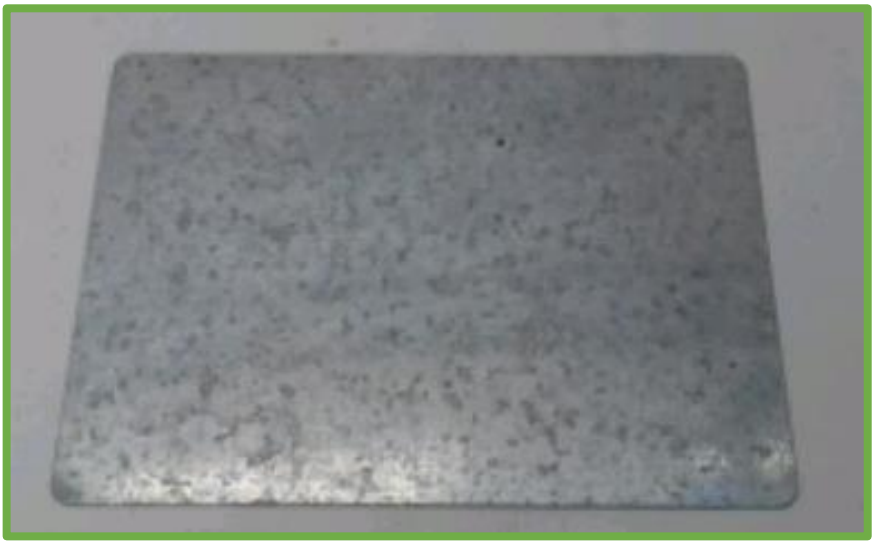


- 01 Lâmpada incandescente; halógena;

- 01 Boquilha com extensão;

- 02 Pedaços de isopor para isolar termicamente os sensores;

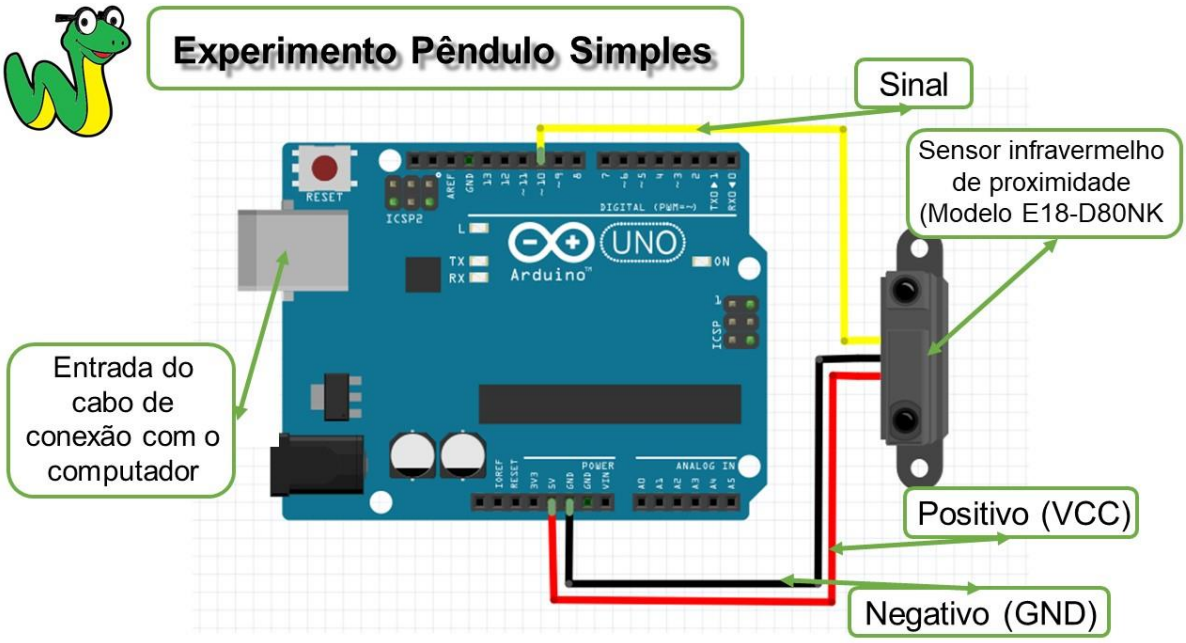
- 02 Placas metálicas de mesmas dimensões pintadas em cores diferentes;



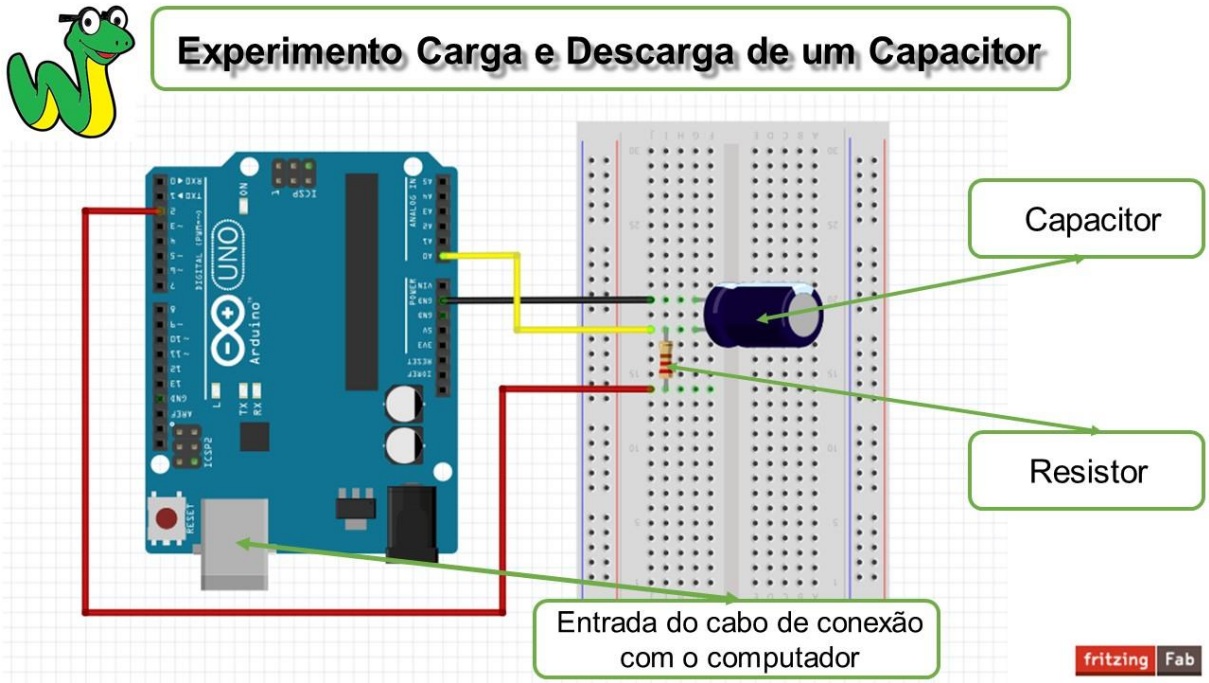
4. FIGURAS E IMAGENS DAS MONTAGENS ELETRÔNICAS

A seguir serão apresentadas as imagens utilizadas no programa referentes montagens eletrônicas de cada experimento.

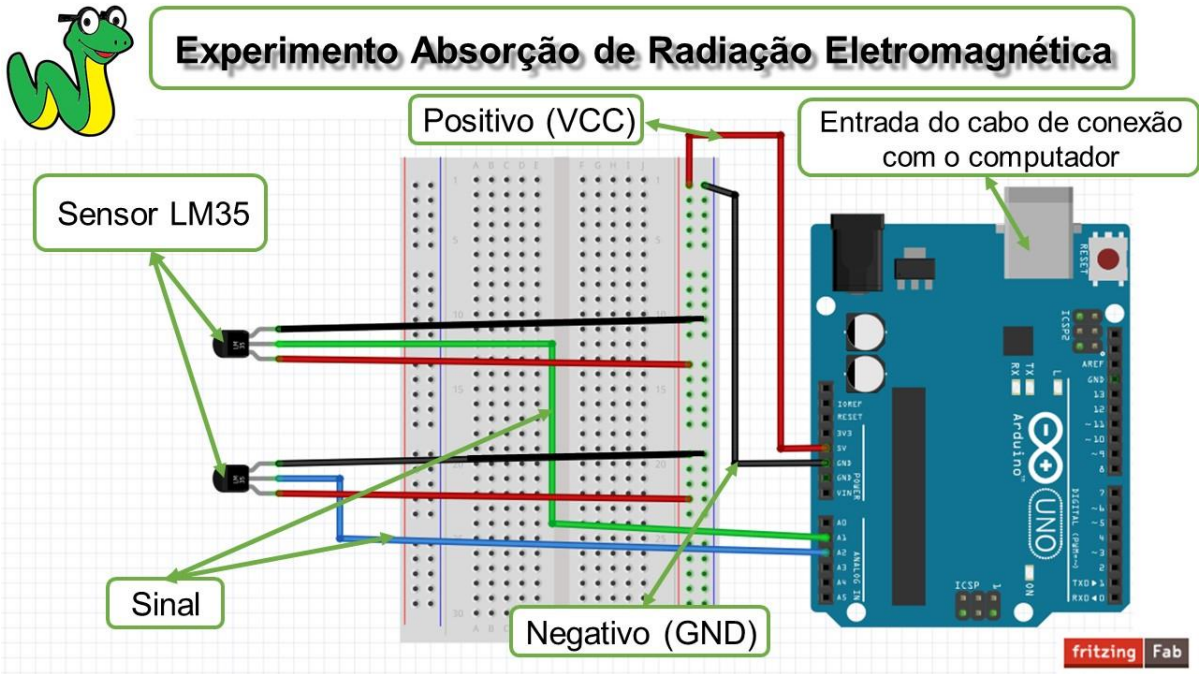
4.1. PÊNDULO SIMPLES



4.2. CARGA E DESCARGA DE UM CAPACITOR



4.3. ABSORÇÃO DE RADIAÇÃO ELETROMAGNÉTICA



5. CÓDIGO DO SOFTWARE PYDUINO (O fluxograma deste código é a figura 6, presente na página 17)

Segue abaixo o código do programa criado:

```
import tkinter as tk, os, time, matplotlib.pyplot as plt, pyfirmata as pf

#apresentado constantes utilizadas:
larg = 1500 #largura da tela principal
alt = 700 #largura da tela principal

#criando janela principal
janela = tk.Tk()
janela.geometry('{}x{}'.format(larg, alt))
janela.title('Aplicativo PyDuino MPNEF')
janela.configure(bg='white')
janela.wm_iconbitmap('Componentes/Icone.ico')

#Criando Janela de Apresentação
def janela_inicial():

    #Função que abre o arquivo com auxílio do protocolo firmata
    def abre_firmata():
        os.startfile('Componentes\Ajuda Firmata.pdf')

    #Função que limpa a tela e executa a função menu_principal
    def iniciar_menu_principal():
        for i in elementos:
            i.destroy()
        menu_principal()

    #criando frame
    frame = tk.Frame(janela, bg='white')
    frame.pack()

    #criando logo
    img_logo = tk.PhotoImage(file='Componentes/Logo Imagem.gif').subsample(1, 2)
    logo = tk.Label(frame, image=img_logo, bg='white')
    logo.pack()

    #colocando imagem com mascote e texto
    img_tela_inicial = tk.PhotoImage(file='Componentes/Tela Inicial.gif').subsample(1,
1)
    mascote = tk.Label(frame, image=img_tela_inicial, bg='white')
    mascote.pack()

    #criando botão começar
    img_comecar = tk.PhotoImage(file='Componentes/Iniciar
```

```
Imagem.gif').subsample(8, 10)
    botao_iniciar = tk.Button(janela, image=img_comecar, cursor='hand2',
command=iniciar_menu_principal, bg='white')
    botao_iniciar.place(x=950, y=550)

    #criando botão para ajuda com o protocolo firmata.
    img_firmata = tk.PhotoImage(file='Componentes/Firmata
Imagem.gif').subsample(8, 10)
    botao_firmata = tk.Button(janela, image=img_firmata, cursor='hand2',
command=abre_firmata, bg='white')
    botao_firmata.place(x=650, y=550)

    elementos=[frame, botao_iniciar, botao_firmata]
    janela.mainloop()

#criando função que encerra o programa
def sair():
    janela.destroy()

#criando função que abre ajuda com a porta de comunicação
def ajuda_USB():
    os.startfile('Componentes\Ajuda USB.pdf')

#criando função que gera janelas de aviso
def aviso(texto):
    aviso = tk.Tk()
    aviso.title('Aviso!')
    aviso.geometry('300x180+500+200')
    aviso.wm_iconbitmap('Componentes/Icone Alerta.ico')
    mensagem = tk.Label(aviso, text=texto, height=7, font=('Arial', 10))
    mensagem.pack()
    ok = tk.Button(aviso, text='OK', command=aviso.destroy)
    ok.pack()
    aviso.mainloop()

#criando função que cria o menu principal
def menu_principal():

    #criando função que especifica o experimento de pêndulo simples
    def pendulo():
        for i in elementos:
            i.destroy()
        menu_experimento('Componentes/Pendulo Imagem.gif', 'Experimento Pendulo
Simples', 'Componentes\Pendulo Material.pdf',
                         'Componentes\Pendulo Montagem.pdf', 'Componentes\Pendulo
Texto Auxilio.pdf', 1)

    #criando função que especifica o experimento de carga e descarga de um
capacitor
    def capacitor():
```

```
        for i in elementos:
            i.destroy()
        menu_experimento('Componentes/Capacitor Imagem.gif', 'Carga e Descarga de
um Capacitor', 'Componentes\Capacitor Material.pdf',
                         'Componentes\Capacitor Montagem.pdf', 'Componentes\Capacitor
Texto Auxilio.pdf', 2)

    #criando função que especifica o experimento de absorção de radiação luminosa
    def radiacao():
        for i in elementos:
            i.destroy()
        menu_experimento('Componentes/Radiacao Imagem.gif', 'Experimento
Absorção de Radiação', 'Componentes\Radiacao Material.pdf',
                         'Componentes\Radiacao Montagem.pdf', 'Componentes\Radiacao
Texto Auxilio.pdf', 3)

    #criando fundo
    img_preenchimento = tk.PhotoImage(file='Componentes/Imagem
Complementar2.png').subsample(1, 1)
    container1 = tk.Label(janela, image=img_preenchimento)
    container1.place(x=0, y=0)
    container2 = tk.Label(janela, image=img_preenchimento)
    container2.place(x=725, y=0)

    # criando logo
    img_logo = tk.PhotoImage(file='Componentes/Logo Imagem.gif').subsample(1, 2)
    logo = tk.Label(janela, text='Menu Principal', font=('Arial', 25, 'bold'),
image=img_logo,
                    compound='top')
    logo.pack()

    # criando texto auxiliar
    texto_auxiliar = tk.Label(janela, text='Escolha o experimento que deseja realizar.',
width=40, height=2, font=('Arial', '14'))
    texto_auxiliar.pack()

    # criando botão1
    img_pendulo = tk.PhotoImage(file='Componentes/Pendulo
Imagem.gif').subsample(3, 7)
    botao1 = tk.Button(janela, text='Pêndulo Simples', font=('Arial', '16'),
image=img_pendulo,
                       compound='top', bg='#b5b5b5', cursor='hand2',
command=pendulo)
    botao1.place(x=190, y=200)

    # criando botão 2
    img_radiacao = tk.PhotoImage(file='Componentes/Radiacao
Imagem.gif').subsample(3, 7)
    botao2 = tk.Button(janela, text='Absorção de Radiação', font=('Arial', '16'),
image=img_radiacao,
```

```
                        compound='top', bg='#b5b5b5', cursor='hand2',
command=radiacao)
    botao2.place(x=190, y=400)

    # criando botão 3
    img_capacitor = tk.PhotoImage(file='Componentes/Capacitor
Imagem.gif').subsample(3, 7)
    botao3 = tk.Button(janela, text='Carga e Descarga de um capacitor', font=('Arial',
'16'),
                        image=img_capacitor, compound='top', bg='#b5b5b5',
cursor='hand2', command=capacitor)
    botao3.place(x=750, y=200)

    # criando botão 4
    img_exp_4 = tk.PhotoImage(file='Componentes/Experimento4
Imagem.gif').subsample(3, 7)
    botao4 = tk.Button(janela, text='Experimento 4', font=('Arial', '16'),
image=img_exp_4,
                        compound='top', bg='#b5b5b5', cursor='hand2')
    botao4.place(x=750, y=400)

    # criando botão sair
    img_sair = tk.PhotoImage(file='Componentes/Sair Imagem.gif').subsample(10, 12)
    botao_sair = tk.Button(janela, text='SAIR', font=('Arial', '12', 'bold'),
image=img_sair,
                           compound='left', bg='#cd3333', cursor='hand2', command=sair)
    botao_sair.place(x=880, y=570)
    elementos = [logo, texto_auxiliar, botao1, botao2, botao3, botao4, botao_sair]
    janela.mainloop()

#criando menu de textos dos experimentos
def menu_experimento(img, texto, material, montagem, txt, x):

    #criando função que volta ao menu principal
    def voltar():
        for i in elementos:
            i.destroy()
        menu_principal()

    #criando função que abre a tela de experimento
    def experimento():
        for i in elementos:
            i.destroy()
            global aviso_cancela
            aviso_cancela = 'não'
        if x == 1:
            menu_conf_pend()
        elif x == 2:
            menu_conf_cap()
        elif x == 3:
```

```
            menu_conf_rad()

    #criando funções que abrem os arquivos PDF
    def abre_material():
        os.startfile(material)
    def abre_montagem():
        os.startfile(montagem)
    def abre_texto():
        os.startfile(txt)

    # criando logo
    img_logo = tk.PhotoImage(file=img).subsample(2, 5)
    logo = tk.Label(janela, text=texto, font=('Arial', 20, 'bold'), image=img_logo,
compound='top')
    logo.pack()

    # criando texto auxiliar
    texto_auxiliar = tk.Label(janela, text='Escolha a opção desejada.', width=20,
height=2, font=('Arial', '14'))
    texto_auxiliar.pack()

    # criando espaço em cima
    container3 = tk.Label(janela, height=1)
    container3.pack()

    # criando botão1
    img_componentes = tk.PhotoImage(file='Componentes/Componentes
Imagem.gif').subsample(10, 12)
    botao1 = tk.Button(janela, text='   Material necessário  ', font=('Arial', '16'),
image=img_componentes,
                       compound='left', bg='#b5b5b5', cursor='hand2',
command=abre_material)
    botao1.pack()

    # criando espaço em cima
    container4 = tk.Label(janela, height=1)
    container4.pack()

    # criando botão2
    img_montagem = tk.PhotoImage(file='Componentes/Montagem
Imagem.gif').subsample(10, 12)
    botao2 = tk.Button(janela, text='Esquema de montagem', font=('Arial', '16'),
image=img_montagem,
                       compound='left', bg='#b5b5b5', cursor='hand2',
command=abre_montagem)
    botao2.pack()

    # criando espaço em cima
    container5 = tk.Label(janela, height=1)
    container5.pack()
```

```
    # criando botão3
    img_texto = tk.PhotoImage(file='Componentes/Texto Imagem.gif').subsample(10,
12)
    botao3 = tk.Button(janela, text='       Texto de apoio       ', font=('Arial', '16'),
image=img_texto,
                       compound='left', bg='#b5b5b5', cursor='hand2',
command=abre_texto)
    botao3.pack()

    # criando espaço em cima
    container6 = tk.Label(janela, height=1)
    container6.pack()

    # criando botão4
    img_exp = tk.PhotoImage(file='Componentes/Experimento
Imagem.gif').subsample(10, 12)
    botao4 = tk.Button(janela, text='         Experimento         ', font=('Arial', '16'),
image=img_exp,
                       compound='left', bg='#b5b5b5', cursor='hand2',
command=experimento)
    botao4.pack()

    # criando espaço na direita
    container8 = tk.Label(janela, width=71)
    container8.pack(side='right')

    # criando espaço na esquerda
    container9 = tk.Label(janela, width=71)
    container9.pack(side='left')

    # criando botão sair
    img_sair = tk.PhotoImage(file='Componentes/Sair Imagem.gif').subsample(20, 20)
    botao_sair = tk.Button(janela, image=img_sair, bg='#cd3333', cursor='hand2',
command=sair)
    botao_sair.pack(side='right')

    # criando botão voltar menu principal
    img_menu = tk.PhotoImage(file='Componentes/Voltar Imagem.gif').subsample(20,
20)
    botao_menu = tk.Button(janela, image=img_menu, bg='#63B8FF', cursor='hand2',
command=voltar)
    botao_menu.pack(side='left')

    elementos = [logo, texto_auxiliar, botao1, botao2, botao3, botao4, botao_sair,
botao_menu,
                 container3, container4, container5, container6, container8, container9]
    janela.mainloop()

#criando janela de configuração do experimento de pêndulo simples
```

```
def menu_conf_pend():

    #definindo função voltar:
    def voltar():
        arquivo1 = open("Componentes/historico.txt", "w")
        cont = 0
        while cont < len(periodos):
            arquivo1.write("{} {}\n".format(comprimentos[cont], periodos[cont]))
            cont += 1
        arquivo1.close()
        for i in elementos:
            i.destroy()
        menu_principal()

    #criando função que cria os marcadores
    def cria_marcadores():

        global marcador1, marcador2, marcador3, marcador4, marcador5, marcador6,
marcador7, marcador8, marcador9,\
            marcador10, var1, var2, var3, var4, var5, var6, var7, var8, var9, var10,
marcadores, variaveis

        var1 = tk.IntVar()
        marcador1 = tk.Checkbutton(frame, text=medidas[0], font=('Arial', '12', 'bold'),
height=1, variable=var1,
                                   onvalue=1, offvalue=0)
        marcador1.pack()

        var2 = tk.IntVar()
        marcador2 = tk.Checkbutton(frame, text=medidas[1], font=('Arial', '12', 'bold'),
height=1, variable=var2,
                                   onvalue=1, offvalue=0)
        marcador2.pack()

        var3 = tk.IntVar()
        marcador3 = tk.Checkbutton(frame, text=medidas[2], font=('Arial', '12', 'bold'),
height=1, variable=var3,
                                   onvalue=1, offvalue=0)
        marcador3.pack()

        var4 = tk.IntVar()
        marcador4 = tk.Checkbutton(frame, text=medidas[3], font=('Arial', '12', 'bold'),
height=1, variable=var4,
                                   onvalue=1, offvalue=0)
        marcador4.pack()

        var5 = tk.IntVar()
        marcador5 = tk.Checkbutton(frame, text=medidas[4], font=('Arial', '12', 'bold'),
height=1, variable=var5,
                                   onvalue=1, offvalue=0)
```

```
        marcador5.pack()

        var6 = tk.IntVar()
        marcador6 = tk.Checkbutton(frame, text=medidas[5], font=('Arial', '12', 'bold'),
height=1, variable=var6,
                                   onvalue=1, offvalue=0)
        marcador6.pack()

        var7 = tk.IntVar()
        marcador7 = tk.Checkbutton(frame, text=medidas[6], font=('Arial', '12', 'bold'),
height=1, variable=var7,
                                   onvalue=1, offvalue=0)
        marcador7.pack()

        var8 = tk.IntVar()
        marcador8 = tk.Checkbutton(frame, text=medidas[7], font=('Arial', '12', 'bold'),
height=1, variable=var8,
                                   onvalue=1, offvalue=0)
        marcador8.pack()

        var9 = tk.IntVar()
        marcador9 = tk.Checkbutton(frame, text=medidas[8], font=('Arial', '12', 'bold'),
height=1, variable=var9,
                                   onvalue=1, offvalue=0)
        marcador9.pack()

        var10 = tk.IntVar()
        marcador10 = tk.Checkbutton(frame, text=medidas[9], font=('Arial', '12', 'bold'),
height=1, variable=var10,
                                   onvalue=1, offvalue=0)
        marcador10.pack()

        global marcadores
        marcadores = [marcador1, marcador2, marcador3, marcador4, marcador5,
marcador6, marcador7, marcador8, marcador9,
                      marcador10]

    #criando função que limpa os icones selecionados
    def limpa():
        variaveis = [var1.get(), var2.get(), var3.get(), var4.get(), var5.get(), var6.get(),
var7.get(), var8.get(),
                     var9.get(), var10.get()]
        for x in marcadores:
            x.destroy()
        for n, i in enumerate(variaveis):
            if i == 1:
                medidas[n] = 'Medida {} : T =          e L =         '.format(n + 1)
                del comprimentos[n]
                del periodos[n]
        cria_marcadores()
```

```
        frame.update()

    #função que gera gráficos do experimento de pendulo simples
    def grafico2():
        x = sorted(comprimentos)
        y = sorted(periodos)
        plt.plot(x, y, 'ro')  # criando gráfico
        plt.plot(x, y, 'k:', color='black')
        plt.title('Experimento Pêndulo Simples - L(m) x T(s)')
        plt.xlabel('Comprimento (m)')
        plt.ylabel('Períodos (s)')
        plt.show()

    #criando função que estabelece as variáveis do experimento e executa a tela de
monitoramento
    def define_pendulo():
        global porta, num_periodos, comprimento
        porta = 'COM' + str(define_porta.get())
        num_periodos = int(campo_periodo.get())
        comprimento = campo_comprimento.get()
        comprimento = comprimento.replace(',', '.')

        #emite uma janela de aviso caso o programa apresente algum erro na porta de
comunicação.
        try:
            # cálculo período teórico
            global periodo_teorico
            periodo_teorico = 2 * 3.14159 * (float(comprimento) / 9.78) ** 0.5  # cálculo
do período teórico:

        except:
            aviso('Comprimento inválido,\n '
                  'favor informe apenas números')

        #emite uma janela de aviso caso o programa apresente algum erro na porta de
comunicação.
        try:
            global placa
            placa = pf.Arduino(porta)
            for i in elementos:
                i.destroy()
            monitoramento_pendulo()

        except:
            aviso('Porta de comunicação informada inválida,\n '
                  'favor verificar qual a porta de comunicação \n'
                          'que o Arduino está conectado')

    #Criando texto principal
    img_conf = tk.PhotoImage(file='Componentes/Conf Imagem.gif').subsample(8, 8)
```

```
    texto_principal = tk.Label(janela, text='Configurações do Experimento',
font=('Arial', '25', 'bold'),
                       image=img_conf, compound='left', bg='#b5b5b5')
    texto_principal.place(x=0, y=0)

    #criando campo para informar a porta usb conectada
    texto_porta = tk.Label(janela, text='Qual a porta de comunicação (COM) utilizada
pelo Arduino?', font=('Arial', '16', 'bold'))
    texto_porta.place(x=30, y=140)
    define_porta = tk.Scale(janela, from_=0, to=20, orient=tk.HORIZONTAL,
font=('Arial', '12', 'bold'))
    define_porta.place(x=240, y=180)
    botao_duvida = tk.Button(janela, text='?', width=3, height=1, font=('Arial', '12',
'bold'),
                             cursor='hand2', command=ajuda_USB)
    botao_duvida.place(x=370, y=192)

    #criando campo para informar quantas oscilações medir
    texto_periodo = tk.Label(janela, text='Quantas oscilações você quer medir?',
font=('Arial', '16', 'bold'))
    texto_periodo.place(x=125, y=250)
    campo_periodo = tk.Scale(janela, from_=0, to=14, orient=tk.HORIZONTAL,
font=('Arial', '12', 'bold'))
    campo_periodo.place(x=255, y=290)

    #criando campo para informar o comprimento do pêndulo
    texto_comprimento = tk.Label(janela, text='Coloque o comprimento do fio em
metros:',
                                 font=('Arial', '16', 'bold'))
    texto_comprimento.place(x=100, y=360)
    campo_comprimento = tk.Entry(janela, bd=5, font=('Arial', '12', 'bold'))
    campo_comprimento.place(x=220, y=410)
    campo_comprimento.focus_set()

    #criando botão começar
    img_exp = tk.PhotoImage(file='Componentes/Iniciar Imagem.gif').subsample(8, 10)
    botao_iniciar = tk.Button(janela, image=img_exp, cursor='hand2',
command=define_pendulo)
    botao_iniciar.place(x=230, y=475)

    # criando botão voltar e sair menu principal
    img_menu = tk.PhotoImage(file='Componentes/Voltar Imagem.gif').subsample(20,
20)
    botao_menu = tk.Button(janela, image=img_menu, bg='#63B8FF', cursor='hand2',
command=voltar)
    botao_menu.place(x=150, y=600)
    img_sair = tk.PhotoImage(file='Componentes/Sair Imagem.gif').subsample(20, 20)
    botao_sair = tk.Button(janela, image=img_sair, bg='#cd3333', cursor='hand2',
command=sair)
    botao_sair.place(x=400, y=600)
```

```
    #criando texto principal
    img_hist = tk.PhotoImage(file='Componentes/Hist Imagem.gif').subsample(8, 8)
    texto_historico = tk.Label(janela, text='  Histórico dos experimentos ',
image=img_hist, compound='left',
                               font=('Arial', '25', 'bold'), bg='#b5b5b5')
    texto_historico.place(x=740, y=0)

    #frame que fica com os resultados
    frame = tk.Frame(janela)
    frame.place(x=920, y=165)

    # Criando Histórico dos experimentos em tela.
    arquivo1 = open("Componentes/historico.txt", "r")
    global periodos, comprimentos
    periodos = []
    comprimentos = []

    for linha in arquivo1:  # trabalhando com cada linha do arquivo por vez.
        termo = linha.split()  # cada vez que há um espaço na linha, gera se uma nova
string, chamada de termo.
        x = float(termo[0])
        y = float(termo[1])
        periodos.append(y), comprimentos.append(x)
    arquivo1.close()
    medidas = []
    for x in range(10):
        if x < len(periodos):
            texto = 'Medida {} : T = {:.2f} e L = {:.2f}'.format((x+1), periodos[x],
comprimentos[x])
        else:
            texto = 'Medida {} : T =          e L =         '.format(x+1)
        medidas.append(texto)

    cria_marcadores()

    # criando botao limpar
    botao_limpar = tk.Button(janela, text='Limpar resultados selecionados.', width=25,
                             height=1, font=('Arial', '14', 'bold'), cursor='hand2',
bg='#b5b5b5', command=limpa)
    botao_limpar.place(x=920, y=520)

    # criando botao plotar grafico T x L
    img_grf = tk.PhotoImage(file='Componentes/Grafico Imagem.gif' ).subsample(15,
10)
    botao_grafico = tk.Button(janela, text='Mostrar Gráfico Período (T) x Comprimento
(L)', image=img_grf,
                              font=('Arial', '13', 'bold'), bg='#b5b5b5', cursor='hand2',
compound='left',
                              command=grafico2)
```

```
    botao_grafico.place(x=820, y=580)

    elementos = [texto_principal, texto_historico, texto_comprimento, texto_periodo,
texto_porta,
                 campo_comprimento, campo_periodo, define_porta, botao_duvida,
botao_iniciar, botao_grafico, botao_menu,
                 botao_sair, botao_limpar, frame]
    if aviso_cancela == 'não':
        aviso('Certifique-se que o Arduino está ligado\n'
              'na porta USB e que todas as conecções\n'
                    'foram feitas corretamente!')

    janela.mainloop()

# Criando janela que monitora o experimento de pêndulo simples
def monitoramento_pendulo():

    # definindo função a ser executa do Arduino no experimento de pêndulo simples
    def pendulo_comecou():

        # função que gera o gráfico
        def grafico_periodos():
            plt.plot(range(len(lista)), lista)  # criando gráfico
            plt.title('Experimento Pêndulo Simples')
            plt.ylim([0, 10])
            plt.xlabel('Ocilação')
            plt.ylabel('Períodos (s)')
            plt.show()

        sensor = placa.get_pin('d:9:i')
        oscilacao_completa = 0
        limitador = 5
        contador = 0
        lista = []
        resultados = '\n'
        botao_começar.destroy()
        texto_resultados.config(text='Aguardando lançamento!')
        botao_grafico.config(command=grafico_periodos)
        janela.update()
        while oscilacao_completa < int(num_periodos):
            detecta = bool(sensor.read())
            if detecta == False:
                contador += 1
                tempo = time.time()
                if contador == 3:
                    t0 = tempo
                elif contador == limitador:
                    periodo = tempo - t0
                    t0 = tempo
                    limitador += 2
```

```
                if periodo <= periodo_teorico * 1.25:
                    lista.append(periodo)
                    periodo = str(periodo)
                    oscilacao_completa += 1
                    resultados += 'Oscilação ' + str(oscilacao_completa) + ' = ' +
periodo[0:5] + 's\n'
                    texto_resultados.config(text=resultados)
                    frame2.update()
            time.sleep(0.2)
        placa.exit()


        # cálculo período experimental:
        global periodo_experimental
        periodo_experimental = sum(lista) / len(lista)

        # exibindo período experimental:
        texto_valor_experimental.config(text='   O resultado experimental é:
\n{:.3f}s'.format(periodo_experimental))

        #acoplando os resultados nas listas e arquivo txt
        periodos.append(periodo_experimental)
        comprimentos.append(comprimento)
        arquivo1 = open("Componentes/historico.txt", "w")
        cont = 0
        while cont < len(periodos):
            arquivo1.write("{} {}\n".format(comprimentos[cont], periodos[cont]))
            cont += 1
        arquivo1.close()

        janela.mainloop()

    # definindo função voltar:
    def voltar():
        for i in elementos:
            i.destroy()
        placa.exit()
        global aviso_cancela
        aviso_cancela = 'sim'
        menu_conf_pend()

    # faz conecção com o Arduíno antes de abrir a janela
    it = pf.util.Iterator(placa)
    it.start()

    # criando frame
    frame = tk.Frame(janela, bg='white')
    frame.pack()

    # criando logo
```

```
    img_mon = tk.PhotoImage(file='Componentes/Monitora Imagem.gif').subsample(3,
6)
    texto = tk.Label(frame, text='Resultados medidos:', image=img_mon,
compound='top', font=('Arial', '20'),
                     bg='#b5b5b5')
    texto.pack()

    # criando frame 2, este é o que atualiza os resultados
    frame2 = tk.Frame(frame)
    frame2.pack()

    # separando espaço texto de resultados
    texto_resultados = tk.Label(frame2, height=14, text='', font=('Arial', '14', 'bold'),
bg='white')
    texto_resultados.pack()
    texto_valor_teorico = tk.Label(frame, text='O resultado teórico previsto é:
\n{:.3f}s'.format(periodo_teorico),
                                   font=('Arial', '14'), bg='#ffccff')
    texto_valor_teorico.pack()
    texto_valor_experimental = tk.Label(frame, text='   O resultado experimental é:  \n'
                                                    '              ', font=('Arial', '14'), bg='#99ccff')
    texto_valor_experimental.pack()

    # criando botões
    img_exp = tk.PhotoImage(file='Componentes/Iniciar Imagem.gif').subsample(10,
12)
    botao_começar = tk.Button(janela, image=img_exp, cursor='hand2',
text='Começar', compound='top',
                          font=('Arial', '10', 'bold'), bg='#b5b5b5',
command=pendulo_comecou)
    botao_começar.place(x=470, y=600)
    img_graf = tk.PhotoImage(file='Componentes/Grafico Imagem.gif').subsample(10,
12)
    botao_grafico = tk.Button(janela, text='Gráfico', image=img_graf, compound='top',
                          font=('Arial', '10', 'bold'), bg='#b5b5b5')
    botao_grafico.place(x=615, y=600)
    img_voltar = tk.PhotoImage(file='Componentes/Voltar
Imagem2.gif').subsample(10, 12)
    botao_voltar = tk.Button(janela, text='Voltar', image=img_voltar, compound='top',
font=('Arial', '10', 'bold'),
                         command=voltar, bg='#b5b5b5')
    botao_voltar.place(x=760, y=600)
    elementos = [frame, botao_grafico, botao_começar, botao_voltar]
    janela.mainloop()

#criando janela de configuração do experimento de carga e descarga de um
capacitor
def menu_conf_cap():

    #definindo função voltar:
```

```
    def voltar():
        for i in elementos:
            i.destroy()
        menu_principal()

    #função que reconhece as informações fornecidas e que vai para a janela de
monitoramento
    def define_capacitor():
        global porta, num_ciclos, resistencia, capacitancia
        porta = 'COM' + str(define_porta.get())
        num_ciclos = int(define_ciclos.get())
        resistencia = define_resistencia.get()
        resistencia.replace(',', '.')
        capacitancia = define_capacitancia.get()
        capacitancia.replace(',', '.')

        try:
            global placa
            placa = pf.Arduino(porta)
            for i in elementos:
                i.destroy()
            monitoramento_capacitor()
        except:
            aviso('Porta de comunicação informada inválida,\n '
                  'favor verificar qual a porta de comunicação \n'
                          'que o Arduino está conectado')

    #Criando texto principal
    img_conf = tk.PhotoImage(file='Componentes/Conf Imagem.gif').subsample(8, 8)
    texto_principal = tk.Label(janela, text='Configurações do Experimento',
font=('Arial', '25', 'bold'),
                   image=img_conf, compound='left', bg='#b5b5b5')
    texto_principal.pack()

    #criando campo para informar a porta usb conectada
    texto_porta = tk.Label(janela, text='Qual a porta de comunicação (COM) utilizada
pelo Arduino?', font=('Arial', '16', 'bold'))
    texto_porta.place(x=355, y=110)
    define_porta = tk.Scale(janela, from_=0, to=20, orient=tk.HORIZONTAL,
font=('Arial', '12', 'bold'))
    define_porta.place(x=600, y=155)
    botao_duvida = tk.Button(janela, text='?', width=3, height=1, font=('Arial', '12',
'bold'),
                             cursor='hand2', command=ajuda_USB)
    botao_duvida.place(x=710, y=169)

    #criando campo para informar quantas ciclos de carga e descarga serão medidos
    texto_ciclos = tk.Label(janela, text='Quantos ciclos de carga e descarga devem
ser medidos?',
                            font=('Arial', '16', 'bold'), justify='center')
```

```
    texto_ciclos.place(x=355, y=215)
    define_ciclos = tk.Scale(janela, from_=0, to=10, orient=tk.HORIZONTAL,
font=('Arial', '12', 'bold'))
    define_ciclos.place(x=595, y=255)

    #criando campo para informar qual a resistência utilizada na montagem
    texto_resistencia = tk.Label(janela, text='Qual a resistência do resistor, coloque o
valor em quilo ohms?',
                                 font=('Arial', '16', 'bold'), justify='center')
    texto_resistencia.place(x=355, y=310)
    define_resistencia = tk.Entry(janela, bd=5, font=('Arial', '12', 'bold'), justify='center')
    define_resistencia.place(x=565, y=360)
    define_resistencia.focus_set()

    #criando campo para informar qual a capacitância do capacitor utilizado na
montagem
    texto_capacitancia = tk.Label(janela, text='Qual a capacitância do capacitor,
coloque o valor em micro farads?',
                                  font=('Arial', '16', 'bold'), justify='center')
    texto_capacitancia.place(x=355, y=405)
    define_capacitancia = tk.Entry(janela, bd=5, font=('Arial', '12', 'bold'),
justify='center')
    define_capacitancia.place(x=565, y=450)

    #criando botão começar
    img_exp = tk.PhotoImage(file='Componentes/Iniciar Imagem.gif').subsample(8, 10)
    botao_iniciar = tk.Button(janela, image=img_exp, cursor='hand2',
command=define_capacitor)
    botao_iniciar.place(x=575, y=495)

    # criando botão voltar e sair menu principal
    img_menu = tk.PhotoImage(file='Componentes/Voltar Imagem.gif').subsample(20,
20)
    botao_menu = tk.Button(janela, image=img_menu, bg='#63B8FF', cursor='hand2',
command=voltar)
    botao_menu.place(x=495, y=600)
    img_sair = tk.PhotoImage(file='Componentes/Sair Imagem.gif').subsample(20, 20)
    botao_sair = tk.Button(janela, image=img_sair, bg='#cd3333', cursor='hand2',
command=sair)
    botao_sair.place(x=745, y=600)

    elementos = [texto_principal, texto_porta, define_porta, botao_duvida,
botao_iniciar, botao_menu,
                 botao_sair, texto_ciclos, define_ciclos, texto_resistencia,
define_resistencia,
                 texto_capacitancia, define_capacitancia]
    if aviso_cancela == 'não':
        aviso('Certifique-se que o Arduino está ligado\n'
              'na porta USB e que todas as conecções\n'
                    'foram feitas corretamente!')
```

```
    janela.mainloop()

# Criando janela que monitora o experimento de carga e descarga de um capacitor
def monitoramento_capacitor():

    #definindo função voltar:
    def voltar():
        for i in elementos:
            i.destroy()
        placa.exit()
        global aviso_cancela
        aviso_cancela = 'sim'
        menu_conf_cap()

    # definindo função a ser executa do Arduino no experimento de carga e descarga
de um capacitor
    def capacitor_comecou():

        #função que gera o gráfico
        def graficos():
            plt.subplot(3, 1, 1)
            plt.plot(periodos, tensao)  # criando gráfico
            plt.title('Experimento Carga e Descarga de um Capacitor')
            plt.xlabel('Tempo (s)')
            plt.ylabel('Tensão no Capacitor(V)')

            plt.subplot(3, 1, 2)
            plt.plot(periodos, corrente_circuito)  # criando gráfico
            plt.title('Experimento Carga e Descarga de um Capacitor')
            plt.xlabel('Tempo (s)')
            plt.ylabel('Corrente no Circuito(A)')

            plt.subplot(3, 1, 3)
            plt.plot(periodos, carga_capacitor)  # criando gráfico
            plt.title('Experimento Carga e Descarga de um Capacitor')
            plt.xlabel('Tempo (s)')
            plt.ylabel('Carga Armazenada (A)')

            plt.tight_layout()
            plt.show()

        botao_grafico.config(command=graficos)
        fonte = placa.get_pin('d:2:o')
        leitor = placa.get_pin('a:0:i')
        x = 0
        ciclos_medidos = 0
        ciclos2 = int(num_ciclos)*2
        tempo_total = 0
        fonte.write(1)
        fonte_status = 'ligada'
```

```
        time.sleep(2)
        corrente_circuito = []
        tensao = []
        carga_capacitor = []
        periodos = [0]
        while ciclos_medidos < ciclos2:
            tempo = time.time()
            leitura = leitor.read() * 5  # a leitura retornada do sensor da um valor entre 0 e
1,
                                         #  como a fonte do arduíno é de 5v, fazemos a converção
            corrente = (5 - leitura) / float(resistencia)
            carga = float(capacitancia) * leitura
            tensao.append(leitura)
            corrente_circuito.append(corrente)
            carga_capacitor.append(carga)
            if x == 0:
                texto_resultados.config(text='Em 0.00s\n a tensão no capacitor é {:.3f} V,\n'
                                             'a corrente elétrica no circuito é {:.3f} mA\n'
                                             'a carga armazenada no capacitor é {:.3f}
uC'.format(leitura,
                                                                          corrente, carga))
                t0 = tempo
                x += 1
            else:
                periodo = tempo - t0
                tempo_total += periodo
                periodos.append(tempo_total)
                t0 = tempo
                texto_resultados.config(text='Em {:.2f}s\n a tensão no capacitor é {:.3f}
V,\n'
                                             'a corrente elétrica no circuito é {:.3f} mA\n'
                                             'a carga armazenada no capacitor é {:.3f}
uC'.format(tempo_total,
                                                                      leitura, corrente, carga))
            if leitura == 4.980 or leitura == 0.010:
                if fonte_status == 'ligada':
                    fonte_status = 'desligada'
                    fonte.write(0)
                else:
                    fonte_status = 'ligada'
                    fonte.write(1)
                ciclos_medidos += 1
            frame.update()
            time.sleep(0.5)
        texto_resultados.config(text='Em {:.2f}s\n a tensão no capacitor é {:.3f} V,\n'
                                     'a corrente elétrica no circuito é {:.3f} mA\n'
                                     'a carga armazenada no capacitor é {:.3f} uC\n\n'
                                      'EXPERIMENTO FINALIZADO!\n'
                                         'Click em GRÁFICOS!'.format(tempo_total,leitura,
corrente, carga))
```

```
        placa.exit()

    #faz conecção com o Arduíno antes de abrir a janela
    it = pf.util.Iterator(placa)
    it.start()

    # criando frame
    frame = tk.Frame(janela, bg='white')
    frame.pack()

    # criando logo
    img_mon = tk.PhotoImage(file='Componentes/Monitora Imagem.gif').subsample(3,
6)
    texto = tk.Label(frame, text='Resultados medidos:', image=img_mon,
compound='top', font=('Arial', '20'),
                     bg='#b5b5b5')
    texto.pack()

    # criando frame 2, este é o que atualiza os resultados
    frame2 = tk.Frame(frame)
    frame2.pack()

    # separando espaço texto de resultados
    texto_resultados = tk.Label(frame2, height=14, text='Clique em START para\n'
                                                        'começar o experimento', font=('Arial', '14', 'bold'),
bg='white', justify='center')
    texto_resultados.pack()

    # criando botões
    img_exp = tk.PhotoImage(file='Componentes/Iniciar Imagem.gif').subsample(10,
12)
    botao_começar = tk.Button(janela, image=img_exp, cursor='hand2',
text='Começar', compound='top',
                              font=('Arial', '10', 'bold'), bg='#b5b5b5',
command=capacitor_comecou)
    botao_começar.place(x=470, y=500)
    img_graf = tk.PhotoImage(file='Componentes/Grafico Imagem.gif').subsample(10,
12)
    botao_grafico = tk.Button(janela, text='Gráfico', image=img_graf, compound='top',
                              font=('Arial', '10', 'bold'), bg='#b5b5b5')
    botao_grafico.place(x=615, y=500)
    img_voltar = tk.PhotoImage(file='Componentes/Voltar
Imagem2.gif').subsample(10, 12)
    botao_voltar = tk.Button(janela, text='Voltar', image=img_voltar, compound='top',
font=('Arial', '10', 'bold'),
                             command=voltar, bg='#b5b5b5')
    botao_voltar.place(x=760, y=500)
    elementos = [frame, botao_grafico, botao_começar, botao_voltar]
    janela.mainloop()
```

```
# Criando janela de configuração o experimento de absorção de radiação luminosa
def menu_conf_rad():

    # definindo função voltar:
    def voltar():
        for i in elementos:
            i.destroy()
        menu_principal()

    # função que reconhece as informações fornecidas e que vai para a janela de
monitoramento
    def define_radiacao():
        global porta, cor1, cor2, tempo_definido
        porta = 'COM' + str(define_porta.get())
        cor1 = define_cor1.get()
        cor2 = define_cor2.get()
        tempo_definido = define_tempo.get().replace(',','.')
        tempo_definido = float(tempo_definido) * 60

        try:
            global placa
            placa = pf.Arduino(porta)
            for i in elementos:
                i.destroy()
            monitoramento_radiacao()
        except:
            aviso('Porta de comunicação informada inválida,\n '
                  'favor verificar qual a porta de comunicação \n'
                              'que o Arduino está conectado')

    # Criando texto principal
    img_conf = tk.PhotoImage(file='Componentes/Conf Imagem.gif').subsample(8, 8)
    texto_principal = tk.Label(janela, text='Configurações do Experimento',
font=('Arial', '25', 'bold'),
                               image=img_conf, compound='left', bg='#b5b5b5')
    texto_principal.pack()

    # criando campo para informar a porta usb conectada
    texto_porta = tk.Label(janela, text='Qual a porta de comunicação (COM) utilizada
pelo Arduino?',
                           font=('Arial', '16', 'bold'))
    texto_porta.place(x=355, y=150)
    define_porta = tk.Scale(janela, from_=0, to=20, orient=tk.HORIZONTAL,
font=('Arial', '12', 'bold'))
    define_porta.place(x=600, y=190)
    botao_duvida = tk.Button(janela, text='?', width=3, height=1, font=('Arial', '12',
'bold'),
                             cursor='hand2', command=ajuda_USB)
    botao_duvida.place(x=710, y=204)
```

```
    #criando campo para informar quanto tempo de duração terá o experimento
    texto_tempo = tk.Label(janela, text='Informe o tempo, em minutos, de duração do
experimento:',
                           font=('Arial', '16', 'bold'), justify='center')
    texto_tempo.place(x=355, y=250)
    define_tempo = tk.Entry(janela, bd=5, font=('Arial', '12', 'bold'), justify='center')
    define_tempo.place(x=565, y=290)

    # criando campos para informar qual a cor dos corpos utilizados na montagem
    #primeira placa
    texto_cor1 = tk.Label(janela, text='Qual a cor da primeira placa?(Conectada na
porta A1)',
                          font=('Arial', '16', 'bold'), justify='center')
    texto_cor1.place(x=355, y=330)
    define_cor1 = tk.Entry(janela, bd=5, font=('Arial', '12', 'bold'), justify='center')
    define_cor1.place(x=565, y=370)

    #segunda placa
    texto_cor2 = tk.Label(janela, text='Qual a cor da segunda placa?(Conectada na
porta A2)',
                          font=('Arial', '16', 'bold'), justify='center')
    texto_cor2.place(x=355, y=410)
    define_cor2 = tk.Entry(janela, bd=5, font=('Arial', '12', 'bold'), justify='center')
    define_cor2.place(x=565, y=450)

    # criando botão começar
    img_exp = tk.PhotoImage(file='Componentes/Iniciar Imagem.gif').subsample(8, 10)
    botao_iniciar = tk.Button(janela, image=img_exp, cursor='hand2',
command=define_radiacao)
    botao_iniciar.place(x=575, y=495)

    # criando botão voltar e sair menu principal
    img_menu = tk.PhotoImage(file='Componentes/Voltar Imagem.gif').subsample(20,
20)
    botao_menu = tk.Button(janela, image=img_menu, bg='#63B8FF', cursor='hand2',
command=voltar)
    botao_menu.place(x=495, y=600)
    img_sair = tk.PhotoImage(file='Componentes/Sair Imagem.gif').subsample(20, 20)
    botao_sair = tk.Button(janela, image=img_sair, bg='#cd3333', cursor='hand2',
command=sair)
    botao_sair.place(x=745, y=600)

    elementos = [texto_principal, texto_porta, define_porta, botao_duvida,
botao_iniciar, botao_menu,
                 botao_sair, texto_cor1, texto_cor2, define_cor2, define_cor1,
define_tempo, texto_tempo]
    if aviso_cancela == 'não':
        aviso('Certifique-se que o Arduino está ligado\n'
              'na porta USB e que todas as conecções\n'
                    'foram feitas corretamente!')
```

```
    janela.mainloop()

# Criando janela que monitora o experimento de absorção de radiação luminosa
def monitoramento_radiacao():

    #definindo função voltar:
    def voltar():
        for i in elementos:
            i.destroy()
        placa.exit()
        global aviso_cancela
        aviso_cancela = 'sim'
        menu_conf_rad()

    # definindo função a ser executa do Arduino no experimento de absorção de
radiação
    def radiacao_comecou():

        #função que gera o gráfico
        def graficos():
            #configurando eixo 1
            plt.plot(periodos, leituras1, color='blue', label='{}'.format(cor1))

            #configurando eixo 2
            plt.plot(periodos, leituras2, color='red', label='{}'.format(cor2))

            plt.ylim([0, 70])
            plt.title('Experimento Absorção de Radiação Eletromagnética')

            plt.grid(True)
            plt.xlabel('Tempo (s)')
            plt.ylabel('Temperatura (ºC)')
            plt.legend()
            plt.show()

        botao_grafico.config(command=graficos)
        sensor1 = placa.get_pin('a:1:i')
        sensor2 = placa.get_pin('a:2:i')

        y = 0 #variável para aparecer o primeiro quadro
        time.sleep(2)
        leituras1 = []
        leituras2 = []
        periodos = [0]
        tempo_total = 0

        while tempo_total <= tempo_definido:
            temperatura1 = sensor1.read() / 0.00201 #conversação para que a leitura
saia em graus celcius
```

```
            temperatura2 = sensor2.read() / 0.00201
            leituras1.append(temperatura1)
            leituras2.append(temperatura2)
            tempo = time.time()
            if y == 0:
                texto_resultados.config(text='Em 0.00s\n a temperatura na placa {} é
{:.2f}ºC e\n'
                                        'a temperatura na placa {} é {:.2f}ºC'.format(cor1,
temperatura1,
                                                                                      cor2, temperatura2))
                t0 = tempo
                y += 1

            else:
                periodo = tempo - t0
                tempo_total += periodo
                periodos.append(tempo_total)
                t0 = tempo
                texto_resultados.config(text='Em {:.2f}s\n a temperatura na placa {} é
{:.2f}ºC e\n'
                                        'a temperatura na placa {} é {:.2f}ºC'.format(tempo_total,
cor1,
                                                                                      temperatura1, cor2, temperatura2))
            frame2.update()
            time.sleep(0.3)
        texto_resultados.config(text='Em {:.2f}s\n a temperatura na placa {} é {:.2f}ºC
e\n'
                                'a temperatura na placa {} é {:.2f}ºC\n\n'
                                'EXPERIMENTO FINALIZADO!\n'
                                'Click em GRÁFICOS!'.format(tempo_total,
cor1,temperatura1, cor2,
                                                            temperatura2))
        placa.exit()

    #faz conecção com o Arduíno antes de abrir a janela
    it = pf.util.Iterator(placa)
    it.start()

    # criando frame
    frame = tk.Frame(janela, bg='white')
    frame.pack()

    # criando logo
    img_mon = tk.PhotoImage(file='Componentes/Monitora Imagem.gif').subsample(3,
6)
    texto = tk.Label(frame, text='Resultados medidos:', image=img_mon,
compound='top', font=('Arial', '20'),
                     bg='#b5b5b5')
    texto.pack()
```

```
    # criando frame 2, este é o que atualiza os resultados
    frame2 = tk.Frame(frame)
    frame2.pack()

    # separando espaço texto de resultados
    texto_resultados = tk.Label(frame2, height=14, text='Clique em START para\n'
                                            'começar o experimento', font=('Arial', '14', 'bold'),
                                            bg='white', justify='center')
    texto_resultados.pack()
    # criando botões
    img_exp = tk.PhotoImage(file='Componentes/Iniciar Imagem.gif').subsample(10,
12)
    botao_começar = tk.Button(janela, image=img_exp, cursor='hand2',
text='Começar', compound='top',
                          font=('Arial', '10', 'bold'), bg='#b5b5b5',
command=radiacao_comecou)
    botao_começar.place(x=470, y=500)
    img_graf = tk.PhotoImage(file='Componentes/Grafico Imagem.gif').subsample(10,
12)
    botao_grafico = tk.Button(janela, text='Gráfico', image=img_graf, compound='top',
                          font=('Arial', '10', 'bold'), bg='#b5b5b5')
    botao_grafico.place(x=615, y=500)
    img_voltar = tk.PhotoImage(file='Componentes/Voltar
Imagem2.gif').subsample(10, 12)
    botao_voltar = tk.Button(janela, text='Voltar', image=img_voltar, compound='top',
font=('Arial', '10', 'bold'),
                         command=voltar, bg='#b5b5b5')
    botao_voltar.place(x=760, y=500)
    elementos = [frame, botao_grafico, botao_começar, botao_voltar]
    janela.mainloop()

janela_inicial()
```