



**PROGRAMA DE MESTRADO PROFISSIONAL EM
ENSINO DE CIÊNCIAS E MATEMÁTICA**

PRODUTO EDUCACIONAL

Curso: Arduino como recurso didático ao ensino de física

Adriana de Andrade

Marcio Vinicius Corrallo

São Paulo (SP)

2021

Catálogo na fonte
Biblioteca Francisco Montojos - IFSP Campus São Paulo
Dados fornecidos pelo(a) autor(a)

a553p Andrade, Adriana de
Produto educacional: curso de arduino como recurso didático ao ensino de física / Adriana de Andrade. São Paulo: [s.n.], 2021.
100 f. il.

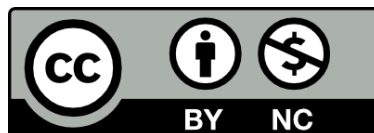
Orientador: Marcio Vinicius Corrallo

Dissertação (Mestrado Profissional em Ensino de Ciências e Matemática) - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, 2021.

1. Arduino. 2. Ensino de Física. 3. Automatização da Coleta de Dados. I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo II. Título.

CDD 510

Este trabalho está licenciado sob uma Licença Creative Commons Atribuição-NãoComercial 4.0 Internacional. Para ver uma cópia desta licença, visite <http://creativecommons.org/licenses/by-nc/4.0/>.



Produto Educacional apresentado como requisito à obtenção do grau de Mestre em Ensino de Ciências e Matemática pelo Programa de Mestrado Profissional em Ensino de Ciências e Matemática do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus São Paulo. Aprovado em banca de defesa de mestrado no dia 17 dezembro 2021.

AUTORES

Adriana de Andrade: Licenciada em Matemática pela Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) Campus Caraguatatuba e Mestre em Ensino de Ciências e Matemática pelo Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP). Pesquisa sobre ambientes virtuais de aprendizagem (Moodle) e Tecnologias para o Ensino (simuladores, softwares para ensino de física e matemática, eletrônica etc.).

Marcio Vinicius Corrallo: Professor do Instituto Federal de São Paulo – IFSP – Campus São Paulo, desde 2010. Doutor em Ensino de Ciências (Modalidade Física) pela Universidade de São Paulo. Professor permanente do programa de mestrado profissional em Ensino de Ciências e Matemática do IFSP. Líder do Grupo de Pesquisa em Inovação Tecnológica para o Ensino de Física – GPITEF. Atua em cursos e projetos de Educação a Distância no IFSP. Investiga principalmente o uso e as aplicações das atividades experimentais, com apoio de tecnologias, para a formação de professores de Física.

Sumário

Apresentação.....	5
Introdução.....	6
Referencial teórico.....	6
Curso: Arduino como recurso didático ao ensino de física.....	8
Objetivos.....	9
Metodologia.....	9
Conteúdo Programático.....	9
Avaliação.....	10
Bibliografia Básica.....	10
Bibliografia Complementar.....	10
Semana 1.....	11
Semana 2.....	14
Semana 3.....	21
Semana 4.....	25
Semana 5.....	42
Semana 6.....	51
Semana 7.....	60
Semana 8.....	69
Semana 9.....	72
Semana 10.....	93
Referências.....	94

Apresentação

O presente produto educacional faz parte do PTT (Produtos Técnicos Tecnológicos) e é parte da dissertação de mestrado intitulada: “Um estudo das Representações Sociais sobre a automatização da coleta de dados no laboratório didático de física durante a formação docente”, do Mestrado Profissional em Ensino de Ciências e Matemática do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus São Paulo.

O objetivo deste material é apresentar o curso de extensão: “Arduino como recurso didático ao ensino de física” para que possa servir de apoio para qualquer professor que queira ministrar um curso de formação continuada. Para as aulas práticas, utilizamos o ambiente virtual Tinkercad¹. Nele é possível simular circuitos eletrônicos por meio de sensores, e realizar medições que auxiliam na compreensão de conceitos físicos. Vale ressaltar que o simulador utiliza o mesmo método de montagem e programação do Arduino físico.

Desse modo, esperamos que ao fim da aplicação do curso deste produto educacional, o cursista possa utilizar o Arduino como ferramenta didática nas principais temáticas da física, como em experimentos de mecânica, termodinâmica, eletrodinâmica e física moderna, tornando o Arduino uma ferramenta de alto potencial para o ensino de física.

Boa Leitura!
Os Autores.

¹ Tinkercad é um produto da empresa Autodesk, Inc, e permite desenvolver projetos 3D, programas e circuitos eletrônicos de forma gratuita. Disponível em: <https://www.tinkercad.com>, Acesso em: 04 mar. de 2021.

Introdução

O curso foi desenvolvido para ser ministrado *on-line* e está dividido em 10 semanas de 4 horas, com sugestão de um encontro síncrono semanal.

Referencial Teórico

O Ambiente Virtual de Aprendizagem (AVA) Tinkercad é um *software* da empresa Autodesk, Inc, que permite desenvolver projetos 3D e circuitos eletrônicos de forma gratuita. Possui uma diversidade de recursos digitais capazes de criar sistemas eletrônicos elaborados, contribuindo, assim, como ferramenta didática para as principais áreas da física, como em experimentos de mecânica, termodinâmica, eletrodinâmica e física moderna. Nesse ambiente, encontramos diversos componentes eletrônicos, tais como: resistores, capacitores, sensores, diodo, indutor, potenciômetro, LED, motor de vibração, sensor infravermelho, placas microcontroladores e diversos instrumentos, como multímetro, fonte de energia, circuitos integrados, entre outros periféricos.

É razoável supor que a utilização do AVA TinkerCAD, sob determinadas estratégias pedagógicas, pode oportunizar ao estudante a realização de atividades experimentais nos moldes “*Do-It-Yourself*”, ou seja, “Faça você mesmo”, remetendo à cultura *maker*, que está articulada com as metodologias ativas de ensino, potencializando o protagonismo do aluno.

A BNCC, na área das Ciências da Natureza e suas Tecnologias, diz que é fundamental a apropriação do conhecimento científico e tecnológico por parte dos estudantes: “Aprender tais linguagens, por meio de seus códigos, símbolos, nomenclaturas e gêneros textuais, é parte do processo de letramento científico necessário a todo cidadão.” (BRASIL, 2017, p. 553). Já Andrade e Corrallo (2020, p. 1) acrescentam que “[...] cabe ao professor, compreender os letramentos² do *maker*, e assim produzir multiletramentos para que os estudantes possam interagir com os conteúdos disciplinares com autonomia, dentro do AVA escolhido.”

²“Os letramentos são, em si mesmos, tecnologias e nos dão as chaves para usar tecnologias de forma mais amplas.” (LEMKE, 2010, p. 456).

A cultura *maker* está fundamentada por nove ideias chaves: “[...] faça, compartilhe, ensine, aprenda, use ferramentas (ou seja, ou tenha acesso seguro às ferramentas necessárias), jogue, participe, apoie e provoque mudanças[.]” (MEIRA; RIBEIRO; PRADO, 2016, p. 3). Os autores acrescentam ainda que é uma “[...] forma de estimular a inteligência colaborativa, a criatividade e o caráter prático do uso das tecnologias[.]”. Fortunato, Tardin, Pinheiro (2020, p. 2) salientam que cultura *maker* se relaciona

[...] as tecnologias e a pedagogia ‘mão na massa’ tendem a cativar estudantes, principalmente porque proporcionam a busca por mais conhecimentos. [...] todos acabam por exercitar sua criatividade ao desafiarem a si mesmos, testando e superando os próprios limites. (FORTUNATO; TARDIN; PINHEIRO, 2020, p. 2).

O AVA Tinkercad dialoga com a cultura *maker*, sobretudo, na automatização da coleta de dados, uma vez que, encontramos no AVA Tinkercad uma bancada para que o estudante construa os próprios circuitos eletrônicos, e, assim, o professor poderá articular uma compreensão dos conceitos físicos, demonstrados no circuito desenvolvido. Nesse contexto, em relação ao Arduino³, Silva (2018, p. 14) destaca que:

Os conceitos e aplicações da Computação Física para a aprendizagem de programação, tal como sua inserção nos processos educacionais; O papel da plataforma Arduino nesse contexto, características do produto e possibilidades de uso, abordando especialmente o uso de simuladores; O papel da motivação sobre o processo de aprendizagem e sua influência no design instrucional [...]. (SILVA, 2018, p. 14).

Nesse sentido, em circuitos, o AVA Tinkercad articula-se ao ensino de física, e ainda conta com os principais benefícios da cultura *maker*, que é a democratização do conhecimento em relação ao “fazer”, proporcionando a adaptação de ferramentas tecnológicas no âmbito educacional.

É notório que o movimento *maker* vem sendo disseminado na educação, podendo enriquecer os processos de aprendizagem na sala de aula desde que haja um planejamento estratégico, com projetos pedagógicos bem elaborados para não

³ Arduino Uno é uma placa microcontrolada baseada no ATmega328P (ficha técnica). Possui 14 pinos de entrada / saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um ressonador de cerâmica de 16 MHz (CSTCE16M0V53-R0), uma conexão USB, um conector de alimentação, um conector ICSP e um botão de reinicialização. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3> Acesso em: 04.jun. 2021

resultar em prejuízo no processo de aprendizagem. Os autores Bacich e Moran (2018) indicam que:

As aprendizagens por experimentação, por design e aprendizagem *maker* são expressões atuais da aprendizagem ativa, personalizada, compartilhada. A ênfase na palavra ativa precisa sempre estar associada à aprendizagem reflexiva, para tornar visíveis os processos, os conhecimentos e as competências do que estamos aprendendo com cada atividade (BACICH; MORAN, 2018, p. 28).

De acordo com Andrade e Corrallo (2020), articular o Tinkercad em apoio ao ensino de física, tanto o estudante quanto o professor ganham em relação ao protagonismo do processo de ensino e aprendizagem. No entanto, os autores alertam sobre a importância da intencionalidade do professor, que precisa estar calcada, sobretudo, no processo de aprendizagem do estudante.

O AVA Tinkercad dialoga tanto com a cultura *maker* quanto com a BNCC, no sentido de desenvolver o protagonismo do estudante, principalmente na compreensão das tecnologias de forma crítica, significativa e reflexiva, produzindo conhecimentos e resolvendo problemas.

[...] a cultura digital tem promovido mudanças sociais significativas nas sociedades contemporâneas. Em decorrência do avanço e da multiplicação das tecnologias de informação e comunicação e do crescente acesso a elas pela maior disponibilidade de computadores, telefones celulares, tablets e afins, os estudantes estão dinamicamente inseridos nessa cultura, não somente como consumidores. (BRASIL, 2017, p. 59).

Dentro desse contexto, a escola terá que formar cidadãos não para ser consumidores das tecnologias digitais, mas para serem cidadãos críticos em relação às tecnologias.



Curso de Arduino como recurso didático ao ensino de física

O curso está dividido em 10 semanas de 4 horas de curso

EMENTA - Curso: O Arduino como recurso didático ao ensino de física:

O curso almeja abordar de forma crítico-reflexiva a inserção de elementos tecnológicos para o ensino de física, e, em particular, a automatização da coleta de dados de experimentos amplamente utilizados nos diversos níveis de ensino. Com a adoção da plataforma de prototipagem Arduino, pretende-se desenhar, de forma colaborativa, itinerários pedagógicos que oportunizem: maior reflexão e discussão na tomada de dados, estratégias com um viés investigativo e testes de hipóteses de modelos subjacentes constituídos a partir dos dados coletados e/ou oriundos de

discussões preliminares. Além da ferramenta supracitada, pretende-se subsidiar o cursista com outras tecnologias que também possam integrar as atividades práticas experimentais, mesmo que de forma exclusivamente virtual.

OBJETIVOS:

O objetivo geral deste curso é contribuir com a formação acadêmica de estudantes/egressos de licenciatura em física e a formação continuada de docentes de física. Concebendo situações que possam permitir a reflexão e a construção de estratégias pedagógicas (apoiadas em recursos digitais) que estejam em consonância com uma participação mais ativa de seus estudantes durante as aulas práticas experimentais.

Como objetivos específicos, o presente curso tem por finalidade:

- compreender os conceitos básicos de funcionamento da plataforma de prototipagem Arduino (físico/virtual);
- compreender a eletrônica básica necessária para automatizar experimentos de física;
- compreender os aspectos físicos e o funcionamento de sensores e de periféricos;
- desenvolver códigos que permitam controlar, obter dados de sensores e periféricos e realizar modelagem;- articular experimentos de física com a plataforma de prototipagem Arduino (físico/virtual);
- planejar e executar projetos para o ensino de física com a plataforma de prototipagem Arduino (físico/virtual).

METODOLOGIA:

Como sugestão, o curso deverá ter duração de 40 horas, reservando ao menos uma aula para encontros síncronos. O material didático deverá ser hospedado na sua totalidade em uma plataforma virtual, como o Moodle Os materiais didáticos apresentados ao longo desse Produto Educacional foram escolhidos a partir de curadoria e deverão estar disponíveis para os cursistas, dentre eles, teremos: artigos, videoaulas (tutoriais de autoria própria e terceiros) e livro em versão digital. Também deverá ser utilizada uma plataforma com acesso gratuito on-line de circuitos elétricos digitais e analógicos, a qual é possível simular: a própria plataforma de prototipagem Arduino, diversos componentes eletrônicos, multímetros, sensores, entre outros, segue link: <https://www.Tinkercad.com> . Cabe ressaltar que os cursistas deverão ser assessorados, tanto nos encontros síncronos quanto pela plataforma virtual ao longo de todo o curso.

CONTEÚDO PROGRAMÁTICO:

- Ambientação no espaço virtual.
- Papel do computador no processo ensino e aprendizagem.
- Ambientação no espaço virtual Tinkercad.
- Eletrônica básica.
- Programação básica. Comandos básicos e variáveis.

- Utilizando sensores/componentes e a plataforma de prototipagem Arduino.
- Automatizando experimentos de física.
- Modelagem e o ensino de física.
- Desenvolvimento de projeto final.

AVALIAÇÃO:

A sugestão é que a avaliação seja contínua e realizada por meio de atividades propostas e desenvolvidas com os recursos das plataformas digitais disponibilizadas aos cursistas, com nota para este item de até 4,0 pontos. A sugestão é que no final do curso aconteça uma apresentação de um projeto elaborado individualmente ou em grupo, versando sobre uma aplicação da plataforma de prototipagem Arduino articulada com o laboratório didático de física, sendo a nota máxima de 6,0 pontos para este item. Para aprovação será necessário a nota final igual ou superior a 6,0 pontos e realização de no mínimo 75% das atividades assíncronas.

BIBLIOGRAFIA BÁSICA:

HAUGEN, A. J.; MOORE, N. T. A model for including Arduino microcontroller programming in the introductory physics lab. **Physics Education**, New York, v.1. p.1-11, 2014.

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

RICARDO, E. C. Problematização e Contextualização no Ensino de Física. In: CARVALHO, A. M. P. de (ed.). **Ensino de Física - Coleção Idéias em Ação**. São Paulo: Cengage Learning, 2010.

VALENTE, J. A. Pesquisa, comunicação e aprendizagem com o computador. O papel do computador no processo ensino-aprendizagem. In: ALMEIDA, M. E. B.; MORAN, J. M. **Integração das Tecnologias na Educação: Salto para o futuro**. Brasília: Ministério da Educação, Seed, 2005, p. 22-31 (Tópico 1.3). Disponível em: <http://portal.mec.gov.br/seed/arquivos/pdf/1sf.pdf>. Acesso em: 20 fev. 2021.

BIBLIOGRAFIA COMPLEMENTAR:

CAVALCANTE M. A.; TAVOLARO, C. R. C.; MOLISANI, E. Física com Arduino para iniciantes, **Revista Brasileira de Ensino de Física**. São Paulo, v. 33, n. .4, 2011.

CORRALLO, M. V.; JUNQUEIRA, A. C.; SCHULER, T. E. Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 35, n. 2, p. 634-659, set. 2018.

CROVADOR, Álvaro. **Física Aplicada à Robótica**. Curitiba: Contentus, 2020.

JESUS, V. L. B. **Experimentos e Videoanálise – Dinâmica**. 1ª ed. São Paulo: Livraria da Física, 2014.

MONK, S.. **Projetos com Arduino e Android**: use seu smartphone ou tablet para controlar o Arduino. Porto Alegre: Bookman, 2014.

MONK, S. **30 projetos com Arduino**. Porto Alegre: Bookman, 2014.

MOURÃO, O. **Arduino & ensino de Física**: automação de práticas experimentais. Tianguá: Clube dos Autores, 2018. Disponível em: <https://ifce.edu.br/sobral/campus-sobral/cursos/posgraduacoes/mestrado-1/mnpef/arquivos/5-produto-educacional-arduino-e-o-ensno-de-fisica.pdf> . Acesso em: 22 fev. 2021.

OLIVEIRA, R. **Informática Educativa**: dos planos e discursos à sala de aula. 17^a ed., Campinas: Papirus, 2015.

PAPERT, S. **A Máquina das Crianças**: repensando a escola na era da informática. Porto Alegre: Artmed, 2008.

SILVA, R. B. *et al.* Estação meteorológicas de código aberto: um projeto de pesquisa e desenvolvimento tecnológico. **Revista Brasileira de Ensino de Física**, São Paulo, v. 37, n. 1, 2015.



O Produto Educacional traz uma versão similar à utilizada no ambiente virtual de aprendizagem – Moodle. As semanas foram separadas em abas (1...10) e, em cada semana, os materiais foram separados por tarjas, como descrito no quadro 1.

Quadro 1: Significado das tarjas utilizada ao longo do curso de extensão.

“**Para assistir**” refere-se à indicação de vídeos.

“**Para ler** - fundamentação teórica de física”

“**Para ler** - aspectos técnicos”.

“**Para ler** - aspectos pedagógicos”

“**Para fazer**” - sugestão de atividades para serem aplicadas ao longo do curso.

“**Para saber mais**” - indicação de materiais complementares.

“**Tarja mesclada em preto e branco**” indica o início da semana

Sugere-se ainda que em todas as semanas tenha um fórum referente aos assuntos tratados para a postagem de dúvidas.



SEMANA 1 – APRESENTAÇÃO DO CURSO

Nesta seção você encontra uma apresentação (<16 min) das potencialidades do Arduino, por Massino Banzi (co-fundador do Arduino) e as atividades propostas para a semana 1, disponível no link <https://youtu.be/UoBUXOOdLXY>. Depois de assistir o vídeo, pede-se a realização da tarefa pré-curso. Na sequência aconselha-se a leitura do material sobre lógica de programação e a realização da tarefa de



elaboração de um algoritmo sobre coleta de dados. Por fim, sugere-se a realização do cadastro na plataforma Tinkercad e a leitura do tutorial de postagem de tarefa via Tinkercad.

Bons estudos!



Para fazer

Tarefa pré-curso

Conte um pouco sobre o seu objetivo em se matricular em nosso curso. (Aqui sugere-se solicitar ao cursista o envio de um pequeno texto sobre suas intenções na realização do curso. Talvez possa ser utilizado para um redirecionamento das atividades e calibração na complexidade das atividades.)



Para ler – aspectos técnicos

Vamos compreender a lógica de programação

Para iniciarmos, vamos entender o que é um algoritmo, e assim entendendo como "escrever um código" para "rodar" no Arduino.

O que é algoritmo?

Vamos começar entendendo por meio de um exemplo do nosso cotidiano. O algoritmo está presente na receita de um bolo. Para fazer o bolo devemos seguir uma sequência de ações para alcançar o objetivo pretendido, que é fazer o bolo.

De maneira análoga, isso se aplica ao Arduino. Para construir um projeto precisamos indicar os passos ao Arduino para que o circuito funcione. Nesse caso, vale ressaltar que se faltar um 'ponto e vírgula' no final da linha do código, o código dará erro. Igualmente no bolo, se não colocarmos o fermento, o bolo não crescerá. Portanto, é importante saber ler o código para perceber, por exemplo porque o LED não está acendendo verificar se os pinos estão conectados corretamente.

Elaboração de Algoritmos⁴:

1. Identificar o problema, no âmbito em como resolver o problema.
2. Identificar as “entradas do sistema”, ou seja, quais informações serão fornecidas.
3. Identificar a “saída do sistema”, ou seja, como será disponibilizado o resultado.
4. Definir os passos a serem realizados, desenvolvendo uma sequência de passos que leve a solução do problema, e serão convertidas em entradas nas saídas).
5. Identificar as regras e limitações do problema;
6. Identificar as limitações do computador;
7. Determinar as ações que serão realizadas pelo computador.

⁴ O texto foi inspirado em aula 7 proferida no CURSO FIC – CIRCUITOS & PROGRAMAÇÃO COM ARDUINO, ministrado pelo Prof. Danilo Henrique Santos, docente no IFSP Campus de Registro em dezembro de 2020.

8. Registrar os comandos, que será a forma de representar os algoritmos.
9. Teste da solução, execute cada passo do algoritmo, seguindo a sequência estabelecida.

Exemplo de uma atividade doméstica: Como trocar uma lâmpada?

Abaixo segue um exemplo dos passos a serem seguidos para a troca de uma lâmpada.

1. Verificar se o interruptor está desligado ou chave geral;
2. Pegar uma escada;
3. Pegar a lâmpada nova;
4. Posicionar a escada logo abaixo da lâmpada;
5. Subir na escada levando a lâmpada nova;
6. Retirar a lâmpada velha;
7. Colocar a lâmpada nova;
8. Descer da escada;
9. Testar a lâmpada nova utilizando o interruptor;
10. Descartar de forma adequada a lâmpada velha.

O que é Codificação?

O processo de codificação é quando o algoritmo é transformado em códigos da linguagem de programação escolhida para se trabalhar (ASCENCIO; CAMPOS, 2012).



Para fazer

Tarefa 1: Elaborar um algoritmo para a coleta de dados de um experimento de física.

Um algoritmo destina-se a resolver um problema, ou seja, fixar um padrão de comportamento a ser seguido. Para esta tarefa, você precisa elaborar o algoritmo sobre como coletar dados de um determinado experimento.

A escolha sobre o tipo de experimento é livre. O que será avaliado é o procedimento em relação a coleta de dados.

Boa atividade!

Cadastro no Tinkercad

Você terá a oportunidade em se cadastrar como educador e/ou aluno ou criar uma conta pessoal. Para tanto, siga os procedimentos, aqui daremos o exemplo de uma conta pessoal que contemplará a criação de projetos.

- Acesse, em seu navegador a URL www.tinkercad.com.
- Clique em "Inscrever-se" no canto superior direito, como indicado na imagem abaixo. Mas se você já tem conta no Tinkercad ou na Autodesk, basta clicar em "Entrar".
- Ao clicar em se inscrever abrirá a primeira tela do seu cadastro, você deverá escolher a opção "Criar uma conta pessoal", e clique em avançar.
- Na sequência, sugiro que escolha "Entrar com o Google".
- O próximo passo será aceitar os termos de serviço do Tinkercad.
- Se você seguiu os passos acima, a sua conta está criada.



SEMANA 2

Na semana 2, sugere-se a leitura dos materiais disponíveis na seção "Para ler". Sendo que os dois primeiros itens tratam de uma abordagem técnica do Arduino e seus componentes. Os dois itens seguintes trazem uma discussão pedagógica sobre as tecnologias no ensino. Já o item "Atividade Blink" é uma proposição de construção de um pequeno *sketch* (programa) no Tinkercad. Na seção "Para fazer", você encontra um espaço para postagem da "Atividade Blink" e outro espaço para a postagem das respostas das questões relacionadas ao texto: "O papel do computador no processo ensino-aprendizagem".



Para ler – aspectos técnicos

Conhecendo a placa de prototipagem Arduino

Arduino é uma plataforma de prototipagem de desenvolvimento associada com uma linguagem de programação intuitiva que se desenvolve utilizando o ambiente de desenvolvimento integrado (IDE). Foi desenvolvido na Itália em 2005, por meio da cultura *open source*, isto é, incide em dispositivos físicos de tecnologia concebidos e oferecidos pelo movimento de código aberto. Por sua rápida difusão e capacidade e adaptabilidade, tornou-se uma excelente alternativa na concepção de projetos de diversas áreas do conhecimento. McRoberts (2013, p. 24) destaca que:

A maior vantagem do Arduino em relação a outras plataformas de desenvolvimento de microcontroladores é a sua facilidade de utilização, o que permite que pessoas que não sejam de áreas técnicas possam aprender o básico e criar seus próprios projetos em um período relativamente curto. Artistas, em especial, parecem considerá-lo a maneira ideal para criar obras de arte interativas rapidamente, sem a necessidade de um conhecimento especializado em eletrônica.

Seu *designer* permite ligá-lo em diferentes sensores, como: temperatura, luz, velocidade, aceleração, força, pressão, umidade do ar entre outros. Também temos como possibilidade de saída o controle de lâmpada de LED (Light Emitting Diode), motores, alto-falantes e display, por exemplo. A interface com o computador pode ser via entrada/saída USB/WiFi/Bluetooth. É possível, também, armazenar os dados em um cartão de memória MicroSD e, posteriormente, recolher os dados, além de disponibilizá-los em servidor próprio ou a partir de Cloud Computing. Isto, por sinal, é bastante usado para a chamada Internet das Coisas (IOT – Internet of Thing), na qual o usuário pode disponibilizar seus dados ou controlar periféricos remotamente. Já existem vários serviços dessa natureza adaptados para a plataforma Arduino, sendo que alguns necessitam de pagamento; no entanto, temos alguns serviços que podem ser contratados, para pequenos projetos, sem custo aos usuários não comerciais. Um desses serviços é o site Thingspeak, disponível em: <https://thingspeak.com> (CORRALLO, CARVALHO, SCHULER, 2018).



Figura 1: Arduino. Fonte: <https://www.flickr.com/photos/adafruit/5027882580>

A plataforma Arduino opera com uma camada simples de *software* implementado na placa, que é um *bootloader* (é uma espécie de sistema operacional que roda no microcontrolador AVR, e que permite a gravação em memória Flash as rotinas geradas pelo Arduino IDE e compiladas pela biblioteca AVR-GCC). A interface pode ser feita por diversos *softwares*, geralmente *open source*, como: Processing (bastante interessante para a confecção de gráficos em tempo real, disponível para download em <https://processing.org>), Arduino IDE (Integrated Development Environment, interface, mais utilizado para programação do Arduino, disponível em <https://www.arduino.cc>) e Scratch (disponível em <https://scratch.mit.edu>). Em Scratch se programa utilizando blocos que se encaixam, tornando-se mais lúdica a programação, enquanto em Arduino IDE e Processing se utiliza uma linguagem baseada em C++, na qual o processo de programação é mais “duro” e requer conhecimento da linguagem e lógica de programação. A versão mais popular do Arduino é o Uno que está baseada no microcontrolador ATmega 328P, 8-bit, 16MHz, CPU com 2KB de memória RAM para execução das rotinas e 32KB de memória Flash para armazenar as rotinas (SOUZA *et al.*, 2011). Vale ainda mencionar que as bibliotecas, que permitem conectar sensores e periféricos ao Arduino, podem ser facilmente encontradas na Web (CORRALLO, CARVALHO, SCHULER, 2018).

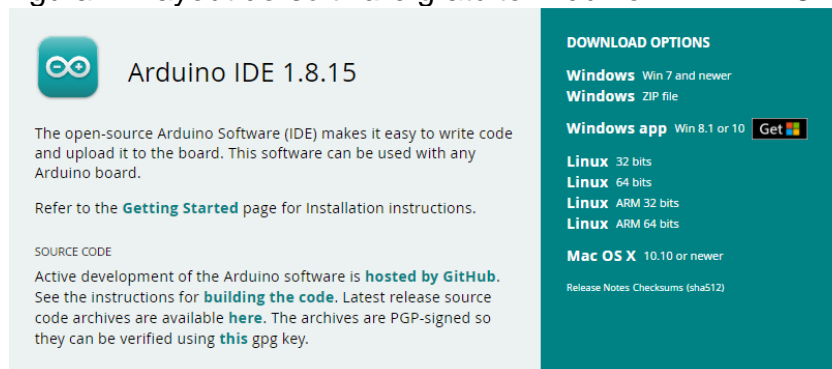
De maneira resumida, a placa de prototipagem Arduino Uno possui 14 pinos de entrada/saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas

analógicas, um ressonador de cerâmica de 16 MHz (CSTCE16M0V53-R0), uma conexão USB, um conector de alimentação, um conector ICSP e um botão de reinicialização.

Como se comunicar com o Arduino?

Para que o Arduino faça o que você deseja, é necessário programá-lo. Para isso, você pode utilizar o *software* gratuito Arduino IDE - v. 1.8.15 (veja figura 2), disponível em <https://www.arduino.cc/en/software>. Antes de baixar, verifique a versão adequada para seu sistema operacional.

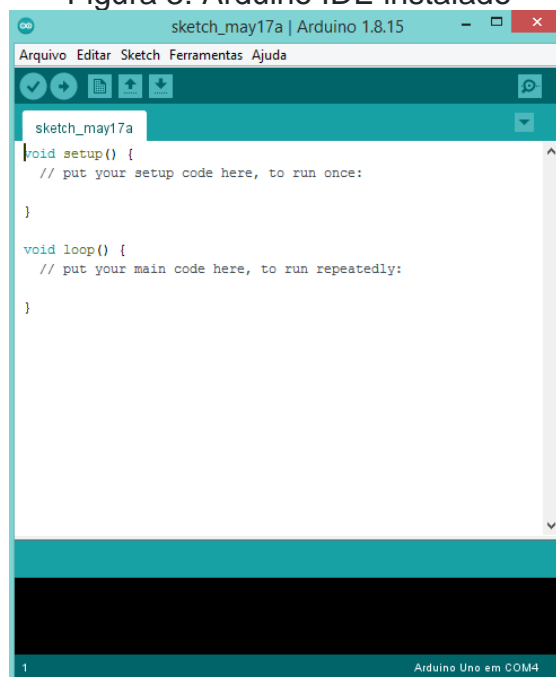
Figura 2: Layout do *software* gratuito Arduino IDE - v. 1.8.15



Fonte: <https://www.arduino.cc/en/software>

Com o Arduino IDE instalado (veja figura 3), você poderá escrever os comandos para os projetos usando uma linguagem de programação, a qual estudaremos nas próximas semanas.

Figura 3: Arduino IDE instalado

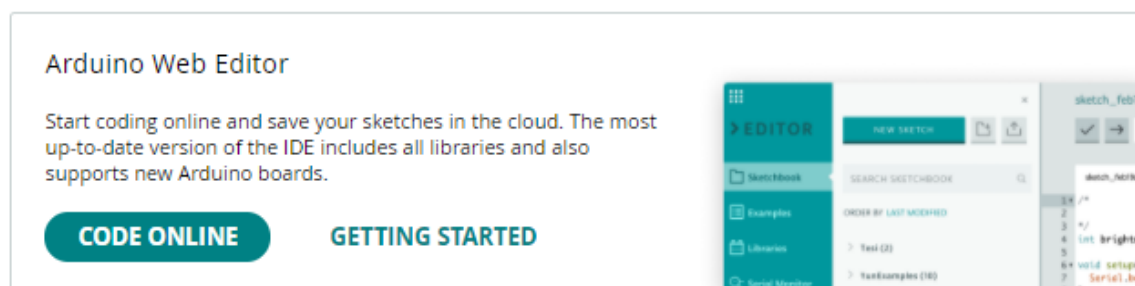


Fonte: <https://www.arduino.cc/en/software>

Arduino Web Editor

O Arduino Web Editor permite ao usuário cadastrado na plataforma seguir [link: https://www.arduino.cc/en/software](https://www.arduino.cc/en/software) "rodarem" seus *sketchs* (programas) de forma *on-line*, sem a necessidade de instalação do Arduino IDE no PC.

Figura 4: Layout Arduino Web Editor



Fonte: <https://www.arduino.cc/en/software>

Protoboard

É uma placa de plástico, também conhecida como matriz de contato, em que os orifícios são conectados internamente por trilhas metálicas. Os orifícios servem para a inserção de terminais de componentes e fios de conexão.

Listagem de componentes utilizados por usuários iniciantes

Abaixo segue listagem com os principais componentes e quantidades utilizados em projetos por usuários iniciantes. Vale mencionar que é possível encontrar no mercado *kits* para os diferentes estágios dos usuários.

Quadro 2: listagem com os principais componentes.

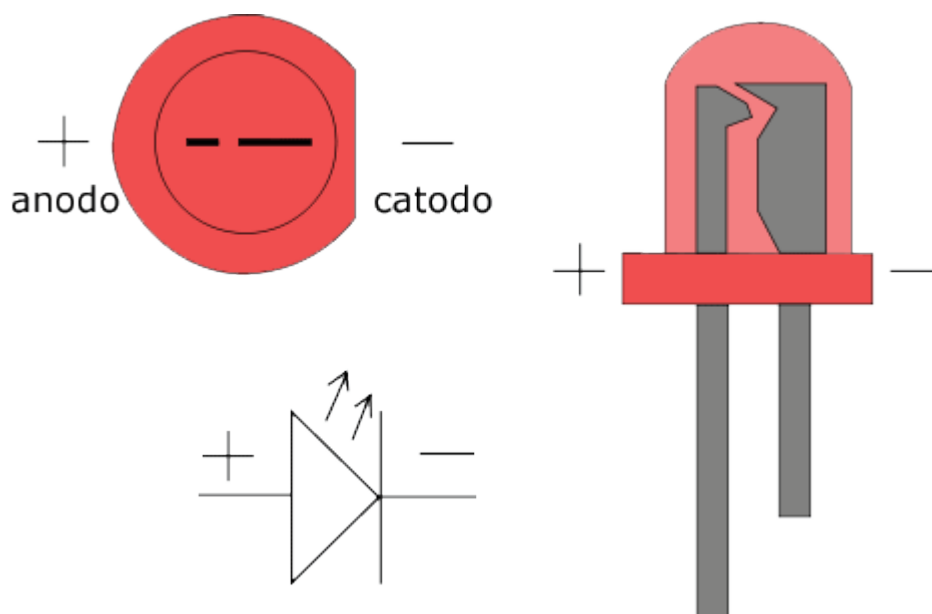
01 - Placa Uno R3	15 - Resistor 330Ω
01 - Cabo USB 30cm	05 - Resistor 1KΩ
01 - Protoboard 400 Pontos	05 - Resistor 10KΩ
30 - Jumper Macho-Macho	04 - Diodo 1N4007
01 - Micro Servo 9g SG90	01 - Potenciômetro 10KΩ
TowerPro	04 - Capacitor Cerâmico 10nF
01 - Sensor de Temperatura NTC	04 - Capacitor Cerâmico 100nF
01 - Sensor de Luz LDR	02 - Capacitor Eletrolítico 10uF
01 - Buzzer Ativo 5V	02 - Capacitor Eletrolítico 100uF
05 - LED Vermelho	05 - Chave Táctil (Push-Button)
05 - LED Amarelo	
05 - LED Verde	

Fonte: autoria própria.

Conhecendo o LED

O Diodo Emissor de Luz (LED = Light Emitting Diode) é um componente eletrônico emissor de luz baseado em material semicondutor. O LED é um componente bipolar, significando que ele conduz corrente elétrica apenas num sentido. Para descobrirmos qual a polarização correta para ligá-lo a uma fonte e acendê-lo, devemos olhar seus terminais. O maior, ou curvado, é o anodo (positivo) e o menor é o catodo (negativo). Na figura 5 encontramos a representação esquemática do LED.

Figura 5: Representação esquemática do LED



Fonte: Autor Desconhecido - licenciado em [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/)

Para não “queimarmos” um LED (a voltagem máxima é diferente para cada tipo de LED, mas não deve ser menor que 2V e muito maior que 3V), devemos ligá-lo em série com um resistor R. Para sabermos o valor mínimo desta resistência, usamos a Lei de Ohm. Suponha que o LED suporte até 2 V de tensão e corrente de 10 mA = 0,010 A e seja ligado em uma fonte de 5 V (geralmente a saída do Arduino). Se ele suporta apenas 2 V, então 3 V da fonte deve ir para o resistor R.

$$V_{res} + V_{LED} = 5 \text{ V} \rightarrow V_{res} = 5 \text{ V} - 2 \text{ V} = 3 \text{ V}$$

Pela Lei de Ohm, o valor mínimo da resistência deve ser:

$$V_{res} = R \cdot i \rightarrow R = V_{res} / i = 3 / 0,01 = 300 \ \Omega$$



Para ler – aspectos pedagógicos

Sugestão de leitura

VALENTE, J. A. Pesquisa, comunicação e aprendizagem com o computador. O papel do computador no processo ensino-aprendizagem. *In*: ALMEIDA, M. E. B.; MORAN, J. M. **Integração das Tecnologias na Educação**: Salto para o futuro. Brasília: Ministério da Educação, Seed, 2005, p. 22-31 (Tópico 1.3). Disponível em: <http://portal.mec.gov.br/seed/arquivos/pdf/1sf.pdf> . Acesso em: 16 nov. 2021.

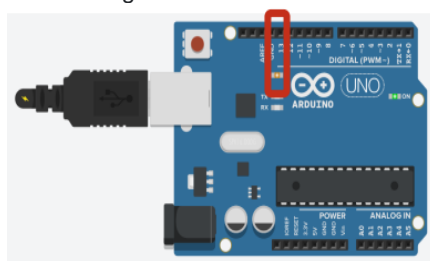
O artigo traz uma análise sobre as questões técnicas e pedagógicas da informática na educação, apresentando os grandes desafios dessa combinação do técnico com o pedagógico, em especial, para o professor saber orientar e desafiar o aluno em atividades computacionais. Sugere-se a leitura do texto da página 22 até 31 - Tópico 1.3.

Para ler – aspectos técnicos

Blink

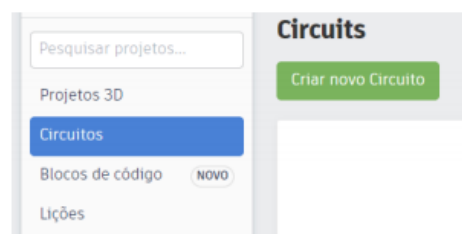
Blink é um exemplo bastante simples de uso do Arduino. Com ele podemos controlar o LED interno do Arduino figura 6, sem montagem de circuito, a partir da porta digital 13. Para iniciar, abra sua conta do Tinkercad.com e clique em “Circuitos”, no menu à esquerda da tela figura 7. Em seguida, clique no botão “Criar novo Circuito”.

Figura 6: LED interno do Arduino



Fonte: imagem gerada via plataforma www.tinkercad.com.

Figura 7: menu à esquerda no Tinkercad



Fonte: imagem gerada via plataforma www.tinkercad.com

Figura 8: Componentes Básico no Tinkercad.

Na tela seguinte, você encontrará, no menu à direita, “Componentes Básico”, localize o Arduino UNO R3 (fig. 8) e arraste-o para o lado esquerdo da tela.



Fonte: imagem gerada via plataforma Tinkercad.com.

Em seguida, clique em “Código” e escolha “Texto”, conforme a figura 8. Na caixa de texto, transcreva o código do quadro 3.

Fonte: imagem gerada via plataforma Tinkercad.com

Quadro 3– Código blink

```
//blink – controle do LE interno do Arduino,.  
Int pino = 13;
```

```

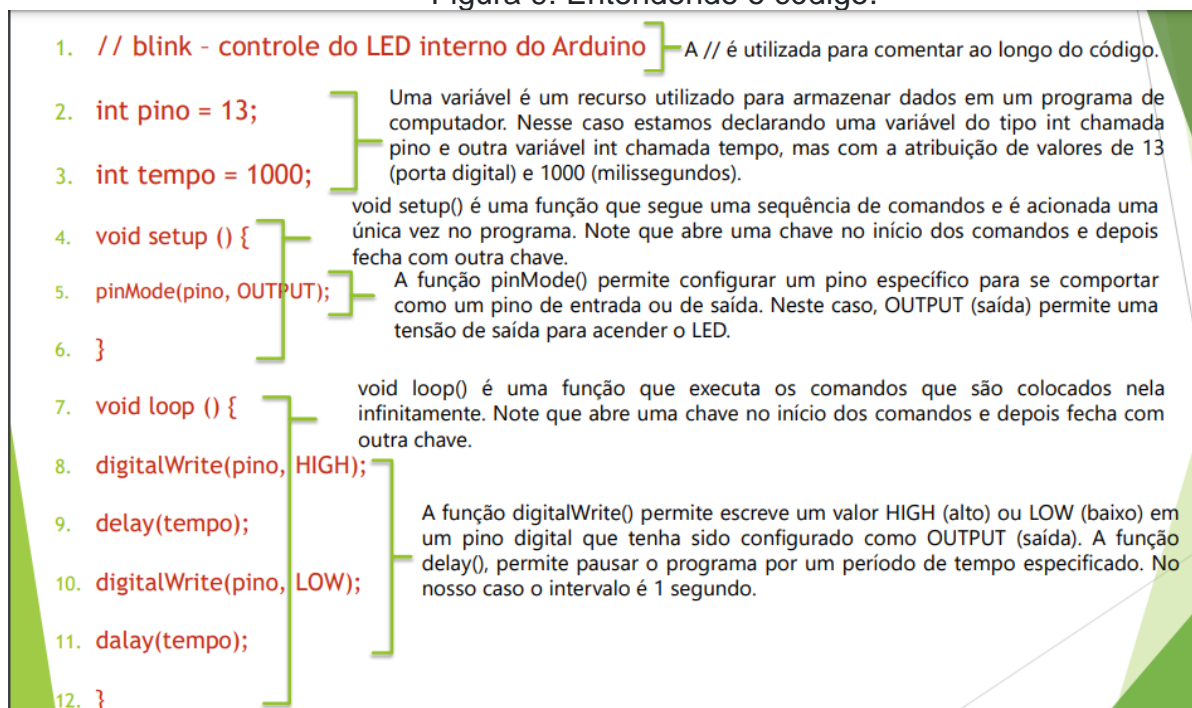
Int tempo =1000;
void setup () {
    pinMode (pino, OUTPUT);
}
void loop() {
digitalWrite(pino, HIGH);
    delay (tempo);
    digitalWrite(pino, LOW);
    delay (tempo);
}

```

Fonte: adaptado do exemplo Blink, disponível em IDE Arduino.

Por fim, clique em “Iniciar simulação” e observe o LED interno do Arduino. Sugere-se realizar alterações no intervalo de tempo de espera.

Figura 9: Entendendo o código.



1. `// blink - controle do LED interno do Arduino` } A // é utilizada para comentar ao longo do código.

2. `int pino = 13;` } Uma variável é um recurso utilizado para armazenar dados em um programa de computador. Nesse caso estamos declarando uma variável do tipo int chamada pino e outra variável int chamada tempo, mas com a atribuição de valores de 13 (porta digital) e 1000 (milissegundos).

3. `int tempo = 1000;` }

4. `void setup () {` } void setup() é uma função que segue uma sequência de comandos e é acionada uma única vez no programa. Note que abre uma chave no início dos comandos e depois fecha com outra chave.

5. `pinMode(pino, OUTPUT);` } A função pinMode() permite configurar um pino específico para se comportar como um pino de entrada ou de saída. Neste caso, OUTPUT (saída) permite uma tensão de saída para acender o LED.

6. `}` }

7. `void loop () {` } void loop() é uma função que executa os comandos que são colocados nela infinitamente. Note que abre uma chave no início dos comandos e depois fecha com outra chave.

8. `digitalWrite(pino, HIGH);` }

9. `delay(tempo);` }

10. `digitalWrite(pino, LOW);` } A função digitalWrite() permite escreve um valor HIGH (alto) ou LOW (baixo) em um pino digital que tenha sido configurado como OUTPUT (saída). A função delay(), permite pausar o programa por um período de tempo especificado. No nosso caso o intervalo é 1 segundo.

11. `dalay(tempo);` }

12. `}` }

Fonte: os autores.



Para assistir

Vídeo de 39 segundos demonstrando no Arduino físico (fig.10), o exemplo Blink, no LED embutido da porta 13 – link: https://youtu.be/q-2_7ijDOck

Figura 10 - Imagem do experimento capturada pelo vídeo.



Fonte: os autores.



Para fazer

Atividade Blink

Na "Atividade Blink" (descrito no tópico anterior "Blink") sugere-se a inserção de intervalos de tempo distintos, como apresentado no vídeo da semana 2.



Para ler – aspectos pedagógicos

Questões referentes ao texto: "O papel do computador no processo ensino-aprendizagem" p.22-31

Sugere-se a leitura do artigo "O papel do computador no processo ensino-aprendizagem", p.22-31, e relacione os aspectos pedagógicos relevantes durante a leitura do texto, disponível no link: <http://portal.mec.gov.br/seed/arquivos/pdf/1sf.pdf>
Acesso em: 17 nov. 2021



SEMANA 3

Nesta seção você encontra indicação de leitura/consulta sobre eletricidade. Na seção "Para fazer" tem a atividade "Simulando semáforos no Tinkercad" e outra sobre modelagem no ensino de física.



Para ler - fundamentação teórica de física



1. Onde não está a eletricidade?
2. Pondo ordem dentro e fora de casa
3. Elementos dos circuitos elétricos
4. Cuidado! É 110 ou 220?
5. A conta de luz
6. Exercícios

Figura 11 – Layout da página do GREF – eletromagnetismo.

Para uma revisão sobre eletricidade e as grandezas físicas envolvida, sugestão dos capítulos 1, 2 e 3 do texto do GREF. Disponível em: <http://www.if.usp.br/gref/eletro/eletro1.pdf>. Acesso em: 01 nov. 2021.

Para ler – aspectos pedagógicos

Problematização e Contextualização no Ensino de Física

Sugestão de texto para leitura.

RICARDO, Elio Carlos. Problematização e contextualização no ensino de física. **Ensino de Física**. São Paulo: Cengage Learning, p. 29-48, 2010.

CÓDIGO DE CORES - RESISTORES

Tabela 1: valores dos resistores

Cor	Dígito	Multiplicador	Tolerância
Prata	-	x 0,01	± 10%
Dourado	-	x 0,1	± 5%
Preto	0	x 1	-
Marrom	1	x 10	± 1%
Vermelho	2	x 100	± 2%
Laranja	3	x 1K	-
Amarelo	4	x 10K	-
Verde	5	x 100K	± 0,5%
Azul	6	x 1M	± 0,25%
Violeta	7	x 10M	± 0,1%
Cinza	8	-	± 0,05%
Branco	9	-	-

Fonte:

https://commons.wikimedia.org/wiki/File:Tabela_de_cores_de_um_resistor.jpg

Esta calculadora de código de cores de resistor é desenvolvida para encontrar os valores de resistência e tolerância, segue link: [:https://br.mouser.com/technical-resources/conversion-calculators/resistor-color-code-calculator](https://br.mouser.com/technical-resources/conversion-calculators/resistor-color-code-calculator)

Os resistores são dispositivos eletrônicos que, limitando a intensidade, conseguem resistir a corrente elétrica. Assim, ela pode transformar energia elétrica em energia térmica, fenômeno que recebe o nome de efeito joule.

Resistores não possuem um polo positivo ou negativo e você não precisa se preocupar em qual posição o resistor estará no seu circuito em série, pois ele funcionará da mesma forma.

Primeira e segunda lei de Ohm

As leis de Ohm permitem calcular importantes grandezas físicas, como a tensão, corrente e a resistência elétrica dos mais diversos elementos presentes em um circuito.

Corrente (i) - Tensão (V) elétrica - Resistência (R). *A relação entre eles é dado pela Lei de Ohm.*

Figura 12: Quadro resumo leis de Ohm.

Primeira lei de Ohm - permite calcular importantes grandezas físicas, como a tensão, corrente e a resistência elétrica dos mais diversos elementos presentes em um circuito.

V → tensão (V)

R → resistência elétrica (Ω)

I → corrente elétrica (A)

$$V = Ri \quad \text{equação 1}$$

Segunda Lei de Ohm - relaciona as características físicas que determinam a resistência de um condutor, ou seja, um condutor depende do tipo do material do qual é construído, de suas dimensões e da temperatura, conforme as equações 2 e 3.

R₂₀ → resistência a 20°C (Ω)

α₂₀ → coeficiente de temperatura

T → temperatura (T)

ρ → resistividade elétrica (Ω.m)

A → área da seção transversal (m²)

L → comprimento (m)

$$R = R_{20}(1 + \alpha_{20}(T - 20)) \quad \text{equação 2}$$

$$R = \frac{\rho L}{A} \quad \text{equação 3}$$

Ligação em série- Na ligação em série, a resistência equivalente é igual à soma das resistências. $R_{EQ} = R_1 + R_2 + R_3 + \dots + R_N$

Ligação em paralelo- A associação em paralelo é obtida quando os resistores são ligados de modo que a **corrente elétrica divide-se ao passar por eles**. Nesse tipo de associação, a resistência elétrica equivalente será sempre menor do que a menor das resistências.

Soma do inverso das resistências individuais:

$$\frac{1}{R_{EQ}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots + \frac{1}{R_N}$$
$$R_{EQ} = \frac{R_1 R_2}{R_1 + R_2} \quad \text{Calculo de dois resistores em paralelo}$$

Fonte: os autores.

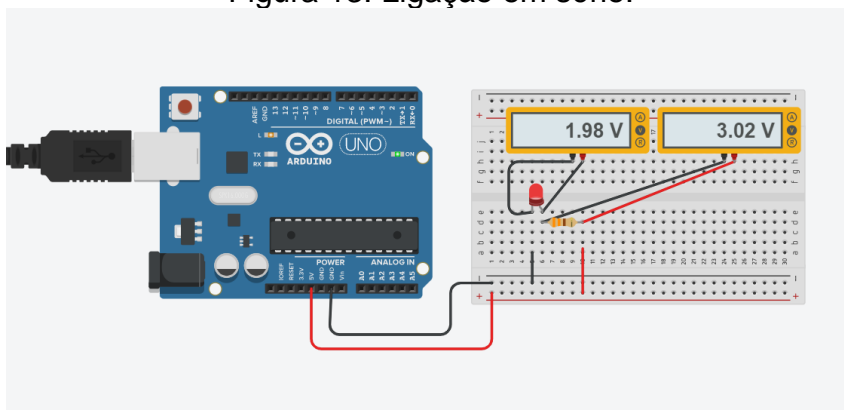
Associação de resistor no Tinkercad

Como visto na semana 2, a associação em série pode ser usada para proteção de componentes em um circuito. Na descrição a seguir veremos como montar uma associação de um LED e resistor em série, e observar a voltagem a partir de voltmímetro ligados em paralelo aos componentes do circuito.

Note na figura 14 que o LED está associado em série com um resistor (código de cor: laranja, laranja e marrom), isto é, 330 ohms. Também é possível observar que os

voltímetros estão associados em paralelo aos componentes. A leitura de cada voltímetro se aproxima de 2,0 V e 3,0 V, respectivamente, permitindo que o LED esteja protegido pelo resistor limitador. Cabe ainda mencionar que realização desta montagem não necessita de um código de programação para sua simulação, bastando, portanto, estabelecer a ligação adequada e, em seguida, clicar em, no menu de controle do Tinkercad.

Figura 13: Ligação em série.



Fonte: imagem gerada via plataforma Tinkercad.com



Para assistir

O vídeo apresenta uma descrição e sugestão de montagem de semáforo do modelo inglês e do modelo utilizado no Brasil. Disponível no link <https://youtu.be/2WOjIX34Ek>. Acesso em 10 jan. 2022.



Para fazer

Como atividade da semana, pede-se a montagem (circuito e código) no Tinkercad de:

- um semáforo do modelo inglês (veja figura 14)

Figura 14: Semáforo modelo inglês.



Fonte: autor desconhecido está licenciado em CC BY-NC-ND.

b) e outro do modelo utilizado no Brasil, conforme descrito no vídeo presente na seção "Para assistir"

Dica: para a montagem do circuito, recomenda-se utilizar a simulação apresentada na proposta "Associação em série e paralelo no Tinkercad", já para a montagem do código de programação, sugere-se retornar à "Atividade Blink" da Semana 2.

Após a montagem dos semáforos (circuito e código), você deve ter um local para que o aluno COMPARTILHE OS LINKS no espaço "Simulando semáforos no Tinkercad".

IMPORTANTE: No Tinkercad, para compartilhar o link não basta copiar a URL que aparece no navegador. Você deve clicar no botão "COMPART.", em seguida "Convidar pessoas" e então copiar o link que aparece. Salve estes os links no espaço indicado para a entrega.

c) A partir da leitura do texto sugerido na seção "Para Ler – aspectos pedagógicos", exemplifique como a modelagem poderia ser implementada em suas aulas de física.



SEMANA 4

É aconselhável a leitura dos materiais disponíveis na seção "Para ler". Neste tópico apresenta-se um resumo sobre as portas digitais e analógicas, variáveis, Serial Monitor, expressões lógicas e o Código Morse. Também, sugere-se a leitura do artigo "A model for including Arduino microcontroller programming in the introductory physics lab".



Para ler – aspectos técnicos

Entendendo as portas digitais

Funções em linguagem de programação são pequenos blocos de instruções ou procedimentos que compõem o programa principal, são escritas para realizar uma tarefa de modo que se possa ser reutilizada, dessa maneira o código do programa fica menor e mais fácil de entender.

Função `setup()` é executada no início do código, tudo que estiver nesta função é executado apenas uma vez. Insere-se aqui as configurações iniciais do programa.

Função `loop()` é uma função que fica em constante execução. Dentro desta função fica o código principal do programa.

Alguns comandos:

- O comando `pinMode` indica a configuração do pino que queremos usar como modo entrada ou saída, isto é, (INPUT, OUTPUT). É importante ressaltar que o default dos pinos digitais é como INPUT. Recomenda-se sempre declarar tanto INPUT quanto OUTPUT. Como corrente elétrica nos pinos temos no máximo 40 mA.

Exemplo:

```
void setup() {  
  pinMode(7, INPUT); // o pino 7 está definido no modo de entrada.  
  pinMode(8, OUTPUT); // o pino 8 está definido no modo de saída.  
}
```

- O comando `digitalWrite` permite enviar para os pinos informados níveis de tensão, que podem ser alto/ligado (HIGH) ou baixo/desligado (LOW).

- O comando `digitalRead`() é utilizado para ler o estado de um pino digital. O `digitalRead`() recebe os parâmetros HIGH ou LOW.

Exemplo:

```
digitalRead(10); //Lê a porta digital 10 (HIGH OU LOW)
```

- O comando `delay`() é utilizado quando necessita de uma pausa de determinado tempo no programa.

É importante destacar que o Arduino executa as instruções linha a linha. Após executar uma instrução, ele passa para a próxima, na linha seguinte. O sinal de “;” (ponto-e-vírgula) avisa que o comando da linha acabou.

Exemplo:

```
void loop() {  
  digitalWrite(13, HIGH);  
  delay(1000);  
  digitalWrite(13, LOW);  
  Delay(1000);  
}
```

É importante destacar que a função `loop`() se repete indefinidamente.

Figura 15: programação: códigos de comando.

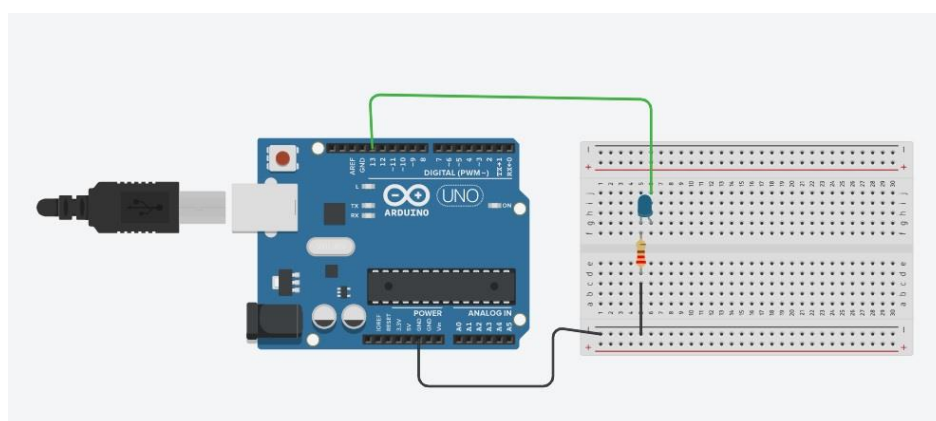
```
Definições {
  Void setup () {
    // inicializa o pino digital como uma saída.
    // O pino 13 tem um LED conectado na maioria das
    placas Arduino:
    pinMode (13,OUTPUT); (Pino de saída)
  }

Laço: Repete indefinidamente {
  Void loop () {
    digitalWrite (13, HIGH); // Saída para LED (HIGH = 5V)
    delay (1000); //Espera 1000ms
    DigitalWrite (13, LOW); //Saída para LED (LOW = 0 V)
    delay (1000); //Espera 1000ms
  }
}
```

Fonte: os autores.

Vamos entender cada parte de código de um LED piscando:

Figura 16: Montagem de um circuito com a ligação de um LED.



Fonte: imagem gerada via plataforma Tinkercad.com

Quadro 4: adaptado do exemplo blink, disponível em IDE Arduino.

```
void setup() // é a primeira função e é executada uma só vez.
{
  pinMode(13, OUTPUT); // é o modo de trabalho do pino 13 e você define se é
  entrada ou saída.
}
```

```
void loop()
{
  digitalWrite(13, HIGH); // aqui está mandando ligar o pino 13
  delay(1000); // Aguarde 1 segundo
  digitalWrite(13, LOW); // aqui está mandando desligar o pino 13
  delay(1000); // Aguarde 1 segundo
}
```

Fonte: os autores

Importante: o Arduino não lê nada após as duas barras (//), ou seja, você pode comentar o programa. Já as chaves iniciam e encerram uma função.

Tipos de Variáveis no IDE Arduino

Variáveis são expressões que você pode usar em programas para armazenar valores, como, por exemplo, a leitura de um sensor a partir de um pino analógico. As variáveis devem ser declaradas informando seu tipo e nome (identificação da variável).

Tabela 2: Tipos de Dados.

Tipos de dados	Definição	RAM	Intervalo numérico
Void	Indica tipo indefinido. Usado geralmente para informar que uma função não retorna nenhum valor.	N/A	N/A
Boolean	Os valores possíveis são true (1) e false (0). Recomenda-se utilizar o bool.	1 Byte	0 a 1 (false)
Char	Ocupa um byte de memória. Pode ser uma letra ou um número. A faixa de valores é de -128 a 127.	1 Byte	-128 a 127
Unsigned char	O mesmo que o char, mas a faixa de valores válida é de 0 a 255.	1 Byte	0 a 255
Byte	Ocupa 8 bits de memória. A faixa de valores é de 0 a 255.	1 Byte	0 a 255
Int	Armazena números inteiros e ocupa 16 bits de memória. A faixa de valores é de -32.768 a 32.767.	2 Bytes	-32.768 a 32.767
Unsigned int	equivalente ao int, mas a faixa é de 0 a 65.535.	2 Bytes	0 a 65.535
Word	Equivalente ao unsigned int	2 Bytes	0 a 65.535
Long	Armazena números até 32 bits. A faixa de valores é de -2.147.483 a 2.147.483.647.	4 Bytes	-2.147.483 a 2.147.483.647.
Unsigned long	Equivalente ao long.	4 Bytes	0 até 4.294.967.295.
Short	Armazena número de até 16 bits	2 Bytes	-32.768 a 32.767
Float	Armazena valores de ponto flutuantes (reais) e ocupa 32 bits	4 Bytes	-3.40282235E-38 a 3.4028235E+38
Double	equivalente ao float	4 Bytes	-3.40282235E-38 a 3.4028235E+38

Fonte: os autores.

Exemplos de uso e sintaxe dos diversos tipos de variáveis podem ser encontrados no link: <https://www.arduino.cc/reference/pt/>

LED piscando S.O.S. em Código Morse

Na figura 17 temos o Código Morse, que é um sistema binário de representação a distância de números, letras e sinais gráficos, utilizando-se de sons curtos e longos, pontos e traços para transmitir mensagens. Os caracteres são representados por uma combinação específica de pontos e traços, conforme exposto na tabela acima. Para formar as palavras, basta realizar a combinação correta de símbolos. (FRANCISCO, 2021)

Figura 17: Código Morse com representação de letras e números.

A ●-	J ●---	S ●●●
B -●●●	K -●-	T -
C -●-●	L ●-●●	U ●●-
D -●●	M --	V ●●●-
E ●	N -●	W ●--
F ●●-●	O ---	X -●●-
G --●	P ●-●●	Y -●--
H ●●●●	Q --●-	Z --●●
I ●●	R ●-●	

Fonte: https://commons.wikimedia.org/wiki/File:Fig5_tipaper.jpg

A letra “S” é representada com 3 pontos curtos e a letra “O” é representada por 3 pontos longos. Representaremos estes pontos em acionamentos do LED, acendendo e apagando 3 vezes rapidamente para representar o “S” e acendendo e apagando 3 vezes lentamente para representar a letra “O”:

Figura 18: representação do SOS em Código Morse.

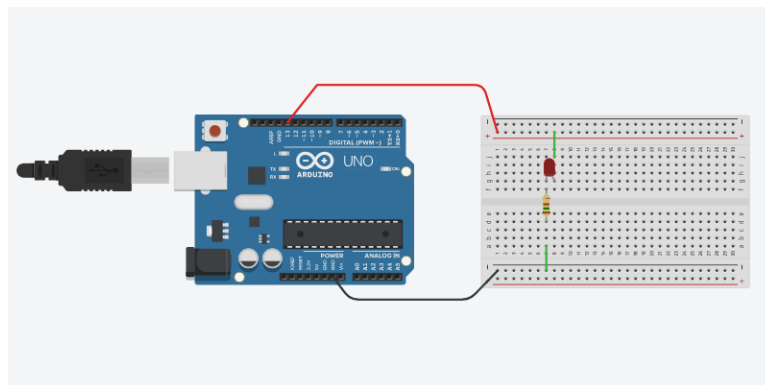


Fonte: os autores.

LED PISCANDO SOS EM CÓDIGO MORSE

Na figura 19, temos a montagem de um circuito no Tinkercad acenda um LED na porta 13 do Arduino e insira o código para ver o LED piscando SOS em código Morse.

Figura 19: Montagem de um circuito com a ligação de um LED.



Fonte: imagem gerada via plataforma TinkerCAD.com.

Quadro 5: Código para o LED piscar SOS em código Morse.

```
void setup()
{
pinMode(13, OUTPUT);
}
void loop()
{
//LETRA S
digitalWrite(13, HIGH);
delay(300);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
```



```
delay(300);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(300);
digitalWrite(13, LOW);
delay(800);

//LETRA O
digitalWrite(13, HIGH);
delay(700);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(700);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(700);
digitalWrite(13, LOW);
delay(800);

//LETRA S
digitalWrite(13, HIGH);
delay(300);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(300);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(300);
digitalWrite(13, LOW);
delay(2000);
}
```

Fonte: os autores

O que é Serial Monitor?

O Monitor Serial é uma ferramenta que auxilia o Arduino no recebimento e envio de dados para a placa sem a necessidade de recorrer a uma ferramenta externa.

Para acessar o Serial Monitor no TinkerCAD, basta clicar em



e, em seguida, no ícone , logo abaixo do código.

Serial.begin()

Esta função deve ser utilizada sempre que formos fazer a comunicação serial. Ela inicializa a porta de comunicação do Arduino com base em uma taxa de transmissão (*baud rate*) definida nesta mesma função:

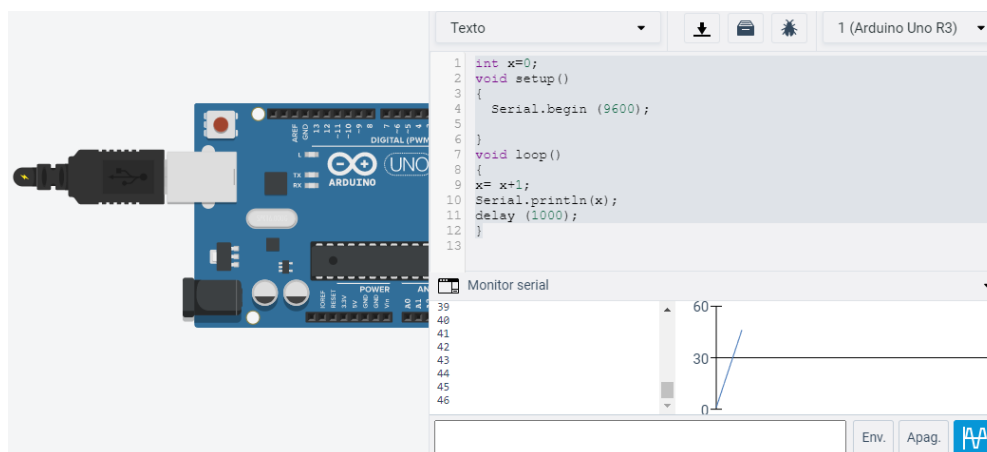
Exemplo: `Serial.begin(9600);`

É comum, por padrão, inicializarmos a comunicação serial dentro da função `Setup()`;

Utilizando o Monitor Serial

O código descrito na figura 20 permite imprimir a partir da função `Serial.println` valores sequenciados de (*x*) no Serial Monitor. Note que *x* é uma variável do tipo inteira (`int`) e inicia com valor zero. Em void loop, temos o incremento de +1 em no valor de *x* a cada passagem. Na sequência, o valor corrente é impresso com a função `Serial.println` (*x*) e ocorre a mudança de linha com a opção "ln". Note que na imagem 1 também há a construção de um gráfico (canto inferior direito) a partir dos valores de *x*.

Figura 20: Código a partir da função `Serial.println` valores sequenciados de (*x*) no Serial Monitor.



Fonte: imagem gerada via plataforma Tinkercad.com

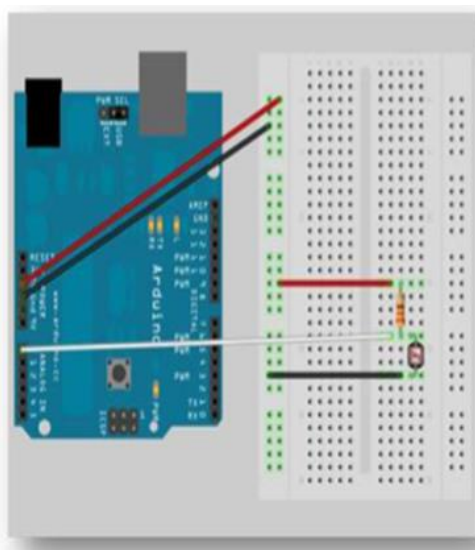
Figura 21 - Fonte: gerado a partir de <http://www.Tinkercad.com>.

No IDE Arduino, o Serial Monitor pode ser acionado pelo botão, presente na barra de ferramenta superior ou pressionando as teclas CTRL+SHIFT+M.

Usando as portas analógicas e LDR

Como exemplo de uso das portas analógica usaremos LDR e um resistor, como indicado na figura 21. O LDR estará conectado na porta analógica.

Figura 21: LDR e as portas analógicas.



Fonte: os autores.

Antes de montar o projeto, vamos discutir um pouco sobre o LDR e as portas analógicas.

O Resistor Dependente de luz (LDR = Light Dependent Resistor) tem sua resistência R mudada de acordo com a luz incidente sobre ele. Basicamente ele converte um sinal luminoso num sinal elétrico (tensão) que pode ser registrado pelo Arduino. A parte do LDR que é sensível à luz na verdade é a trilha de sulfeto de cádmio, como indicado na figura 23.

Figura 22: LDR – fonte: <https://pixabay.com/pt/vectors/resistor-eletr%C3%B4nico-ldr-40611/>



O LDR tem resistência típica de cerca de $0,5\text{ k}\Omega$ quando exposto à luz a valores típicos de cerca de $200\text{ k}\Omega$ quando na escuridão.

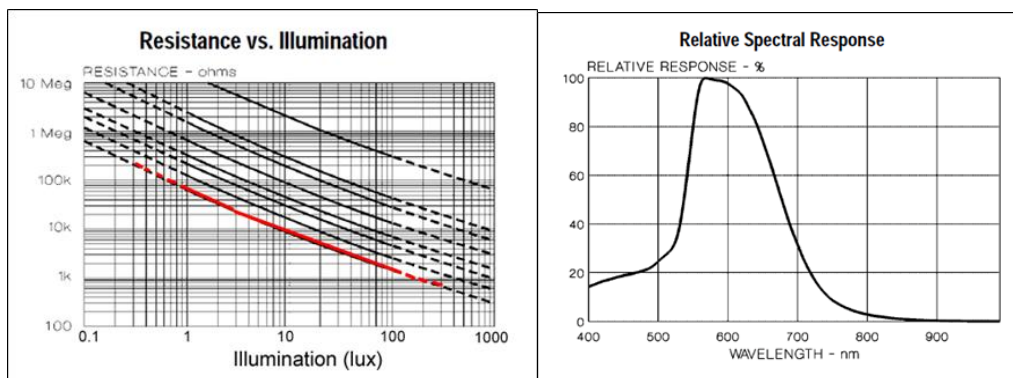
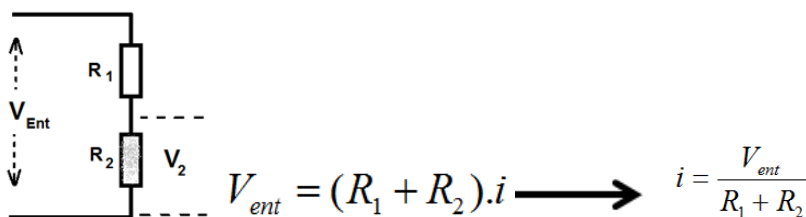


Gráfico 1 - Fonte: <https://learn.adafruit.com/photocells?view=all>

Para entender um pouco melhor o LDR no circuito, vamos lembrar o esquema de divisão de tensão. No circuito abaixo, temos uma tensão V_{ent} aplicada a dois resistores em série: R_1 e R_2 . Vamos calcular a tensão no resistor R_2 .

Pela Lei de Ohm, podemos calcular a corrente pelos resistores



E a tensão no resistor 2 é uma fração de V_{ent} dada por: $V_2 = R_2 \cdot i$

$$V_2 = \left(\frac{R_2}{R_1 + R_2} \right) \cdot V_{ent}$$

Vamos colocar no lugar de R_2 um LDR. Suponha para exemplificar que $R_1 = 10 \text{ kohm}$ e $V_{ent} = 9 \text{ V}$. Desta forma, as tensões no LDR com luz e sem luz serão:

Com luz ($R_{LDR} = 0,5 \text{ k}\Omega$) $V_{LDR} = \left(\frac{0,5}{10 + 0,5} \right) \cdot 9 = 0,43 \text{ V}$

Sem luz ($R_{LDR} = 200 \text{ k}\Omega$) $V_{LDR} = \left(\frac{200}{10 + 200} \right) \cdot 9 = 8,6 \text{ V}$

Na presença de luz, o LDR tem BAIXA tensão; na escuridão, o LDR tem ALTA tensão.



Para ler - fundamentação teórica de física

Uso do LDR: Movimento Oscilatório Amortecido (*)

Sugestão de leitura do artigo: “Um experimento simples para medir a lei do inverso do quadrado da luz em condições de luz do dia”, que trata sobre Movimento Oscilatório Amortecido. No artigo é usado um espelhinho preso a uma régua que pode

oscilar, um Laser e um LDR. O Arduino registra então a variação de tensão no LDR, refletindo o movimento amortecido da régua.

Para ler – aspectos técnicos

Portas analógicas e o Monitor Serial

Retomando o projeto, iremos usar um LDR em uma porta analógica e enviar a leitura para o Monitor Serial do IDE Arduino ou Tinkercad. Veja figura 23.

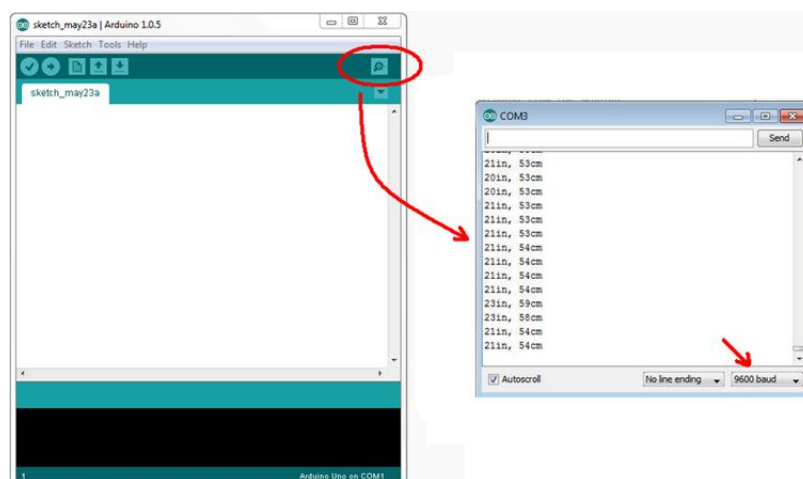


Figura 23 - IDE Arduino e Monitor Serial. Fonte: os autores.

Um pouco sobre as portas analógicas

Para discutirmos as portas analógicas, devemos entender a diferença entre informações digital e analógica.

Informações analógicas: são valores cuja variação no tempo é contínua e proporcionais a uma variável. Por exemplo, o velocímetro analógico de um automóvel (ele assume todos os valores de 0 Km/h até 100 km/h quando acelera do repouso até esta última velocidade).

Informações digitais: aqueles expressos por números binários, compostos pelos dígitos “0” e “1” (chamados bits). Por exemplo, podemos representar sinais de tensão 0 V e 5 V usando um bit de informação (havendo $2^1 = 2$ números de 1 bit):

$$0 = 0V \text{ e } 1 = 5V$$

Porém, se quisermos representar mais valores intermediários, devemos acrescentar mais bits, podendo se criar, por exemplo, números binários de dois bits (totalizando $2^2 = 4$ números de 2 bits).

00 = 0 V
01 = 1,67 V
10 = 3,34 V
11 = 5 V

E assim por diante, até chegarmos a um conjunto de 8 bits, chamado de byte. Desta forma, com uma memória de 1 byte, podemos ter disponível $2^8 = 256$ números binários.

$$T_{porta}(V) = \frac{5 * \text{valor_lido_na_porta}}{1023}$$

A porta analógica do Arduíno na verdade faz uma conversão de informação analógico (tensão) para informação digital. O conversor do Arduíno apresenta 10 bits. Desta forma, ele pode “ler” os sinais analógicos com resolução de $2^{10} = 1024$ “canais” (do canal 0 até o canal 1023).

10 bits permitem o registro de 1024 números binários: de 000000000 até 1111111111.

A função “analogRead(porta)” lê na porta especificada a tensão analógica e converte para informação digital. Assim, se fornecermos tensão de 0 V até 5 V (analógico), os valores serão convertidos como 0 V = canal 0 e 5 V = canal 1023. Desta forma, podemos fazer uma relação direta entre a tensão analógica e o valor do canal: 5V - 1023

$T_{porta} (V)$ - Valor_lido_na_porta

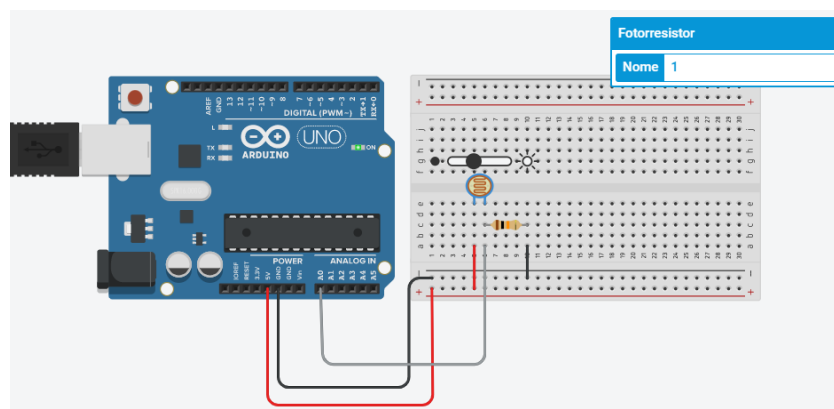
Resolvendo, temos o valor da tensão (em volts) na porta:

$$T_{porta}(V) = \frac{5 * \text{valor_lido_na_porta}}{1023}$$

Retomando o projeto - LDR

Relembrando a montagem do projeto, vamos usar o LDR conectado a uma porta analógica. Agora escrevemos os códigos de comandos no IDE Arduino ou TinkerCAD. Os comandos que usaremos são indicados abaixo.

Figura 23 - Montagem no Tinkercad - Projeto LDR e resistor.



Fonte: imagem gerada via plataforma TinkerCAD.com

Quadro 6: código usando o LDR e resistor

```
Int LDR; // declara a variável

Void setup ()
{
pinMode(A0, INPUT); // define a porta analógica 0 como entrada
Serial.begin(9600); //velocidade de envio de dados ao monitor
}

Void loop()
{
LDR=analogRead(A0); // faz a leitura do valor do LDR
Serial.print ("Valor lido no LDR:"); //imprime no monitor serial a frase entre aspas
Serial.println(LDR); // imprime na tela valor de LDR e pula a linha
Delay(250);
}
```

Fonte: os autores

Sugerimos realizar a simulação do código acima no IDE Arduino ou TinkerCAD.

Expressões Lógicas

Denomina-se expressões que os operadores são lógicos ou relacionais e os operandos são variáveis, constantes ou expressões aritméticas.

As expressões lógicas retornam valores lógicos: verdadeiro (true) se a expressão for verdadeira; ou falso (false) se a expressão for falsa.

Operadores Relacionais

Utiliza-se operadores relacionais para a comparação entre dois valores do mesmo tipo. Os valores podem ser de variáveis, constantes ou resultantes de expressões aritméticas.

Tabela 3: operadores relacionais.

Operador	Exemplo	Descrição
==	$x==y$	x é igual a y
!=	$x!=y$	x é diferente de y
<	$x<y$	x é menor do que y
>	$x>y$	x é maior do que y
<=	$x<=y$	x é menor ou igual a y
>=	$x>=y$	x é maior ou igual a y

Fonte: os autores.

Estruturas Condicionais IF e ELSE

Os testes lógicos são essenciais para podermos deixar os nossos algoritmos mais dinâmicos, possibilitando que os nossos algoritmos realizem ações diferentes com base em comparação de valores.

Estrutura IF

O condicional simples if permite executar um bloco de instruções caso o resultado do teste lógico retorne o valor verdadeiro (true), e não realiza nenhuma ação caso o teste lógico retorne o valor falso (false).

```
if (teste lógico ou pergunta lógica)
{
bloco de código executado se teste lógico retornar
verdadeiro
}
```

Exemplo de teste lógico

```
if (variável1 > variável2)
{
Serial.println("Variável 1 é maior");
delay(1000);
}
```

Estrutura IF ELSE

Já o condicional composto if else permite executar um bloco de instruções caso o resultado do teste lógico retorne o valor verdadeiro (true), e um bloco de instruções diferente caso o teste lógico retorne o valor falso (false).

Exemplo 1:

```
if (teste lógico ou pergunta lógica)
{
bloco de código executado se teste lógico retornar
VERDADEIRO
} else
{
bloco de código executado se teste lógico retornar FALSO
}
```

Exemplo 1:

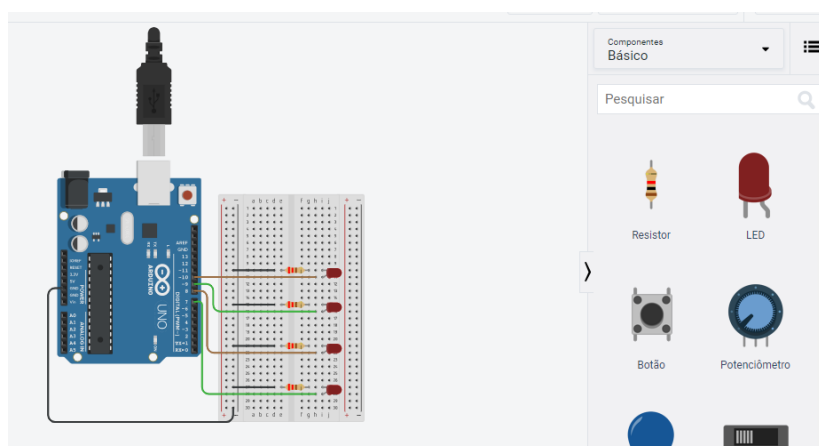
```
if (variavel1 > variavel2)
{
```



```
Serial.println("Variável 1 é maior");  
delay(1000);  
} else  
{  
Serial.println("Variável 2 é maior");  
delay(1000);  
}
```

No exemplo da Figura 24 temos uma aplicação de IF ELSE e com saída para o Serial Monitor. Sugere-se reproduzir no Tinkercad.

Figura 24: Exemplo de IF e ELSE.



Fonte: imagem gerada via plataforma www.Tinkercad.com.

Quadro 6: Exemplo de IF e ELSE.

```
int LED1 = 7;  
int LED2 = 8;  
int LED3 = 9;  
int LED4 = 10;  
  
void setup() {  
Serial.begin(9600);  
pinMode(LED1, OUTPUT);  
pinMode(LED2, OUTPUT);  
pinMode(LED3, OUTPUT);  
pinMode(LED4, OUTPUT);  
Serial.begin(9600);  
}  
  
void loop() {  
if (Serial.available())  
{  
char escolha = Serial.read();  
if (escolha == '1')  
{  
digitalWrite(LED1, HIGH);  
}
```



```
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
}
else if (escolha == '2')
{
digitalWrite(LED1, LOW);
digitalWrite(LED2, HIGH);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
}
else if (escolha == '3')
{
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, HIGH);
digitalWrite(LED4, LOW);
}
else if (escolha == '4')
{
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, HIGH);
}
else
{
Serial.print("Valor ");
Serial.print(escolha);
Serial.println(" incorreto.");
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
}
}
```

Fonte: os autores



Para fazer

Atividade 1 - Imprimindo seu nome no Serial Monitor

Esta atividade é bem simples e servirá para treinar a utilização do Serial Monitor:

Crie um código que envie o seu nome completo a cada linha, em *loop*, no Serial Monitor.

Importante: Você deverá utilizar somente 1 comando Serial para cada parte de seu nome, e entre cada comando deverá ter um *delay* de 1 segundo. O circuito a ser utilizado é apenas o Arduino no Tinkercad. Não há a necessidade de mais nenhum componente. Salve seu projeto no Tinkercad e compartilhe o *link* no local recomendado pelo curso.

Atividade 2 - Piscar o LED com as iniciais do seu nome em código Morse

O Código Morse é um sistema binário de representação à distância de números, letras e sinais gráficos, utilizando-se de sons curtos e longos, além de pontos e traços para transmitir mensagens.

Figura 25: Alfabeto e números em Código Morse.

A	.-	J	.---	S	1	-----
B	-...	K	--- ..	T	- .-	2-
C	-....	L	U	3	...--
D	-... ..	M	-- ..	V-	4-
E	. -	N	-- ..	W-	5
F	O	--- ..	X	--- ..	6	-....
G	P	Y-	7	--... ..
H	Q	Z	8	-----
I	.. -	R			9	-----
						0	-----

Fonte:

<https://www.publicdomainpictures.net/es/viewimage.php?image=35529&picture=morse>

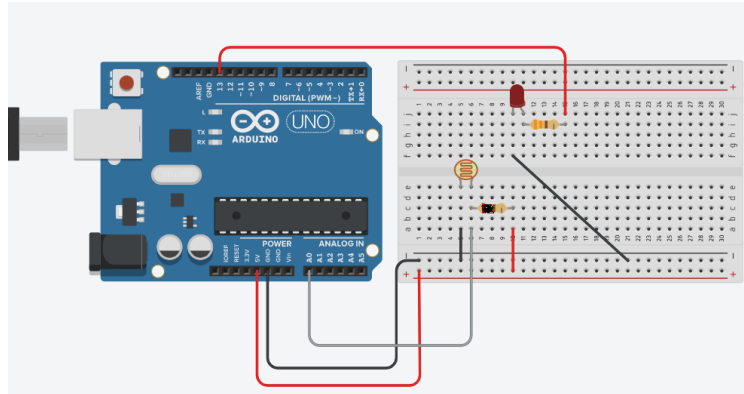
Como atividade, monte um circuito de um LED, equivalente ao utilizado na semana 3. Depois de pronto e testado o circuito do LED piscando, identifique na tabela de Código Morse (acima), as iniciais do seu nome, e ajuste o código da programação, de forma que o LED fique piscando em conformidade com as iniciais do seu nome e o Código Morse. Salve seu projeto no Tinkercad e compartilhe o *link* no local recomendado pelo curso.

Atividade 3 - LDR e luz ambiente

Nesta atividade, iremos usar um LDR acoplado a um LED, como mostrado na figura abaixo. O sinal do LDR irá acionar o LED em função da luz ambiente. Estabeleça um limite mínimo de 4,00 volts de leitura no LDR, o qual irá ligar o LED quando a luminosidade do ambiente esteja em aproximadamente 50% (ligue o LED quando tiver menor quantidade de luz). Estime o valor do resistor (R) ligado ao LDR, realize a montagem do projeto no Tinkercad e desenvolva o código para controle do dispositivo.

Importante: na configuração do circuito apresentada abaixo, teremos a leitura na porta analógica (A0) → 0V quando a luminosidade for máxima a resistência no LDR será mínima (RLDR) → 0. Em contrapartida, quando a luminosidade for mínima, teremos a resistência no LDR (RLDR) com valor máximo e a leitura na porta analógica (A0) → 5,0V. Outras configurações devem ser utilizadas, adequando, naturalmente, o sketch e o resistor associado ao LDR.

Figura 26: LDR e luz ambiente.



Fonte: imagem gerada via plataforma www.Tinkercad.com.

Semana 5

Nesta seção você encontra as atividades propostas para a semana 5. É aconselhável a leitura dos materiais disponíveis na seção "Para ler". Neste tópico apresentamos um resumo sobre comandos de repetição, resistores *pull-down* e *pull-up*, estratégias para interação a partir do teclado via Monitor Serial e o uso de vetores e matrizes. Neste tópico "Para saber mais", sugere-se a leitura do artigo "Estação meteorológicas de código aberto: um projeto de pesquisa e desenvolvimento tecnológico", o qual traz um exemplo de aplicação do Arduino na educação básica. E por fim, temos três atividades para a semana 5.



Para ler – aspectos técnicos

Comando de repetição

Os comandos de repetição se mantêm em um ciclo, um ou mais instruções, até que uma condição esteja sendo satisfeita. São três opções de comandos de repetição, seguem:

while ()

O comando *while ()* analisa uma expressão no início do laço (ciclo) e executa as instruções, caso as condições sejam satisfeitas.

A sintaxe do comando:

```
while (condição) {  
  comandos1;  
  comando2;  
}
```

Vale lembrar que só haverá execução dos comandos internos do laço, caso a condição seja satisfeita.

for ()

Deve ser utilizado quando se tem ideia do número (isto é, usa-se um contador) de laços que serão executados.

A sintaxe do comando:

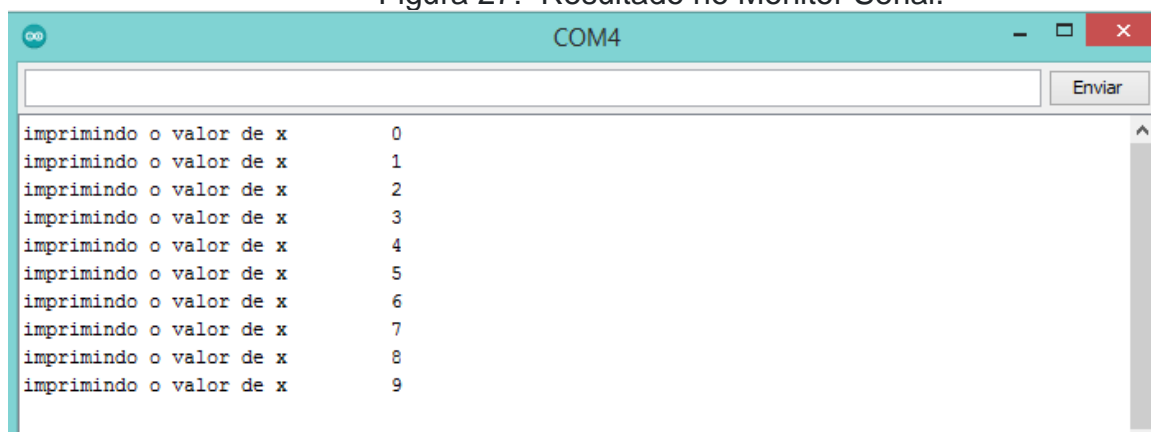
```
for (inicialização do contador; condição; incremento do contador)
{
comando 1;
comando 2;
}
```

Quadro 7: código com o comando for ().

```
int x;
void setup(){
  Serial.begin(9600);
  for (x = 0; x < 10; x++){
    Serial.print("imprimindo o valor de x \t");
    Serial.println(x);
  }
}
void loop(){
}
```

Fonte: os autores.

Figura 27: Resultado no Monitor Serial.



do while ()

Fonte: os autores.

Este comando é similar ao *while ()*, contudo a condição é avaliada no final do laço. Abaixo segue a sintaxe do comando:

```
do {
comando 1;
comando 2;
} while (condição);
```

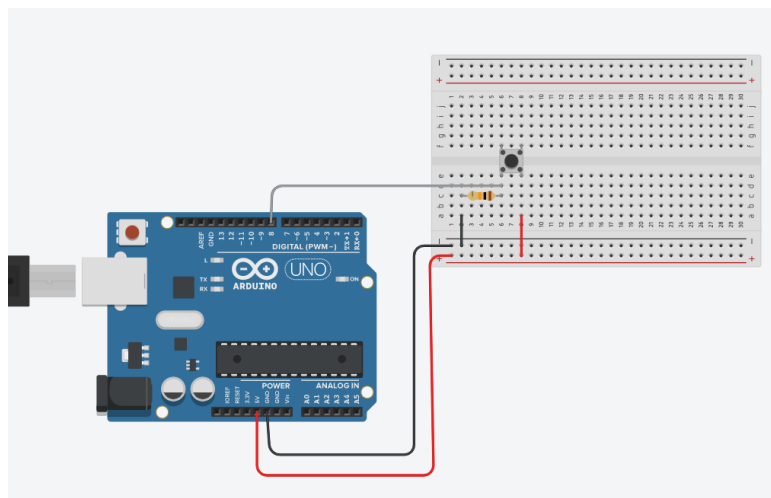
Resistores *pull-down* e *pull-up*

Resistores *pull-down* e *pull-up* são resistores usados em circuitos eletrônicos para garantir que entradas se ajustem em níveis lógicos esperados, para o caso do Arduino, temos 0V ou 5V; portanto, têm a função de minimizar as interferências eletromagnéticas que poderia chegar às portas digitais. Na figura 1, temos o esquema de uma ligação de um botão e resistor de 10kΩ. Para o caso do esquema *pull-down*,

quando o botão é pressionado, a porta digital (I/O pin) receberá 5V (*HIGH* ou 1), já com o botão solto, a porta digital receberá 0V (*LOW* ou 0). O raciocínio é o mesmo para o esquema *pull-up*, tendo como leitura na porta exatamente o oposto da recebida com o esquema *pull-down*.

Na figura 28 temos um exemplo de um circuito com um botão e resistor *pull-down*.

Figura 28: Montagem resistor *pull-down*.



Fonte: figura gerada no site www.Tinkercad.com.

No quadro 8 temos o *sketch* que permite informar via Monitor Serial se o botão foi pressionado.

Quadro 8: *Sketch* para interação com o botão e saída no Monitor Serial.

```
void setup(){
  pinMode(8,INPUT);
  Serial.begin(9600);
}
void loop(){
  if (digitalRead(8) == HIGH){
    Serial.println("Voce esta pressionando o botao ");
    delay(1000);
  }
}
```

Figura 28: Saída no Monitor Serial.

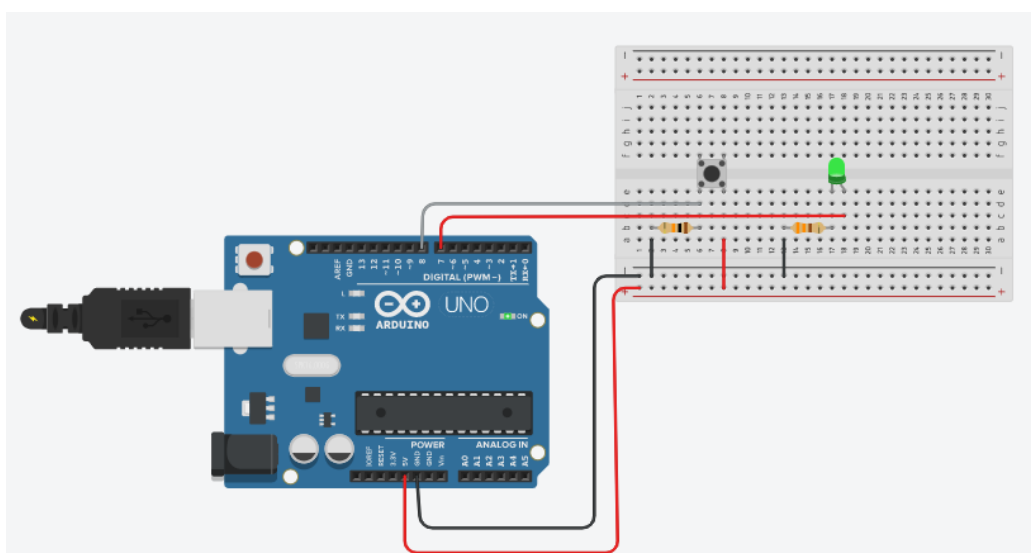


Fonte: figura gerada no site www.TinkerCAD.com.

Como exemplo da figura 29, temos o controle de um LED a partir do acionamento de um botão (interruptor tátil). A proposta é ligar o LED na porta 7 ao 5V. Usando um resistor de 10 k Ω , puxaremos o nível da porta 8 (*pull-down*) para baixo (*LOW*). Ao ligar o GND no botão e este na porta 8, quando pressionado, a corrente elétrica percorre o caminho com menos resistência, jogando o nível lógico da porta para cima (*HIGH*).

Materiais: 1 botão; 1 LED; 2 resistores (330 Ω e 10 k Ω); 1 placa de ensaio; 1 placa de Arduino Uno.

Figura 29: Esquema da montagem do resistor *pull-down* e controle do LED.



Fonte: figura gerada no site www.TinkerCAD.com.

Quadro 9: *Sketch* para acionamento de um LED com um interruptor tátil.

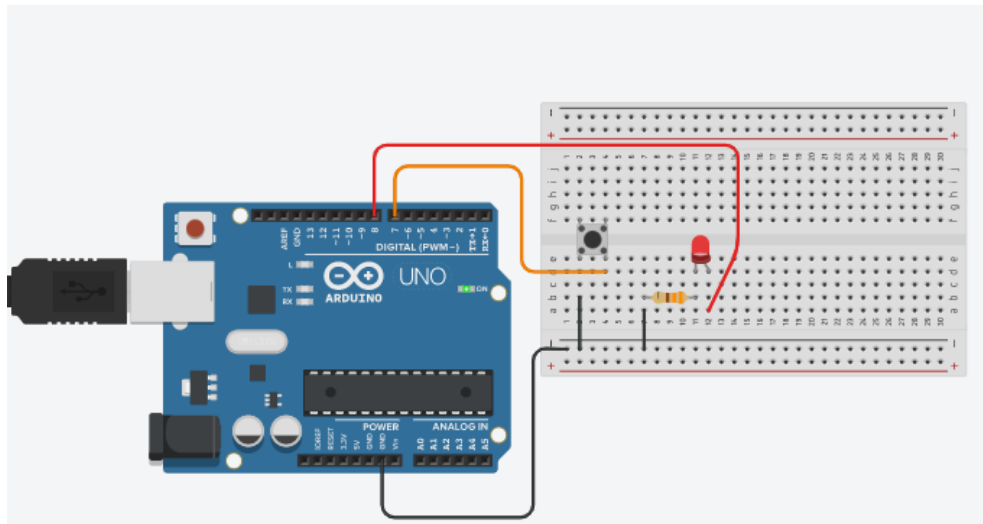
```
//controlando um LED com interruptor tátil
void setup(){
  pinMode(8,INPUT);
  pinMode(7,OUTPUT);
  Serial.begin(9600);
}
void loop(){
  if (digitalRead(8) == HIGH){
    digitalWrite(7, HIGH);
  }
  else {
    digitalWrite(7, LOW);
  }
}
```

Fonte: os autores.

O Arduino também possui um resistor interno que variam entre 20kΩ a 50kΩ (depende do modelo) que pode ser acessado em linha de comando. Para isso, usa-se *pinMode* (nº da porta, *INPUT_PULLUP*), ou seja, teremos uma porta de entrada e um resistor *pull-up* ativados.

A figura 38 ilustra o *sketch* e montagem de controle de LED com um interruptor tátil. Por ser tratar de um resistor *pull-up*, quando o botão é pressionado temos a leitura na porta 7 como *LOW* e o botão solto temos o sinal *HIGH*.

Figura 29: Esquema da montagem.



Fonte: figura gerada no site www.Tinkercad.com

Quadro 10: *Sketch* utilizando o comando *INPUT_PULLUP*

```
// controlando um LED com interruptor tátil
Void setup()
{
pinMode (7, INPUT_PULLUP);
pinMode (8, OUTPUT);
}
Void loop()
{
If (digitalRead(7) == LOW) {
digitalWrite (8, HIGH);
}
else{
digitalWrite (8, LOW);}
}
```

Fonte: os autores

Recebendo dados pelo Monitor Serial

Conforme já abordado na semana 4, o Monitor Serial é utilizado para comunicação entre o Arduino e o computador. Entre suas funções, algumas nos possibilitam enviar dados do ambiente externo para o Arduino, por exemplo, por meio do nosso teclado utilizando o Monitor Serial.

Serial.available()

A função `serial.available()` permite interação Arduino e teclado, via Serial Monitor. No exemplo abaixo o `sketch` retorna a tecla que foi digitado no Serial Monitor.

```
char caracter_digitado;
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available()) { //Aqui é avaliado se algum dado foi enviado via Monitor
    Serial.
    caracter_digitado = Serial.read(); //Aqui lê o dado recebido.
    Serial.print("Imprimindo o que foi digitado:\t");
    Serial.println(caracter_digitado);
  }
}
```

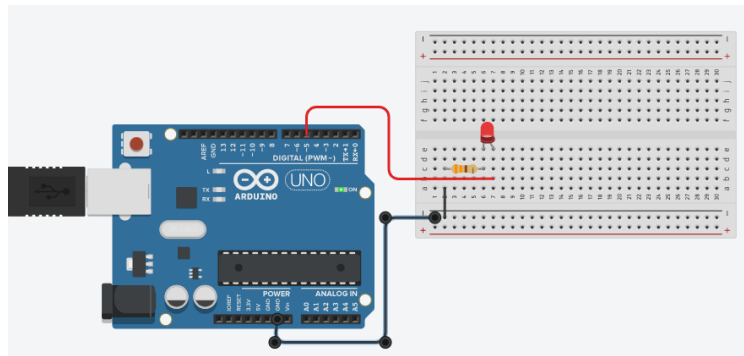
Note que no quadro 11 que a linha `if(Serial.available())` é avaliado se o teclado foi pressionado. Na sequência, a função `Serial.read()` devolve a tecla pressionada finalmente `Serial.print()` imprime o que foi digitado no Serial Monitor.

Quadro 11: *Sketch* acionando o LED pelo teclado.

```
//aciona um LED pelo teclado
int ledpin = 5;
char letra;
void setup(){
  pinMode(ledpin, OUTPUT);
  Serial.begin((9600);
}
void setup(){
  if(Serial.available()) {
    letra= Serial.read();
    if(letra == 'r'){
      digitalWrite(ledpin, HIGH);
    }
    if(letra == 'a') {
      digitalWrite(ledpin, LOW)
    }
  }
}
```

Fonte: os autores.

Figura 30: Montagem LED controlado pelo teclado.



Fonte: figura gerada no site www.Tinkercad.com

Sugere-se como atividade complementar a utilização dos códigos descritos na figura 30, no TinkerCAD. Cabe lembrar que é necessário arrastar primeiramente o Arduino para a bancada e, em seguida, transcrever o *sketch*.

Vetor e Matriz

As variáveis podem assumir valores diferentes ao longo da execução de um *sketch*; porém, os valores atribuídos ao longo da execução não podem ser recuperados posteriormente. Para situações como a descrita, o uso de variáveis tipo vetor ou matriz seria a solução. Vetor (*array*) é um tipo de variável (variável estruturada ou indexada) que permite armazenar valores com o um número índice incorporado. Dessa forma, é possível resgatar seu valor a qualquer momento a partir de sua localização específica atribuída pelo seu índice.

Como exemplo, suponha que as notas de 8 alunos estejam armazenadas em uma variável composta, identificada por *nota*, segue tabela abaixo:

Tabela 3 - Exemplo de uso de um *array*.

índice	0	1	2	3	4	5	6	7
nota	10	8	5	2	8	9	7	4

Fonte: os autores.

Para resgatar o valor da nota armazenada do quarto aluno, deve-se escrever `nota [3]`. Dessa forma, o valor devolvido será "2". A mesma ideia vale para o caso de um vetor bidimensional, isto é, uma matriz. Neste caso, a posição deve possuir dois índices: linha e coluna, por exemplo: `matriz [3][2]`, ou seja, estamos apontando para a linha 3 e coluna 2.

Utilizando os vetores e matrizes

Para definirmos um *array*, basta indicar o tipo de variável seguida de colchetes (indicando tamanho ou não). Também é possível atribuir os valores *a priori*, como o exemplo a seguir:

```
int pino[ ] = {6, 7, 8, 10, 11}; // o vetor pino armazena os valores de 5 portas.
```

```
digitalWrite (pino [0], HIGH); //aqui é solicitado que o porta 6 fique em HIGH.
```

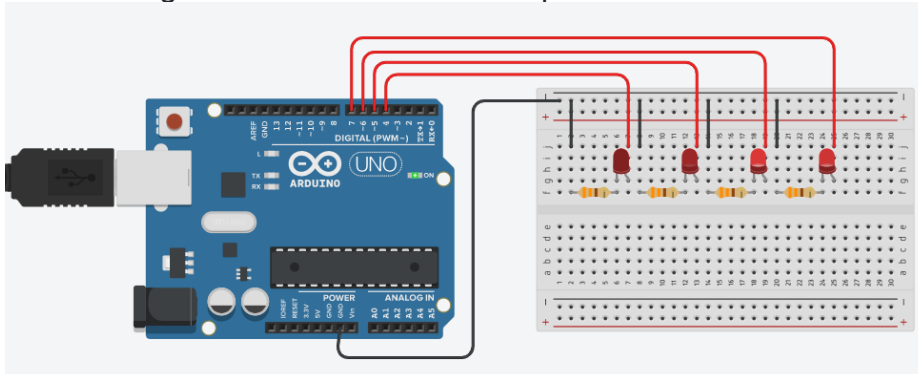
Para o caso de matrizes, é necessário indicar o tipo da variável, como no caso dos vetores, mas indicando as dimensões (linha e coluna). No *sketch* do quadro 12 temos uma variável matriz definida como "matrizpino [][]", que recebe os valores das portas digitais que serão acionadas na execução do algoritmo. Após a definição da matriz com seus valores, as portas são colocadas como saída, em *void setup ()* a partir de dois comandos de repetição *for ()*. Por fim, em *void loop ()*, as portas vão para *HIGH* de forma randômica, com os valores dos índices (i e j), sendo atribuídos aleatoriamente pelo comando *random()*, permitindo que os LEDs sejam acessos um a um, mas de forma imprevisível.

Quadro 12: *Sketch* com exemplo de uso de variável matriz para portas digitais

```
int matrizpino[2][2]={{4,5},{6,7}};
int i, j;
void setup() {
  for (i = 0; i < 2; i++){
    for (j = 0; j < 2; j++){
      pinMode(matrizpino[i][j], OUTPUT);
    }
  }
}
void loop(){
  i = random(2);
  j = random(2);
  digitalWrite(matrizpino[i][j], HIGH);
  delay(250);
  Serial.println(matrizpino[i][j], LOW);
}
```

Fonte: os autores.

Figura 31: Montagem do circuito com LEDs piscando de forma aleatória.



Fonte: figura gerada no site www.Tinkercad.com.

Formas de declaração de um *array*

Todos os métodos abaixo são formas válidas de criar um vetor.

1. `char texto[8] = "Arduino";`
2. `int var[6];`
3. `int var[] = {4, 6, 83, 33, 67};`
4. `int var[5] = {4, 6, -83, 33, 67,};`
5. `float var_real[4];`

Note que na situação 1 é atribuído uma *array* tipo *char*. Este deve manter um elemento a mais para armazenamento de caractere *null* necessário. Na situação 2 declaramos um vetor sem definir os valores. Na situação 3 não foi definido o tamanho do *array*, mas somente os valores atribuídos. Na situação 4 temos os valores e tamanho atribuídos ao *array* "var". Por fim, na situação 5 temos um vetor que aceita valores reais com 4 posições, mas sem defini-los *a priori*.



Para saber mais

Como sugestão de leitura para esta semana, recomenda-se o artigo "Estação meteorológicas de código aberto: um projeto de pesquisa e desenvolvimento tecnológico".

SILVA, Renan Bohrer da *et al.* Estações meteorológicas de código aberto: um projeto de pesquisa e desenvolvimento tecnológico. **Revista Brasileira de Ensino de Física**, São Paulo, v. 37, 2015.



Para fazer

1) Controlando LEDs pelo Teclado

Como atividade, pede-se a montagem no Tinkercad de um circuito com 4 LEDs, cada um com seu resistor e cada um ligado em um pino digital diferente. Utilize o teclado para acender e apagar cada um deles. Escolha letras diferentes para acender e apagar cada LED. Informe o menu uma vez no Monitor Serial. Exemplo: a - acende o LED 1 e b apaga o LED 1.

Importante: você deve utilizar aspas simples 'desse jeito' nas letras a serem comparadas dos IFs.

Exemplo: `if(letra == 'z'){ ...`

2) Dois interruptores táteis e dois LEDs

Desenvolva um sketch e circuito no Tinkercad com dois LEDs e dois botões, cada um em um pino digital. Cada botão deverá acionar um LED.

Exemplo: Botao1 aciona LED1; Botao2 aciona LED2

3) Contador com botões Tarefa

Pede-se nesta atividade a montagem de um circuito e um sketch no Tinkercad. Para tal, deve-se usar um botão para incremento e outro para decremento de uma variável. O valor corrente da variável deve ser apresentado continuamente no Monitor Serial.



Semana 6

Nesta seção você encontra as atividades propostas para a semana 6. É aconselhável a leitura dos materiais disponíveis na seção "Para ler – aspecto técnico". Sendo que os dois primeiros itens tratam da incorporação de componentes (potenciômetro e *Buzzer*) ao Arduino, já o terceiro item apresenta uma ferramenta de integração do IDE Arduino ao Microsoft Excel. Como leitura pedagógica, sugere-se o artigo: "Por que e como introduzir a aquisição automática de dados no laboratório didático de Física? Na seção "Para fazer", você encontra a descrição da atividade proposta para a semana. Por fim, sugerimos que nesta semana discuta sobre a elaboração do projeto final, que deverá ser entregue na semana 10.

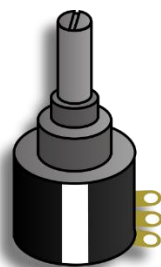


Para ler - fundamentação teórica de física

Utilizando o potenciômetro

Um potenciômetro é um componente eletrônico que possui resistência elétrica variável. O modelo mais comum possui um resistor de três terminais ("início" e "fim"). O terminal central é chamado de cursor, pois pode deslizar em contato com a pista. Ao aplicar 0V em um dos terminais e 5V no outro, teremos uma leitura no terminal central que varia em função de sua localização na pista, portanto, temos na prática um divisor de tensão, quando utilizamos os três terminais, como apresentado na figura 31.

Figura 31: Potenciômetro.



Fonte: <https://pixabay.com/pt/vectors/potenci%C3%B4metro-reostato-resist%C3%A2ncia-40381/>

O funcionamento de um potenciômetro é baseado no fato da resistência de um material (por exemplo, um bastão de carvão) ser proporcional ao seu comprimento. O cursor pode deslizar sobre a trilha de carvão, modificando o comprimento da trilha que será percorrida pela corrente (entre o ponto de contato do cursor e um terminal fixo). Na figura 43 o comprimento é indicado por L (desprezando o tamanho do cursor). Lembrando que a segunda lei de Ohm é dada por: $R = \frac{\rho L}{A}$. Com a variação da resistência e utilizando a porta analógica, é possível interagir com o Arduino. Na sequência temos uma aplicação na qual é possível ligar um LED utilizando do potenciômetro.

A montagem é composta de um resistor de calibração (R_C) e o resistor variável – potenciômetro (R_{pot}). A associação é em série, portanto temos a tensão total como a soma das tensões dos elementos do circuito. Lembrando que $V_{ent} = 5,0V$. Pela Lei de Ohm, podemos calcular a corrente pelos resistores:

$$V_{ent} = (R_C + R_{pot})i$$

A tensão no resistor variável (V_{pot}) é dada pela porta analógica.

Por fim, podemos obter R_{pot} (resistência variável) utilizando a corrente elétrica e a tensão V_{pot} , segue:

$$R_{pot} = \frac{V_{pot}}{i}$$

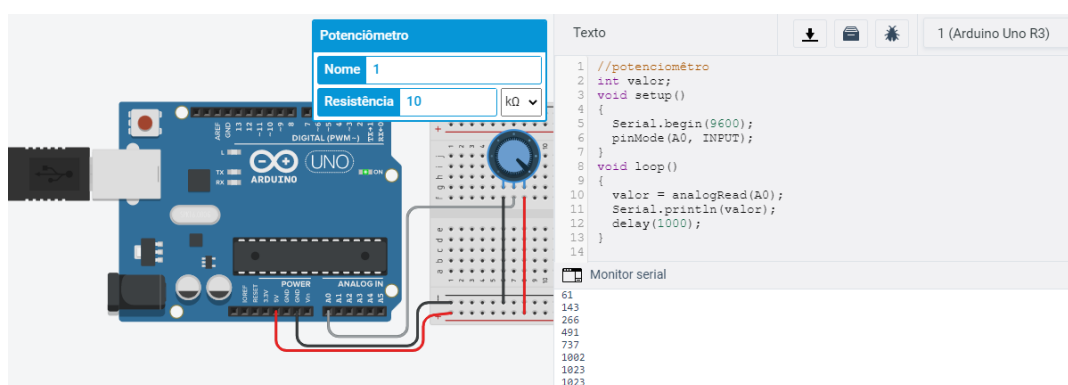


Para ler – aspectos técnicos

Utilizando o potenciômetro no Tinkercad.

O circuito da figura 32 e o sketch do quadro 13 permitem variar a resistência do potenciômetro e realizar sua leitura por meio do Monitor Serial, com valores entre 0 e 1023. Também é possível converter o valor da saída em volts, para tal, basta multiplicar saída da porta A0 pela razão 5/1023.

Figura 32: Esquema da montagem.



Fonte: imagem geral pelo site www.Tinkercad.com.

Quadro 13: Sketch e saída no Monitor Serial.

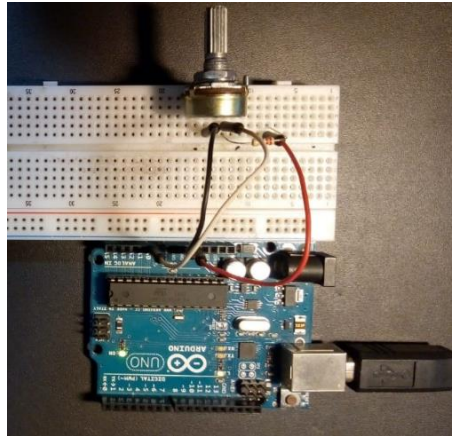
```
// potenciômetro.  
int valor;  
void setup()  
{  
  Serial.begin(9600);  
  pinMode(A0, INPUT);  
}  
void loop()  
{  
  valor = analogRead(A0);  
  digitalprintln(valor);  
  delay(1000);  
}
```

Fonte: os autores

Medindo a resistência variável e controle pelo teclado.

Para a montagem, usaremos o potenciômetro e um resistor de calibração de 220Ω, como indicado na figura 33. O potenciômetro estará conectado na porta analógica (A0).

Figura 33: Esquema da montagem.



Fonte: os autores.

Neste projeto iremos usar um potenciômetro (5,5, k Ω) ligado em série a um resistor de calibração (220 Ω). Veja o *sketch* do quadro 14.

Quadro 14: *Sketch* - Medido resistência variável e controle pelo teclado.

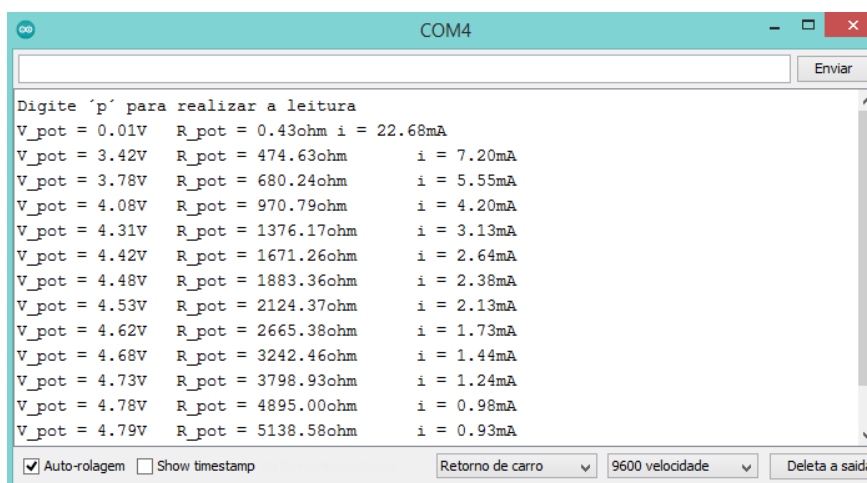
```
float Rpot; // variável - Resistência variável - potenciômetro
float Rc = 220; // valor da resistência de calibração
float Vpot; // variável - tensão no potenciômetro
float Valor_lido_porta;
float i; // variável - corrente no circuito
char leitura; // define a variável leitura que corresponde a uma letra do teclado
void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT); // entrada A0
  Serial.print("Digite 'p' para realizar a leitura");
  Serial.println();
}
void loop() {
  leitura = Serial.read(); // a variável leitura será obtida através do teclado
  if (leitura == 'p') { //se apertar p (minúsculo) no teclado ativa o if
    Valor_lido_porta = analogRead(A0);
    Vpot = 5*Valor_lido_porta/1023;
    i = (5-Vpot)/Rc;
    Rpot = Vpot/i;
    Serial.print("V_pot = ");
    Serial.print(Vpot); // tensão no potenciômetro
    Serial.print("V");
    Serial.print("\t");
    Serial.print("R_pot = ");
    Serial.print(Rpot); // Resistência variável (potenciômetro)
    Serial.print("ohm");
    Serial.print("\t");
    Serial.print("i = "); //corrente no circuito
    Serial.print(i*1000);
    Serial.print("mA");
    Serial.println();
  }
}
```

Fonte: os autores

Vale destacar que o uso da função *Serial.read()* permite interpretar a entrada de um caracter pelo teclado. Na sequência, o *if* permite verificar se a tecla correta foi pressionada. Indo mais além com o *sketch*, temos a conversão do valor lido na porta analógica em volts. Depois temos o cálculo de *i* (corrente elétrica) a partir da tensão aplicada no resistor de calibração e sua resistência. Por fim, temos a obtenção do valor de R_{pot} , ou seja, a resistência variável.

Com a execução do *sketch* e após o pressionamento da tecla *p*, teremos como resposta os valores de: tensão no potenciômetro (V_{pot}); resistência no potenciômetro (R_{pot}); e corrente no circuito (*i*). Vale lembrar que a alteração nas configurações do potenciômetro permite alterar as demais leituras. Na figura 34 temos a saída do *sketch* no Monitor Serial.

Figura 34: Saída no Monitor Serial.

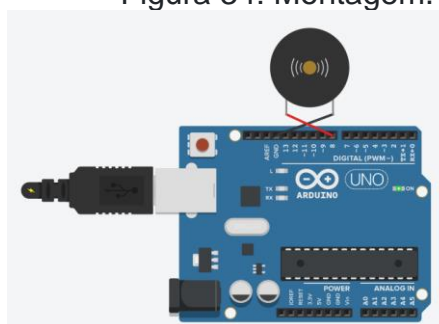


Fonte: os autores.

Sonificador

Na figura 34 e o *sketch* no quadro 15, temos um sonificador (“buzzer”) ligado a uma saída digital para simular um som tipo sirene. A montagem do TinkerCAD e *sketch* são mostrados na figura 1. Note que o som emitido terá dependência com a variável "sinVal", isto é, será uma função senoidal e depois convertida para "toneVal", passando como variável do tipo "int".

Figura 34: Montagem.



Fonte: imagem gerada em www.Tinkercad.com



Quadro 15: sketch do sonorizador.

```
Float sinVal;
Int toneVal;

Void setup() {
pinMode (8, OUTPUT);}

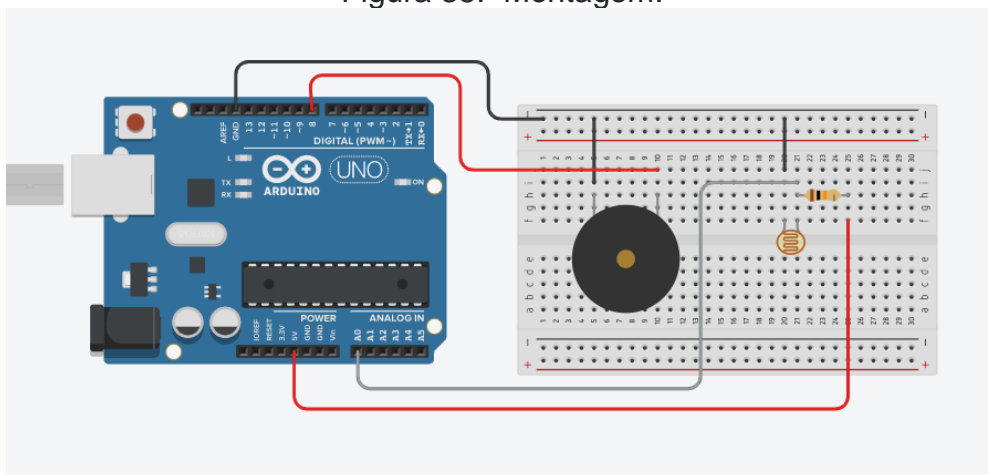
Void loop() {
For (int x=0; x < 180; x++) {
sinVal = (sin(x*(3.1416/180))); // converte graus para radianos
toneVal = 2000 + (int (sinVal*1000)); // gera frequência a partir de sinVal
tone (8, toneVal);
delay (2);}
}
```

Fonte: os autores

LDR e Sonorizador

Neste projeto temos um sonorizador (“buzzer”) ligado ao Arduino, porém o toque depende da luminosidade no LDR por meio do variável "ldrValue", a qual define o *delay* imposto à *void loop ()*, veja a figura 35.

Figura 35: Montagem.

Fonte: imagem gerada em www.Tinkercad.com.

Quadro 16: Sketch do sonorizador e LDR

```
//disponível em “Arduino Básico” de McRoberts, Ed. Novatec
int piezoPin = 8; // sonorizador na porta 8
int ldrPin = 0; // LDR na porta analógica 0
int ldrValue = 0; //valor lido no LDR

void setup() { //nada a fazer aqui }

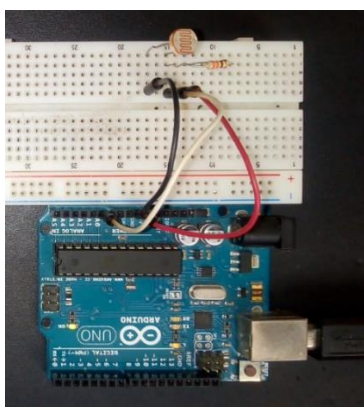
Void loo () {
ldrValue = analogRead (ldrPin); // lê o valor do LDR
tone (piezoPin, 1000); //toca tom de 1000 Hz no sonorizador
delay (25); // aguarda um pouco
noTone (piezoPin); // interrompe o tom
delay(ldrValue); aguarda t milissegundos, com t = ldrValue
```

Integrando o IDE Arduino ao Microsoft Excel

A integração do Arduino com o Microsoft Excel pode ser feita utilizando a macro PLX-DAQ. O termo "DAQ" é o acrônimo em inglês de aquisição de dados e o "PLX" é abreviatura de Parallax (empresa proprietária da ferramenta). Além de permitir a conexão ao microprocessador Parallax, o PLX-DAQ também permite a integração com outros microprocessadores, como o caso do Arduino. O PLX-DAQ possibilita a comunicação com portas seriais que variam entre a COM1 a COM15 e com taxa de transmissão de até 128kbps. As leituras realizadas pelas portas do Arduino podem ser pareadas em tempo real com as células do Microsoft Excel. A planilha pode possuir até 65000 linhas e 26 colunas. Além da apresentação dos dados em colunas, é possível visualizar a construção de gráficos em tempo real.

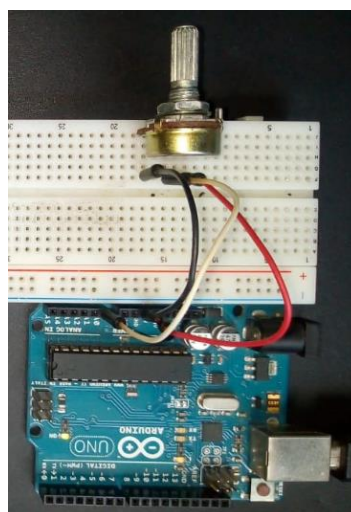
A montagem da figura 36 permite a leitura da tensão no LDR a partir da porta analógica A0. De forma similar, a montagem da figura 2 permite a leitura da tensão no potenciômetro.

Figura 36: Montagem do circuito com resistor limitador e um LDR.



Fonte: os autores.

Figura 37: Montagem do circuito com um potenciômetro.



Fonte: os autores.

Na sequência (quadro 17) temos o *sketch* que possibilita a integração com a planilha Excel e os circuitos das montagens das figuras 36 e 37. Note que em *void setup ()*, linha 8, temos a passagem do comando pelo *Serial.println* de "CLEARDATA", indicando à macro para limpar a planilha e iniciar o processo de transferência de dados. Na linha seguinte encontramos o comando *Serial.println("LABEL,HORA,tensao (v)")*, que executa a impressão do cabeçalho da planilha, veja figura 4. Em *void setup ()* se inicia a impressão dos dados, limitado em 20 linhas (pode ser alterado). No quadro 17, nota-se também que o gráfico foi configurado para ser plotado em tempo real a partir dos dados coletados.

Quadro 17: *Sketch* exemplo para integração do IDE Arduino com o Microsoft Excel.

```
// explorando dados para Excel
int x = 2; //contador de linhas.
int LABEL = 1; //define a 1ª linha na planilha
float tensao = 0;
void setup(){
  pinMode(A0, INPUT);
  Serial.begin(9600);
  Serial.println("CLEARDATA"); //Reset da comunicação serial
  Serial.println("LABEL,HORA,tensao (v)"); //Define o cabeçalho
}
void loop(){
  tensao = analogRead(A0)*5.0000/1023;
  Serial.print("DATA,TIME,"); //inicia a impressão de dados na planilha.
  Serial.print(tensao,4);
  Serial.println(",");
  x++; //incrementa o contador de linhas
  if(x > 21 ){ // limitador do número de linhas
    x = 2;
    Serial.println("ROW,SET, 2"); // retorno à linha 2
  }
  delay(200);
}
```

Fonte: os autores.

Iniciando o PLX-DAQ

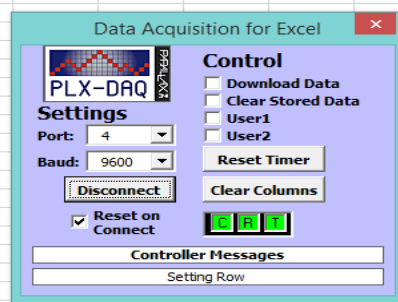
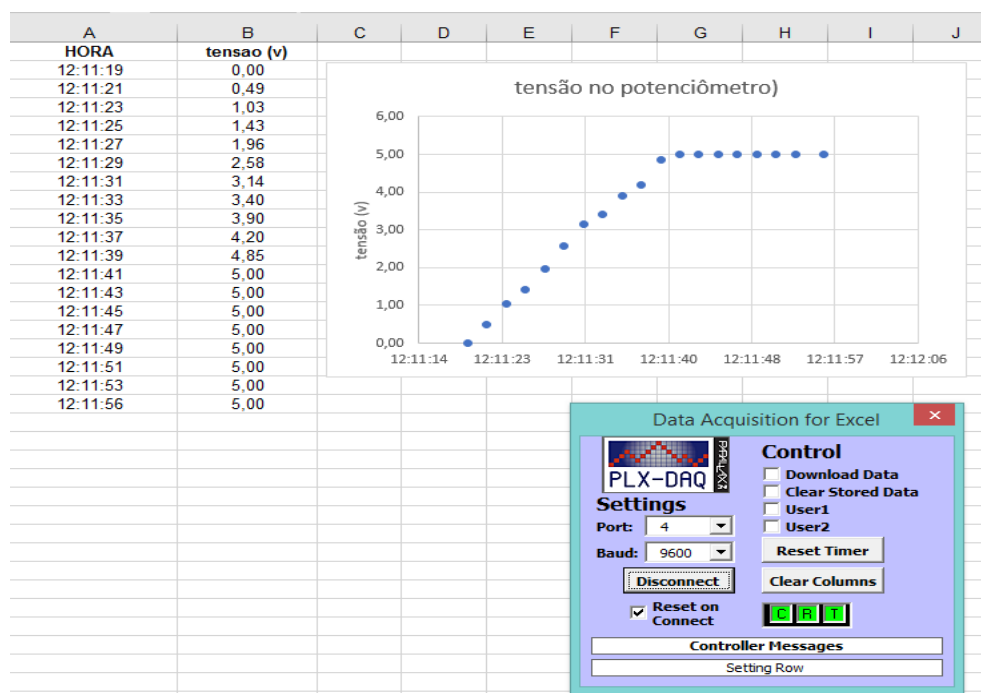
Para realizar o *download* da macro PLX-DAQ em formato (.zip), <https://www.parallax.com/package/plx-daq/>. Depois de realizar a extração da pasta "PLX-DAQ", localize e clique no arquivo executável "plx-daq_install". Ao término do processo de instalação você terá uma pasta chamada "PLX-DAQ", geralmente na área de trabalho. Clique na planilha chamada "PLX-DAQ Spreadsheet". Este arquivo será usado para a coleta de dados. É importante destacar que não se deve remover da pasta "PLX-DAQ" o arquivo "PLX-DAQ Spreadsheet".

Após a abertura da planilha "PLX-DAQ Spreadsheet", carregue o *sketch* da figura 3 no Arduino. Com a planilha aberta, escolha a porta, em "Port", (a mesma que o IDE Arduino está conectado) na tela de comando do "DATA Acquisition Excel" e, em seguida, ajuste a taxa de transmissão "Baud" para o mesmo valor que foi usado no *sketch* "*serial,begin(9600)*". Por fim, clique em "Connect". Você verá os dados sendo atualizados na planilha. Também você poderá inserir um gráfico, como na figura 38. Para isso, selecione as colunas e, em seguida, clique no comando "inserir", no

menu superior do Excel, e escolha um tipo de gráfico mais adequado para seu estudo. Note que na figura 4 escolhemos um gráfico de dispersão, com somente pontos. É o mais indicado para dados experimentais.

Importante: Até a presente data, o PLX-DAQ não permite integração ao TinkerCAD.

Figura 38: Planilha com dados da tensão no potenciômetro coletados pela macro PLQ-DAQ.



Fonte: os autores.

Para ler – aspectos pedagógicos

Sugestão de Leitura:

Como sugestão de leitura para esta semana, recomenda-se o artigo "Por que e como introduzir a aquisição automática de dados no laboratório didático de Física?".

HAAG, Rafael; ARAUJO, Ives Solano; VEIT, Eliane Angela. Por que e como introduzir a aquisição automática de dados no laboratório didático de física? **Física na escola**. São Paulo. Vol. 6, n. 1 (maio 2005), p. 69-74, 2005.

Para fazer

1) **Atividade 1** - Medido a tensão no potenciômetro e acendendo um LED

Nesta atividade pede-se o desenvolvimento do sketch e do circuito apresentado na seção sobre o potenciômetro (Medindo resistência variável e controle pelo teclado) e inclua no projeto dois LED (um vermelho e outro verde) para indicar valores de tensão maiores ou menores de 2,5V no potenciômetro.

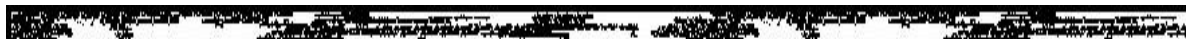
LED verde acende e LED vermelho apaga quando $V_{pot} \leq 2,5$;

LED vermelho acende e LED verde apaga quando $V_{pot} > 2,5$.

Crie seu projeto no Tinkercad.

2) **Atividade 2** - Alarme

Nesta atividade pede-se o desenvolvimento de um sketch e um circuito que acione o Buzzer e acenda um LED vermelho quando a luminosidade no LDR atingir $\approx 1/4$ do valor máximo. Crie seu projeto no Tinkercad.



Semana 7

Nesta seção você encontra as atividades propostas para a semana 7. É aconselhável a leitura dos materiais disponíveis na seção "Para ler", sendo que os itens tratam de sensores capazes de aferir a movimentação de objetos, temperatura e umidade do ar e temperatura de líquidos. Como leitura pedagógica, sugere-se o texto "Contribuições do Arduino no ensino de Física: uma revisão sistemática de publicações na área do ensino", o qual traz uma reflexão sobre as diversas formas de utilização do Arduino como elemento didático-pedagógico. Na seção "Para fazer" você encontra a descrição das atividades pedidas para a semana 7.



Para ler – aspectos técnicos

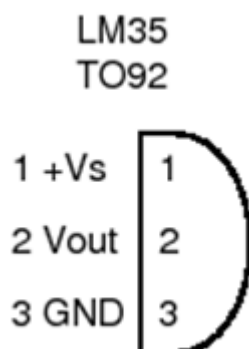
Medindo temperatura com o Arduino

Inicialmente vamos discutir o sensor de temperatura chamado LM35. Abaixo seguem algumas especificações desse tipo de sensor:

- sensor tipo junção npn;
- apresenta uma saída de tensão linear relativa à temperatura;
- tensão de alimentação: 2,2V – 5,5 V, VCC e GND;
- sinal de saída de 10mV/°C;
- trabalha dentro da faixa de temperatura de – 55 a 150 °C;
- não necessita de calibração externa;
- apresenta valores de temperatura com variações de $\frac{1}{4}$ °C ou até mesmo $\frac{3}{4}$ °C.

Figura 38: Sensor de temperatura LM35.

Pino da esquerda (1) - coletor - Pino do meio (2) - base - Pino da direita (3) - emissor



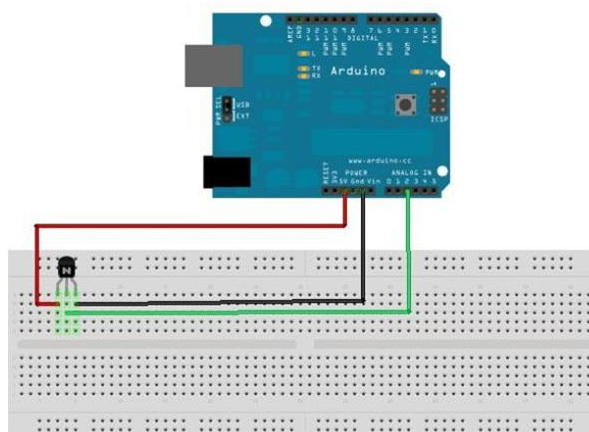
Fonte: os autores.

Para conectar o sensor LM35 ao Arduino basta seguir o esquema da figura 39. Ou seja, alimentar o pino da esquerda em 5,0V (jumper vermelho), o central (jumper

verde) em uma porta analógica e o da direita no GND (jumper preto). Pronto, temos nosso circuito com o LM35.

Cuidado para não inverter os conectores 5,0V e GND, isso pode danificar seus equipamentos. Você poderá notar que há inversão pela leitura da temperatura com valores elevados e/o aquecimento no sensor LM35.

Figura 39: Esquema do circuito com o sensor de temperatura LM35.



Fonte: os autores.

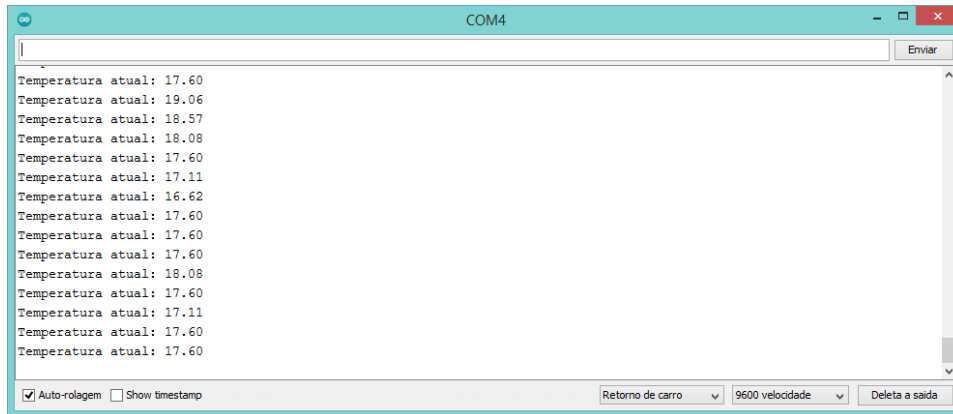
No quadro 17 temos o *sketch* para leitura do sensor LM35 e a saída do Monitor Serial. Note que o valor em volts precisa se multiplicado por 100 para que a saída seja em graus Celsius.

Quadro 17: *Sketch* para leitura do sensor LM35.

```
int pinoSensor = 0;
int valorLido = 0;
float temperatura = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  valorLido = analogRead(pinoSensor);
  temperatura = (valorLido * 5.00/1023);
  temperatura = temperatura * 100;
  Serial.print("Temperatura atual: ");
  Serial.println(temperatura);
  delay(1000);
}
```

Fonte: os autores.

Figura 39: Saída no Monitor Serial.

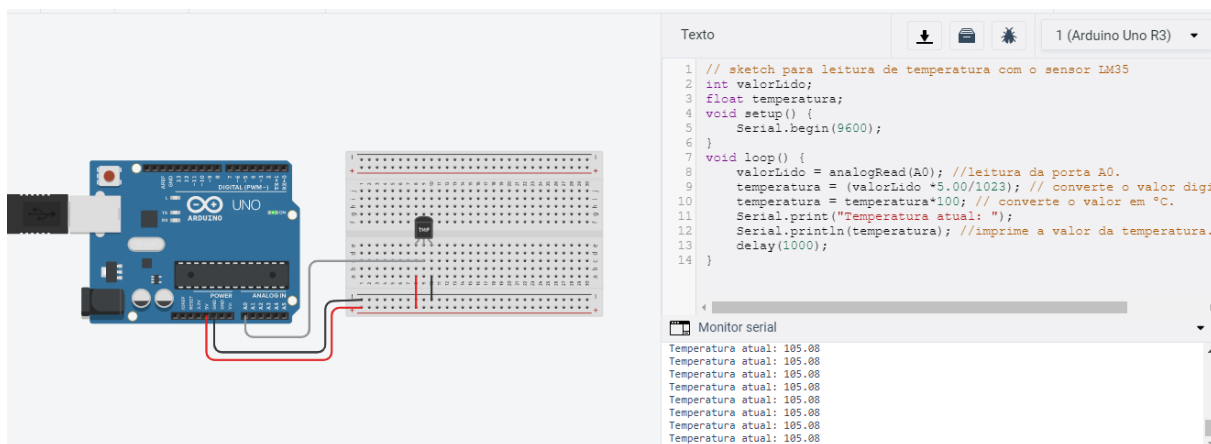


Fonte: os autores.

Note que na saída do Monitor Serial da figura 39 temos flutuações do valor da temperatura. Isso se deve às perturbações, tanto térmicas quanto eletromagnéticas. Para minimizar essa situação, é aconselhável usar um valor médio da temperatura, obtido após algum tempo de coleta de dados. Essa abordagem será solicitada como atividade 1 da semana, veja na seção "Para fazer".

No Tinkercad temos um sensor análogo ao LM35 que é o TMP36. A montagem do circuito é semelhante ao da figura 39 e o *sketch* pode ser o mesmo apresentado para o LM35, figura 39. Note que a leitura (por se tratar de um simulador) traz valores sem flutuação alguma.

Figura 39: Montagem do sensor TMP36, *sketch* e saída do Monitor Serial.



Fonte: Imagem gerada via TinkerCAD.com.

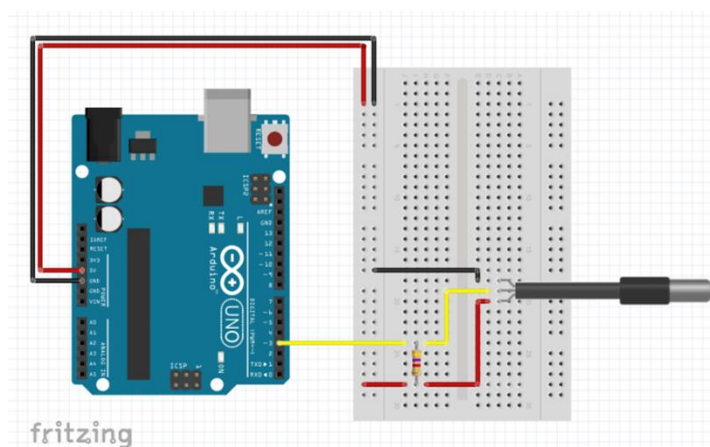
Nota: assumindo que o Tinkercad simula o intervalo entre -40°C até 125°C para o TMP36, é possível incluir uma linha de comando ao *sketch* sugerido para que você tenha no Serial Monitor todo o intervalo. Ente a linha 10 e 11 você pode fazer um deslocamento simples na variável temperatura, segue: "temperatura = temperatura - 50;". Outras possibilidades podem ser propostas a partir da conversão utilizada na linha 9. Ressalto ainda que a manutenção da escala com valores positivos tenha como objetivo manter o paralelismo entre o LM35 (a inclusão de valores negativos é um pouco mais complexa e não foi abordada na semana 7).

Em ambos os casos apresentados anteriormente temos os sensores (LM35 e TMP36) medindo a temperatura do ar. Mas para medir a temperatura de líquidos? Uma solução é usar encapsulamento. O sensor DS18B20 apresenta uma proteção para líquidos bastante interessante, veja a figura 40. Note que em sua ponta há uma blindagem em aço inoxidável. Outra possibilidade é encapsular o LM35 (ou similar) usando o espaguete termo retrátil. Leia o artigo recomendado para mais informação do encapsulamento do LM35.

CORRALLO, M. V.; JUNQUEIRA, A. de C. A Lei de esfriamento de Newton utilizando a automatização da tomada dos dados por meio do Arduino. *In: XXI Simpósio Nacional de Ensino de Física, 2015, Uberlândia. Anais [...]* São Paulo: Sociedade Brasileira de Física, 2015. Disponível em: <https://bitly.com/0QOGNb> Acesso em: 16 nov. 2021.

O sensor DS18B20 possui as seguintes especificações: tensão de operação: 3V-5,5V; faixa de medição: -55°C a $+125^{\circ}\text{C}$; precisão: $\pm 0.5^{\circ}\text{C}$ entre -10°C e $+85^{\circ}\text{C}$. O cabo vermelho deve ser ligado em VCC; cabo preto em GND; cabo amarelo em uma porta digital. Note na figura 59 (esquema da montagem) que usamos um resistor de proteção de $4,7\text{k}\Omega$.

Figura 40: Esquema de circuito para o sensor DS18B20.



Fonte: imagem gerada via Fritzing.

No quadro 18 temos o *sketch* para o sensor DS18B20. Como diferencial temos a incorporação de duas bibliotecas ("OneWire.h" e "DallasTemperatura.h") necessárias para o funcionamento do sensor. É importante observar previamente se ambas estão instaladas; atualizadas no IDE Arduino. Para isso clique em "Ferramentas" no menu superior do IDE Arduino. Em seguida, localize "Gerenciar Bibliotecas...", a partir daí você poderá localizar as duas bibliotecas e fazer a instalação e/ou a atualização para as versões mais recentes. A descrição mais detalhada do *sketch* pode ser observada nos comentários da própria figura 60.

Quadro 18: *Sketch* para o sensor DS18B20.

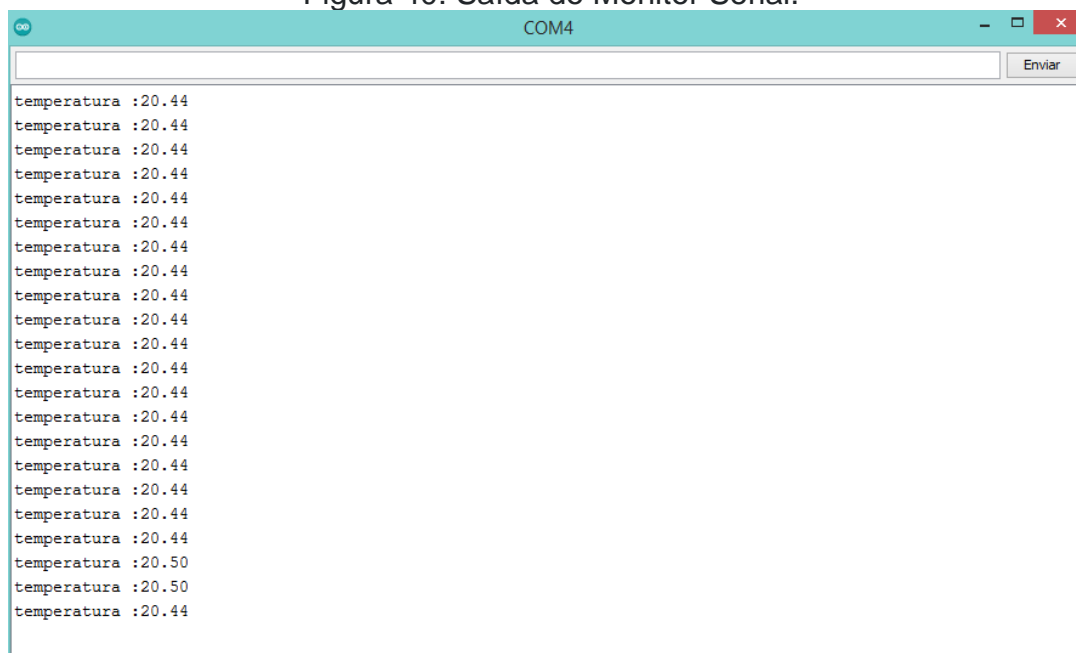
```
#include <OneWire.h> // inclusão da biblioteca OneWire
```

```
#include <DallasTemperature.h> // inclusão da biblioteca DallasTemperatura
#define ONE_WIRE_BUS 2 // O pino de dados é conectado na porta digital 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire); // passa uma referência oneWire para a biblioteca
DallasTemperature
void setup()
{
  Serial.begin(9600);
  sensors.begin(); // inicia a biblioteca
}
void loop()
{
  sensors.requestTemperatures(); // lê a temperatura
  Serial.print("temperatura :");
  Serial.println(sensors.getTempCByIndex(0));
  delay(1000);
}
```

Fonte: os autores

Com o *sketch* embarcado no Arduino, o resultado do Monitor Serial será o apresentado na figura 40. Observe que os valores flutuam muito pouco, diferentemente do LM35. Isso traz maior segurança para os resultados.

Figura 40: Saída do Monitor Serial.



```
COM4
Enviar
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.44
temperatura :20.50
temperatura :20.50
temperatura :20.44
```

Fonte: os autores.

No quadro 19 temos um *sketch* que permite apresentar a leitura no LM35 no navegador de Internet (veja a figura 68). Novamente temos o uso de duas bibliotecas "SPI.h" e "Ethernet.h". Para a transmissão dos dados foi utilizado *shield Ethernet* (veja figura 62), que permite a conexão de Internet via cabo RJ45.

Quadro 19: *Sketch* para apresentar a leitura no LM35 no navegador de Internet.

```
#include <SPI.h>
#include <Ethernet.h>
```

```
float time;
int pinoSensor = 0; // pino que está ligado o terminal central do LM35 (porta analógica)
0)https://eadcampus.spo.ifsp.edu.br/
int valorLido = 0; // valor lido na entrada analógica
float temperatura = 0; // valorLido convertido para temperatura
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED // MAC
};
IPAddress ip(192, 168, 0, 17);
EthernetServer server(80);
// Inicializa a Biblioteca Ethernet
// Define IP e porta 80
void setup() {
  Serial.begin(9600); //Inicializa comunicação serial
  Ethernet.begin(mac, ip); // inicia a comunicação Ethernet e Servidor
  server.begin();
}
void loop() {
  EthernetClient client = server.available();
  if (client){
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) { // envia configuração http
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close"); // a conexão se fechara depois de receber a resposta
          client.println("Refresh: 5"); // Atualiza página em 5 segundos
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          client.println("<body style=background:#DEE5F6>"); //cor fundo da tela
          client.println("<span style=color:#010101>"); //cor texto
          client.println("<font size=10 face=Times>"); //tamanho e fonte
          valorLido = analogRead(pinoSensor);
          temperatura = (valorLido*5.00/1023);
          temperatura = temperatura*100; //Converte milivolts para graus celcius, lembrando que a
          cada 10mV equivalem a 1 grau celcius
          client.print ("Temperatura da sala : ");
          client.print(temperatura,0);
          client.println ("<br />");
          client.println("</html>");
          break;
        }
      }
    }
  }
  client.stop();
  Serial.println("cliente desconectado");
}
```

Fonte: os autores.

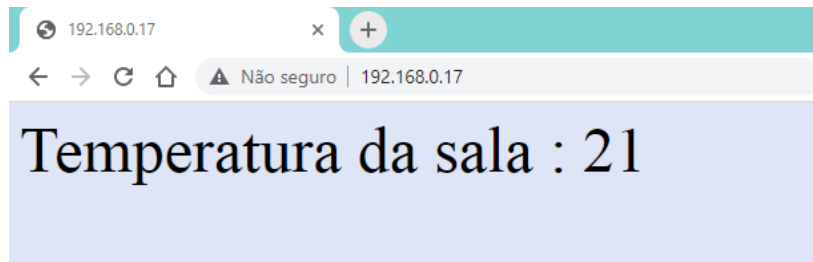
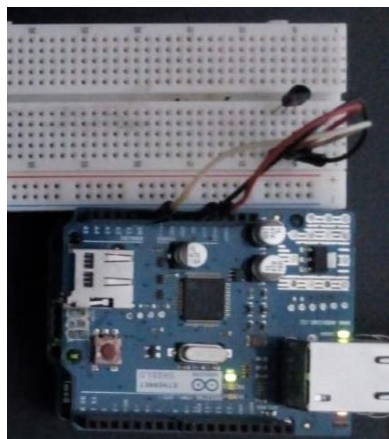


Figura 40: Montagem do circuito - LM35 com *Shield Ethernet*.



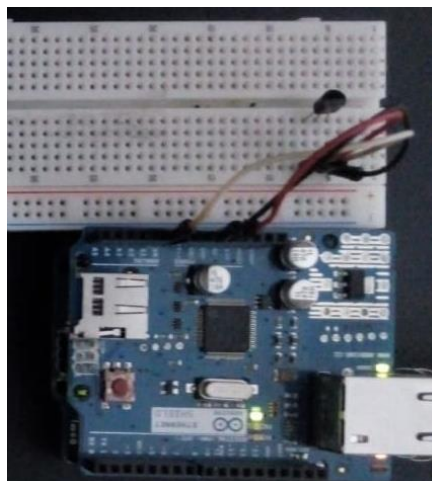
Fonte: os autores.

Sensor de Umidade e Temperatura DHT22 / AM2302

É possível encontrar sensores no mercado que, além de trazer a temperatura, podem apresentar outras grandezas físicas. O sensor DHT22 tem essa funcionalidade de medir a temperatura do ar, bem como a umidade relativa do ar. Ele permite medidas de temperatura de -40°C até 80°C e a umidade relativa do ar entre 0 e 100%. Possui uma incerteza de $0,5^{\circ}\text{C}$ para a temperatura e 2% para a umidade relativa. Sua faixa de operação é 3,0V até 5,0V e tem um tempo de resposta de 2 segundos.

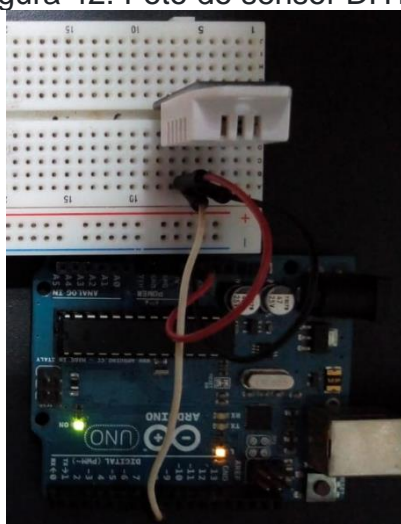
O circuito para DHT22 é bastante similar ao apresentado para o sensor DS18B20, mas não necessita de resistor de proteção. Para ligação, utiliza-se o VCC e GND (nas extremidades) e o pino central ligado em uma porta digital, veja o esquema da montagem na figura 65. O *sketch* também é bastante simples, mas é necessário fazer uso da biblioteca "DHT.h". Esta deve ser instalada/atualizada para o perfeito funcionamento do *sketch*. Na figura 66 temos o *sketch* e uma descrição dos principais comandos utilizados. E, finalmente, a figura 43 apresenta os valores da umidade relativa do ar e a temperatura do ar. Note que também existe pouca flutuação nos dados apresentados.

Figura 41: Montagem do circuito para DHT22.



Fonte: os autores.

Figura 42: Foto do sensor DHT22



Fonte: os autores.

Quadro 19: *Sketch* dos principais comandos utilizados.

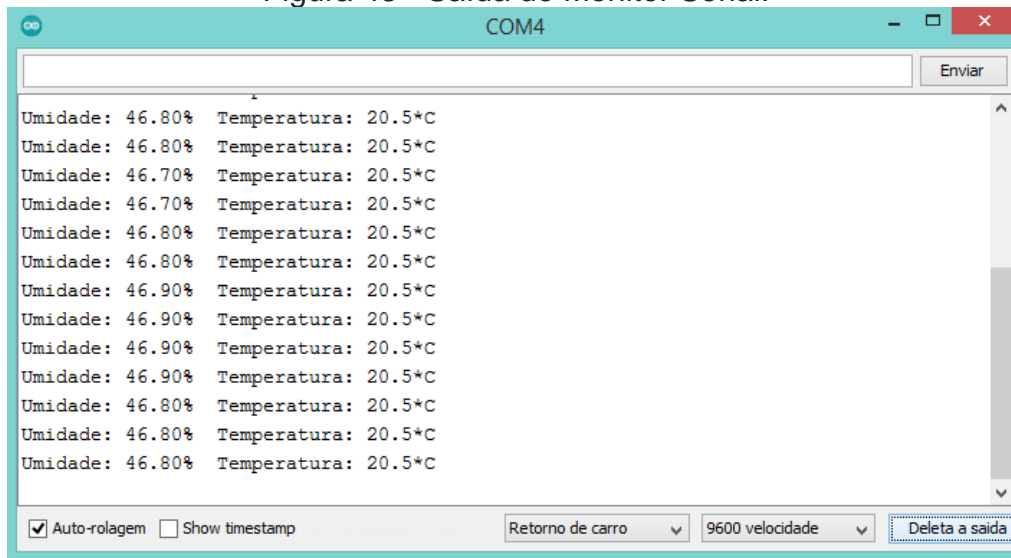
```
// Sensor de temperatura e umidade DHT22.
#include <DHT.h> //chama a biblioteca do sensor DHT.
DHT dht(7, DHT22); //passa os parâmetros para a função DHT, como o número da porta (7) e
                    //modelo do sensor (DHT22).

void setup(){
  Serial.begin(9600);
  dht.begin(); //inicializa a função DHT(
}

void loop(){
  Serial.print("Umidade: ");
  Serial.print(dht.readHumidity()); //imprime o valor da umidade relativa do ar.
  Serial.print("%");
  Serial.print(" Temperatura: ");
  Serial.print(dht.readTemperature(), 1); //imprime o valor da temperatura.
  Serial.println("°C");
  delay(2000);
}
```

Fonte: os autores.

Figura 43 - Saída do Monitor Serial.



Fonte: os autores.



Para ler – aspectos pedagógicos

Como sugestão de leitura para esta semana, recomenda-se o artigo "Contribuições do Arduino no ensino de Física: uma revisão sistemática de publicações na área do ensino".

MOREIRA, Michele Paulino Carneiro *et al.* Contribuições do Arduino no ensino de Física: uma revisão sistemática de publicações na área do ensino. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 35, n. 3, p. 721-745, 2018.



Para fazer

1) **Atividade 1** - Medindo temperatura com o sensor TMP36

Como apresentado na seção "Para ler", o sensor de temperatura LM35 traz alguma flutuação em suas leituras. Nesse sentido, pede-se que se desenvolva, como atividade da semana, um sketch no Tinkercad capaz de mitigar este problema, isto é, possa apresentar valores mais estáveis. Lembre-se que o Tinkercad não possui o sensor LM35, portanto, você deverá usar o TMP36. Como sugestão, deve-se medir muitos dados em um curto intervalo de tempo e apresentar, no Monitor Serial a cada 2 segundos, um valor médio dessas leituras. Outras estratégias podem ser usadas na resolução desse desafio.

Nota: assumindo que o TinkerCAD simula o intervalo entre -40°C até 125°C para o TMP36, é possível incluir uma linha de comando ao sketch sugerido para que você tenha no Serial Monitor todo o intervalo. Ente a linha 10 e 11 você pode fazer um deslocamento simples na variável temperatura, segue: "temperatura = temperatura - 50;". Outras possibilidades podem ser propostas a partir da conversão utilizada na linha 9. Ressalto ainda que a manutenção da escala com valores positivos tenha como objetivo manter o paralelismo entre o LM35 (a inclusão de valores negativos é um

pouco mais complexa e não foi abordada na semana 7. Para saber mais acesse <https://www.manualdaeletronica.com.br/sensor-temperatura-lm35-caracteristicas-aplicacoes/>) e o TMP36. Use o Tinkercad para criação do projeto.



Semana 8

Nesta seção você encontra as atividades propostas para a semana 8. É aconselhável a leitura dos materiais disponíveis na seção "Para ler – aspectos técnicos", sendo que o item trata de sensores e componentes capazes de aferir o movimento de objetos. Como leitura pedagógica, sugere-se o texto: "Ciclos de Modelagem: uma proposta para integrar atividades baseadas em simulações computacionais e atividades experimentais no ensino de física", que apresenta uma descrição do ciclo de modelagem de Hestenes. Na seção "Para fazer", pede-se a submissão do título e uma pequena descrição do projeto final do curso.

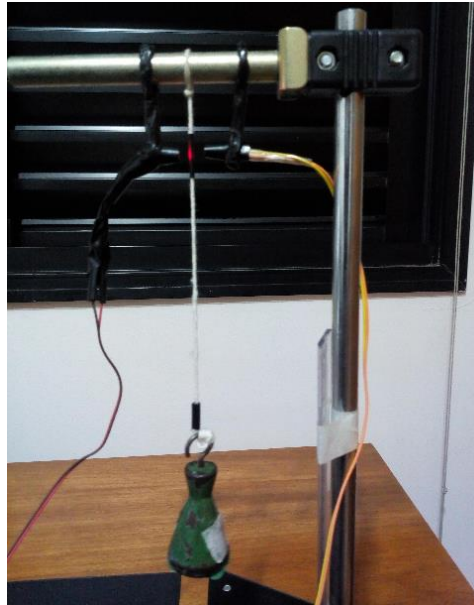


Para ler – aspectos técnicos

Medindo a posição de um objeto com LDR e LASER

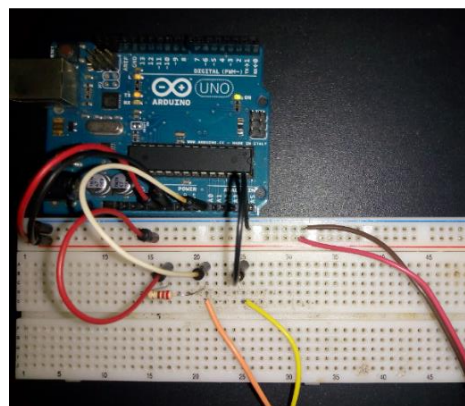
Dando continuidade aos estudos do movimento com o auxílio do Arduino, faremos agora a análise de um MHS (conhecido como pêndulo simples). A ideia é medir o período de oscilação utilizando um LASER e um LDR acoplados, como detector de posição (detalhes da montagem, veja figura 69 e 70). Posicionando ambos na região central do pêndulo (região tracejada), podemos interromper a luz LASER que chega ao LDR com a passagem do fio. O processo de coleta de dados se inicia com a soltura do pêndulo, com um ângulo $\theta < 20^\circ$ (condição para validação da equação 1). Com a primeira passagem em frente ao conjunto LDR+LASER, é acionado o contador do tempo. Cabe mencionar que apesar do fio passar rapidamente em frente ao conjunto LDR+LASER, esse tempo é bastante grande do ponto de vista do *clock* do Arduino. O tempo de passagem pelo conjunto LDR+LASER será considerado um tempo "morto", isto é, o *sketch* deve permanecer parado. Com o retorno do pêndulo à posição tracejada, tempo então 1/2 período contabilizado. É importante ressaltar que a passagem pela face oposta do feixe deve ser incluída também no tempo "morto", tendo em vista que poderá ser confundido com o retorno do pêndulo. O movimento se inverte e com o retorno, temos mais 1/2 período medido. Logo, a somatória das partes do período resulta no período completo (T) do nosso pêndulo simples. Mantendo a condição de $\theta < 20^\circ$ e um tempo relativamente curto (evitando perdas de energia), podemos considerar que a aceleração da gravidade é dada pela equação 1.

Figura 43: Montagem do aparato experimental.



Fonte: os autores.

Figura 44: Montagem do circuito com o Arduino.



Fonte: os autores.

Para a obtenção dos períodos e aceleração da gravidade, podemos usar, no IDE Arduino, o *sketch* abaixo. Como atividade complementar, disponibilizamos 6 vídeos com 5 ensaios cada. A ideia é explorar o estudo do pêndulo simples com o *software* Tracker, e comparar os resultados obtidos com o IDE Arduino e a sugestão é apresentar no encontro síncrono. A sugestão é utilizar o processo de linearização construindo o gráfico T^2 versus L . Obtenha uma reta ajustada e o seu coeficiente angular. A partir dele determine a aceleração da gravidade (g).

Quadro 20: *Sketch* par determinação da aceleração da gravidade.

```
1. //período do pêndulo simples
2. unsigned long tempoinicial, tempoatual;
3. double g, periodo_total;
4. const float pi = 3.1415; //valor de PI
5. const float h = 0.28; // comprimento do pêndulo em metros
6. const int tempo_morto = 10; // intervalo de passagem do fio pelo sensor
7. const float valor_limite = 1000; // valor limite para acionamento
8. bool primeirapassagem; //controla a passagem do fio
```



```
9. void setup() {
10. pinMode(A0, INPUT);
11. Serial.begin(9600);
12. primeirapassagem = true;
13. }
14. void loop() {
15.   if (analogRead(A0) > valor_limite && primeirapassagem == true) {
16.     primeirapassagem = false;
17.     tempoinicial = millis();
18.     delay(tempo_morto); // tempo morto
19.   }
20.   if (analogRead(A0) > valor_limite && primeirapassagem == false){
21.     tempoatual = millis();
22.     periodo_total = 2*(tempoatual-tempoinicial);
23.     periodo_total = periodo_total/1000;
24.     g=(4*pi*pi*h)/(periodo_total*periodo_total);
25.     delay(tempo_morto); // tempo morto
26.     Serial.print("Periodo = ");
27.     Serial.print(periodo_total);
28.     Serial.print(" s\t");
29.     Serial.print("gravidade local = ");
30.     Serial.print(g);
31.     Serial.println(" m/s2");
32.     primeirapassagem = true;
33.     delay(2000);
34.   }
35. }
```

Fonte: os autores.



Para ler – aspectos técnicos

Medindo a posição de um objeto com LED emissor IR e fototransistor IR

Outra possibilidade de determinação da posição de um objeto é utilizar o par LED emissor infravermelho (IR) e o fototransistor (receptor) IR. Eles operam na faixa de comprimento de onda de 940nm. A tensão de operação do LED emissor é de 1,2 VCC (depende do modelo) com uma corrente de 20mA. Já o fototransistor IR opera na faixa de 1,1 a 1,4VCC (depende do modelo).

Uma montagem simples para determinar a presença ou movimento de um objeto pode ser feita código abaixo iniciando na linha 1. Note que o receptor continuamente receberá o sinal IR emitido pelo LED. Com a passagem de um objeto, temos uma informação que pode ter interpretada pelo *sketch* abaixo:

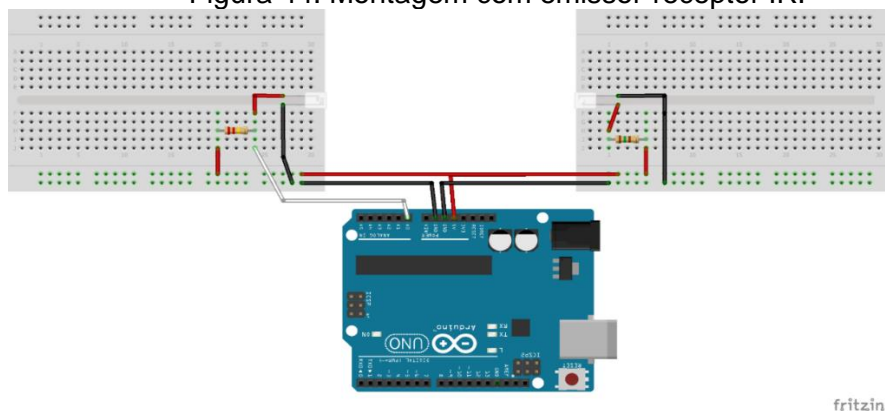
```
1. void setup(){
2.   Serial.begin(9600);
3.   pinMode(A0, INPUT);
4. }
5. void loop(){
6.   if(analogRead(A0) < 1023/2){
7.     Serial.println("Recebendo sinal de IR");
8.   }
9.   else
10.  {
11.    Serial.println("Sinal de IR interrompido");
12.  }
```

13. }

Note que há um limiar de corte na porta analógica A0 (linha 6) que pode ser ajustado para as condições do seu experimento. Para o exemplo, utilizou-se a tensão de operação no emissor por volta de $1,2V_{CC}$ (é importante observar as especificações do fabricante). Sugerindo, portanto, um resistor limitador com valores entre 150Ω e 180Ω (montagem da direita). No exemplo optamos por 150Ω . Por outro lado, no receptor IR utilizamos um resistor de $220k\Omega$ (montagem da esquerda).

Da mesma forma que utilizamos o par LDR-LASER para o pêndulo simples, podemos utilizar o par emissor-receptor IR para realizar a medição do período do pêndulo.

Figura 44: Montagem com emissor-receptor IR.



Fonte: Fritzing.

fritzing



Para ler – aspectos pedagógicos

Como sugestão de leitura para esta semana, recomenda-se o artigo "Ciclos de Modelagem: uma proposta para integrar atividades baseadas em simulações computacionais e atividades experimentais no ensino de física".

HEIDEMANN, L. A.; ARAUJO, I. S.; VEIT, E. A.. Ciclos de modelagem: uma proposta para integrar atividades baseadas em simulações computacionais e atividades experimentais no ensino de física. **Caderno Brasileiro de Ensino de Física**, Florianópolis,. Vol. 29, esp 2 (out. 2012), p. 965-1007, 2012.



Para fazer

1) Atividade - Descrição do projeto final

Como atividade assíncrona da semana 8, pede-se a submissão do título do projeto final com uma descrição sucinta.

Semana 9

Nesta seção você encontra as atividades propostas para a semana 9 (S9). É aconselhável a leitura dos materiais disponíveis na seção "Para ler", sendo que nos

dois primeiros itens você encontra uma discussão sobre o ciclo de Modelagem de Hestenes e uma aplicação do ciclo de Modelagem na física térmica. No terceiro item é discutido a integração do Arduino com a plataforma ThingSpeak Como leitura extra sobre o tema do ciclo de Modelagem, sugere-se o texto: "Modeling methodology for Physics teachers". Nesta semana sugere-se não ter atividade "Para fazer" devido a elaboração do projeto final.



Para ler - fundamentação teórica de física

Ciclo de Modelagem de Hestenes

Sobre o ciclo de Modelagem de Hestenes e seu desenvolvimento podem ser encontrados no artigo: "A Teoria da Modelagem de David Hestenes no Ensino de Ciências e Matemática", Disponível em: <https://revistapos.cruzeirodosul.edu.br/index.php/rencima/article/view/1154> Acesso em 23 out. 2021.

No quadro 21, proposto por (BREWE, 2008), encontramos um comparativo entre o ensino tradicional e o ensino centrado na Modelagem.

Quadro 21: Comparação entre o ensino focado na modelagem e os cursos tradicionais (BREWE, 2008).

Ensino centrado na Modelagem	Cursos Tradicionais
Modelos são construídos baseados em leis físicas e em condições de contorno.	Leis são apresentadas na forma de equações e são usadas na resolução de problemas.
Modelos são construídos com o auxílio de ferramentas de representação e, então, são usados para resolver problemas.	A resolução de problemas é predominantemente uma atividade de manipulação de equações.
Modelos são temporários e podem ser validados, refinados e expandidos.	O conteúdo é permanente; a validação já foi realizada.
Modelos são aplicados em situações físicas específicas.	Leis são aplicadas em situações físicas específicas.
A modelagem é um processo que é aprendido pelo acúmulo de experiência.	A resolução de problemas é um jogo que requer truques e é aprendida pela resolução de um grande número de problemas.
Modelos são distintos dos fenômenos que representam e podem incluir elementos causais, descritivos e preditivos.	O conteúdo é indistinguível do fenômeno físico.

FONTE: Heidemann, Araujo e Veit , 2012, p. 971.

No quadro 3, temos um exemplo de um exercício clássico utilizado nos cursos de física que foi adaptado para uma proposta visando a Modelagem.

Quadro 22: Exemplo de problema adaptado para atividades de modelagem (BREWE, 2008).

Problema Padrão	Problema de Modelagem
<p>Uma corda é usada para puxar um bloco de 3,57 kg ao longo de 4,06 m com velocidade constante em um piso horizontal. A força exercida pela corda sobre o bloco tem uma magnitude de 7,68 N e forma um ângulo de 15,0° com a superfície. Quais são (a) o trabalho realizado pela força da corda, (b) o aumento da energia térmica do sistema bloco-piso, e (c) o coeficiente de atrito cinético entre o bloco e o piso?</p>	<p>Construa o modelo mais completo que conseguir da seguinte situação: Um bloco de 3,57 kg é puxado com velocidade constante ao longo de 4,06 metros de um piso horizontal por uma corda. A força exercida pela corda sobre o bloco tem uma magnitude de 7,68 N e forma um ângulo de 15,0° com a superfície.</p>

FONTE: Heidemann, Araujo e Veit, 2012, p. 974.

Abaixo segue um extrato do artigo "Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores", que traz uma descrição das etapas do ciclo de Modelagem de Hestenes e sua articulação com a automatização de experimentos para o ensino de física.

Referência do artigo completo

CORRALLO, Marcio Vinicius; DE CARVALHO JUNQUEIRA, Astrogildo; SCHULER, Tunísia Eufrausino. Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 35, n. 2, p. 634-659, 2018.

O ciclo de Modelagem de Hestenes é constituído de dois estágios principais: 1º Estágio (Desenvolvimento do Modelo) e 2º Estágio (Implementação do Modelo) (HEIDEMANN; ARAUJO; VEIT, 2012). O 1º Estágio consiste de três divisões. Inicialmente, o problema é apresentado pelo professor (geralmente com uma demonstração ou discussão acerca do problema/questão), em seguida, os alunos (em pequenos grupos e de forma colaborativa) iniciam o processo de investigação e planejamento de experimentos que possam responder o problema proposto ou demanda. Finalmente, a apresentação e conclusões de forma oral/escrita das possibilidades encontradas pelos pequenos grupos. Indicando, assim, a formulação de modelos para o fenômeno em estudo, bem como sua avaliação do modelo a partir dos dados. O 2º Estágio é o momento de aplicar o modelo construído em novas situações (JACKSON; DUKERICH; HESTENES, 2008).

A fase da investigação é o momento que, possivelmente, os alunos irão se deparar com a necessidade de novos conhecimentos e novas ferramentas; logo, é nesse ponto que a introdução, pelo professor, de ferramentas como o Arduino e softwares são necessárias e passam a ter uma importância para a concretude das ideias elencadas pelos componentes dos grupos. Nesse sentido, Hestenes (1996) salienta que o professor deve estar atendo às necessidades dos grupos, fornecendo ferramentas que viabilizem o aprimoramento dos modelos e fomente a qualidade do discurso. Para o autor, é fundamental que o professor tenha em mente, durante a proposição do problema, o uso de habilidades e ferramentas de modelagem. Destaca, ainda, que a familiarização com a modelagem desenvolve uma visão sobre a estrutura do conhecimento científico durante a análise dos modelos e seu encaixe na teoria, e permite uma leitura melhor da epistemologia da Ciência com suas limitações e incertezas subjacentes. A cada etapa do processo é necessária a interação constante entre aluno-aluno e aluno-professor, permitindo um processo evolutivo de apropriação do conhecimento, ao mesmo tempo, que lhe dá sentido, haja vista que surgem necessidades impostas pelos próprios representantes do grupo. Nesse momento, são construídos esquemas mentais que permitem atuar sobre a situação posta, bem como rever suas concepções acerca do problema e, conseqüentemente, as soluções possíveis. Em outras palavras, o sujeito pode se deparar com conflitos cognitivos e, a partir de processos de resignificação, encontrar alternativas diferentes das convencionais e, em muitos casos, diferentes da que o professor possa ter como expectativa. Cabe destacar, ainda, que para assessorar na elaboração dos modelos, na apresentação dos resultados experimentais e da solução do problema, Hestenes sugere a adoção, pelos alunos, de uma pequena lousa branca (small whiteboard). Essa resignificação da atuação apoiada sobre pressupostos de uma ação reflexiva faz com que o ciclo de Modelagem de David Hestenes seja bastante adequado para a implementação com automatização de experimentos, juntamente com uma proposta problematizadora de atividades práticas experimentais.

Finalmente, temos no quadro 5 um resumo das etapas e os dois estágios sugeridos por Hestenes. Cabe salientar que a etapa final pode convergir com uma comunicação dos resultados, como a apresentação e publicação de artigo em congresso, participação em feira de ciências, publicação de artigo em periódico científico, entre outros.

Quadro 23: Resumo dos estágios e das fases dos ciclos de Modelagem propostos por Hestenes.



Primeiro Estágio: Desenvolvimento do modelo	<p>1) Discussão pré-laboratorial: professor apresenta o problema.</p> <p>2) Investigação: em pequenos grupos, os alunos trabalham no planejamento e na condução de experimentos.</p> <p>3) Discussão pós-laboratorial: em conjunto, os alunos apresentam e justificam as suas conclusões na forma oral e escrita por meio dos quadros brancos.</p>
Segundo Estágio: Implementação do modelo	<p>Alunos implementam o modelo recém confeccionado em outras situações.</p> <ul style="list-style-type: none"> • Problemas • Novos experimentos • Implementação computacional

FONTE: Heidemann; Araujo; Veit , 2012, p. 977.

Referências

BREWE, E. Modeling theory applied: Modeling Instruction in introductory physics. **American Journal of Physics**, v. 76, n. 12, p. 1155-1160, 2008. Disponível em: <http://dx.doi.org/10.1119/1.2983148>. Acesso em 03 fev. 2017.

HEIDEMANN, L. A.; ARAUJO, I. S.; VEIT, E. A. Ciclos de Modelagem: uma alternativa para integrar atividades baseadas em simulações computacionais e atividades experimentais no Ensino de Física. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 29, n. Especial 2, p. 965–1007, 2012.

JACKSON, J.; DUKERICH, L.; HESTENES, D. Modeling Instruction: An Effective Model for Science Education. *Science Educator*, v. 17, n. 1, p. 10–17, 2008.



Para ler - fundamentação teórica de física

Determinação do calor específico de um sólido sem calorímetro

Nesta semana exemplificaremos o processo de automatização da coleta de dados com um experimento clássico da física térmica, isto é, a determinação do calor específico de um sólido, mas sem o calorímetro. Cabe mencionar que a proposta do experimento pode ser utilizada em um episódio didático apoiado no ciclo de Modelagem de Hestenes, como descrito no artigo "Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores".

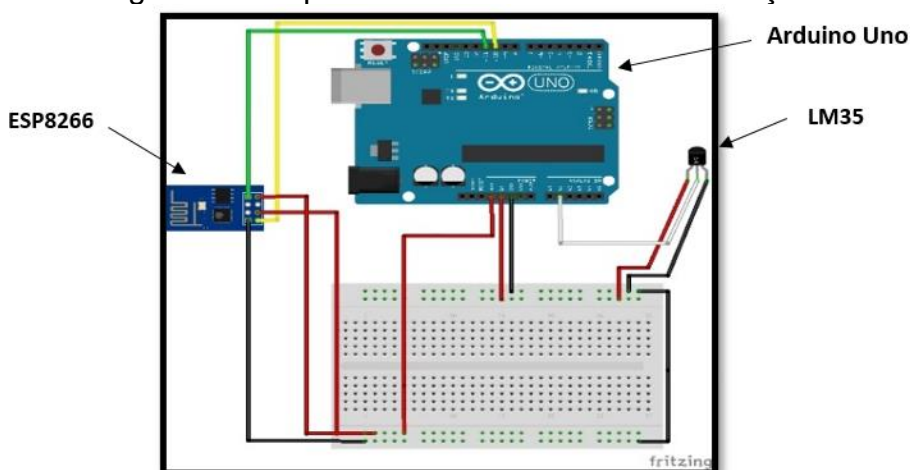
É importante lembrar que durante o processo de automatização de um experimento didático de física, principalmente com o apoio do Arduino e sensores acoplados, nos deparamos com uma grande quantidade de dados. Nesse sentido, as ferramentas computacionais, como planilhas eletrônicas ou *softwares* de tratamentos de dados científicos (por exemplo *software* SciDAVis - Clique no link: <http://scidavis.sourceforge.net/> para baixá-lo) são ferramentas poderosas e podem auxiliar na resolução de problemas e na construção de modelos matemáticos para um determinado fenômeno físico.

Voltando ao experimento mencionado, temos abaixo o extrato do artigo "Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores", que apresenta de forma pormenorizada um modelo matemático possível para determinação do calor específico de um bloco de alumínio imerso em um copo com água. Note que sua aplicação em sala de aula, sob os preceitos do ciclo de Modelagem de Hestenes, deve seguir um processo gradativo de construção e em total harmonia com as propostas levantadas pelo grupo de alunos. Cabe ao professor fornecer novos conhecimentos (matemáticos, computacionais ou físicos) à medida que o grupo evolui em suas hipóteses e necessidades. Evitando, portanto, a apresentação de uma proposta única e infalível durante a aula experimental.

Apesar da proposição de determinação do calor específico sem calorímetro já ter sido palco de outros trabalhos, como: Mattos e Gaspar (2003); Sias e Ribeiro-Teixeira (2006) e Silva *et al.* (2003), acreditamos haver espaço para novas abordagens. Nesse sentido, a inserção do referencial do ciclo de Modelagem de David Hestenes associado com elementos de automatização poderiam trazer o aprendizado de novos elementos procedimentais. Estes, por sua vez, poderiam trazer um pouco das vivências dos processos investigativos engendrados nas Ciências. Em geral, as atividades experimentais de Calorimetria se baseiam no uso do calorímetro como elemento auxiliar aos experimentos. Dessa forma, justificado como dispositivo que dificulta a troca de calor entre o meio externo e o experimento. Contudo, para a determinação do calor específico de um sólido, nota-se grande dificuldade em determinar a temperatura de equilíbrio entre o sólido e o líquido contidos no calorímetro (MATTOS; GASPAS, 2003). Exatamente por essa dificuldade, a proposição do experimento passou a ser interessante, especialmente no curso superior, pois a temperatura de equilíbrio pode ser determinada por modelagem e ajuste de curvas. Cabe ainda ressaltar que, apesar da complexidade matemática, encontramos em Sias e Ribeiro-Teixeira (2006) uma aplicação também para alunos da escola básica. Nesse contexto, entendemos que a proposição de determinação de calor específico sem calorímetro poderia ser um elemento desafiador aos nossos alunos do curso de extensão, uma vez que, para muitos dos professores cursistas, a modelagem e ajustes de curvas não são temas usuais na sala de aula da escola básica. Assim sendo, propusemos, como a fase inicial do 1º Estágio do ciclo de Modelagem de Hestenes, o desafio, aos cursistas, da determinação do calor específico de um sólido sem calorímetro, com o subsídio dos artigos Mattos e Gaspar (2003); Sias e Ribeiro-Teixeira (2006) e Silva *et al.* (2003). Após a leitura dos textos, optamos por uma discussão oral sobre o plano de trabalho, sem a obrigatoriedade de seguir a proposição de Hestenes (Hestenes, 1996), o qual sugere que o registro das estratégias seja realizado com o auxílio de uma pequena lousa branca. Assim, depois que os grupos apresentaram suas proposições, eles decidiram pela realização de uma sequência única de montagem e coleta de dados. Apesar de que os cursistas poderiam sugerir alternativas aos artigos, optaram em seguir a sequência experimental semelhante ao que foi sugerida pelos autores. Cabe comentar que os cursistas se asseguraram em estratégias já consolidadas, mesmo diante de liberdade para proposições diversas. Dessa forma, os cursistas propuseram o esquema de montagem utilizando um Becker (mas poderia ser um copo de vidro) para suporte para o líquido e o sólido (no caso foi utilizado um bloco cilíndrico de alumínio), balança, aquecedor e o sensor de temperatura (LM35) conectado ao Arduino. Na figura 2, temos o esquema de montagem do sistema de automatização de coleta dos dados desenvolvida pelos grupos. Nota-se a presença do Arduino conectado, via protoboard, ao sensor de temperatura LM35 e ao módulo WiFi ESP8266. Dando sequência à realização dos

experimentos, os cursistas aqueceram a água até 60°C, aproximadamente. Já com o sensor de temperatura LM35 imerso em água e devidamente impermeabilizado, inseriram o bloco de alumínio, e continuando a leitura da temperatura durante alguns minutos, de acordo com a sugestão de Mattos e Gaspar (2003). É importante mencionar que foram discutidas, durante os encontros, algumas limitações do experimento apresentadas em Mattos e Gaspar (2003). Vale ressaltar que os cursistas disponibilizaram também os dados via plataforma Thingspeak, permitindo que todos os grupos pudessem visualizar os dados dos demais grupos, em tempo real (tendo aqui uma alternativa à pequena lousa branca, para apresentação de resultados). A placa Arduino foi conectada via WiFi utilizando o módulo ESP8266 (veja Figura 2), e as rotinas adaptadas para isso estão disponíveis no Apêndice B. Porém, vale a pena frisar que a montagem dessa rotina se deu em um processo de construção durante todo o curso, isto é, discutimos a aplicação e a inserção de controle dos dispositivos LM35 e ESP8266 em outras situações experimentais (a programação do curso de extensão pode ser encontrada no Apêndice A). É preciso ponderar que a utilização de diversos elementos “novos”, simultaneamente, tende a aumentar a carga cognitiva, e, assim, comprometendo o propósito da atividade, isto é, a modelagem de um fenômeno físico. De posse dos dados, os cursistas passaram à fase de análise de gráficos e ajustes de curvas, isto é, a modelização propriamente dita. Para tal, fizeram uso do software livre SciDAVis (Scientific Data Analysis and Visualization), que também foi objeto de discussão em outros momentos do curso. A título de exemplificação, utilizaremos, na sequência, dados experimentais de um dos grupos para ilustrar o processo de modelização e comparação com o modelo teórico. Nesse sentido, faremos uma descrição pormenorizada dos procedimentos e ferramentas utilizadas para a modelagem. Contudo, gostaríamos de reiterar que não buscamos, nessa atividade, um novo modelo para explicar um fenômeno amplamente estudado, mas, sim, uma sequência didática que possa minimamente contribuir para refletir a respeito de paradigmas consolidadas sobre a Ciência, dentre eles a visão rígida e dogmática, tendo o método científico como um caminho único e infalível, como assinalam Gil-Pérez et al. (2001).

Figura 44: Esquema do sistema de automatização.



Fonte: Fritzing.

Após a coleta dos dados realizada pelo processo de automatização, esses dados foram salvos em um arquivo de texto simples em ASCII, com duas colunas separadas por tabulação. Em seguida, foram exportados para o software SciDAVis (veja no Apêndice C os detalhes de exportação e manipulação dos dados). Em acordo com as especificações técnicas dos fabricantes do sensor LM35 e do Arduino Uno, utilizamos 0,01s e 0,25°C de

incerteza para o tempo e temperatura, respectivamente. O fabricante do LM35 destaca que a incerteza pode variar entre o intervalo de $\frac{1}{4}$ até $\frac{3}{4}$ de 1°C . Em nosso estudo optamos por utilizar o valor mínimo da incerteza. Assumindo que a lei de resfriamento de Newton pode ser escrita como

$$\Delta T = T - T_R = (T_0 - T_R)e^{-\sigma t}, \quad (1)$$

em que T_0 é a temperatura inicial do líquido; T_R é a temperatura do reservatório (ambiente) e σ é a constante do tempo, que nos permite prever o tempo necessário para o resfriamento ou aquecimento de um corpo. Com o intuito de linearizarmos a equação 1, aplicamos o logaritmo natural em ambos os lados, cujo resultado segue:

$$\ln(T - T_R) = \ln(T_0 - T_R) - \sigma t. \quad (2)$$

Dessa forma, temos uma equação de 1° grau com as seguintes atribuições de x , y , coeficiente linear (B) e coeficiente angular (A)

$$\ln(T - T_R) = y, \quad (3)$$

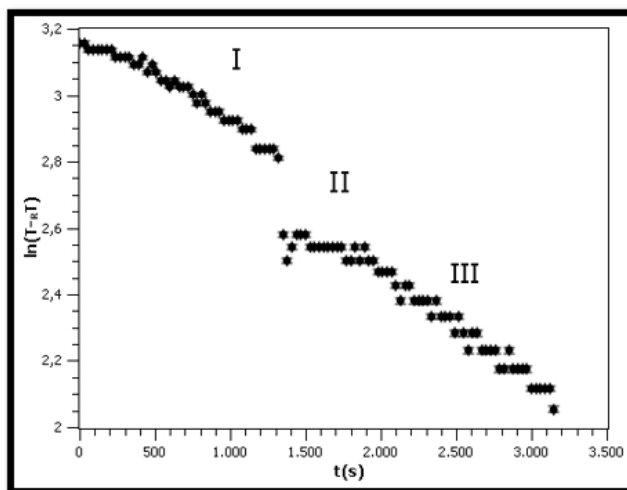
$$t = x, \quad (4)$$

$$\ln(T_0 - T_R) = B \quad (5)$$

$$-\sigma = A. \quad (6)$$

De posse dessas informações, foi possível plotar o gráfico – Curva de Resfriamento - (Gráfico 2), isto é, $\ln(T-TR)$ vs t . Nota-se que o gráfico possui três regiões distintas: I - resfriamento da água; II - introdução do sólido na água; III - resfriamento da água com o sólido. As retas da região I e III (Gráficos 3 e 4) podem ser ajustadas e prolongadas (este processo é descrito no Apêndice C), permitindo por extrapolação a determinação da temperatura da água no instante em que o sólido foi introduzido.

Gráfico 2: Curva de resfriamento.



Fonte: os autores.

Na sequência, com os valores obtidos a partir dos gráficos das Figuras 75 e 76 determinamos as temperaturas inicial da água e de equilíbrio. Assim, do gráfico da Figura 4 encontramos $y = 2,707$ e do gráfico da Figura 5 temos $y' = 2,542$. É importante ressaltar que o processo de automatização trouxe um número grande de pontos, permitindo, portanto, maior rigor na determinação da temperatura de equilíbrio (T_e). Também não se pode deixar de mencionar que esse sutil aprimoramento foi observado pelos cursistas, frente às proposições dos trabalhos de Mattos e Gaspar (2003); Sias e Ribeiro-Teixeira

(2006) e Silva et al. (2003), utilizados como base para o desenvolvimento do projeto.

Para o nosso ensaio temos: $m_{Al} = (68 \pm 1)$ g - massa do alumínio; $m_{\text{água}} = (103 \pm 1)$ g - massa da água; $c_{\text{água}} = (1,00)$ cal/g°C – calor específico da água; $T_R = (20,50 \pm 0,25)$ °C - temperatura ambiente; $T_0(\text{água}) = (35,5 \pm 0,2)$ °C – temperatura inicial da água na região II decorre de $y = 2,71 \pm 0,01$ (detalhes do cálculo da incerteza, vide Apêndice C); $T_e = (33,2 \pm 0,2)$ °C – temperatura de equilíbrio na região II decorre de $y' = 2,54 \pm 0,01$. Mesmo diante de um sistema aberto, presume-se que houve pouca variação na troca de calor antes e depois da introdução do bloco de alumínio na água. Logo, é razoável afirmar que $\Sigma Q = 0$ é válida na região II do gráfico da Figura 1 (MATTOS; GASPARGAS, 2003), portanto, temos:

$$C_{Al} = - \frac{m_{\text{água}} c_{\text{água}} (T_e - T_0(\text{água}))}{m_{Al} (T_e - T_R)} \quad (7)$$

Gráfico 3: Curva de resfriamento temperatura inicial da água.

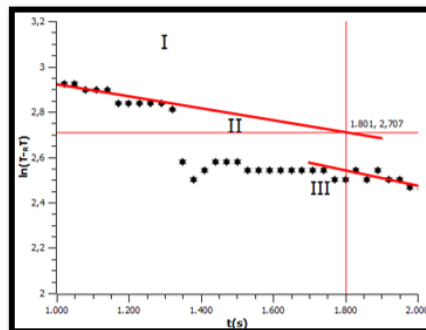
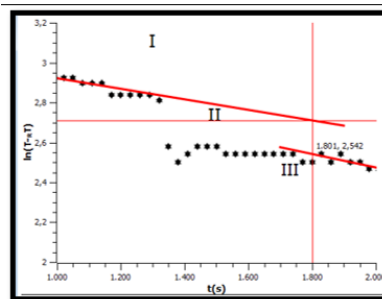


Gráfico 4: Curva de resfriamento temperatura de equilíbrio.



Fonte: os autores.

Baseado na fórmula de propagação de incertezas, conforme Voulo (1996, p. 113) aplicada na equação 7, realizamos a propagação de incertezas, conforme indicado a seguir:

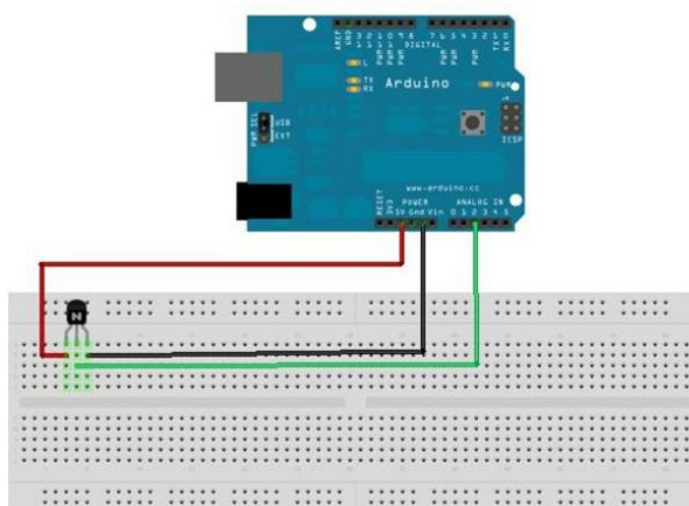
$$\sigma_{C_{Al}}^2 = \left(\frac{\partial C_{Al}}{\partial m_{\text{água}}} \right)^2 \sigma_{m_{\text{água}}}^2 + \left(\frac{\partial C_{Al}}{\partial T_e} \right)^2 \sigma_{T_e}^2 + \left(\frac{\partial C_{Al}}{\partial T_0(\text{água})} \right)^2 \sigma_{T_0(\text{água})}^2 + \left(\frac{\partial C_{Al}}{\partial m_{Al}} \right)^2 \sigma_{m_{Al}}^2 + \left(\frac{\partial C_{Al}}{\partial T_R} \right)^2 \sigma_{T_R}^2 \quad (8)$$

Desenvolvendo as derivadas parciais da equação 8 e substituindo os respectivos valores, encontramos que: $\sigma_{AL} = 0,05$ cal/g°C, assim o calor específico do bloco de alumínio pode ser descrito como: $CAL = 0,22 \pm 0,05$ cal/g°C. No apêndice C do mesmo artigo é apresentada uma pequena descrição/tutorial para utilização do SciDAVis na condução da análise dos dados do experimento.

Sketchs para medir a temperatura do líquido+sólido utilizando o LM35 ou DS18B20

A proposta aqui é coletar a temperatura de 100g de água acondicionada em um recipiente aberto com valor inicial de temperatura por volta de 60°C, como mencionado na descrição do experimento de determinação do calor específico de um sólido sem o uso de um calorímetro. A ideia é providenciar a imersão de um sólido (cilindro de alumínio), por exemplo de 4 g, no recipiente com água com valor por volta de 40°C. No quadro 24, sugerimos um *sketch* devolve, via Serial Monitor, os dados do instante e da temperatura do líquido. Note que o algoritmo calcula a média de um *array* de 5000 posições. Isto trará valores mais fidedignos da temperatura do líquido+sólido.

Figura 45: Esquema da montagem do projeto com LM35.



Fonte: Imagem gerada via Fritzing.

No quadro 6 sugerimos um *sketch* devolve, via Serial Monitor, os dados do instante e da temperatura do líquido.

Quadro 24: *sketch* devolve, via Serial Monitor

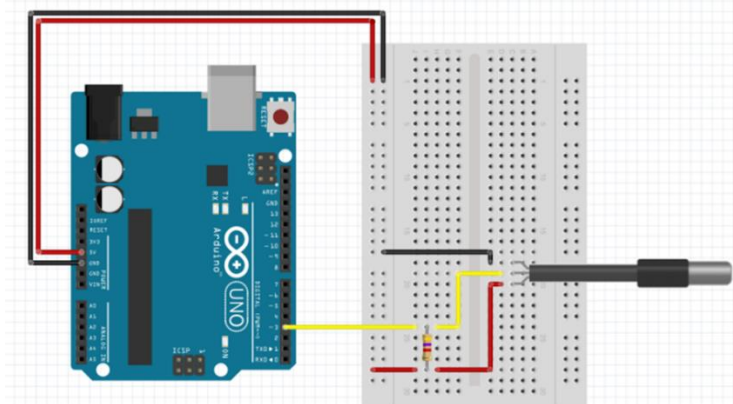
```
// curva de resfriamento - Determinacao do calor especifico de um solido sem calorimetro utilizando
o LM35.
float instante;
float tempinc [5000]; // array para armazenagem da temperatura
float somatemp=0; // acumula os valores da temperatura
void setup() {
  Serial.begin(9600);
}
void loop() {
  for (int x = 0; x < 5000; x++)
  {
    tempinc[x] = analogRead(A0); //faz a leitura do pino A0.
    tempinc[x] = (tempinc[x]*5.0/1023)*100; // 5V / 1023 = 0.00488 (precisao do A/D) e converte
milivolts para graus celcius, lembrando que a cada 10mV equivalem a 1 grau celcius
    somatemp+=tempinc[x]; // acumula o valor da temperatura
  }
  instante = millis(); // base de tempo para a coleta
  Serial.print (instante/1000,2);
  Serial.print("\t");
  Serial.print(somatemp/5000,2); //divide por 5000 a variavel somatemp
  Serial.println();
  somatemp = 0; // zera a variavel somatemp
  delay(5000); //espera 5 segundos para fazer nova leitura
}
```

Fonte: os autores.

Outra possibilidade para a coleta de dados do experimento supracitado é utilizar o sensor DS18B20 (a montagem pode ser encontrada na descrição da S7, mas sem a necessidade do uso do *array*. O *sketch*:

```
// curva de resfriamento - Determinacao do calor especifico de um solido sem calorimetro utilizando o
DS18B20
#include <OneWire.h> // inclusao da biblioteca OneWire
#include <DallasTemperature.h> // inclusao da biblioteca DallasTemperatura
#define ONE_WIRE_BUS 2 // O pino de dados eh conectado na porta digital 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire); // passa uma referencia oneWire para a biblioteca
DallasTemperature
void setup()
{
  Serial.begin(9600);
  sensors.begin(); // inicia a biblioteca
}
void loop()
{
  sensors.requestTemperatures(); // faz a leitura da temperatura
  Serial.print("temperatura :");
  Serial.println(sensors.getTempCByIndex(0));
  delay(5000);
}
```

Figura 45: Esquema da montagem do projeto com DS18B20.



Fonte: Fritzing.

No artigo "Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores", Apêndice B, disponível em você encontra uma sugestão de *sketch* que inclui a coleta de dados para o experimento "determinação do calor específico de um sólido sem calorímetro", mas com a possibilidade de compartilhamento, via Internet, dos dados em tempo real utilizando o módulo Wifi ESP8266 e integrado à plataforma ThingSpeak.

CORRALLO, M. V.; DE CARVALHO J. A.; SCHULER, T. E. Ciclo de Modelagem associado à automatização de experimentos com o Arduino: uma proposta para formação continuada de professores. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 35, n. 2, p. 634-659, 2018.

Referências

GIL-PÉREZ, D. *et al.* Para uma imagem não deformada do trabalho científico. **Ciência & Educação**, Bauru, v. 7, n. 2, p. 125–53, 2001.

HESTENES, D. Modeling Methodology for Physics Teachers. *In: International Conference on Undergraduate Physics Education, 1996, United States. Anais College Park*, 1996. Disponível em: <http://modeling.asu.edu/modeling/MODELING.PDF>. Acesso em 17 nov. 2021

MATTOS, C.; GASPAR, A. Uma Medida de Calor Específico sem Calorímetro. **Revista Brasileira de Ensino de Física**, São Paulo, v. 25, n. 1, p. 45–48, 2003.

SIAS, D. B.; RIBEIRO-TEIXEIRA, R. M. Resfriamento de um corpo: a aquisição automática de dados propiciando discussões conceituais no laboratório didático de física no ensino médio. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 23, n. 3, p. 360–381, 2006.

SILVA, W. P. da *et al.* Medida de Calor Específico e Lei de Resfriamento de Newton: Um Refinamento na Análise dos Dados Experimentais. **Revista Brasileira de Ensino de Física**, São Paulo, v. 25, n. 4, p. 392–398, dez. 2003.

VOULO, J. H. **Fundamentos da teoria de erros**. São Paulo: Blucher, 1996.

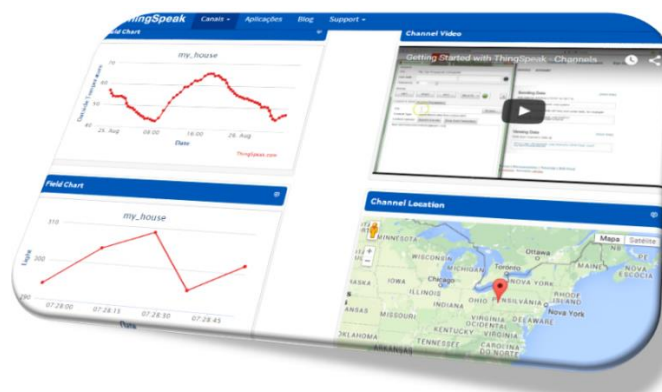


Para ler – aspectos técnicos

Plataforma de transferência de dados

Uma possibilidade relativamente simples de compartilhamento de dados, via Internet, é a partir da plataforma [ThingSpeak](https://thingspeak.com). É uma plataforma baseada na Internet das Coisas (IoT), que permite: o compartilhamento de dados em tempo real; a análise de dados usando o MATLAB; a visualização de dados a partir do MATLAB; a inserção de *plugins* (estes permitem a criação de *templates* para visualização); dentre outras funcionalidades para análise e compartilhamento de dados. O uso não comercial é gratuito, mas exige um intervalo de 15s entre os envios dos dados. Além do Arduino, a plataforma ThingSpeak permite também a integração com o módulo Wifi ESP8266 e o Raspberry Pi. Na figura 46 temos um exemplo de compartilhamento em um canal público hospedado no ThingSpeak, no qual é possível observar a leitura de temperatura, luminosidade, canal de vídeo e a localização dos sensores.

Figura 46: Exemplo de um canal público com leitura de sensor de temperatura, luminosidade, YouTube e localização.



Fonte: www.thingSpeak.com.



Para fazer uso da plataforma ThingSpeak é exigido um cadastro inicial. Após essa etapa, o passo seguinte para compartilhamento de dados é a criação de um novo canal. Para isso, basta clicar em "New Channel". Em seguida, preencha os campos, conforme figura 47. É importante lembrar que os campos que serão utilizados para transferência de dados dos sensores devem ser ativados. A plataforma ThingSpeak permite a coleta de dados de no máximo 8 campos por canal. Também é possível cadastrar sua localização geográfica a fim de realizar o compartilhamento público.

Figura 47: Criação de canal no ThingSpeak.



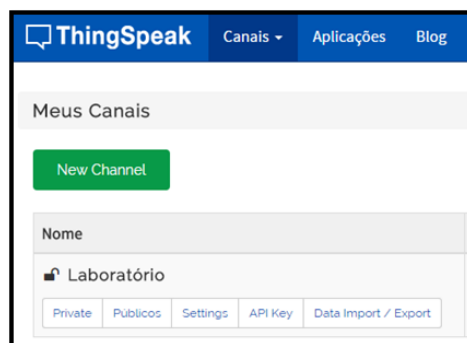
The image shows a web form titled "New Channel" with the following fields and options:

- Nome:
- Descrição:
- Campo 1: (containing "Resultado do campo 1") with a dropdown arrow.
- Campo 2: with a dropdown arrow.
- Campo 3: with a dropdown arrow.
- Campo 4: with a dropdown arrow.
- Campo 5: with a dropdown arrow.
- Campo 6: with a dropdown arrow.
- Campo 7: with a dropdown arrow.
- Campo 8: with a dropdown arrow.
- Metadata:
- Tags: (with a note "TAGS ARE COMMA SEPARATED")
- Latitude:
- Longitude:
- Altitude:
- Tomar Público?:
- URL:
- ID do vídeo: (with radio buttons for YouTube and Vimeo)
- Save Channel:

Fonte: www.thingspeak.com.

Depois de criado um novo canal, clique em Canais → Meus Canais. Você terá, nessa tela, um menu de opções, conforme a figura 48. Clicando no nome do seu canal é possível adicionar as visualizações pretendidas em seu canal. Nesse mesmo espaço é possível inserir as aplicações: MATLAB Analysis, MATLAB Visualization e outras aplicações. Em "Setting", é possível observar as configurações do canal, além do ID do canal (esse pode ser usado para leitura via *smatphone*). Em "API Key", encontramos as chaves para leitura e escrita. Essas chaves são necessárias para estabelecer comunicação com o IDE Arduino.

Figura 48: Configuração do canal.

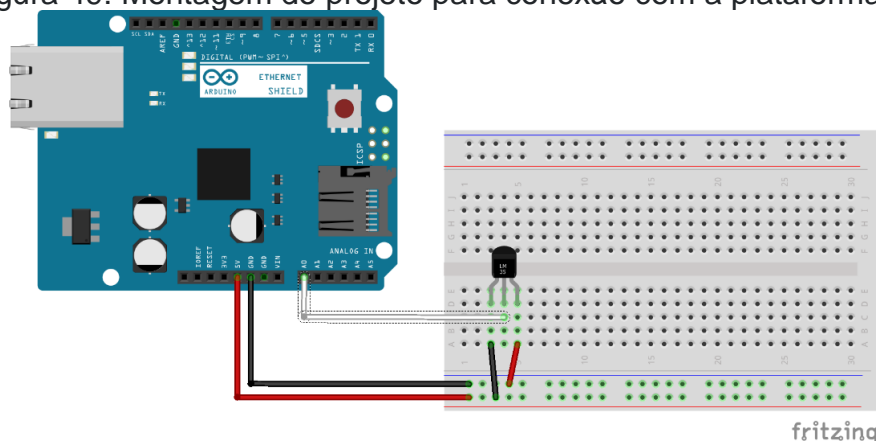


Fonte: www.thingspeak.com.

Projeto de integração do Arduino à Plataforma ThingSpeak

Depois de finalizada as configurações de seu canal na plataforma ThingSpeak, pode-se fazer uso do canal com um simples projeto de compartilhamento de dados de um sensor de temperatura LM35. A montagem está representada na figura 49. Note que temos o Arduino e um shield Ethernet para conexão à Internet.

Figura 49: Montagem do projeto para conexão com a plataforma ThingSpeak.



Fonte: Fritzing.

Quadro 25: Sketch que fará a comunicação com seu canal no ThingSpeak:

```
//Rotina baseada em https://github.com/iobridge/ThingSpeak-Arduino
//Examples/blob/master/Ethernet/Arduino_to_ThingSpeak.ino
#include <SPI.h> // chama a biblioteca SPI
#include <Ethernet.h> // chamada de biblioteca
byte mac[] = { 0xD4, 0x28, 0xB2, 0xFF, 0xA0, 0xA1 }; // configuracao - Rede Local
// Configuracao - ThingSpeak
char thingSpeakAddress[] = "api.thingspeak.com";
String writeAPIKey = "XXXXXXXXXXXXXXXXXX"; //o codigo deve ser copiado de seu canal
const int updateThingSpeakInterval = 15000; // delay para atualizacao do ThingSpeak (15 segundos)
float temperatura = 0;
long lastConnectionTime = 0;
boolean lastConnected = false;
int failedCounter = 0;
EthernetClient client; // Inicializa Ethernet Client
void setup()
{
  Serial.begin(9600);
  startEthernet(); // Start Ethernet
```



```

}
void loop()
  if (client.available())
  {
    char c = client.read();
    Serial.print(c);
  }
  if (!client.connected() && lastConnected)
  {
    Serial.println("...disconnected");
    Serial.println();
    client.stop();
  }
  // Atualiza o ThingSpeak
  if(!client.connected() && (millis() - lastConnectionTime > updateThingSpeakInterval))
  {
    String analogValue0 = String(analogRead(A0)*5.0/1023);
    updateThingSpeak("field1="+analogValue0);
  }
  // Check se o Arduino Ethernet necessita de atualizacao
  if (failedCounter > 3 ) {startEthernet();}
  lastConnected = client.connected();
}
void updateThingSpeak(String tsData)
{
  if (client.connect(thingSpeakAddress, 80))
  {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");
    client.print(tsData);
    lastConnectionTime = millis();
    if (client.connected())
    {
      Serial.println("Conectando com ThingSpeak...");
      Serial.println();
      failedCounter = 0;
    }
  }
  else
  {
    failedCounter++;
    Serial.println("Falha na conexao com ThingSpeak (" +String(failedCounter, DEC)+")");
    Serial.println();
  }
}
else
{
  failedCounter++;
  Serial.println("Falha na conexao com ThingSpeak (" +String(failedCounter, DEC)+")");
  Serial.println();
  lastConnectionTime = millis();
}
}
void startEthernet()
{

```



```
client.stop();
Serial.println("Conectando o Arduino com a Rede...");
Serial.println();
delay(1000);
if (Ethernet.begin(mac) == 0)
{
  Serial.println("DHCP falhou, atualizando o Arduino");
  Serial.println();
}
else
{
  Serial.println("Arduino conectando usando DHCP");
  Serial.println();
}
delay(1000);
failedCounter = 0;
}
```

Fonte: baseada em <https://github.com/iobridge/ThingSpeak-Arduino>

Na figura 50 podemos observar o compartilhamento de dados no canal 1 da leitura do LM35.

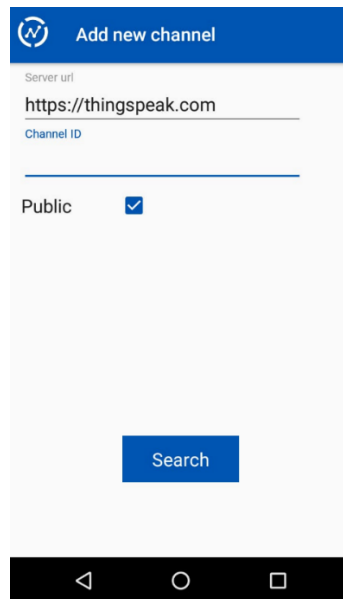
Figura 50: Leitura do canal 1 do LM35.



Fonte: www.thingspeak.com.

É importante destacar que os dados ficam armazenados e podem ser baixados como arquivo de extensão (.csv). Também é possível fazer a leitura via *smartphone*. Na figura 50 apresentamos o App ThingViewer que fazem a leitura em tempo real. A configuração é bastante simples, sendo necessário que o usuário insira o *Channel ID*. Lembrando que o *ID* é obtido na plataforma ThingSpeak.

Figura 50: Configuração do App do App ThingViewer.

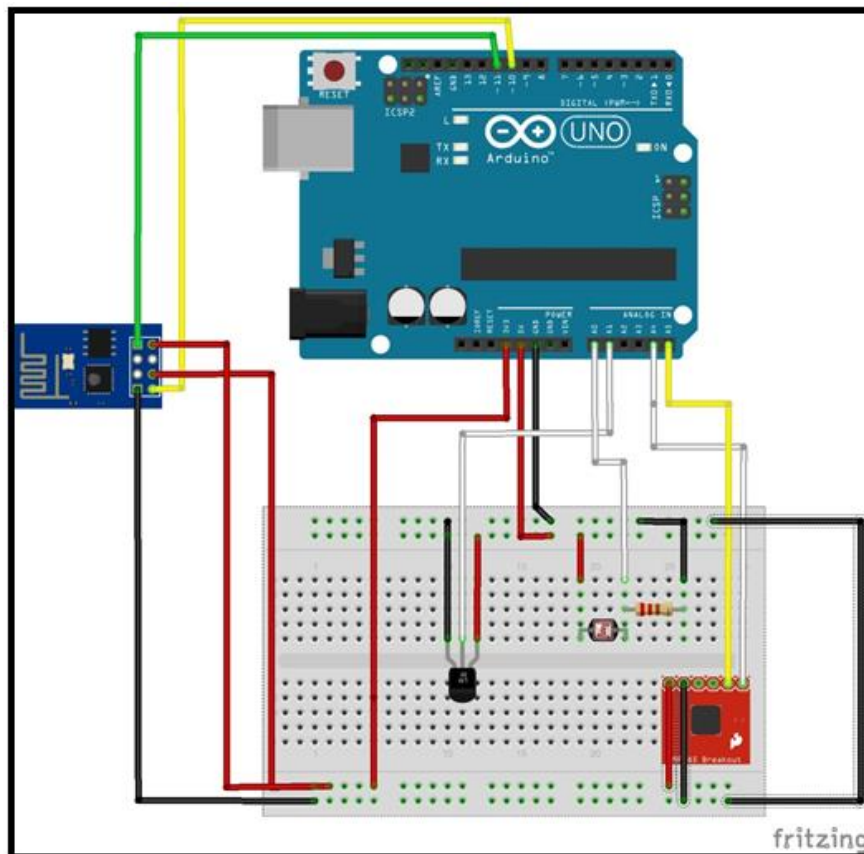


Fonte: App App ThingViewer

Comunicação via WiFi ESP8266 com ThingSpeak

O esquema apresentado na figura 51 tem por finalidade transmitir dados de sensores para a plataforma ThingSpeak. Para tal, utilizou-se os seguintes componentes: sensor barométrico BMP085, sensor de temperatura LM35, LDR, e um resistor de 220Ω.

Figura 51: Montagem do Projeto.



Fonte - Fritzing

Sketch de comunicação com o Thingspeak e o módulo Wifi ESP8266 (veja no quadro 26);

Quadro 26: Comunicação via WiFi ESP8266 com ThingSpeak

```
// Adaptado de http://electronut.in/an-iot-project-with-esp8266/
// conectar: BMP085 SCL - pino A5; SDA - pino A4; VCC - 3,3V /LDR leitura em A0, LM35 leitura em
A1
// Lembre-se que o modulo ESP8622 deve ser ligado em 3,3V.
#include <SoftwareSerial.h>
#include <stdlib.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp; // define funcao bmp
String apiKey = "XXXXXXXXXXXXXXXX"; //Chave para escrita ThingSpeak. Deve ser copiada do do
seu canal.
SoftwareSerial ser(10, 11); // TX, RX (comunicacao Arduino e ESP8622 via RX TX)
void setup()
{
  Serial.begin(9600);
  ser.begin(9600);
  !bmp.begin(); // inicia funcao BMP
  // inicia conexao rede WiFi
  String cmd="AT+CWJAP="; // a variavel String rede o comando AT para conexao Wifi
  cmd+="Digite nome da Rede WiFi"; // Rede Wifi
  cmd+="\",";
  cmd+="Senha da Rede WiFi"; // Senha rede Wifi
  cmd+="\",";
  ser.println(cmd); // executa a partir da ser o comando cmd
}
void loop() {
  String pressao = String(bmp.readPressure(), DEC);
  String temperatura_BMP = String(bmp.readTemperature(), DEC);
  String analogValue0 = String(analogRead(A0)*5.0/1023, DEC);
  String analogValue1 = String(analogRead(A1)*5.0*100/1023, DEC);
  Serial.print("LDR_(V): ");
  Serial.println(analogValue0);
  Serial.print("Temp_LM35 (C): ");
  Serial.println(analogValue1);
  Serial.print("Pressao_BMP085 (Pa): ");
  Serial.println(pressao);
  Serial.print("Temp_BMP085 (C): ");
  Serial.println(temperatura_BMP);
  // conexao via TCP
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
  ser.println(cmd);
  if(ser.find("Error")){
    Serial.println("AT+CIPSTART error");
    return;
  }
}
// preparando GET string
String getStr = "GET /update?api_key=";
getStr += apiKey;
getStr += "&field1=";
getStr += String(analogValue0);
getStr += "&field2=";
```



```
getStr += String(analogValue1);
getStr += "&field3=";
getStr += String(pressao);
getStr += "&field4=";
getStr += String(temperatura_BMP);
getStr += "\n\n\n";
// mandando dados length
cmd = "AT+CIPSEND=";
cmd += String(getStr.length());
ser.println(cmd);
if(ser.find(">")){
  ser.print(getStr);
}
else{
  ser.println("AT+CIPCLOSE");
  // alerta usuario
  Serial.println("AT+CIPCLOSE");
}
// thingspeak necessita de 15 segundos para a proxima atualizacao
delay(15000);
}
```

Fonte: Adaptado de <http://electronut.in/an-iot-project-with-esp8266/>

Diferentemente do *shield Ethernet* apresentado anteriormente, o módulo Wifi ESP8266 utiliza comandos AT, via Monitor Serial, para estabelecer a comunicação com a um rede Wifi e um roteador. Veja as informações sobre os comandos AT na figura 51.

Figura 51: Características do módulo Wifi ESP8266.

Conhecendo um pouco mais sobre o módulo ESP8266 e comando AT

Steps and note

AT+RST restart the module, received some strange data, and "ready"
AT+CWMODE=3 change the working mode to 3, AP+STA, only use the most versatile mode 3
(AT+RST may be necessary when this is done.)

Join Router

AT+CWLAP search available wifi spot
AT+CWJAP="you ssid", "password" join my mercury router spot
AT+CWJAP=? check if connected successfully, or use AT+CWJAP?

TCP Client

AT+CIPMUX=1 turn on multiple connection
AT+CIPSTART=4,"TCP","192.168.1.104",9999 connect to remote TCP server 192.168.1.104 (the PC)
AT+CIPMODE=1 optionally enter into data transmission mode
AT+CIPSEND=4,5 send data via channel 4, 5 bytes length (see socket test result below, only "elect" received), link will be "unlink" when no data go through

TCP Server

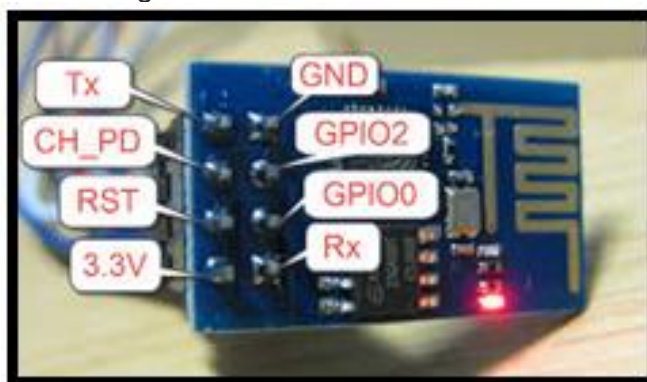
AT+CIPSERVER=1,9999 setup TCP server, on port 9999, 1 means enable
AT+CIFSR check module IP address
Fonte: <http://www.electrodragon.com/w/ESP8266>



Módulo EPS8266

Reset do ESP8266 pode ser acionado em GND.
Deve-se parar a execução da rotina no Arduino. Uma possibilidade é carregar uma instrução nula.
Conexão RX – porta Digital 0; TX - porta Digital 1.
Comando AT com Monitor Serial: Ambos, NL e CR e 9600bps

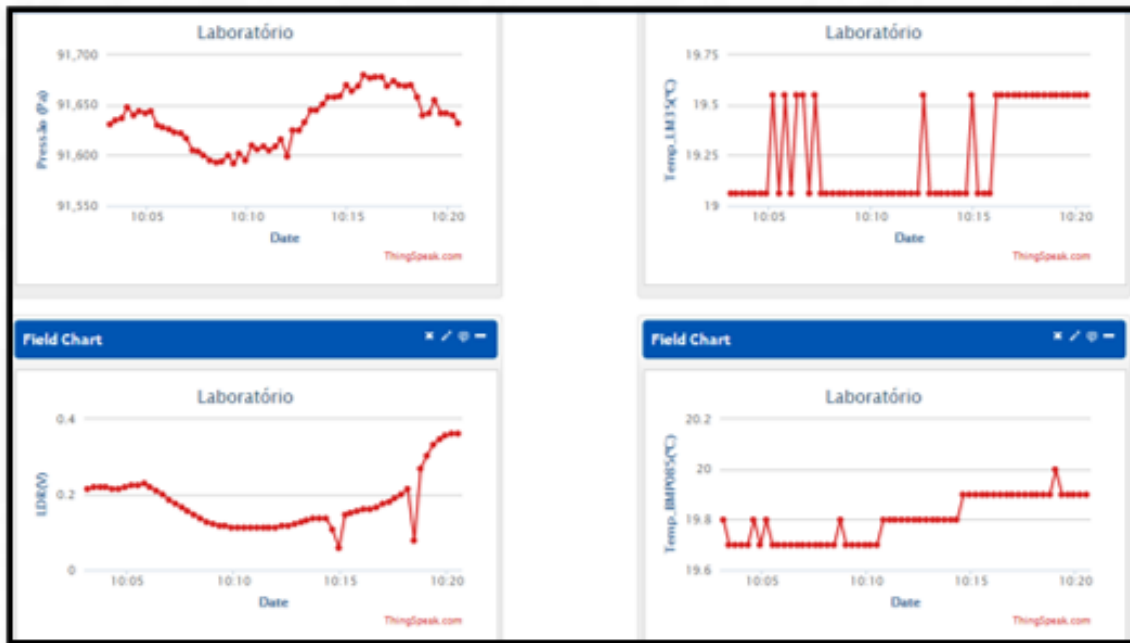
Figura 52: Módulo Wifi ESP8266.



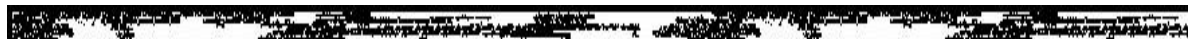
Fonte: www.electrodragon.com

Como resultado do *sketch*, teremos o compartilhamento no canal do ThingSpeak, conforme a figura 53.

Figura 53: Dados em tempo real do canal do ThingSpeak.



Fonte: www.thingspeak.com.



Semana 10

Finalizamos o curso com a semana 10. Sugere-se deixar um espaço para a postagem do *link* e exibição dos vídeos com os projetos desenvolvidos pela turma.

O projeto pode ser apresentado em dupla ou individual, conforme a metodologia escolhida. Reserve o espaço para o aluno compartilhar o *link* do projeto final na plataforma e deixe por um determinado tempo para visitaç o.

Fim

REFERÊNCIAS

ANDRADE, A.; CORRALLO, M. V. Uma Análise das Influências da cultura maker e TinkerCAD no ensino de física. *In: Seminário de Iniciação Científica do Litoral Norte, 2020, Caraguatatuba, SP. Anais do SICLN IFSP CAR, 2020. Disponível em: <https://ocs.ifspcaraguatatuba.edu.br/sicln/x-sicln/paper/view/339/108>. Acesso em: 14 mar 2021.*

BACICH, L.; MORAN, J. **Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática.** Porto Alegre, RS, Penso Editora, 2018.

BRASIL. Base Nacional Comum Curricular (BNCC). **Educação é a Base.** Brasília, MEC/CONSED/UNDIME, 2017. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_sit_e.pdf. Acesso em: 13 mar. 2021.

BRASIL. Base Nacional Comum Curricular (BNCC). **Educação é a Base.** Brasília, MEC/CONSED/UNDIME, 2017. Disponível em http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=79611-anexo-texto-bncc-aprovado-em-15-12-17-pdf&category_slug=dezembro-2017-pdf&Itemid=30192 Acesso em: 24 out. 2021

FORTUNATO, I.; TARDIN, M. L. P. Um inventário das teses e dissertações sobre cultura maker. **Ciências em Foco**, Campinas, SP, v. 13, p. e020016-e020016, 2020.

LEMKE, Jay L. Letramento metamidiático: transformando significados e mídias. **Trabalhos em Linguística Aplicada**, v. 49, n. 2, p. 455-479, dez. 2010.

MEIRA, S. L. B.; RIBEIRO, J.L. P. A Cultura Maker no ensino de física: construção e funcionamento de máquinas térmica. **Fablearn Brazil**, Stanford, EUA, v. 1, 2016.

SILVA, W. C. **Aplicando a computação física e o arduino para o apoio ao ensino de programação com base na abordagem motivacional ARCS: uma proposta de curso a distância com o uso de simulador.** 2018, 94f. TCC - Universidade Federal da Paraíba, Paraíba, 2018. Disponível em: <https://repositorio.ufpb.br/jspui/bitstream/123456789/15756/1/WCS08022019.pdf> Acesso em: 17 nov. 2021.

SCOTT, R. M.; DORTMANS, D.; RATH, C.; BOIN, J.; MEEUSSEN, N. Makerspace in the Primary Grades: Best Fieldtrip Ever. **Teaching & Learning**, United States, v. 12, n. 1, p. 1-14,