Experimentos Didáticos com Arduino para o ensino de Física

Guia com propostas de uso da Cultura Maker com Arduino para auxilio didático nos conteúdos de física

PPGEEProf - Programa de Pós-Graduação em Educação Escolar
Mestrado e Doutorado Profissional

Silas Jader Pereira Silva
Juracy Machado Pacífico

UNIR
FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA

Porto Velho - RO, 2022.

Autores
Silas Jader Pereira Silva
Juracy Machado Pacífico

Projeto Gráfico
Silas Jader Pereira Silva

Capa e ilustrações
Silas Jader Pereira Silva

Dados Internacionais de Catalogação na Publicação
Fundação Universidade Federal de Rondônia
Gerada mediante informações fornecidas pelo(a) autor(a)

S585e Silva, Silas Jader Pereira.
Experimentos didáticos com Arduino para o ensino de física. / Silas Jader Pereira Silva. -- Porto Velho, RO, 2022.
52p. : il.

1. Educação escolar. 2. Tecnologias educacionais. 3. Cultura Maker e Arduino. 4. Ensino de física. 5. Didática desenvolvimental I. Produto final de dissertação - Programa de Pós-Graduação em Educação Escolar - Mestrado e Doutorado Profissional. II. Fundação Universidade Federal de Rondônia.

CDU 37:53

Bibliotecária Ozelina do Carmo de Carvalho Saldanha CRB 11/ 486

UNIR
FUNDAÇÃO UNIVERSIDADE
FEDERAL DE RONDÔNIA

Experimentos Didáticos com Arduino para ensino de Física

Guia com propostas de
uso da Cultura Maker com Arduino
para auxilio didático nos conteúdos de física

Silas Jader Pereira Silva
Juracy Machado Pacífico

Descrição técnica do produto

- **Título:** Experimentos didáticos com Arduino para o ensino de física
- **Origem do Produto:** Trabalho de dissertação denominado "Tecnologias e Educação: A Cultura Maker com Arduino como Possibilidade para o Ensino no Curso de Licenciatura em Física do IFRO."
- **Área de Conhecimento:** Educação.
- **Público Alvo**: Licenciandos e professores de física, docentes de áreas diversas.
- **Tipo de Produto:** Material didático. Conforme definição da área de educação, trata-se de"Produto de apoio/suporte com fins didáticos na mediação de processos de ensino e aprendizagem em diferentes contextos educacionais."
- **Sub-tipo de Produto**: Texto em formato digital
- **Finalidade**: Apresentar o uso da ferramenta maker Arduino, como proposta didática para o ensino de física.
- **Aplicabilidade e Organização do Produto**: O produto foi organizado em oito experimentos práticos com Arduino para o ensino de física, com o intuito de propiciar a licenciandos e docentes de física e disciplinas afins, um guia passo-a-passo com experimentos que podem ser replicados em sala de aula. Cada experimento neste guia está organizado em tópicos com orientações, sendo: componentes em destaque, diagrama de montagem/ funcionamento, código-fonte, comentário do código, desdobramentos e material para consulta.
- **Registro do Produto/Ano:** Biblioteca da UNIR - Porto Velho, 2022.
- **Avaliação do Produto:** O produto foi avaliado pela banca de defesa da dissertação, composta por três doutores das áreas relacionadas.
- **Disponibilidade:** Todo material neste guia deve ser usado dando-se o respectivo crédito. Uso para fins educacionais. É proibido o uso comercial deste material.
- **Divulgação:** Em formato digital.
- **URL:** Produto acessível no site PPGEEProf-UNIR (www.mepe.unir.br)
- **Instituições envolvidas:** UNIR/IFRO
- **Idioma:** Português
- **Cidade:** Porto Velho

Resumo

Trata-se de um guia com propostas de uso do Arduino como ferramenta da Cultura Maker para o ensino de física. O trabalho foi elaborado como um dos requisitos acadêmicos e formativos para obtenção do título de Mestre do Programa de Pós-graduação em Educação Escolar, Mestrado e Doutorado Profissional, da Universidade Federal de Rondônia. A partir das conclusões da dissertação, foram extraídas experiências para o auxílio didático dos conteúdos de física básica, como forma de levar aos licenciandos e docentes da área uma maneira prática, didática e descomplicada de realizar, com seus alunos, experiências simples, porém enriquecedoras do ponto de vista da didática. Par isso foram escolhidas, com ajuda de um profissional docente da área de física, oito experiências descritas passo-a-passo no guia, objetivando introduzir os docentes, ou futuros docentes, na cultura maker com uso de uma ferramenta acessível e largamente difundida. Cada uma das experiências são apresentadas, divididas didaticamente em tópicos, fáceis de serem executados, em etapas claras, para tornar a reprodução e também os desdobramentos (avanços) acessíveis a qualquer professor, mesmo, e principalmente, aqueles que não são da área de informática ou tecnologia. Almeja-se que este material possa, de fato, auxiliar professores e alunos a vivenciarem, na prática, conceitos mostrados em sala de aula, quase sempre de forma apenas teóricos, e, com isso, desenvolver uma conceituação mais profunda e desenvolvimental.

Palavras-chave: Educação Escolar. Tecnologias Educacionais. Cultura Maker e Arduino. Ensino de Física. Didática Desenvolvimental.

Abstract

This is a guide with proposals for using Arduino as a Maker Culture tool for teaching physics. The work was prepared as one of the academic and training requirements for obtaining the title of Master of the Postgraduate Program in School Education, Masters and Professional Doctorate, at the Federal University of Rondônia. From the conclusions of the dissertation, experiences were extracted for the didactic aid of the contents of basic physics, as a way of bringing to the undergraduates and teachers of the area a practical, didactic and uncomplicated way of carrying out, with their students, simple but enriching experiences of the didactic point of view. For this, eight experiences described step-by-step in the guide were chosen, with the help of a physics professor, with the aim of introducing professors, or future professors, into the maker culture using an accessible and widely disseminated tool. Each of the experiences is presented, didactically divided into topics, easy to perform, in clear steps, to make the reproduction and also the unfoldings (advances) accessible to any teacher, even, and especially, those who are not in the area of computing. or technology. It is hoped that this material can, in fact, help teachers and students to experience, in practice, concepts shown in the classroom, almost always in a theoretical way, and, with that, develop a deeper and developmental conceptualization.

Keywords: School Education. Educational Technologies. Maker Culture and Arduino. Physics Teaching. Developmental Didactics.

Índice

Considerações iniciais 06
Arduino 08
Arduino UNO 09
Cultura maker 12
Ferramentas 13
Materiais 13
Estrutura do guia 14
Barra lateral 15
8 – Experimentos 16
1 – Pisca LED: “Olá Mundo” 17
2 – Sistema de luz e cores 21
3 – Velocidade do som 25
4 – Cálculo de velocidade 28
5 – Plano inclinado 32
6 – Roda de cores 37
7 – Frequência sonora 39
8 – Equilíbrio estático 43
Considerações Finais 49
Referências 50


DIGITAL (PWM~)
SMD EDITION
UNO
ARDUINO
ON
ICSP
ARDUINO.ORG
MADE IN ITALY
POWER
ANALOG IN
GND

Considerações iniciais

Este produto educacional é um guia prático de como montar, programar e testar alguns sketches (experimentos) com uso de Arduino, que podem ajudar o professor de física, no caminho da didática.

Ele se origina na pesquisa de mestrado intitulada Tecnologias e Educação: A Cultura Maker com Arduino como Possibilidade para o Ensino no Curso de Licenciatura em Física do IFRO. A pesquisa é fundamentada na Didática Desenvolvimental, que tem como fundamento a Psicologia Histórico Cultural.

Para a elaboração deste guia, contou-se com a ajuda do professor de física Fabrício Araújo de Sousa, coordenador do laboratório de Física Aplicada do IFRO, campus porto Velho Calama e também professor de física na E.E.E.F.M Barão do Solimões em Porto velho – RO.

Neste sentido, este guia pretende ajudar na formação de professores para o enriquecimento de sua prática no dia-a-dia da sala de aula com a ajuda da cultura maker.

É sabido que as formas convencionais de ensino, em que recursos como "quadro e giz" ainda são largamente utilizados, são cada vez mais questionadas. Os questionamentos, e por conseguinte, as pressões são demandas de alunos cada vez mais acostumados com recursos tecnológicos do seu próprio tempo.

Acostumados a celulares e computadores, esses alunos demandam didáticas outras, capazes de prenderem sua atenção e, acima de tudo, estabelecer uma relação dos conceitos apresentados com a realidade que os cercam. A didática desenvolvimental prima especialmente por esse aspecto: os conceitos científicos não podem estar desligados da realidade.

> Trata-se de uma perspectiva de qualidade da educação centrada no desenvolvimento humano para uma sociedade justa e democrática, amparada em princípios da teoria histórico-cultural, concebendo a educação escolar como lugar de apropriação dos conceitos científicos e da formação do pensamento teórico enquanto meios instrucionais para a promoção do desenvolvimento intelectual e da formação global da personalidade dos estudantes. (FREITAS; LIBÂNEO, 2019, p. 382)

Este trabalho tem como público alvo professores de física, mas nada impede que docentes de várias áreas se beneficiem dele, afinal, uma educação que pretenda fazer interconexões dos conceitos científicos com a realidade concreta, não está circunscrita a nenhuma disciplina de forma exclusiva.

Os experimentos apresentados utilizam o Arduino que é uma conhecida plataforma de prototipação eletrônica, muito utilizada para o ensino de robótica desenvolvido com o intuito de facilitar a aprendizagem desta área. Com o tempo passou a ser adotada por hobbystas e artistas que observavam na facilidade de uso um aliado para seus projetos. Isto significa que não há a necessidade de aprofundamentos em ciências como elétrica ou eletrônica. Claro que, para quem pretenda ser um especialista, quanto mais se conhece

Considerações iniciais - Continuação...

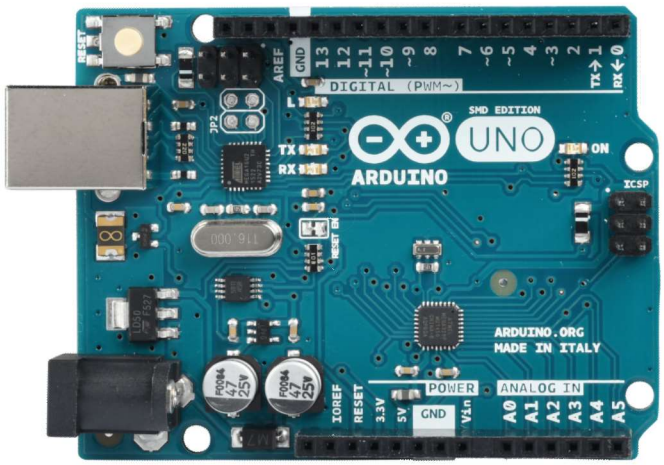
destas áreas, melhor.

O Arduino é uma plataforma livre, que significa que seu hardware (parte física e projeto eletrônico) e seu software (programa utilizado por ele) são de código aberto e podem ser utilizados, reproduzidos, copiados e até comercializados livremente1. A cultura maker pressupõe o “fazer” como princípio e utilizar softwares proprietários implicaria no pagamento de licenças e/ou direito de uso, impossibilitando o uso por muitos.

Para que a cultura maker com Arduino para o ensino de física possa ser uma realidade aplicada às escolas públicas, alvo maior deste trabalho, não basta apenas formar professores. É preciso que os recursos sejam acessíveis. Portanto, a liberdade proporcionada por essa categoria de plataforma reflete também em custos acessíveis, o que as tornam inclusivas para escolas com pouco recursos disponíveis, em que um laboratório de física aplicada seria impraticável. Outra grande vantagem da plataforma é a quantidade gigantesca de atuadores e sensores disponíveis para a mesma.

Sensores nada mais são que dispositivos eletrônicos que interagindo com o meio físico ambiente, conseguem captar certas grandezas físicas, como temperatura e pressão, por exemplo, e enviá-las ao Arduino que atua como controlador, o cérebro do projeto. Uma vez no Arduino, estes dados são tratados pelo programa feito pelo usuário (professor, aluno, etc.), para produzirem os resultados desejados. Um bom exemplo é um sensor de temperatura, capaz de captar o calor no ambiente e mandar essa informação para o controlador que através do programa toma a decisão do que fazer com essa informação (temperatura), como, por exemplo, ligar um dispositivo de refrigeração caso a temperatura atinja um determinado valor.

Atuadores , ao contrário dos sensores, recebem ordens do controlador (Arduino) e interagem com o ambiente físico. Normalmente são motores, telas displays (telas), artefatos de produção de som, movimento, etc.



Arduino UNO

Fonte: Produzida por Silas J. P. Silva 2022

1 Apenas o nome e a logo do Arduino são marcas registradas.

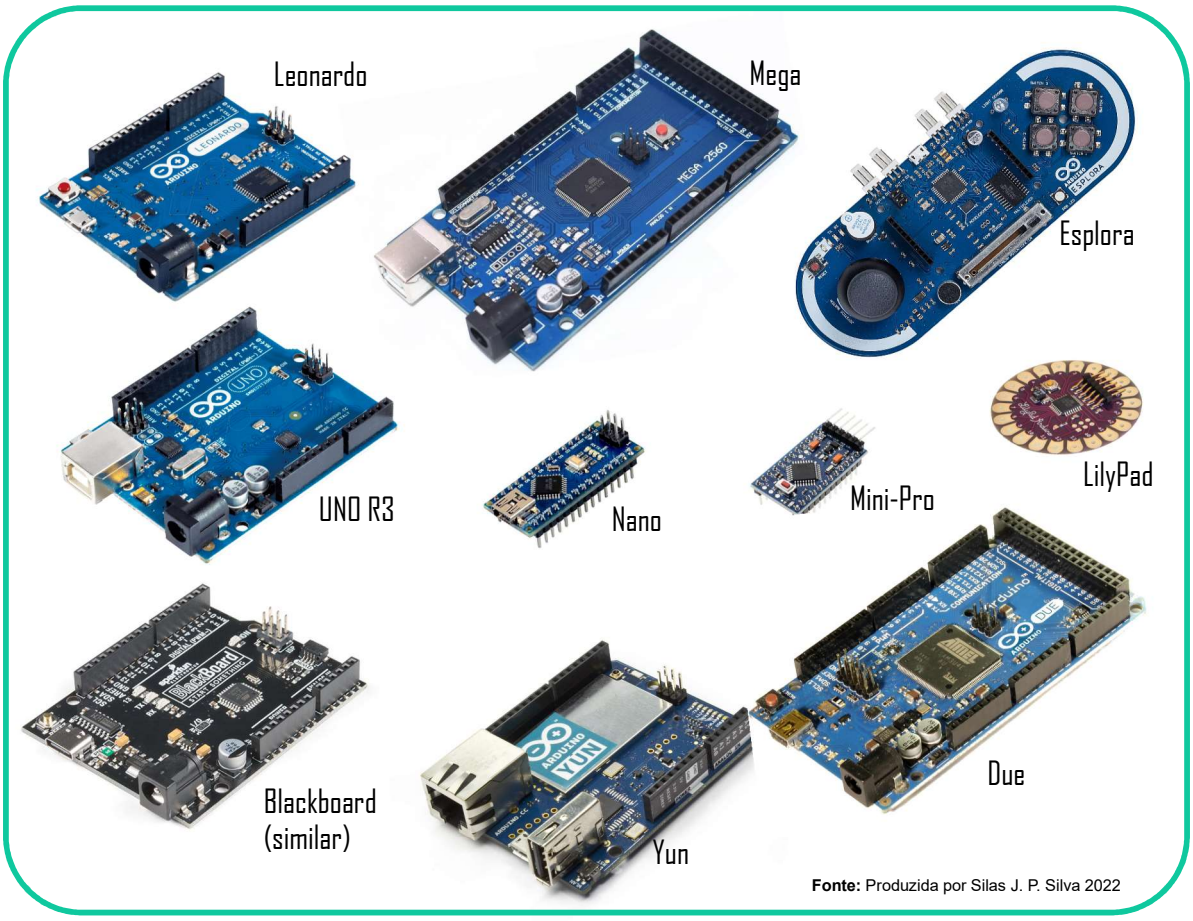
Considerações iniciais - Continuação...

Este guia traz oito experimentos que podem ser facilmente replicados a um baixíssimo custo. Cada um deles conta com a lista dos sensores e/ou atuadores utilizados, uma breve explicação do seu funcionamento, um diagrama de ligação destes componentes ao Arduino e uma aplicação possível no ensino de física. Alguns destes projetos podem trazer também possíveis incrementos, desdobramentos que podem ser experimentados por professores e alunos.

Embora verse sobre Arduino, este guia não se trata de um curso do mesmo. Quem desejar se aprofundar no assunto deve buscar ajuda em um dentre as centenas de sites que tratam sobre o assunto. No entanto, apresentaremos, nos próximos itens, algumas definições e explicações.

Arduino

O modelo indicado aqui é o Arduino UNO, o modelo básico de onde derivam todos os outros. Caso você use outro modelo, nenhuma adaptação será necessário no código, porém algumas pequenas mudanças podem ocorrer no hardware. Os modelos genéricos funcionam tão bem quanto os originais e, se baseados no UNO, serão exatamente iguais quanto aos pinos, alimentação e disposição das portas. Abaixo alguns modelos de Arduino.



Fonte: Produzida por Silas J. P. Silva 2022

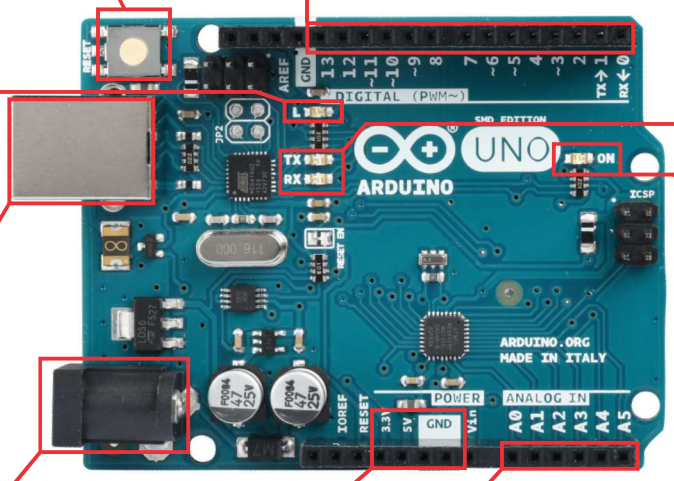
Arduino UNO

O modelo específico apresentado em todos os experimentos neste guia é o Arduino UNO. Traremos logo abaixo algumas características do projeto (desenho) da placa. Os pormenores do seu funcionamento podem ser encontrados em www.arduino.cc. Vale destacar mais uma vez, que nada impede que qualquer outro modelo seja utilizado, pois, o código é idêntico para qualquer um deles, bastando apenas identificar onde ficam localizadas cada parte na placa e informar o referido modelo na "IDE" do Arduino, a ser visto mais adiante.

LED BUILT-IN: LED ligado à porta 13. Para fazer um LED piscar para teste do Arduino. Não precisa ligar nada a ele, uma vez que já possui este LED ligado a porta 13. Caso você utilize a porta para outra função é provável que este LED fique piscando, o que é normal.

Botão de RESET: provoca um *reset* na placa. É como se o Arduino fosse desligado e ligado novamente

Portas digitais: as portas de 0 a 13 são digitais. Além delas, as portas de A0 a A5 podem também ser configuradas como digitais de entrada caso necessário.



Porta USB: usada para conectar o Arduino ao computador. É através desta porta que o programa (código) é enviado ao Arduino e também é a porta usada para enviar e receber informação entre o computador e nossa placa

Saída de 3.3V e 5V: qara ligar pequenos sensores que necessitem destas voltagens para funcionar. O GND é o negativo (terra)

Portas de entradas Analógicas: de A0 a A4.

Alimentação Externa: quando se liga o Arduino ao PC ele já estará com energia e pronto pra uso. Porém, se precisar ligá-lo sem o uso do computador é aqui que você ligará uma bateria ou uma fonte de alimentação entre 7v e 12v

LED ON: informa quando o Arduino estiver ligado.

LEDs TX RX: informa quando o Arduino está se comunicando com o computador

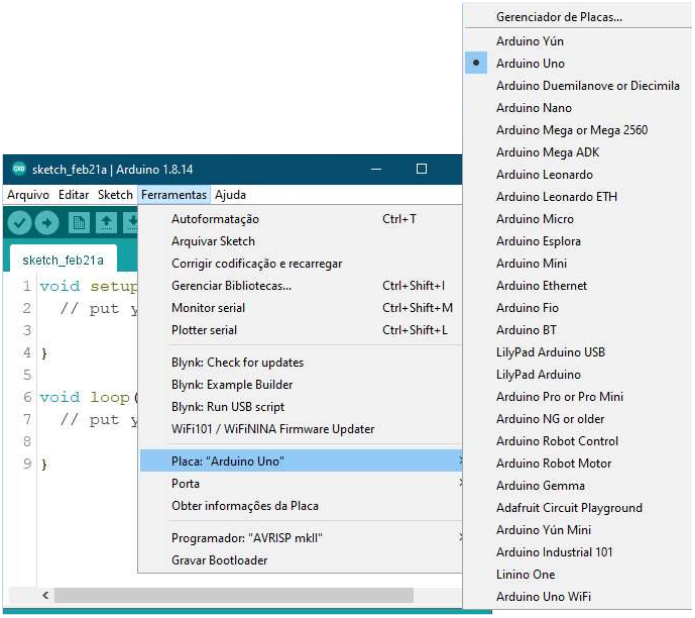
Fonte: Produzida por Silas J. P. Silva 2022

Arduino continuação...

Existe um pequeno programa onde se digita o código-fonte e que é usado para fazer a carga ou gravação deste no Arduino: a IDE - Integrated Development Environment ou Ambiente de Desenvolvimento Integrado. Esse programa pode ser baixado gratuitamente em https://www.arduino.cc/en/software. Abaixo a tela inicial da IDE do Arduino.

Fonte: Produzida por Silas J. P. Silva 2022

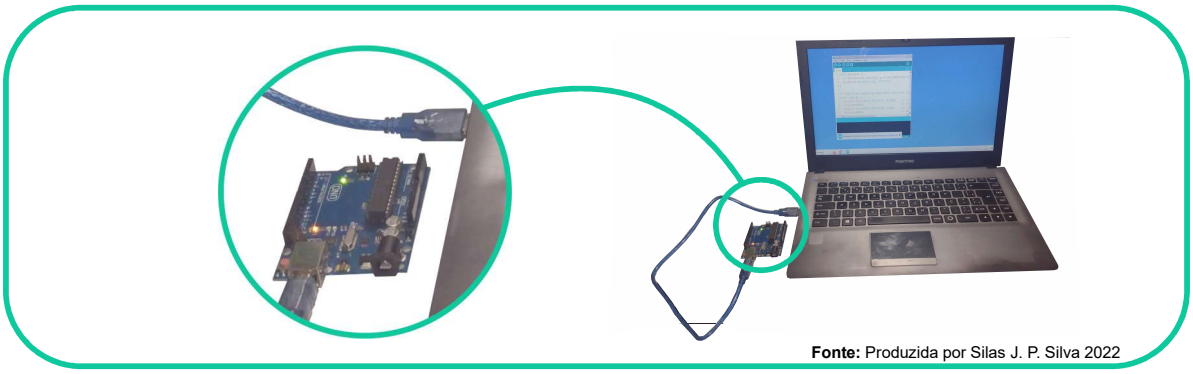
Caso esteja trabalhando com outro modelo de Arduino, este modelo deve ser definido na IDE para que a compilação e a gravação do programa ocorra normalmente, como se segue na imagem.



Fonte: Produzida por Silas J. P. Silva 2022

Arduino continuação...

O Arduino pode ser alimentado por baterias ou fontes de alimentação de 7,5V a 12V, mas na maioria das vezes pode trabalhar diretamente ligado a um notebook. Para fazer a carga (gravar) o código do projeto no Arduino, ele tem que estar ligado ao computador, é claro.

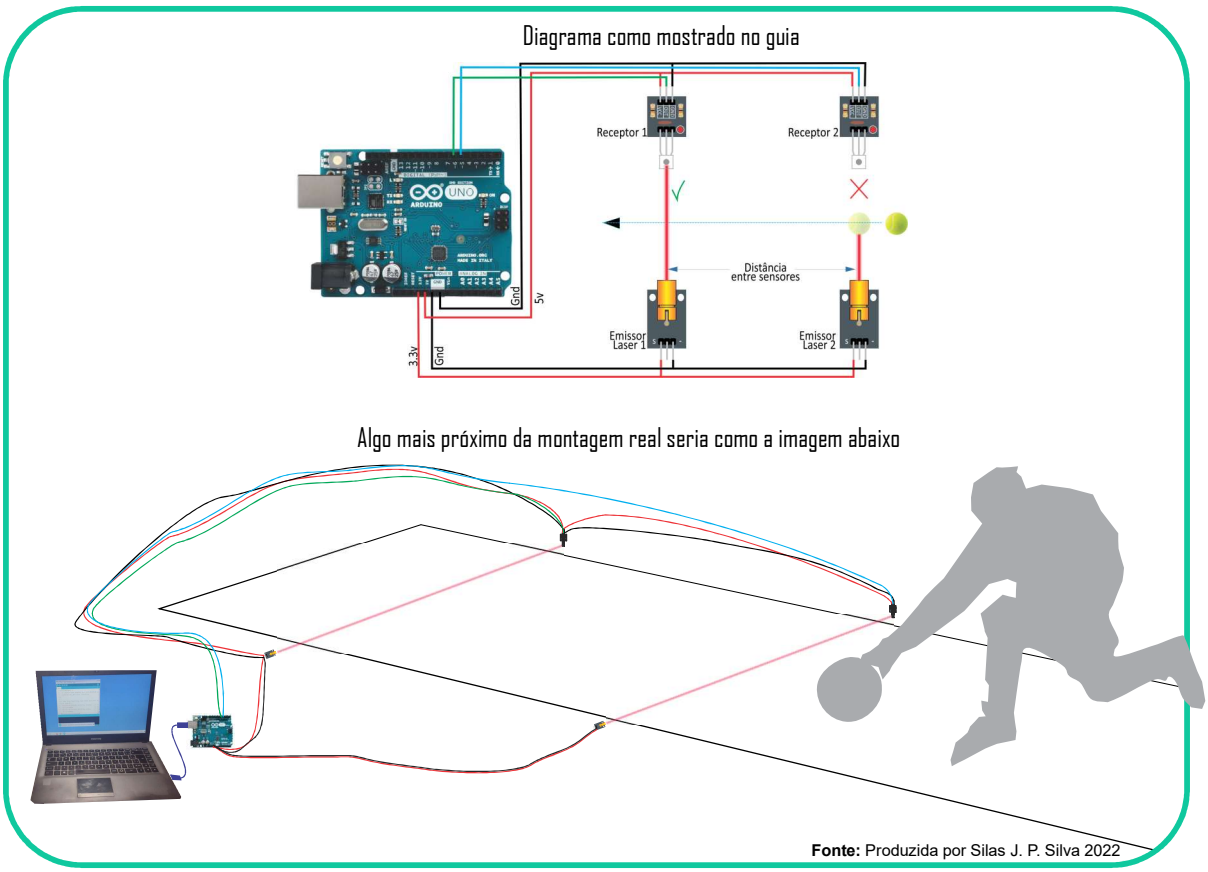


Fonte: Produzida por Silas J. P. Silva 2022

Observações:

- Para a descrição dos componentes não serão inclusos o Arduino, fios ou *protoboard*. O Arduino e os *jumpers* (fios) são obrigatórios em todos os experimentos e a protoboard é dispensável.

- Os diagramas deste guia estão propositalmente fora de escala, para que seja possível enquadrar em uma imagem ilustrativa no documento. As escalas corretas devem refletir a intenção do experimento como no exemplo da imagem do experimento abaixo:



Fonte: Produzida por Silas J. P. Silva 2022

Cultura Maker

O movimento ou cultura maker, propõe uma visão de mundo em que qualquer pessoa pode construir, reformular ou reparar os objetos físicos do mundo que os cerca. Não apenas utilizar coisas, mas, principalmente, descobrir como estas coisas foram feitas e como elas funcionam.

No início, o movimento *maker* não estava diretamente relacionado à educação. Porém, seu potencial nas experiências de aprendizagem foi rapidamente descoberto, e no Brasil, a cultura maker ganhou corpo com a implantação de espaços destinados especificamente para esse fim. Os FabLabs surgiram, inicialmente, em universidades como um laboratório onde equipamentos eram disponibilizados e compartilhados pela comunidade local para a criação e construção de artefatos diversos. Hoje tem se difundido fora dos espaços *fablabs* e tem entrado nas salas de aulas regulares em escolas de vários níveis.

O movimento maker nasce como forma genuína de experimentação e na educação, esta prática valoriza a forma criativa da aprendizagem e a resolução de problemas como processo de retenção do conhecimento. Outro aspecto bem desejável na interação dos estudantes com o conteúdo é a autonomia que o fazer "mão da massa" da cultura maker proporciona.

Desta maneira a cultura maker traz uma boa possibilidade para o professor em termos de recurso didático, em que os conceitos teóricos são construídos, não apenas a partir dos conteúdos teóricos de livros e apostilas, mas também da experimentação destes conteúdos.

> Neste sentido, o movimento maker vem sendo considerado como o próximo salto educacional e tecnológico, apresentando-se como alternativa às aulas tradicionais, que priorizam as metodologias expositivas consideradas passivas e repetitivas pela maioria dos estudantes. (BROCKVELD, TEIXEIRA e SILVA, 2017, p. 6)

Desta forma, a cultura maker se apresenta como proposta de didática e traz o Arduino como ferramenta tecnológica que pode promover uma aproximação dos conteúdos de física ao mundo das tecnologias da informação, tão presentes na vida dos estudantes modernos.

Esta proposta de conciliar o ensino de física com tecnologia da informação através de uma ferramenta da cultura maker também tenta refazer caminhos de um ensino mais desenvolvimental e completo, atuando como incentivo aos alunos na busca de um uso mais crítico das tecnologias que lhes são apresentadas como prontas e acabadas.

... cultura maker

Ferramentas

A cultura *maker* pressupõe uma prática de construção, criação, confecção de artefatos de vários tipos e para vários fins. Isso significa que alguma interação com ferramentas é necessária e para construir os experimentos trazidos neste guia, não será diferente.

Para adentrar no mundo *maker*, é preciso ter em mente que a afinidade com o uso de ferramentas acontece de forma natural e faz parte do processo de apropriação do conhecimento. Nenhuma habilidade profissional será exigida e a maioria das necessidades poderá ser supridas por ferramentas que a maioria das pessoas já possui, como tesoura, alicate, serra de mão, martelo e furadeira, por exemplo. Vale lembrar que a falta de uma ferramenta não inviabiliza a construção, apenas pode torná-la mais lenta.

Abaixo algumas ferramentas que pode ser útil de maneira geral.

Fonte: Produzida por Silas J. P. Silva 2022

Materiais empregados

Os materiais empregados vai depender do *sketch* que se deseja realizar. A criatividade e necessidades de cada projeto são quem vão determinar os materiais a serem utilizados. Abaixo, seguem-se alguns exemplos:

Fonte: Produzida por Silas J. P. Silva 2022

OBS. Materiais reciclados e recuperados de sucata, podem e devem ser usados nos projetos. Coisas como madeira, papeis, fios e qualquer material que possam ser re-utilizados ou dado um novo uso. A cultura *maker* prima pela reciclagem, pois muitos materiais que podem gerar bons projetos são jogados no lixo diariamente, por pessoas e empresas.

Estrutura deste guia

O guia traz, para cada experimento, uma breve síntese do funcionamento e logo depois a apresentação propriamente dita, estruturada nos seguintes: Componentes em destaque; Diagrama e montagem/funcionamento; Código fonte; Comentário do código; Desdobramentos; Para Consulta caso haja.

Componentes em destaque

Apresenta quais componentes serão empregados no experimento e o funcionamento dos mesmos. Como já referido, Arduino, *proatboard* e fios, não serão destacados. Além disso, poderá ser requerido o uso de papelão, pequenos pedaços de madeira, cola, tesoura, alicate, martelo e outras ferramentas básica. Lembrem-se se tratar de cultura *maker,* ou seja, mão na massa. A lista completa será mostrada na Barra Lateral vista mais adiante.

Diagrama e montagem/funcionamento

O diagrama traz (em escala adaptada) a forma como os componentes devem ser ligados entre si. Também será apresentado um texto com a sugestão da montagem física do experimento. Essa montagem pode (e deve) ser modificado pelo professor a qualquer momento caso ele ache interessante para a sua aula.

Código-fonte

O programa que deve ser gravado no Arduino é referido como "código-fonte". Isso porque o funcionamento de todo o experimento depende deste código. Ele será apresentado em forma de texto e poderá ser copiado diretamente deste documento e colado na IDE do Arduino.

Comentário do código

Neste tópico será comentado a função de cada linha, ou grupo de linhas, do código-fonte, para que o usuário possa se familiarizando com a programação.

Desdobramentos

Diz respeito à possibilidade de outros arranjos físicos com os mesmo componentes. Caso o experimento possa ser utilizado de mais de uma forma para o ensino de física, essa possibilidade de uso será apresentada neste tópico.

Para consulta

Alguma informação que pode precisar ser acessada constantemente para a construção do experimento, como tabelas de valores, links para outras informações, etc.

As próximas páginas trazem os experimentos como proposta didática para o ensino de física. Esperamos que este guia sirva de ponto de partida para os docentes que queiram iniciar os estudos na área.

... estrutura deste guia

- Ensino de física
- Lista de materiais e ferramentas
- Observações

Barra lateral

A barra lateral traz:

Ensino de Física

Possibilidades do *sketch* para o ensino de física. Neste tópico será apresentado uma sugestão de aplicação do experimento para a didática no ensino de física em sala de aula.

Leva-se sempre em consideração neste trabalho, que o docente é senhor de sua prática, e embora as sugestões tenham sido elaborados com a contribuição de profissionais da área de física, nada impede e até se incentiva, que o professor vislumbre outra aplicação didática para o experimento, diferente daquele sugerido por esse guia. Na verdade, isso é desejável e é também objetivo deste trabalho: que o professor possa caminhar e experimentar cada vez mais as possibilidades da cultura maker com Arduino nas disciplinas de física.

Lista de Materiais e ferramentas

- A lista completa de materiais e algumas ferramentas que podem ser necessárias para a confecção do experimento. A única barreira para o uso de materiais e ferramentas é a criatividade de cada um.

Já falamos sobre o de ferramentas e materiais, mas vale destacar novamente, que o uso de materiais reciclados, descartes e sucatas em geral devem ser incentivado para despertar nos estudantes uma visão mais "desenvolvimental", a medida em que refletem sobro o uso consciente de recursos e materiais.

A questão do ferramental envolvido nos projetos deste manual, também refletem a proposta de tornar a cultura *maker* acessível a escolas públicas, que não dispõem de recursos para montar laboratórios sofisticados. Ferramentas simples de uso comum no dia-a-dia dos próprios alunos e professores têm preferência nos projetos apresentados aqui.

Observações

Nesta secção serão apresentadas observações importantes. Observações em geral podem aparecer em qualquer parte deste manual, mas sempre que algo merecer destaque, aparecerá como observações na barra lateral.

Oito Experimetos

Seguem-se os experimentos propostos neste guia de Cultura maker com Arduino para o Ensino de Física

1

Pisca LED: "Olá Mundo"

Ensino de física

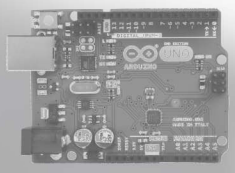
Neste primeiro experimento o professor de física pode explorar os conteúdos relacionados às leis de Ohm, fazendo cálculos sobre qual resistor usar em série com o LED, em função da voltagem/amperagem que se deseja.

Lista de materiais e ferramentas

01 - LED vermelho difuso
01 - Resistor de 220 Ω
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

Observações

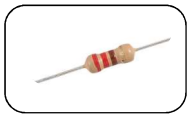
1 - As informações sobre voltagens e amperagens dos componentes utilizados neste manual estão largamente disponíveis na internet. No caso do LED, basta fazer uma busca pelo "datasheet"do mesmo. Datasheets são as informações técnicas disponibilizadas pelos fabricantes de componentes eletrônicos. Os LEDs de cores difusas têm voltagem que variam de 1.8V a 2.8V em média. Caso queira fazer uma busca mais exata, basta digitar no Google "datasheet LED vermelho difuso".
2 - Os resistores são identificados pelas suas cores. No tópico "Para Consulta" é apresentada uma tabela de cores/valores de resistores.



Este primeiro experimento pretende iniciar o professor na plataforma Arduino bem como neste manual. O *sketch* apresentado a seguir pode ser totalmente simulado em www.tinkercad.com.

Componentes principais

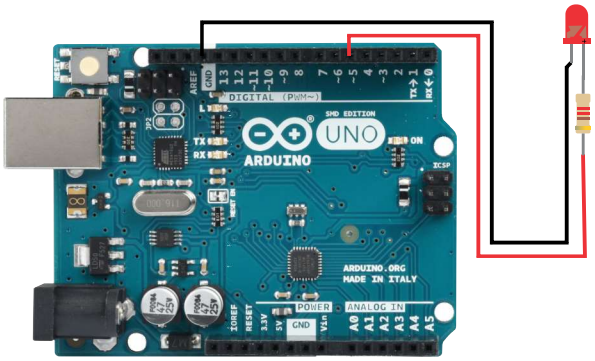
LED Vermelho difuso de 5mm.



Resistor de 220Ω ou 330Ω

Além de *jumppers* e do próprio Arduino, serão utilizados dois componentes bastante conhecidos: o LED (*Ligth Emitter Diode*), que consiste em uma espécie de lâmpada usada para sinalização e o resistor, que tem a função de adequar a tensão sobre o LED.

Diagrama e montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Montagem

Os LEDs estão disponíveis em diversas cores e modelos. Cada tipo possui suas próprias características de tensão e corrente. Aqui utilizaremos um LED de 5mm vermelho que, segundo a documentação do fabricante[3]. Opera com tensão entre 2V e 2,5V e com corrente de 20mA, por isso a necessidade de utilizar um resistor para limitar a tensão, visto que o Arduino envia sinais de 5V aos dispositivos ligados a ele. O ideal, segundo a Lei de Ohm, seria utilizar um resistor de 150 Ω, porém esse não é um valor comercial, ou seja, não é um resistor fácil de encontrar, por isso utilizaremos um de 220 Ω.

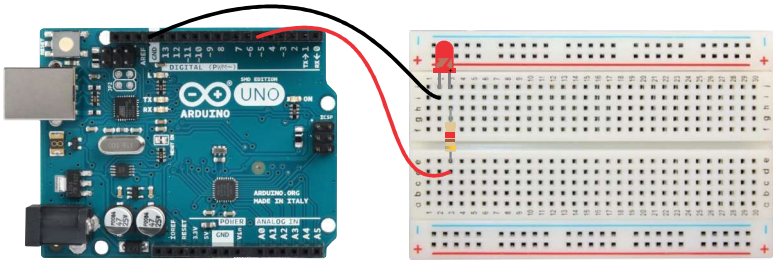
Acima trouxemos o diagrama com as ligações feitas de forma direta, com os fios soldados diretamente nos componentes, porém, é sempre indicado o uso de *protoboards*, que dispensa o uso de solda e mantém a característica de reaproveitamento dos componentes. Os dois primeiros experimentos que apresentaremos, mostrará também

3 Informações disponíveis em www.asdasda.com.br.

... montagem

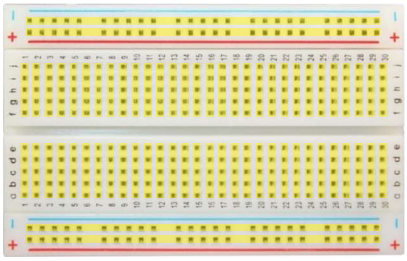
uma versão da montagem feita com o uso de *protoboard* e *jumppers*.

A seguir trazemos o mesmo *sketch* com o uso destes recursos.



Fonte: Produzida por Silas J. P. Silva 2022

A *protoboard* é um recurso muito importante na prototipação (criação de modelos funcionais), pois proporciona a ligação de componentes de maneira rápida e fácil, dispensando o uso de solda. Para ligar componentes entre si, basta introduzi-los nos furos que estão internamente interligados, como mostra a figura abaixo:



Fonte: Produzida por Silas J. P. Silva 2022

O destaque em amarela na figura acima, mostra como estão ligados os furos internamente. Para ligar um componente a outro, basta garantir que seus terminais estejam em furos interligados. Com o uso de *jummpers* ou fios comuns, a quantidade de furos pode ser expandido para outra barra de furos ou até para outra *protoboard*.

Código-fonte para o Arduino

```
1  void setup() {              // inicializa a função setup()
2    pinMode(5, OUTPUT);       // define a porta 5 como saída digital
3  }                           // fim da função setup()
4                               // linha em branco não tem função
5  void loop() {               // inicio da função loop()
6    digitalWrite(5, HIGH);    // Liga o LED com HIGH (voltagem = 5v)
7    delay(1000);              // Aguarda 1000 milissegundos (1 segundo)
8    digitalWrite(5, LOW);     // Desliga o LED com LOW (voltagem) = Ov)
9    delay(1000);              // Aguarda 1000 milissegundos (1 segundo)
10 }                           // fim da função loop()
```

OBS. do código. Os números em vermelho no início de cada linha são usados apenas para referenciar a linha e não fazem parte do código em si.

Comentário do código

Antes de falar sobre as funções de cada linha, é importante observar que as linhas do código apresentado trazem comentários precedidos por '//' (barra barra). Sempre que duas barras aparecem no código do Arduino, significa que o texto que vem logo depois será ignorado completamente. Então, estas linhas têm a função apenas de comentar, instruir, fazer uma observação. Essas linhas ou trechos de linhas, assim como as linhas em branco não serão consideradas quando o programa for enviado ao Arduino.

`1 void setup()` Inicia a função *setup()* com o comando *void*. Na programação do Arduino, pelo menos duas funções são obrigatórias: a função *setup()* e função *loop()*. Na função *setup()* são colocados os comandos que o Arduino deverá executar somente uma vez, quando for ligado e na função *loop()* os comandos que ele deve executar em *loop*, ou seja, eternamente ou até ser desligado. As funções *loop()* e *setup()* são delimitadas por "{}" (abre chaves no começo e fecha chaves no final), como nas linhas 1 e 3 para *setup()* e 5 e 10 para *loop()*.

`2 pinMode(5, INPUT);` Define o pino 5 Arduino como entrada. Isso porque as portas ou pinos que vão de 0 a 13 no Arduino, podem ser configuradas para receberem dados (INPUT) ou enviarem dados (OUTPUT). Neste caso vai ser enviado um sinal para a porta 05 onde está conectado o LED. Esse sinal de saída pode ser HIGH que corresponde a 5V ou LOW que corresponde a 0V.

`3 }` Finaliza a função setup().

`5 void loop()` Inicia a função *loop()*. Esta função, compreendida entre as linhas 5 e 10, delimitadas pelas chaves, executa em *loop* eterno (enquanto o Arduino estiver ligado) os comandos contidos nela.

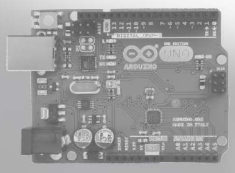
`6 digitalWrite(5, HIGH);` Envia uma "escrita" digital com valor HIGH para a porta 5, onde esta ligado o LED. Isso equivale a enviar 5V para essa porta e por consequência, para o LED, o que faz com que ele acenda.

`7 delay (1000);` Esta linha diz ao Arduino que aguarde 1 segundo antes de continuar. O comando "*delay*" determina uma parada, fazendo o microcontrolador congelar por um tempo determinado. Nenhuma outra tarefa é realizada até que transcorra o tempo especificado em miliseegundos, neste caso (1000).

`8 digitalWrite(5, LOW);` Envia uma 'escrita' digital com valor LOW para a porta 5, onde esta ligado o LED. Isso equivale a enviar OV para essa porta o que faz com que o LED apague.

`9 delay (1000);` Aguarda um segundo novamente.

`10 }` Delimita o fim da função loop(). Isso não significa que o



... cometário do código

programa se encerra, ao contrário, neste ponto, o fluxo (andamento do programa) é novamente enviado à linha 5 onde inicia o loop() novamente. Toda sequência a partir desta linha será repetida para sempre enquanto o Arduino permanecer ligado.

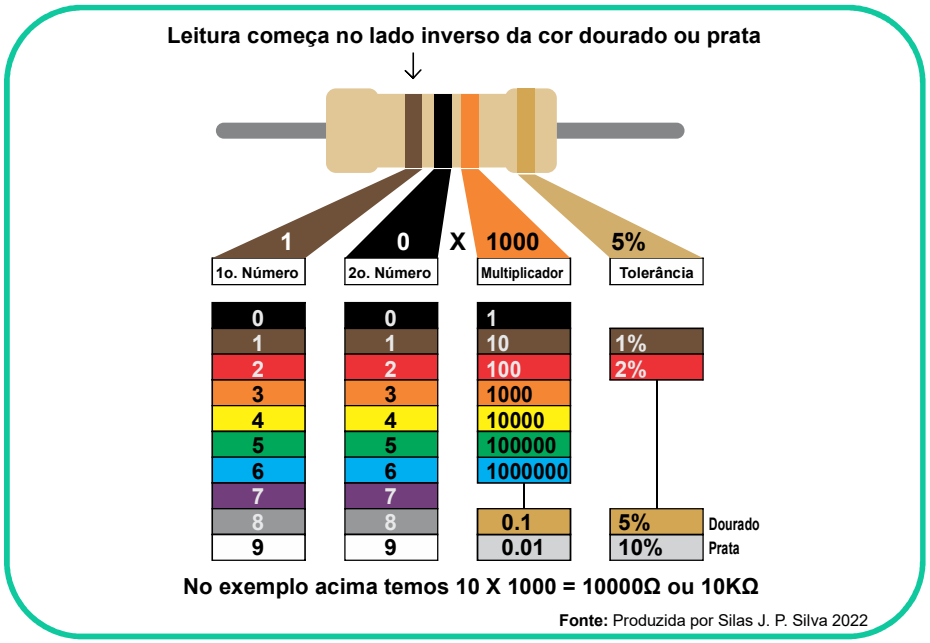
Desdobramentos

O simples piscar de um LED pode suscitar várias outras experiências. Acrescentando outros LEDs de cores diferentes, pode-se, por exemplo, simular o funcionamento de um semáforo, com tempos definidos para cada etapa, sabendo que para cada cor de LED torna-se necessário calcular o valor dos resistores, embora, valores como 220Ω ou 330Ω funcionem normalmente para LEDs em geral.

Na programação em geral, existe um termo usado para designar o programa mais básico possível num determinado contexto, este termo é "Hello World", ou, "Olá! Mundo". Foi precisamente isto que apresentamos neste primeiro contato com o Arduino: uma visão geral e introdutória para "quebrar o gelo".

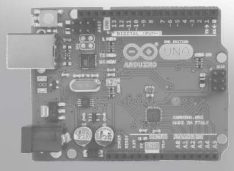
Para consulta

Tabela de cores para valores de resistores



Fonte: Produzida por Silas J. P. Silva 2022

Algumas vezes você pode encontrar a seguinte dica: coloque os dois primeiros números e o terceiro à quantidade de zeros no final. Funciona como uma boa forma de fixar os valores da tabela. ■



2

Sistema de luz e cores

Ensino de física

Neste experimento o professor ou o aluno podem trabalhar variando a intensidade da luz atuando nos potenciômetros para fazer a "mistura" de luzes para a obtenção de uma cor específica.

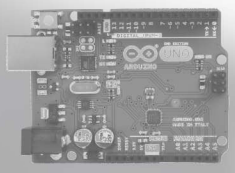
Na tela do computador é mostrado o valor para cada cor que os LEDs estão recebendo e estes valores variam de 0 a 255 para cada cor.

Lista de materiais e ferramentas

02 - LEDs RGB catodo comum
02 - Resistores de 100Ω
03 - Potenciômetros de 10KΩ
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

Observações

- Podem ser usados potenciômetros de valores diferentes dos empregados aqui, deste que sejam de valores iguais entre si.



LEDs RGB (Red-vermelho, Green-verde e Blue-azul) são LEDSs que possuem as três cores (luzes) primárias num mesmo LED. A intensidade de cada luz pode ser controlada separadamente por um potenciômetro, para gerar valores entre 0(zero) e 16.581.375 (mais de dezesseis milhões) de cores diferentes.

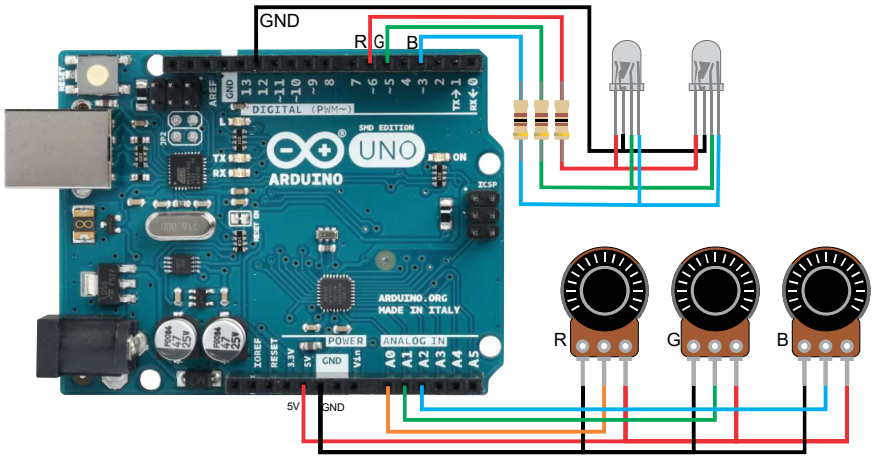
Componentes principais

LED RGB

Potenciômetro 50 KΩ

Resistor de 100 Ω

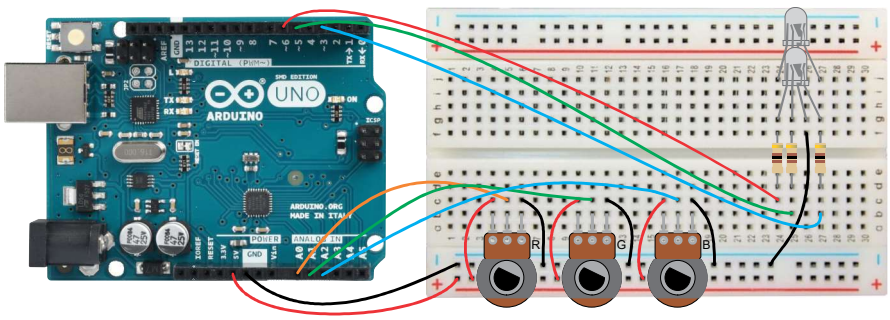
Diagrama e montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Este será nosso último experimento cujo diagrama será exibido também com a utilização da *protoboard*. Como mencionado anteriormente, os diagramas serão trazidos fora de escala, pois a maioria deles não permite enquadrar, em uma única imagem, a montagem de todos os componentes e ainda exibir o Arduino e *protoboard* mantendo a escala.

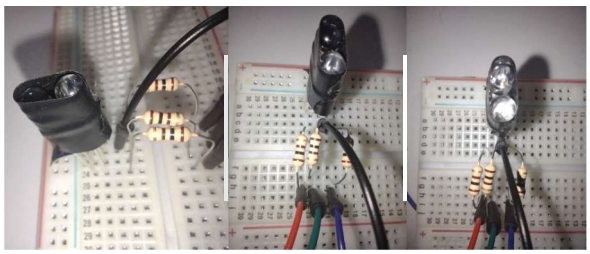
Segue-se o mesmo diagrama utilizando a protoboard:



Fonte: Produzida por Silas J. P. Silva 2022

Montagem

Os LEDs então ligados em paralelo, ou seja, como se fossem um só e o que acontece a um, acontece a todos. Para essa montagem pode ser usado apenas um único LED RGB e o resultado será praticamente o mesmo. Optamos aqui por dois LEDs apenas para intensificar a luz e caso optem por usar dois LEDs, é indicado que eles sejam colocados bem próximos, de preferência amarrados juntos com fita isolante, como mostrado abaixo:



Fonte: Produzida por Silas J. P. Silva 2022

Código-fonte para o Arduino

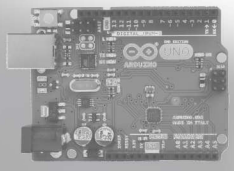
```
1  void setup(){                    // inicializa a função setup()
2    Serial.begin(9600);            // liga a comunicação com o computador
3    pinMode(A0, INPUT);            // define os pinos que serão usado e
4    pinMode(A1, INPUT);            // sua forma de funcionamento
5    pinMode(A1, INPUT);
6    pinMode(3, OUTPUT);
7    pinMode(5, OUTPUT);
8    pinMode(6, OUTPUT);
9  }
10 void loop(){                     // linha em branco loop()
11   int pt1 = analogRead(A0);
12   int pt2 = analogRead(A1);
13   int pt3 = analogRead(A2);
14   analogWrite(6, pt1/4);
15   analogWrite(5, pt2/4);
16   analogWrite(3, pt3/4);
17   Serial.print("R =");
18   Serial.println(pt1/4);
19   Serial.print("G =");
20   Serial.println(pt2/4);
21   Serial.print("B =");
22   Serial.println(pt3/4);
23   delay(30);
24 }                                // finaliza a função setup()
```

Comentário do código

As linhas 1 e 9 iniciam e finalizam, respectivamente, a função *setup*(), que tem a tarefa de fazer as configurações iniciais do Arduino.

2 `Serial.begin(9600);` Inicia a comunicação serial com o computador. Essa função é necessária, pois precisamos mostrar as informações com os valores recebidos dos potenciômetros, correspondentes a cada cor que será enviada aos LEDs.

3 a 8 – As linhas com os `pinMode` - Estas são as linhas que preparam cada



... cometário do código

porta ou "pino" do Arduino para receber ou enviar dados: as portas A0, A1 e A2 como entradas (receberão dados) e 3, 5, 6 com saídas (enviarão dados) . Logo depois, temos a linha 9 que finaliza a função setup().

`10 void loop(){` – Inicia a função loop(), que roda repetidamente enquanto o Arduino estiver ligado.

`11, 12 e 13` – Criam as variáveis do tipo "int" (número inteiro) que vão armazenar temporariamente os valores lidos em cada potenciômetro: "pt1" guarda o valor do potenciômetro destinado à cor "R" (vermelho), "pt2" para o "G" (verde) e "pt2" para o "B" (azul).

`14, 15 e 16` – Enviam os valores recebidos dos potenciômetros para cada terminal dos LEDs RGB. Esses valores são divididos por 4 porque a leitura dos potenciômetros retornam valores entre 0 e 1023 e, os valores enviados aos terminais dos LEDs precisam estar entre 0 e 255, o que dá uma razão de 4 para 1. Por exemplo, quando a leitura do potenciômetro estiver próximo a 512, representando metade do valor, o Arduino estará enviando 128 ao terminal do LED o que também representa metade do valor a ser enviado.

`17 a 22` – As linhas deste bloco exibem no monitor serial, na tela do computador, os valores enviados aos LEDs. Esses valores serão usados pelo professor como subsídios de para discussão com seus alunos, pois são respectivos à mistura de luz que acontecerá no LED. O resultado destas linhas será algo como:

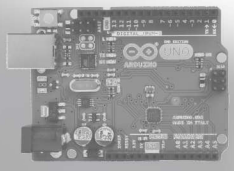
```
R = 130
G = 85
B = 101
```

`23 delay(30);` Aguarda 30 milissegundos antes de realizar novo *loop*.

Desdobramentos

O professor pode expandir este *sketch*, aumentando a quantidade de LEDs, para a produção de painéis luminosos, como o da figura abaixo, que também estão amplamente disponíveis para aquisição.

Fonte: Adaptada de https://1.bp.blogspot.com/-_BEArM8Nw1k/Vvhf3t-sbJI/AAAAAAAAE50/pML2dJs2LgwLin99-BBQw1F66wunnCL0g/s1600/Neopixel-Matriz-led-ws2812.jpg, por Silas J. P. Silva 2022



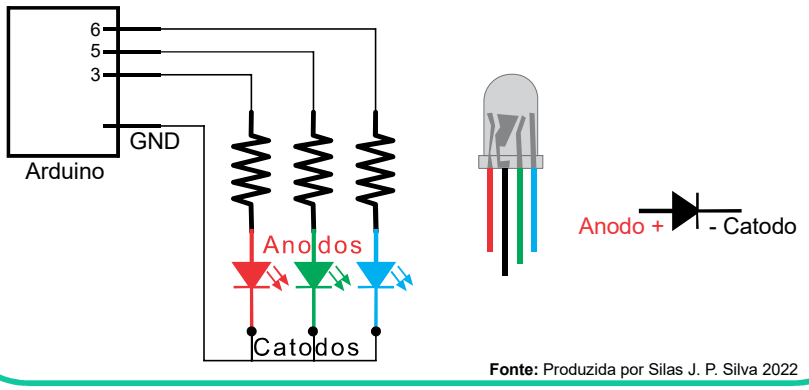
... desdobramentos

Este experimento traz o princípio de funcionamento das telas dos diversos dispositivos tecnológicos utilizados por todos atualmente, o que pode ensejar vários experimentos, por se tratar de uma tecnologia presente na vida de alunos dos vários níveis de ensino.

Para consulta

Os LEDs comuns possuem um catodo (terminal negativo) e um anodo (terminal positivo), como já visto anteriormente. No caso dos LEDs RGB, temos três terminais independentes que representam as cores e um terminal comum para os três. Portanto, os LEDs RGB podem do tipo "catodo comum" quando os positivos são independentes e os negativos ligados juntos, ou "anodo comum" quando os negativos são independentes e os positivos são juntos. No nosso caso usamos um LED RGB de catodo comum. Abaixo o funcionamento deste tipo LED.

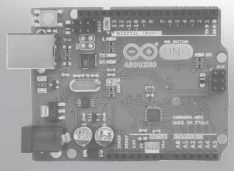
Funcionamento LED RGB - Catodo comum



Fonte: Produzida por Silas J. P. Silva 2022

Existem vários tipos e combinações de LEDs RGB. Abaixo algumas das formas em que eles podem ser encontrados.

Fonte: Adaptada de https://blog.baudaeletronica.com.br/leds-rgb/ por Silas J. P. Silva 2022



3

Velocidade do Som

Ensino de física

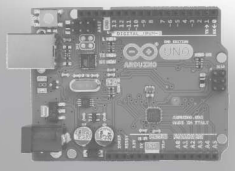
O objetivo é que o aluno possa jogar uma bola entre os dois sensores possibilitado que o Arduino meça o tempo decorrido entre as passagens entre eles e assim poder calcular a velocidade média ou a distância. A leitura do tempo é mostrada na tela de um notebook ligado ao Arduino.

Lista de materiais e ferramentas

01 - Módulo Ultrassônico HC-SR04
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

Observações

Os módulos de sensores ultrassônicos não têm grande precisão, mas proporcionam boa usabilidade em vários projetos. Para melhor funcionamento o objeto deve se colocado perpendicularmente em frente ao sensor e preferencialmente a distâncias entre 3 e 300 centímetros.

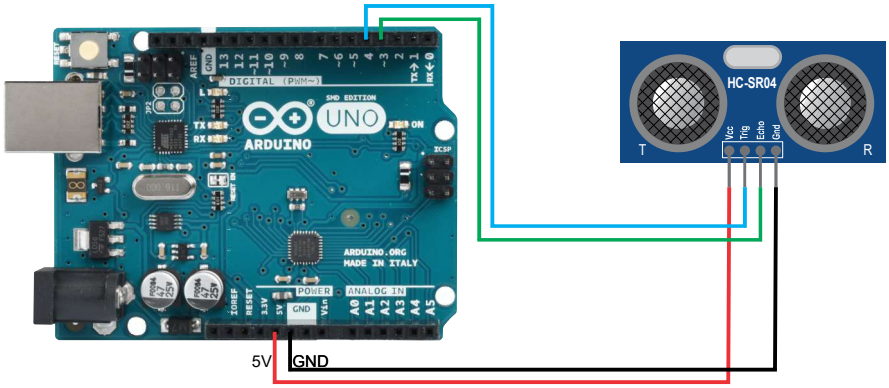


Os sensores ultrassônicos funcionam enviando um sinal sonoro que ao se chocar com um objeto, retorna ao sensor permitindo calcular a distância entre o sensor e o objeto.

Componentes principais

Sensor Ultrassônico HC-SR04

Diagrama de montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Código-fonte para o Arduino

```
1  unsigned int duracao = 0;
2  unsigned int distancia = 0;
3  void setup(){
4    Serial.begin(9600);
5    pinMode(5, INPUT);  //echo
6    pinMode(6, OUTPUT); //trig
7  }
8  void loop(){
9  digitalWrite(5, HIGH);
10 delayMicroseconds(10);
11 digitalWrite(6, LOW);
12 duracao = pulseIn(5, HIGH);
13 distancia = duracao * 0.017;
14 Serial.print("Distância =");
15 Serial.println(distancia);
16 }
```

Comentário do código

`1 unsigned int duracao = 0;` e `2 unsigned int distancia = 0;` - Criamos variáveis para dois dados a serem calculados, que receberam os nomes “duracao” e “distancia” e que terão como valores números inteiros, por isso o uso de “unsigned int”. Seu valor inicial é zero.

... cometário do código

3 `void setup() {` – Inicia a função de preparação *setup()*.

4 `pinMode(5, INPUT);` – Define o pino 5 do Arduino como entrada. Este pino vai receber valores enviados pelo pino "Echo" do sensor, que receberá o retorno do som.

5 `pinMode(6, OUTPUT);` – Define o pino 6 do Arduino como saída. Este pino vai enviar sinal para o pino Trig do sensor para que ele dispare o som.

6 `Serial.begin(9600);` – Inicia a função que estabelece a comunicação com o computador, habilitando tanto o envio quanto a recepção de informação entre eles. Neste sketch, as informações de distância vão ser mostrados na tela do comutador, por isso este comando é necessário.

7 `}` – Finaliza a função setup().

8 `void loop() {` – Inicia a função *loop()*.

9 `digitalWrite(6, HIGH);`

10 `delayMicroseconds(10);`

11 `digitalWrite(6, LOW);` – O conjunto dessas três linhas (9, 10 e 11), pretende: na linha 9, mandar ligar o envio de ultrassom pelo sensor; na linha 10, aguardar 10 microssegundos, equivalendo a 1/1.000.000 segundos, ou seja, 1 segundo dividido por um milhão; e, por último, na linha 11, desligar o envio de ultrassom no sensor. O que acontece resumidamente nestas três linhas, é que o Arduino comanda o ultrassom a ficar pulsando um som inaudível de tempos em tempos.

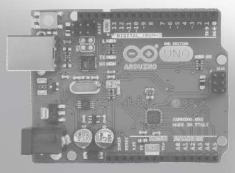
12 `duracao = pulseIn(5, HIGH);` – Aqui a variável "duracao" recebe um valor em microssegundos, que representa o tempo entre o momento do envio do sinal pelo sensor e a recepção do retorno deste sinal, caso o ultrassom enviado tenha encontrado um obstáculo em que refletiu.

13 `distancia = duracao * 0.017;` – Atribui à variável “distancia” o cálculo resultante da variável “duração” multiplicado por 0.017.

14 `Serial.println(distancia);` – Envia ao computador o valor resultante armazenado na variável “distancia”.

15 `delay(100);` – Aguarda 100 milessegundos, ou seja, permanece parado por um décimo de segundo antes de seguir adiante.

16 `}` - Marca o fim da função *loop()*. Ao chegar aqui, *loop()* é reinicializada e começa tudo novamente.

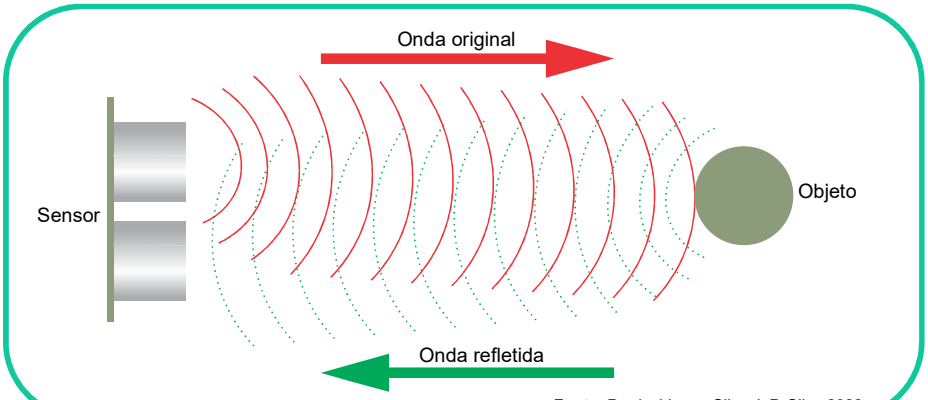


Desdobramentos

O funcionamento do sensor ultrassônico pode ensejar vários desdobramentos. Toma-se como ponto de partida o fato de o som viajar, se chocar com um objeto qualquer e retornar para o sensor. Baseado neste funcionamento pode-se, por exemplo, usar como sensor de passagem (de objetos), pois quando nada fica diante do sensor ele registra distâncias acima de 3 m. Assim, quando algo passa na frente dele, registra-se a passagem, independentemente da distância. Pose ser usado, por exemplo, para verificar: se uma porta está aberta ou fechada, se alguém passou por um determinado local, se um objeto está presente ou não, etc.
Devemos ter em mente que este dispositivo trata-se de um sonar.

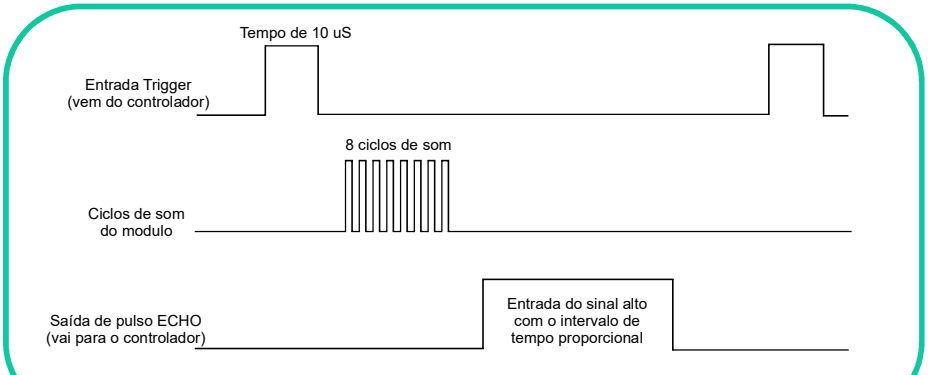
Para consulta

Princípio de funcionamento sensor ultrassônico



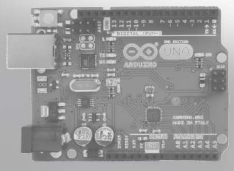
Fonte: Produzida por Silas J. P. Silva 2022

Detalhe técnico do funcionamento do sensor ultrassônico



Fonte: Adaptada de https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf por Silas J. P. Silva 2022

Um artigo detalhado com o funcionamento do sensor ultrassônico usado neste experimento pode ser encontrado em https://www.filipeflop.com/blog/medindo-distancia-com-frdm-kl05z-e-sensor-hc-sr04/



4

Cálculo de velocidade

Ensino de física

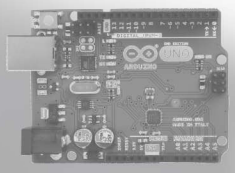
Bem intuitivo, este projeto pode servir para vários experimentos. No exemplo trazido aqui, o aluno poderá jogar uma bola entre dois laseres/sensores e o Arduino registra o tempo decorrido entre estes dois momentos, permitindo ao professor trabalhar os conceitos de cálculo de velocidade.

Lista de materiais e ferramentas

01 - Módulo Diodo Laser Ky-008 5v, 5mW
01 - Módulo receptor laser Arduino
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

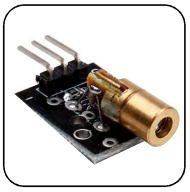
Observações

1 - Os laseres fabricados para funcionarem com Arduino e similares, têm uma potência bem baixa. Porém, por precaução, em hipótese alguma, deve ser apontado diretamente para o olho humano ou de animais, ou apontar por período prolongado para a pele ou outras partes do corpo.

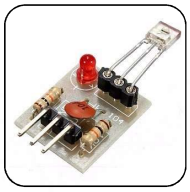


Este experimento pode tornar o aprendizado sobre velocidade média bem divertido, pois o aluno poderá comparar os resultados dos cálculos com sua experiência em sala de aula.

Componentes principais

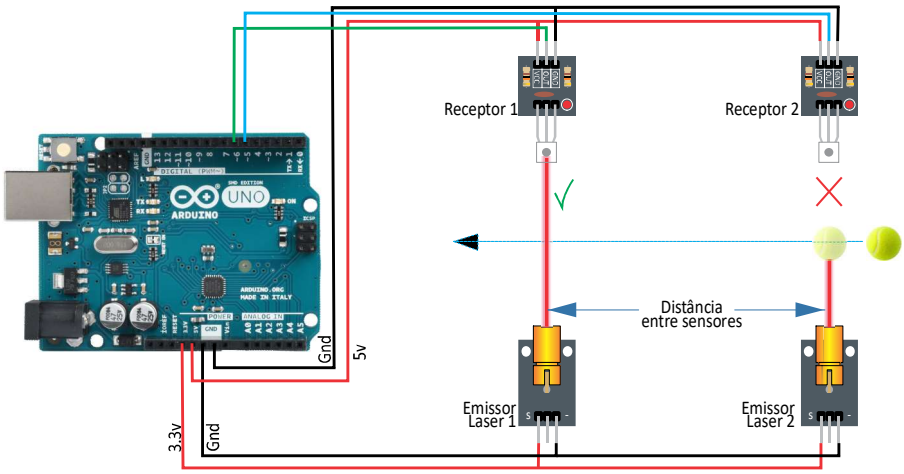


Módulo laser - emite um laser a algumas dezenas de metros



Módulo receptor laser - registra a incidência ou não do laser

Diagrama e montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Montagem.

O emissor e receptor laser, devem ficar perfeitamente alinhados e a fiação que liga os dispositivos ao Arduino devem ficar fora do caminho onde a bola deve correr e esta pista (caminho), pode ser deste uma mesa até uma quadra de futebol.

Com o aumento da distância entre o emissor e o receptor, pode ficar difícil fazer o apontamento do laser. Sendo assim, é recomendado que primeiro o emissor laser seja ligado na direção desejada e depois seja posicionado o receptor.

Código-fonte para o Arduino

```
unsigned long tempo_decorrido = 0;
unsigned long tempo_tomado = 0;
bool inicia = true;
void setup()
{
  pinMode(6, INPUT);
  pinMode(5, INPUT);
  Serial.begin(9600);
}
```

... código-fonte para o Arduino

```
10 void loop()
11 {
12  if ((digitalRead(5) == LOW) and (inicia == true))
13   {
14     tempo_tomado = millis();
15     inicia =  false;
16   }
17  if ((digitalRead(6) == LOW) and (inicia == false))
18   {
19     tempo_decorrido = millis() - tempo_tomado;
20     Serial.print("Tempo.: ");
21     Serial.print(tempo_decorrido);
22     Serial.println(" Milessegundos");
23     inicia = true;
24   }
25 }
```

Cometário do código

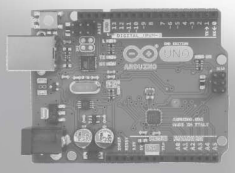
1 e 2 `unsigned long tempo_decorrido = 0;` e `unsigned long tempo_tomado = 0;`

Criam variáveis de números longos. Essas variáveis se assemelham ao 'x' ou 'y' das equações matemáticas, ex.: (x = y + 2). Eles possuem determinados valores que podem ou não mudar ao longo do tempo. Em resumo, variáveis armazenam esses valores para serem usados em algum momento. No caso das linhas 1 e 2 criam varáveis do tipo número longo ou *unsigned long*, chamadas "tempo_tomado" que armazenará o tempo inicial para o cálculo, e "tempo_decorrido" que armazenará o tempo decorrido entre o momento de acionamento de primeiro sensor e o segundo. Elas começam com valor 0 (zero), pois no início do programa não possuem valores prévios.

3 - `bool inicia = true;` Cria uma variável chamada inicia, do tipo lógica (booleana), capaz de armazenar apenas os valores *true* (verdadeiro) ou *false* (falso). Muito utilizadas em programação esse tipo de dado serve por exemplo fazer comparações entre valores. Dizemos por exemplo que 5 == 5 (cinco é igual a cinco) tem valor *true* e 5 < 3 (cinco menor que três) tem valor *false*. No caso a variável 'inicia' começa com valor *true.*

4 `void setup()` Inicia a função *setup()* com o comando *void*. Na programação do Arduino, pelo menos duas função são obrigatórias: a função *setup()* e função *loop()*. Na função *setup()* são colocados os comando que o Arduino deverá executar somente uma vez quando for ligado e na função *loop()* os comandos que ele deve executar em *loop*, ou seja, eternamente ou até ser desligado. As funções *loop()* e *setup()* são delimitadas por "{}" (abre chaves no começo e fecha chaves no final, como nas linhas 5 e 9 para *setup()* e 11 e 25 para *loop().*

6 e 7 `pinMode(6, INPUT);` e `pinMode(5, INPUT);` Define os pino 6 e 7



... cometário do código

do Arduino como entrada. Isso porque as portas ou pinos que vão de 0 a 13 no Arduino, podem ser configuradas para receberam ou enviarem dados. Neste caso vão receber os dados do sensores de recepção laser de acordo com a incidência ou não do laser sobre eles.

`8 Serial.begin(9600);` Esta linha habilita a comunicação necessário para que o Arduino possa mandar informações ao computador e também possa receber dados oriundos do computador. Neste projeto, o tempo decorrido entre os sensores será mostrado na tela do computador.

`10 void loop()` Inicia a função *loop()*. Esta função, compreendida entre as linhas 9 e 24 definidas pelas chaves, executa em l*oop* eterno (enquanto o Arduino estiver ligado) os comandos contidos nela.

`12 if ((digitalRead(5) == LOW) and (inicia == true))` verifica o sensor ligado ao pino 5. Este sensor está organizado para ser o primeiro no caminho que o objeto (bola) deve seguir. Este programa não funciona se esta ordem for invertida. Por padrão, enquanto o laser estiver atingido o sensor, o valor será *HIGH*. Quando a luz é interrompida, o valor muda pra *LOW*. Somente quando o pino 5 for *LOW* e o valor de 'inicia' for *true* ele executará o que estão nas linhas 14 e 15.

`14 tempo_tomado = millis();` - Esta linha guarda na variável 'tempo_tomado' o tempo atual do Arduino que é baseado em um relógio interno e é dado em milissegundos.

`15 inicia = false;` Guarda o valor *false* na variável *inicia*. É necessário para controle do fluxo. Desta forma o que está posto no bloco iniciado na linha 12 não será executado até que o bloco compreendido entra as linhas 17 e 24 seja executado.

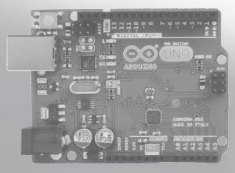
`17 if ((digitalRead(6) == LOW) and (inicia == false))` Verifica se o segundo sensor ligado ao pino 6 teve sua luz interrompida. Caso isso tenha ocorrido e valor de "inicia"' for *false* ele executa os comandos entre as linhas 19 e 23.

`19 tempo_decorrido = millis() - tempo_tomado;` guarda o na variável "tempo_decorrido" o valor do tempo atual que está sempre em *millis()*, subtraído do tempo tomado anteriormente, que havia sido guardado na variável "tempo_tomado". Com isso o valor de "tempo_decorrido" será justamente o tempo que o objeto (bola) percorreu entre o primeiro e o segundo sensor.

`20 Serial.print("Tempo.: ");,`

`21 Serial.print(tempo_decorrido);,`

`22 Serial.println(" Milessegundos");` Estas três linhas envia o tempo decorrido para o computador no formato "Tempo.: xxxx



... cometário do código

Milessegundos".

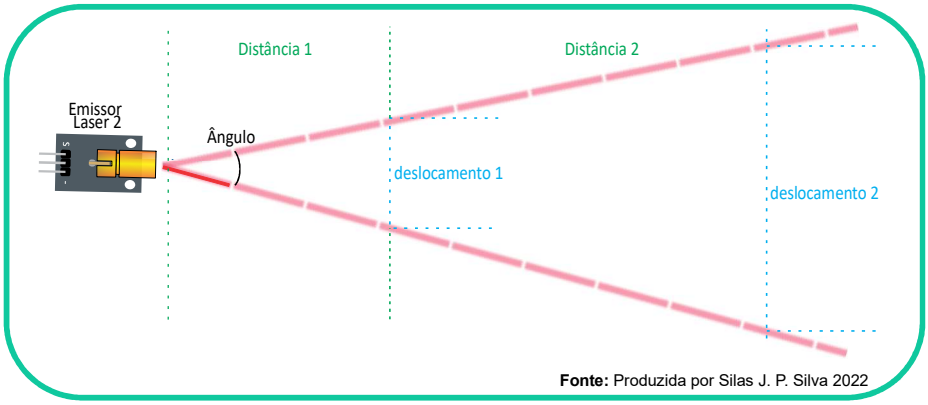
23 `inicia = true;` Torna a variável 'inicia' novamente como *true* para que seja feita uma nova verificação se um objeto interromper a luz do primeiro sensor.

25 `{` Esta chave da linha 25 demarca o fim do bloco de linhas pertencentes à função *loop()*. Neste ponto o Arduino retorna à linha 10 e recomeça a partir dali. Isto acontecerá até que o Arduino seja desligado.

Desdobramentos

Um desdobramento possível para esse experimento, pode ser o acréscimo de mais sensores para calcular a aceleração/desaceleração. Para isso deve ser usado um programa semelhante ao que aparece no experimento 5.
A escala também pode ser alterada, podendo funcionar desde em cima de uma mesa até em uma quadra de esportes, onde os próprios alunos podem correr entre os sensores para produzir os dados.

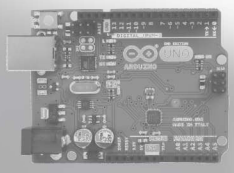
O laser também pode ser usado para experimentos que envolvam medida angular, agindo como uma grande reta em que o professor pode sugerir aos alunos que realizem medições no deslocamento do feixe próximo ao emissor e medições do mesmo deslocamento a uma distância maior, com ajuda de transferidor e fitas métricas, como sugere a imagem abaixo.



Fonte: Produzida por Silas J. P. Silva 2022

Para consulta

Detalhe spbre o funcionamento e montagem podem ser encontrados em: https://blogmasterwalkershop.com.br/arduino/arduino-utilizando-o-modulo-receptor-de-laser. Ou você pode consultar a oágina oficial do Arduino que traz vários exemplos do uso de laser, em: https://create.arduino.cc/projecthub/projects/tags/lasers



5

Plano inclinado

Ensino de física

O objetivo é passar uma bola-de-gude pelo cano inclinado, e a media em que ela passa pelos sensores, o tempo vai sendo registrado, e enviado à tela do computador ligado ao Arduino, permitindo assim o cálculo da aceleração.

Lista de Materiais e ferramentas

- 05 LEDs infravermelhos
- 05 foto-transistores Til78
- 05 resistores 10kΩ
- 05 resistores 100Ω
- Cola quente para fixação
- fios para ligação
- 1m de cano PVC de 1"
- um bola-de-gude (ou similar)

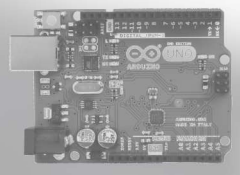
* Arduino UNO

Observações

1 - O plano inclinado é essencialmente o que o nome diz e pode ser usado como tal ou pode ser colocado totalmente perpendicular ao solo e se fazer a tomada de tempos para uma queda livre, visto que o atrito da bola-de-gude com as paredes do cano devem mínimas.

2 - Os fototransistores podem ser de 3mm e 5mm. Ao furar o cano para fixá-los basta usar uma broca da medida exata do fototransistor e dos LED, que quase nenhuma quente será necessária.

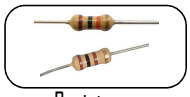
3 - Lembre-se de que os LED são polarizados, têm um lado negativo e um positivo



O plano inclinado é utilizado para experimentos envolvendo aceleração. Neste experimento utilizaremos LEDs infravermelhos e foto-receptores para registrar o movimento de um corpo e assim poder medir sua aceleração.

Componentes principais

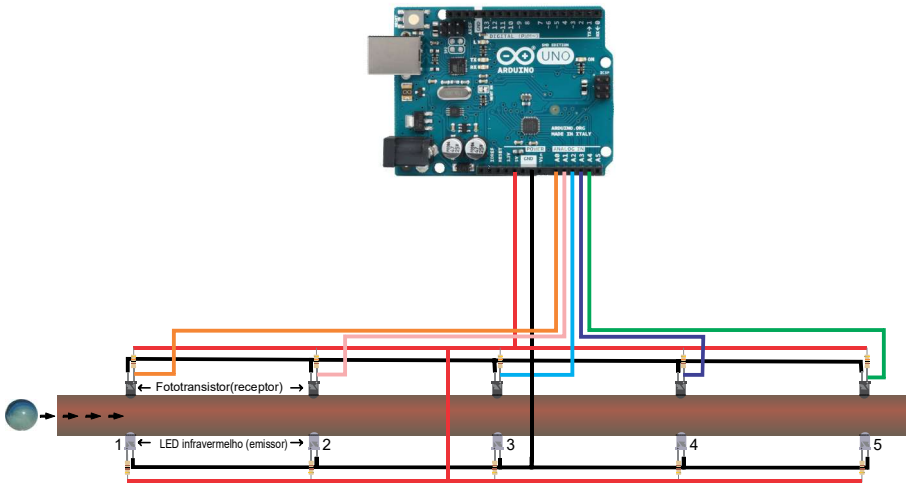
LED infra-vermelho:
(emissor)



Resistores:
10 KΩ e 100Ω

Foto-transistor:
(receptor)

Diagrama de montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Montagem.

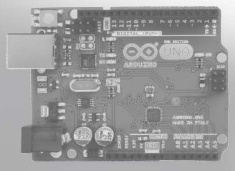
O cano PVC deve ser perfurado de forma que o LED infravermelho e o fototransistor sejam fixos apontados diretamente um para o outro, gerando um sinal baixo no receptor, por padrão, e um sinal alto quando a bola-de-gude passar entres eles, assim o Arduino pode registrar essa interrupção na luz e fazer a tomada de tempo. O ideal é que os leds e fototransistor não fiquem muito profundos para não dificultar a passagem da esfera.

Os fios de ligação podem ser distribuídos livremente pela superfície do cano, visto que a leitura vai acontecer no interior do mesmo.

A distância entre os furos, para esse experimento, devem ser idênticas e fica a critério de cada um. Alguns experimentos podem exigir furos em distâncias progressivas ou não uniformes, e isso pode ser feito conforme a necessidade de cada um. O importante é ter em mente que o Arduino sempre vai mostrar no computador o tempo que a bola-de-gude leva para percorres a distância entre os sensores.

Código-fonte para o Arduino

```
1  // cria as varáveis que registrarão cada tempo
2  // e também a variável tc para registrar sempre o tempo atual
3  unsigned long ta, t1, t2, t3, t4;
4
5  void setup()
6  {
7    // as 4 linhas abaixo definem os pinos A0 a A4 como
8    // entradas analógicas
9    pinMode(A0, INPUT);
10   pinMode(A1, INPUT);
11   pinMode(A2, INPUT);
12   pinMode(A3, INPUT);
13   pinMode(A4, INPUT);
14   // incia a comunicação Serial que permite ao
15   // arduino enviar dados ao computador
16   Serial.begin(9600);
17 }
18
19 void loop() // inicia a função loop()
20 {
21   while (analogRead(A0) < 100){
22         // aguarda aqui até que o 1o. sensor ser acionado
23   }
24   ta = millis(); // registra o tempo atual
25
26   while (analogRead(A1) < 100){
27         // aguarda aqui até que o 2o. sensor seja acionado
28   }
29   t1 = millis() - ta ;  // registra o 1o. tempo decorrido
30   ta = millis();        // registra o tempo atual novamente
31
32   while (analogRead(A2) < 100){
33         // aguarda aqui até que o 3o. sensor seja acionado
34   }
35     t2 = millis() - ta ; // registra o 2o. tempo decorrido
36     ta = millis();       // registra o tempo atual novamente
37
38   while (analogRead(A3) < 100){
39         // aguarda aqui até que o 4o. sensor seja acionado
40   }
41     t3 = millis() - ta; // registra o 3o. tempo decorrido
42     ta = millis();      // registra o tempo atual novamente
43
44   while (analogRead(A4) < 100){
45         // aguarda aqui até que o 5o. sensor seja acionado
46   }
47     t4 = millis() - ta; // registra o 4o. tempo decorrido
48
49   // mostra no terminal serial o tempo 1
50   Serial.print("Tempo 1: ");
51   Serial.print(t1);
52   Serial.println(" ms");
53   // mostra no terminal serial o tempo 2
54   Serial.print("Tempo 2: ");
55   Serial.print(t2);
56   Serial.println(" ms");
57   // mostra no terminal serial o tempo 3
58   Serial.print("Tempo 3: ");
59   Serial.print(t3);
60   Serial.println(" ms");
61   // mostra no terminal serial o tempo 1
62   Serial.print("Tempo 4: ");
63   Serial.print(t4);
64   Serial.println(" ms");
65
66 } // retorna ao inicio e começa novo ciclo
67   // de tomada de tempo
```



Comentário do código

O código-fonte (programa) deste experimento, como visto, possui 67 (sessenta e sete) linhas e uma análise "linha a linha" ficaria muito extensa e cansativa ao leitor.

O código traz muitos comentários (começam com '//', duas barras) e essas linhas, ignoradas pelo Arduino, são utilizadas para descrever a função que cada linha deseja implementar e o texto a seguir funciona como complemento a eles. Linhas em branco também são ignoradas sendo colocadas no código apenas para melhorar o visual.

3 `unsigned long ta, t1, t2, t3, t4;` – Esta linha cria as variáveis do tipo ''numero inteiro longo'', ou seja, variáveis que vão receber os valores numéricos inteiros para armazenarem os tempos, sendo: "ta" para tempo atual e "t1" a "t4" para os tempos registrados pelos sensores 1 a 4.

5 a 17 – Função setup(), que como já visto, traz as configurações iniciais para o programa. Neste caso, as portas A0 a A4 como entradas analógicas que receberão os sensores e registrarão valores de 0 a 1023, dependendo da luz infra-vermelha incidindo sobre o foto-transistor. Quando mais claridade menor o valor recebido e vice-versa. Nesta função também temos a linha 16 que inicial a comunicação com o computador, para ele poder receber os dados enviados pelo Arduino.

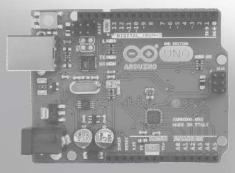
19 a 66 - Essa linhas compreendem a função loop(), que será executada em ciclos, enquanto o arduino permanece ligado.

21 `while (analogRead(A0) < 100) {` - Vamos nos concentrar especialmente nesta linha, pois ela se repete por mais quatro vezes no código, mudando apenas a porta de leitura, de A1 a A4. O que for dito sobre ela, vale para todas as outras.

Esta linha serve para que a Arduino fique aguardando (while) até que o valor da leitura no fototransistor seja maior que 100 (cem). Este valor no Arduino varia de 0 a 1023, dependendo da luz que incide sobre o fototransistor e corresponde à função analogRead(A0). Com a luz diretamente sobre ele, retornará um valor bem baixo e quando essa luz interrompida pela bola-de-gude, esse valor sobe rapidamente acima dos 100.

É importante observar que, para o experimento funcionar com este código deve comecar sempre pela leitura na porta A0, pela própria característica do programa que começa a leitura por ela. Portanto, o sensor (fototransistor) ligado a ela deve estar sempre voltado para cima.

Enquanto o valor de A0 não sobe acima de 100, o Arduino permanece parado nas linhas 21 a 23 e nada acontece. Assim que isso muda, o andamento (fluxo) do código continua à partir da linha 24.



... cometário do código

50 a 64 – Estas linhas se encarregam de enviar ao computador ligado ao Arduino, através do Monitor Serial os valores com os tempos tomados no experimento. Essas linhas resultam em algo como mostrado abaixo.

```
Tempo 1: 401 ms
Tempo 2: 380 ms
Tempo 3: 365 ms
Tempo 4: 340 ms
```

De posse destes valores na tela o professor poderá trabalhar com seus alunos, com os cálculos pertinentes à aceleração dos corpos, além de poder também alternar o ângulo de inclinação do cano, para obter valores diferentes.

66 } – Retorna a linha 19 e reinicia a função loop().

Desdobramentos

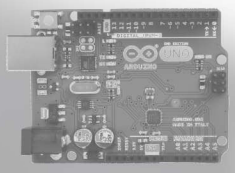
Este experimento, como o professor de física já pode ter notado, tem muito potencial.

Ao inclinar o cano, pode-se tomar leituras diferentes, visto que a aceleração vai mudar gradativamente até que o cano fique perpendicular ao solo. Outra coisa que pode ser feita, a mudança da posição dos sensores ao longo do cano. Para isso é indicado que seja feito, já de início, uma linha de furos equidistantes no cano, em que tanto o LED infravermelho quanto o fototransistor possam ser fixos em distâncias conforme a necessidade de cada experimento.

Como sugestão, foi utilizado um cano PVC, porém nada impede que este experimento seja montado sobre outras bases, como calhas, trilhos e até mesmo pêndulos, dado que o princípio de funcionamento é a tomada de tempo entre os sensores.

Para consulta

A maioria dos experimentos encontrados na internet usando estes componentes utilizam com a função de reflexão. A luz enviada pelo LED se choca com um objeto sendo captado de volta pelo fototransistor. Este uso pode ensejar outras montagens também interessantes. Para o uso aqui proposto pode-se encontrar um bom material em :https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-led-emissor-ir-e-fototransistor-ir



... cometário do código

`24 ta = millis();` - Esta linha registra o tempo em que ocorre a primeira leitura, guardando esse valor em na variável "ta" que havia sido criada anteriormente e agora tem o valor correspondente ao relógio interno do Arduino no exato momento em que a bola-de-gude interrompe a luz em A0. Sempre que o Arduino é ligado, um relógio interno começa uma contagem dos milissegundos transcorridos. Esse valor começa em 1 e vai sendo incrementado com o passar do tempo. A função millis() no Arduino será sempre igual a esse valor, portanto a variável "ta" terá o valor dos milissegundos naquele momento. Essa informação servirá para que no próximo sensor, transcorrido algum rempo, possa ser tomado outro valor e esse novo valor seja subtraído do valor de "ta" para se obter o valor dos milissegundos decorridos entre um sensor e o outro.

Vamos tomar como exemplo o caso hipotético em que ao ser acionado o primeiro sensor em A0, millis() tinha o valor de 52563. Esse valor é guardado em "ta". No próximo sensoro valor lido de millis() foi 52964. Para saber o tempo decorrido, subtraímos "ta" de millis() e guardamos esse valor em "t1". Para entendermos melhor, vamos dividir essa dinâmica em "momentos" distintos e acompanhar o que acontece em cada momento:

Momento 1 -> ta = millis(); // agora "ta" será igual a 52563 (primeiro tempo tomado)

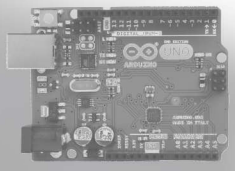
Momento 2 -> t1 = millis() – ta; // neste momento millis() será igual a 52964, porque millis() não para de aumentar e "t1" será igual 52964-52563, ou seja "t1" = 401, que é o tempo decorrido me milissegundos entre a leitura feita em A0 e A1. Esse processo se repete até o último sensor em A4. Vale ressaltar que antes de cada nova leitura o valor de "ta" é novamente igualado ao millis() atual, para então, servir de base para o cálculo da próxima leitura.

`26` a `28` `while (analogRead(A1) < 100) {` - Aguarda pela passagem do objeto na frente do sensor ligado a porta A1, para só então seguir a diante.

`29 t1 = millis() - ta ;` - Atribui a "t1" o valor do tempo atual menos o tempo arterior armazenado em "ta", o que equivale ao tempo decorrido entre A0 e A1.

`30 ta = millis();` - Atualiza o novo tempo atual em "ta", para que sirva de parâmetro para o próximo tempo a ser calculado.

`32` a `47` – Nesta linhas a dinâmica mostrada acima vai se repetir mais duas vezes. Ao final destas linhas teremos os valores de "t1" a "t4" definidos.



6

Roda de cores

Ensino de física

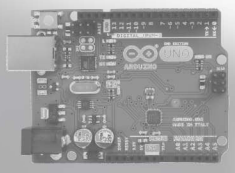
A mistura das cores nesta roda em uma determinada velocidade vai se tornar branco. O experimento permite controlar a velocidade até atingir o objetivo, tanto para a roda de cores como para roda estroboscópica.

Lista de materiais e ferramentas

01 - Módulo drive para motor AC L298N
01 - Potenciômetro de 50kΩ
01 - Motor 3V a 6V alta rotação
01 - Fonte de alimentação de 5v a 9v, indicada 7.5v
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

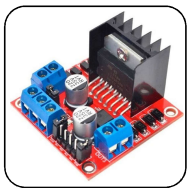
Observações

1 - Os efeitos para a roda cor têm melhor resultado se exposta a luz branca ou do sol, enquanto que para o efeito estroboscópico é necessária uma luz deste tipo ou no mínimo uma luz fluorescente.
2 - A velocidade atingida depende do motor utilizado, motores de baixa rotação podem não dar o efeito desejado.
3 - O módulo drive e a fonte externa são necessários porque o Arduino não conseguem alimentar o motor diretamente em suas portas.
4 - Para as duas rodas pode ser usado um CD ou DVD com uma adaptação no centro para fixar o eixo do motor



A roda de cores ou roda de Newton, é uma boa experiência visual para trabalhar conceitos como luz e cores. No final é apresentado um modelo de roda que pode ser usado com uma luz fluorecente para um efeito estroboscópico.

Componentes principais



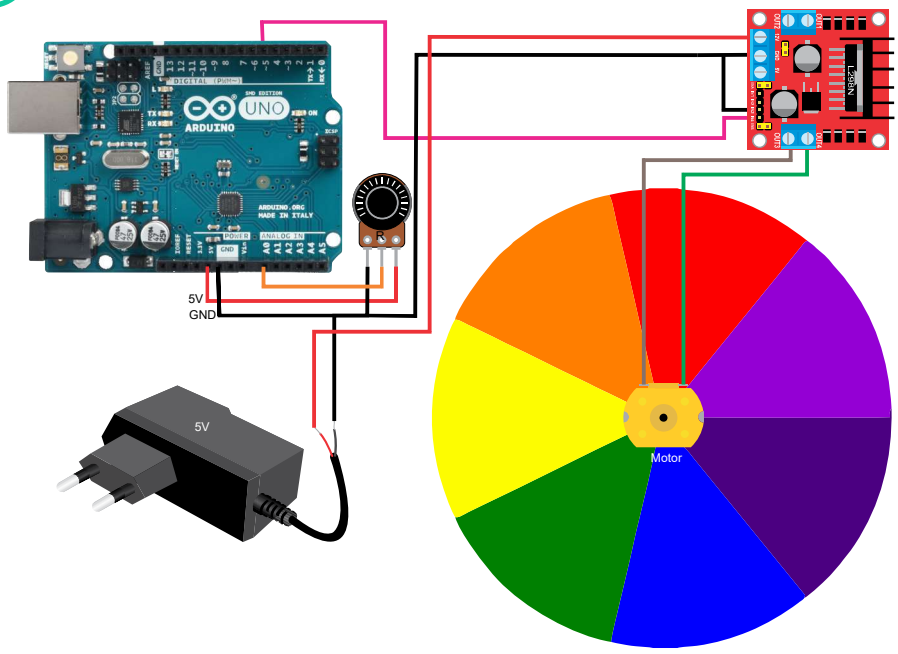
Módulo drive para motor AC L298N

Potenciômetro 50KΩ

Motor DC 3v-6v alta rotação

Fonte de alimentação de 5v a 9v.. Indicada 7.5v.

Diagrama de montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Montagem

Essa montagem é bem intuitiva. Inicialmente corta-se um disco de papelão onde vai ser colado uma folha impressa com o padrão de cores. É importante que o furo em que vai ser encaixado no eixo do motor esteja bem centralizado, pois, precisa alcançar altas velocidades e um furo deslocado do centro pode causar trepidações. O motor indicado aqui, pode ser substituído por qualquer motor que funcione entre 3v e 6v, preferencialmente reciclado.

6 Roda de cores

Código-fonte para o Arduino

```
1  void setup(){
2   Serial.begin(9600);
3   pinMode(5, OUTPUT);
4   pinMode(A0, INPUT);
5  }
6
7  void loop(){
8   analogWrite(5, analogRead(A0)/4);
9  }
```

Comentário do código

Apesar deste sketch possuir muitos componentes o funcionamento do código é bem simples.

1 `void setup() {` - Inicio da função setup() que configura os parâmetros inicias do Arduino.

2 `Serial.begin(9600);` - Inicia a comunicação do Arduino com o computador através do Monitor Serial via cabo USB.

3 `pinMode(5, OUTPUT);` - Declara que o pino 5 é do tipo saída. Este pino vai ser ligado ao controlador do motor para enviar-lhe a velocidade em que o motor deverá girar. Este, será um valor analógico entre 0 e 255.

4 `pinMode(A0, INPUT);` - Configura a porta analógica A0 como entrada. Isso significa que ela vai receber os valores de voltagens enviados pelo potenciômetro e convertê-las em valores numéricos que vão de 0 a 2013.

5 `}` - Finaliza a função setup().

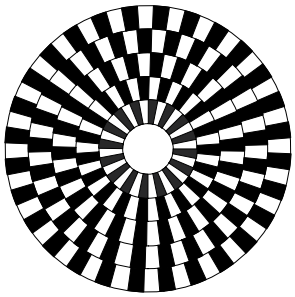
7 `void loop() {` - Inicio da função loop() que é executada constantemente.

8 `analogWrite(5, analogRead(A0) / 4);` - Faz uma escrita analógica na porta 5, com o valor da leitura da porta A0 já convertida e dividida por 4(quatro). Essa divisão é necessária pois, os valores analógicos de entrada têm 1024 graduações, enquanto os valores analógicos de saída têm apenas 256. O ideal é fazer uma regra de três para a conversão, mas como 1024 é múltiplo de 256, basta fazer uma divisão simples por 4. (quatro). Assim, por exemplo, quando a leitura do potenciômetro for 512, metade da escala possível para entrada analógica, estaremos enviando 128, que é a metade do valor possível para saída analógica.

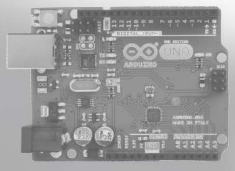
9 `}` – Finaliza loop() retornando a linha 7 e iniciando novamente.

Desdobramentos

Pode-se trocar o disco de cores por este outro ao lado e o resultado pode ser bem surpreendente à medida que se altera a velocidade. Com esta roda, o efeito é melhor percebido se visto sob luz fluorescente ou estroboscópica, ou ainda filmado por um celular, ou outro tipo de câmera.



Fonte: Produzida por Silas J. P. Silva 2022



7

Frequência sonora

Ensino de física

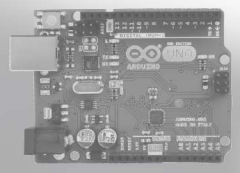
Cada nota musica, corresponde a uma frequência sonora. Este projeto, "brinca" com estas frequências, formando sons, bipes e até musicas.

Lista de materiais e ferramentas

01 - Busser 5v 12mm ou
01 - Altofalante 0,5W 8Ω
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

Observação

1 - A qualidade do som deixa a desejar, mas a composição e frequência de cada nota é fiel.
2 - Pode-se tentar gerar sons de alertas como sirenes buzinas, por exemplo.



Este *sketch* introduz o uso de tons e frequências sonoras com Arduino, para executar a marcha de Darth Vader.

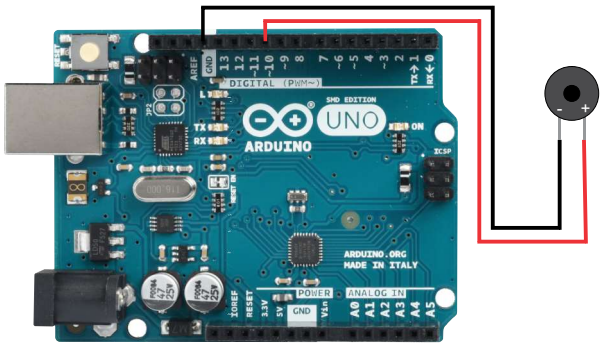
Componentes principais

Buzzer 5v 12mm

ou

Autofalante 0,5W 8Ω

Diagrama de montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Código-fonte para o Arduino

```
#define B0   31
#define C1   33
#define CS1  35
#define D1   37
#define DS1  39
#define E1   41
#define F1   44
#define FS1  46
#define G1   49
#define GS1  52
#define A1   55
#define AS1  58
#define B1   62
#define C2   65
#define CS2  69
#define D2   73
#define DS2  78
#define E2   82
#define F2   87
#define FS2  93
#define G2   98
#define GS2  104
#define A2   110
#define AS2  117
#define B2   123
#define C3   131
#define CS3  139
#define D3   147
#define DS3  156
#define E3   165
#define F3   175
#define FS3  185
#define G3   196
#define GS3  208
#define A3   220
#define AS3  233
#define B3   247
#define C4   262
#define CS4  277
#define D4   294
#define DS4  311
#define E4   330
#define F4   349
#define FS4  370
#define G4   392
#define GS4  415
#define A4   440
#define AS4  466
#define B4   494
#define C5   523
#define CS5  554
#define D5   587
#define DS5  622
#define E5   659
#define F5   698
#define FS5  740
#define G5   784
#define GS5  831
#define A5   880
#define AS5  932
#define B5   988
#define C6   1047
#define CS6  1109
#define D6   1175
#define DS6  1245
#define E6   1319
#define F6   1397
#define FS6  1480
#define G6   1568
#define GS6  1661
#define A6   1760
#define AS6  1865
#define B6   1976
#define C7   2093
#define CS7  2217
#define D7   2349
#define DS7  2489
#define E7   2637
#define F7   2794
#define FS7  2960
#define G7   3136
#define GS7  3322
#define A7   3520
#define AS7  3729
#define B7   3951
#define C8   4186
#define CS8  4435
#define D8   4699
#define DS8  4978
#define ZZ   0
```

...código-fonte para o Arduino

```
91 #define ALTO_FALANTE 10 // alto-falante ou buzzer
92 int NUM_NOTAS = 44;     // número de notas incluídas em melodia[]
93
94 int melodia[] = {
95     G4, G4, G4, DS4, AS4, G4, DS4, AS4, G4,
96     D5, D5, D5, DS5, AS4, FS4, DS4, AS5, G4,
97     G5, G4, G4, G5, FS5, F5, E5, DS5, E5,ZZ,
98     GS4, CS5, C5, B5, AS5, A5, AS5, ZZ,
99     DS4, FS4, DS4, AS4, G4, DS4, AS5, G4};
100
101// duração: 1 = semibreve, 2 = mínima, 4 = seminima, 8 = colcheia,
102etc.:
103int duracoes[] = {
104     4, 4, 4, 6, 8, 4, 6, 8, 2,
105     4, 4, 4, 6, 8, 4, 6, 8, 2,
106     4, 6, 8, 4, 6, 8, 16, 16, 8, 8,
107     8, 4, 6, 8, 16, 16, 8, 8,
108     8, 4, 6, 8, 4, 6, 8, 2
109};
110void setup() {
111     for (int nota = 0; nota < NUM_NOTAS; nota++) {
112         int duracao = 1000 / duracoes[nota]; // divide 1000 por cada
113                                              //duração
114         tone(ALTO_FALANTE, melodia[nota], duracao);
115         int pausa = duracao * 1.30;
116         delay(pausa);
117         noTone(ALTO_FALANTE);
118     }
119}
120void loop() {}
```

Comentário do código

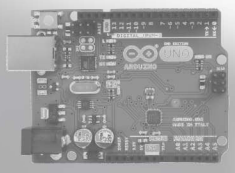
As linhas de 1 a 90 fazem as definições das notas musicais. É verdade que a maioria delas não foram utilizadas neste *skecth*, mas foram deixadas de propósito para que o professor possa explorar com seus alunos as possibilidades musicais. Como exemplo deste grupo de linhas vamos ver o conteúdo da linha 2.

2 `#define C1  33` – Esta linha, semelhante às outras 89, traz a definição de C1 como tendo o valor 33. C1 funciona como se fosse uma variável ela "meio que" armazena o valor 33, e este, define a frequência em Hz (Hertz) do som que será tocado pelo Arduino. Assim, cada nota tem sua própria frequência e as notas configuradas nestas linhas representam 7 oitavas das notas musicais, com seus sustenidos, por este motivo são 89 notas.

Se o professor, assim como eu, não entende estes conceitos musicais tão bem, pode simplesmente procurar na internet por códigos-fontes para Arduino com músicas, e conseguirá um acervo bem vasto. Outra coisa a se ter em mente é que a qualidade do som é bem baixo, servindo apenas como sinais a serrem emitidos e não como música para ouvir.

91 `#define ALTO_FALANTE  10` – Esta linha cria a definição com o valor 10. Sempre que se fizer referência a ALTO_FALANTE, ela será substituída por 10, que é o número da porta onde o *buzzer* ou auto-falante será ligado.

92 `int NUM_NOTAS = 44;` - Cria uma variável chamada NUM_NOTAS com o valor 44, que representa o número total de notas que a melodia



... comentário do código

terá. Caso queira aumentar ou diminuir o tamanho da melodia, este número precisa ser alterado.

94 a 99 – Cria uma variável tipo lista (ou vetor) contendo todas as notas da melodia na ardem em que devem ser tocadas. A lista melodia[] conterá todas as notas escolhidas, sendo que cada uma nota pode ser referenciada pela sua posição na lista, que começa em 0 (zero). Por exemplo, melodia[0] = G4, melodia[4] = AS4 e melodia[41] = DS4.

103 a 109 – Cria uma lista ou vetor, chamado "durações[]", contendo os valores correspondentes ao tempo (musical) que cada nota deverá ser tocada. Pela lógica apresentada, o valor de cada um dos tempos será o dividendo de 1000. Portanto, quanto menor o número, maior será o tempo de execução da nota.

110 a 119 – Função setup(). Diferente dos outros experimentos, neste, toda ação acontece na função setup(). Isso porque, não queremos que a melodia fique sendo repetida infinitamente enquanto o Arduino está ligado. Ao contrário, ela será executada apenas uma vez quando ligarmos o Arduino e não mais. Caso queiramos executar mais de uma vez, basta desligar e ligar novamente ou pressionar o botão *reset*.

`111 for (int nota = 0; nota < NUM_NOTAS; nota++) {` - Esta linha inicia uma contagem que coloca as linhas 112 a 117 em repetição. Esta repetição começa em 0 (zero) e se dará até o valor de NUM_NOTAS, definida na linha 92 como 44(quarenta e quatro). A cada repetição a variável nota assume os valores referentes àquela repetição. Pro exemplo, na primeira repetição nota = 0, na próxima nota = 1 e assim por diante até 43.

`112 int duracao = 1000 / duracoes[nota];` - Divide 1000(mil) que é o tempo básico, por cada um dos valores definidos em duracoes[] e armazena esse valor em duracao (note que duracoes[] e duração são variáveis diferentes). Como exemplo podemos citar que nas primeiras três repetições duracao vai ser igual a 1000 / 4 e, na quarta repetição vai ser 1000/6, depois 1000/8 e assim por diante, conforme os valores definidos ente as linhas 1o4 e 108.

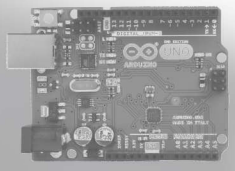
`114 tone(ALTO_FALANTE, melodia[nota], duracao);` - O comando "tone" é quem executa de fato a nota com os parâmetros que estão entre parênteses. Assim, na primeira repetição teremos um resultado em que substituindo as variáveis pelos seus respectivos valores, teremos "tone (10, 394,250)".

`115 int pausa = duracao * 1.30;` - Cria uma variável chamada pausa e atribui a ela o valor de duracao * 1.3. No caso da primeira repetição duracao = 250, e pausa seria 325 (250 + 1.3)

`116 delay(pausa);` - Executa uma pausa com duração definida no cálculo acima. Como exemplo, da linha acima teremos "delay(350)".

`117 noTone(ALTO_FALANTE);` - Gera um silêncio no alto-falante.

`118 }` – Verifica se a repetição iniciada na linha 111 chegou ao final. Se chegou, esse trecho do código é encerrado, se não, é acrescentado



... comentário do código

1(um) ao valor de “nota” e a repetição continua a partir da linha 111. No nosso caso essa repetição vai acontecer 44 vezes, correspondente ao número de notas. A cada vez uma nota com uma duração diferente é executada.

`120 void loop() {}` – Como dito anteriormente, na função loop() nada acontece. Por esse motivo essa linha termina com “{}”, o que significa que a função está vazia. Neste código exemplo, toda ação acontece na função setup().

Desdobramentos

Em se tratando de sons, as possibilidades são muitas. Sirenes, toques, bipes e muito mais podem ser implementados pela função "tone()" do Arduino. Pode-se por exemplo, desenvolver um *sketch* em conjunto com um sensor ultrassônico de distância em que um bipe toca a uma determinada distância e vai diminuindo o intervalo de toque à medida que o objeto se aproxima. As possibilidades so dependem da criatividade de professores e alunos.

Para consulta

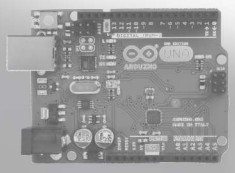
As notas musicais neste projeto estão substituídas em seus nomes pelas letras respectivas usadas em cifras de músicas. Não se trata, partitura e sim de cifra.
Abaixo uma tabela com as notas e a letra respectiva que normalmente aparecem e cifras musicais.

Dó	Ré	Mi	Fá	Sol	Lá	Si
C	D	E	F	G	A	B

Fonte: Produzida por Silas J. P. Silva 2022

Outra forma de fazer musica com arduino pode ser encontrado em: https://dragaosemchama.com/2019/02/musicas-para-arduino/

Um projeto de um teclado musical, bem básico e rústico pode ser encontrado em :https://create.arduino.cc/projecthub/GlauberSantiago/tgl-teclado-musical-com-8-notas-e-1-tecla-de-funcao-04965a



8

Equilíbrio estático

Ensino de física

O objetivo é deslocar os pesos entre os dois sensores para que os resultados sejam enviados ao computador. Com a variação dos dados de acordo com os pesos, professor pode explorar muitos dos conteúdos envolvendo massa.

Lista de materiais e ferramentas

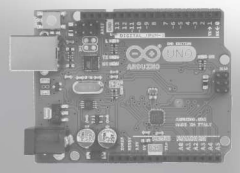
01 - Módulo sensor e placa de massa (peso) Hx711 de 5kg
* Arduino UNO
* Jumppers ou fios
* Protoboard (opcional)

Observações

1 - Existem sensores para faixas de peso e o experimento pode mudar de acordo com a necessidade do professor. Só é preciso lembrar que para cada faixa de peso é importante fazer a calibração com um peso bem menor, mais não muito. Caso use uma célula de 50Kg, por exemplo, não faz sentido fazer a calibração com algumas gramas. O idela neste caso seria usar um peso de alguns quilos.

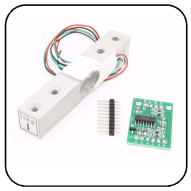
2 - Este sketch é complexo na sua execução, apenas devido à calibração. Passado esta fase, os experimentos podem fluir mais rapidamente, pois a calibração é feita apenas uma única vez para cada sensor.

3 - Anote no sensor o número de calibração conseguido após testado no programa de teste, para uso futuro.



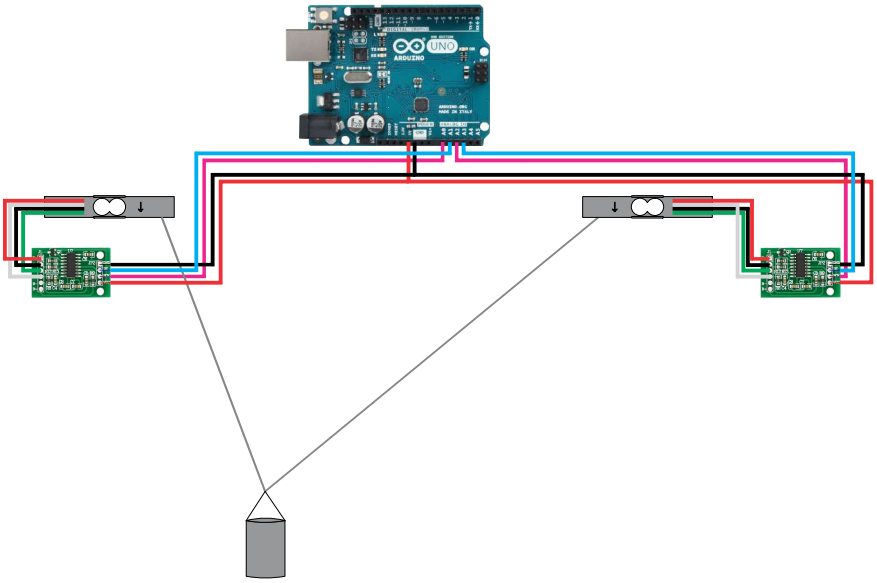
Este experimento, talvez seja o mais complexo de todos os apresentados aqui. Isso não significa que qualquer um não possa executá-lo. Trata-se de utilizar dois sensores tipo balança para medir a massa (peso) de objetos e vários desdobramentos podem advir dele. Na secção "Para Consulta", estará disponível um código que deverá ser usado apenas para calibração do sensor. Se isso for feito corretamente, basta realizar a calibração uma única vez. Mas isto implica que haverá um código que será executado primeiro para calibragem e outro que ficará rodando me definitivo no projeto.

Componentes principais



Módulo sensor e placa de massa (peso) Hx711 de 5kg

Diagrama de montagem/funcionamento



Fonte: Produzida por Silas J. P. Silva 2022

Montagem. Esse *sketch* precisará de suportes feitos de madeira. Pode ser usado madeira de *palet* para fazer um quadro de 1m X 1m (um metro por um metro) para fixar as células de carga. Para prender o peso pode ser usado qualquer tipo de barbante e o peso em si, pode ser uma pequena lata cheia de areia ou qualquer outra coisa

Notar que existe uma seta de indicação do sentido da carga aplicada. No caso do exemplo acima a seta deve ficar virada para baixo. Isso é de extrema importância para o funcionamento do sensor.

Código-fonte para o Arduino

Apresentamos agora o código definitivo que vai rodar no experimento. Como dito, existe um código de calibração que precisa ser gravado e executado primeiramente no Arduino, para só depois esse código abaixo deve ser gravado. Por se tratar do código definitivo, é este código que será comentado. O código de calibração será apenas apresentado, sem comentários.

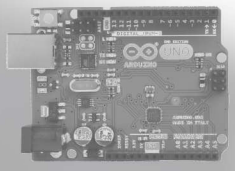
```
1   #include "HX711.h"
2
3   #define DT1  A1
4   #define SCK1 A0
5   #define DT2  A3
6   #define SCK2 A2
7   HX711 pesa1;
8   HX711 pesa2;
9   void setup() {
10    Serial.begin(9600);
11    pesa1.begin (DT1, SCK1);
12    pesa2.begin (DT2, SCK2);
13    Serial.print("Leitura do Valor Inicial:  ");
14    Serial.println(pesa1.read());   // Aguarda até pesa1 estar pronto
15    Serial.println(pesa2.read());   // Aguarda até pesa2 estar pronto
16    Serial.println("Nao coloque nada na balanca!");
17    Serial.println("Iniciando...");
18    pesa1.set_scale(397930.55);   // Substituir 397930.55 pelo valor
19  encontrado na calibração do sensor 1
20    pesa1.tare(5);                // faz 5 pesagens e reseta o peso
21  pra 0
22    pesa2.set_scale(397930.55);   // Substituir 397930.55 pelo valor
23  encontrado na calibração do sensor 2
24    pesa2.tare(5);                // faz 5 pesagens e reseta o peso
25  pra 0
26    Serial.println("Amarre o peso nos sensores");
27  }
28   void loop() {
29   Serial.print("Peso sensor 1: ");
30   Serial.print(pesa1.get_units(5), 3); // faz a 5 medidas e pega a
31  média com 3 digitos decimais
32   Serial.println(" kg  ");
33   Serial.print("Peso sensor 2: ");
34   Serial.print(pesa2.get_units(5), 3); // faz a 5 medidas e pega
35  amédia com 3 decimais decimais
36   Serial.println(" kg  ");
37   delay(1000);
38  }
```

Comentário do código

1 `#include "HX711.h"` - Esta linha faz a inclusão de algo chamado aqui na programação do Arduino de biblioteca. Muitos sensores precisam de uma biblioteca específica para funcionar e geralmente esta biblioteca é disponibilizada pelo fabricante e precisa ser instalada à parte para que o Arduino possa utilizá-la. Para fazer a instalação, vá na seção "Para Consulta".

3 `#define DT1  A1`, **4** `#define SCK1 A0`, **5** `#define DT2 A3` e **6** `#define SCK2 A2` - Trazem as definições dos pinos do Arduino que receberão os pinos `DT` e `SCK` de cada sensor. `DT1` para peso1 e `DT2` para peso2 e assim por diante.

7 `HX711 pesa1;` e **8** `HX711 pesa2;` - Estas linhas dizem ao Arduino que



... comentário do código

vamos trabalhar com dois sensores. Um vai se chamar pesa1 e outro, pesa2.

9 `void setup() {` - Incia função setup() que será finalizada na linha 23

10 `Serial.begin(9600);` - Inicia a comunicação com o computador, necessário pois os valores dos pesos serão enviados a ele.

11 `pesa1.begin (DT1, SCK1);` e 12 `pesa2.begin (DT2, SCK2);` - Inicia o funcionamento dos sensores nomeados anteriormente, cada um nos seus terminais: DT1 e SCK1 para sensor de nome pesa1 e DT2 e SCH2 para o sensor de nome pesa2.

13 `Serial.print("Leitura do Valor Inicial: ");` 14 `Serial.println(pesa1. read());` 15 `Serial.println(pesa2. read());` 16 `Serial.println("Nao coloque nada na balanca!");` e 17 `Serial.println("Iniciando...");` - Estas linhas enviam para balança o valor inicial de peso que será descartada assim que a balança for inicializada.

18 `pesa1.set_scale(397930.55);` - Configura a balança com o valor gerado pelo programa de calibração para o sensor um, aqui chamado "pesa1". Lembre-se que a balança precisa ser calibrada antes e o processo para isso é mostrado em "Para Consulta".

19 `pesa1.tare(5);` - Zera o peso da balança (tara) do sensor pesa1, com base no valor de calibração passado na linha acima.

20 `pesa2.set_scale(397930.55);` - Configura a balança com o valor gerado pelo programa de calibração para o sensor um, aqui chamado "pesa2". Lembre-se que a balança precisa ser calibrada antes e o processo para isso é mostrado em "Para Consulta".

21 `pesa2.tare(5);` - Zera o peso da balança (tara) do sensor peso2, com base no valor de calibração passado na linha acima.

22 `Serial.println("Amarre o peso nos sensores");` Neste momento o peso deverá ser preso aos sensores.

23 `}` - Finaliza setup().

24 `void loop() {` - Inicia a função loop() que será executada repetidamente durante o experimento.

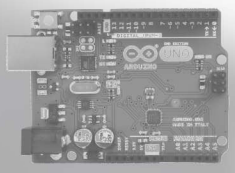
25 a 30 - Mostra na tela uma mensagem com os pesos dos dois sensores, peso1 e peso2, no seguinte formato:

```
Peso sensor 1: X,xxx kg
Peso sensor 2: X,xxx kg
```

A instrução `(pesaX.get_units(5), 3)`, faz 5 leituras rápidas de peso e tira a média, com 3 dígitos decimais, em Kg(quilogramas). Assim tem-se mais precisão e o valor mostrado, será no formato X,xxx, onde "X" é a unidade em Kg e xxx são as unidade em gramas.

31 `delay(1000);` - Aguarda um segundo antes da fazer a próxima média. Deve-se destacar porém, que a precisão das medidas não serão exatas, sempre haverá uma variação de algumas poucas gramas para cada quilo.

32 `}` - Finaliza o bloco da função loop() o que faz com que o programa reinicie a partir da linha 24.

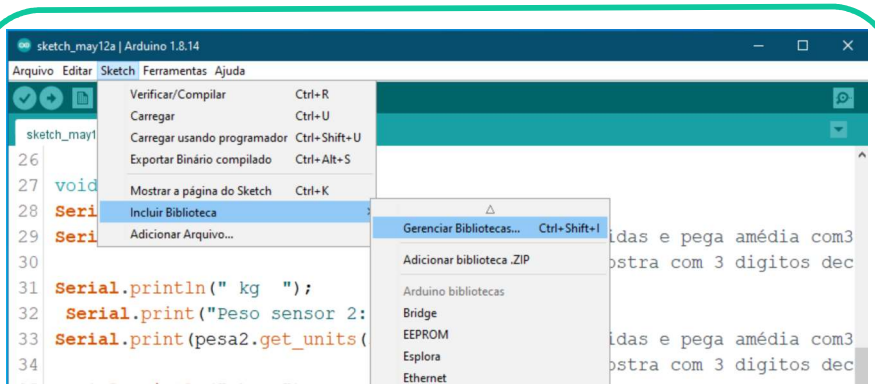


Desdobramentos

Como se trata basicamente de uma balança, os desdobramentos para este código pode ser qualquer coisa que envolva pesagem. Embora o código seja um pouco pesado para um iniciante, dominar este *sketch* vai proporcionar riscas experiências e projetos envolvendo massa.

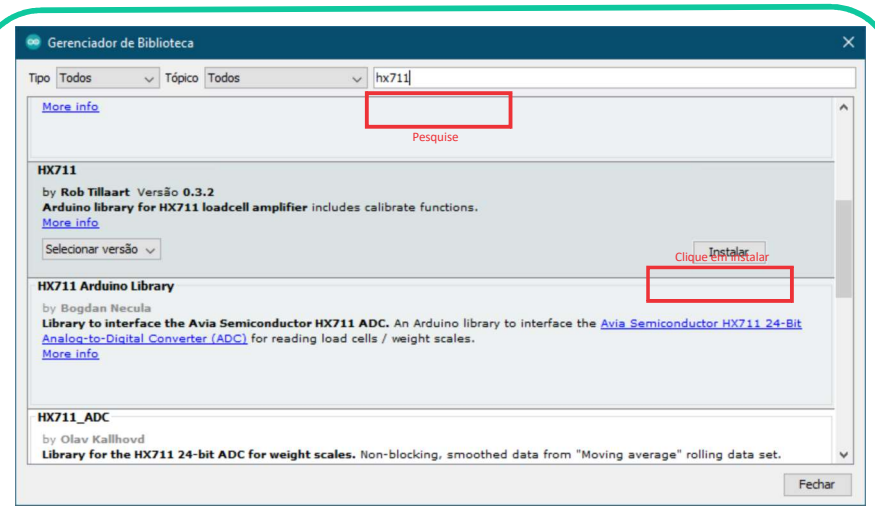
Para consulta - Instalar uma biblioteca no Arduino

Neste código temos a linha #include "HX711.h". Isso indica ser necessário disponibilizar esta biblioteca antes de mandar este código para o Arduino. Vamos aos passos.
Primeiro clique no menu "Sketch", "Incluir Biblioteca" e depois escolha a opção "gerenciar Bibliotecas", como mostrado na figura abaixo.

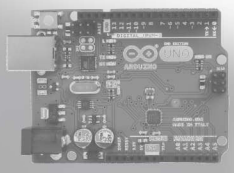


Fonte: Produzida por Silas J. P. Silva 2022

Ao aparecer a janela abaixo, pesquise pelo nome "XH711". Dos nomes que podem aparecer, opte sempre pelo mais simples e clique em instalar. Depois de alguns instantes a biblioteca estará instalada. Fecha a janela e reinicie a IDE do Aduino, para que a biblioteca instalada fique disponível. Após instalada, uma biblioteca permanece disponíveis para os projetos futuros.



Fonte: Produzida por Silas J. P. Silva 2022



... para consulta

procedimento de calibração dos sensores

Antes de ter este experimento com o código definitivo que vai ficar efetivamente rodando no Arduino, temos que instalar um código temporário para calibrar os sensores. Isto é necessário, pois este sensor trabalha com deformação de metal e as leituras sempre dão diferença de um para o outro, além do fato que, sem peso, os sensores produzem leituras aleatórias. Por isso você precisa fazer a calibração de cada um dos sensores de massa separadamente. Não vamos comentar o código, linha a linha, mas vamos disponibilizá-lo aqui com os comentários que o ajudarão neste processo de calibração.
Para realizar a calibração você vai precisar de um objeto de peso conhecido que seja bem menor que o valor máximo do sensor, neste caso menor que 5Kg. Vamos supor que você possui um objeto cujo peso seja de 180g (cento oitenta gramas) ou 0.180Kg.

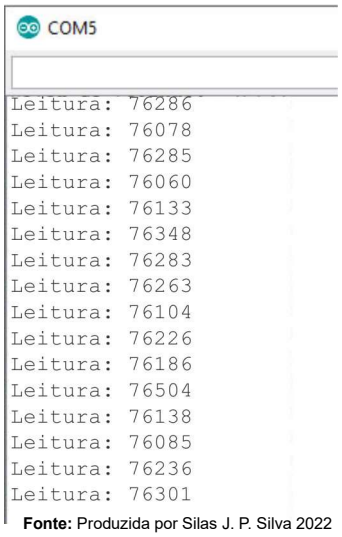
código-fonte para calibração dos sensores

De posse deste peso, pegue o código abaixo e grave no Arduino.

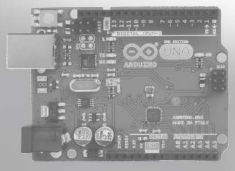
```
#include "HX711.h"

#define DT A1
#define SCK A0
HX711 calibra;
void setup() {
  calibra.begin (DT, SCK);
  Serial.begin(9600);
  Serial.print("Tara:  ");
  Serial.println(calibra.read());
  Serial.println("Espere um pouco!!!");
  Serial.println("Iniciando ...");
  calibra.set_scale();
  valibra.tare(20);
  Serial.println("Pendure o peso conhecido");
}
void loop() {
  Serial.print("Leitura: ");
  Serial.println(escala.get_value(10),0);
  delay(100);
}
```

Ao executar este código e observar os resultados no Monitor Serial no computador, você vai perceber que o sensor emite números aleatórios e com valores baixos, o que é perfeitamente normal. Ao aparecer a mensagem para pendurar o peso, faça isso.
Neste momento devem aparecer os números relacionados ao peso utilizado, em valores bem altos, como no exemplo ao lado que utilizaremos como exemplo para os cálculos .



Fonte: Produzida por Silas J. P. Silva 2022



... para consulta

procedimento de calibração dos sensores

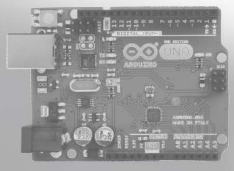
Com os valores em tela, como no exemplo, deve-se fazer uma análise empírica e pegar 10 (dez) destes resultados entre os mais altas e os mais baixos, para se construir uma média. Assim teríamos no nosso exemplo: (76060+76133+76348+76283+76263+76104 +76226+76186+76504+76138)/10 = 76224.5, este último nosso número de ouro (76224.5). Agora, para obtermos o números definitivo da calibração, dividimos este valor pelo peso conhecido que utilizamos em quilogramas (0.180), assim temos 76224.5/ 0.180 = 423469.44. Este é valor que deverá ser colocado no programa final para este sensor.

```
Leitura: 76286
Leitura: 76078
Leitura: 76285
Leitura: 76060
Leitura: 76133
Leitura: 76348
Leitura: 76283
Leitura: 76263
Leitura: 76104
Leitura: 76226
Leitura: 76186
Leitura: 76504
Leitura: 76138
Leitura: 76085
Leitura: 76236
Leitura: 76301
```

Fonte: Produzida por Silas J. P. Silva 2022

Na linha 18, do nosso programa principal, temos a instrução:
`pesa1.set_scale (397930.55);` O valor contido nela deverá ser substituído pelo numero final encontrado na calibração ficando assim:
`pesa1.set_scale (423469.44);`
Repita a calibração para o segundo sensor e substitua o número contido na instrução da linha 22 polo encontrado na segunda calibração e tudo deve funcionar perfeitamente.

Pata maiores informações veja o artigo em:
https://www.usinainfo.com.br/blog/balanca-arduino-com-celula-de-peso-e-hx711-tutorial-calibrando-e-verificando-peso/
Ou assista o excelente vídeo sobre o assunto em:
https://www.youtube.com/watch?v=-qLfybfvsHw



Considerações finais

O uso da Cultura Maker com Arduino para o ensino de física está apenas no início. Neste sentido buscou-se aqui muito mais trazer ao professor da área, um despertar para as possibilidades que esta plataforma pode proporcionar para sua prática. Os experimentos trazidos aqui, podem suscitar outros tantos, que nos, autores deste trabalho, por não sermos da área de física só podemos imaginar. Nem de longe buscou-se esgotar o assunto ou trazer um número grande de exemplos, pois eles estão vastamente disponibilizados na internet.

A intenção essencial deste trabalho é mesmo o de plantar uma semente. Trazer a tona uma possibilidade tão pouco explorada nas escolas, principalmente públicas, do país. Intenciona-se também apresentar a docentes que ministram a disciplina de física subsídios para poderem contornar, na medida do possível, a deficiência na disponibilidade de laboratórios de física em escolas estaduais e municipais das comunidades em que o poder público ainda não implementou laboratórios adequados e estes espaços são limitados ou mesmo inexistentes. Por ser uma plataforma livre e de baixo custo, o Arduino pode ocupar esta lacuna e levar às salas de aula uma experiência prática ao ensino dos conteúdos e dar sentido aos conteúdos dos livros didáticos.

Por último, e não menos importante, a intenção é também aproximar os alunos às tecnologias da informação para a facilitar a apropriação dos conceitos que as envolvem, de forma crítica e emancipadora.

Pode parecer bem pretensioso, mas toda semente não é?

Dúvidas e sugestões sobre este conteúdo por parte de docentes da área de física, podem ser enviadas para silas.silva@ifro.edu.br com o assunto "Arduino no ensino de Física".

Referências

BROCKVELD, M. V. V.; TEIXEIRA, C. S.; SILVA. M. R. **A Cultura Maker em prol da inovação: boas práticas voltadas a sistemas educacionais**. Conferência Rio +30, 2017. Disponível em: http://via.ufsc.br/wp-content/uploads/2017/11/maker.pdf. Acesso em: 20 set. 2021.

FREITAS, E. A. M. M.; LIBÂNEO, J. C. **Didática desenvolvimental e políticas educacionais para a escola no Brasil. In: Linhas Críticas**. Brasília, 2019. Disponível em: https://periodicos.unb.br/index.php/linhascriticas/article/view/21850. Acesso em: 14 out. 2021.

MCROBERTS, M. **Arduino básico**. São Paulo: Novatec Editora, 2011.