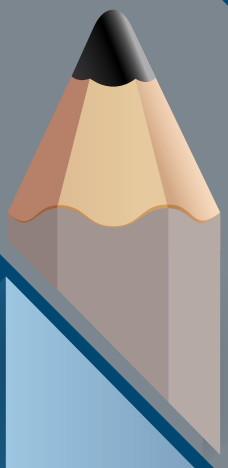
ROBÓTICA



ESCOLA DE TEMPO INTEGRAL

CADERNO DO PROFESSOR

ENSINO MÉDIO

GOVERNO DO ESTADO DE SÃO PAULO
SECRETARIA DA EDUCAÇÃO

MATERIAL DE APOIO
AO PROGRAMA ENSINO INTEGRAL
DO ESTADO DE SÃO PAULO

ROBÓTICA

ENSINO MÉDIO
CADERNO DO PROFESSOR

Primeira edição

2014

São Paulo

Governo do Estado de São Paulo

Governador

Geraldo Alckmin

Vice-Governador

Guilherme Afif Domingos

Secretário da Educação

Herman Jacobus Cornelis Voorwald

Secretária-Adjunta

Cleide Bauab Eid Bochixio

Chefe de Gabinete

Fernando Padula Novaes

Subsecretária de Articulação Regional

Raquel Volpato Serbi Serbino

Coordenadora da Escola de Formação e Aperfeiçoamento dos Professores – EFAP

Silvia Andrade da Cunha Galletta

Coordenadora de Gestão da Educação Básica

Maria Elizabete da Costa

Coordenadora de Gestão de Recursos Humanos

Cleide Bauab Eid Bochixio

Coordenadora de Informação, Monitoramento e Avaliação Educacional

Ione Cristina Ribeiro de Assunção

Coordenadora de Infraestrutura e Serviços Escolares

Dione Whitehurst Di Pietro

Coordenadora de Orçamento e Finanças

Claudia Chiaroni Afuso

Presidente da Fundação para o Desenvolvimento da Educação – FDE

Barjas Negri

Prezado(a) professor(a),

Em dezembro de 2011, a Secretaria da Educação do Estado de São Paulo instituiu o Programa Educação – Compromisso de São Paulo, que tem como um de seus pilares expandir e aperfeiçoar a política de Educação Integral, como estratégia para a melhoria da qualidade do ensino e, portanto, para o avanço na aprendizagem dos alunos.

Nesse contexto, foi criado, em 2012, o Programa Ensino Integral, com o objetivo de assegurar a formação de jovens autônomos, solidários e competentes por meio de um novo modelo de escola. Esse novo modelo, entre outras características, prevê jornada integral de alunos, currículo integrado, matriz curricular diversificada, Regime de Dedicação Plena e Integral dos educadores e infraestrutura que atenda às necessidades pedagógicas do Programa Ensino Integral. Essa estrutura visa proporcionar aos alunos as condições necessárias para que planejem e desenvolvam o seu Projeto de Vida e se tornem protagonistas de sua formação. O Programa, inicialmente direcionado a escolas de Ensino Médio, teve sua primeira expansão em 2013, quando passou a atender também os anos finais do Ensino Fundamental. O Programa deverá continuar sua expansão nos segmentos que já atende e ampliar sua atuação na Educação Básica, compreendendo também escolas dos anos iniciais do Ensino Fundamental.

Esta série de cadernos contempla um conjunto de publicações que se destina à formação continuada dos profissionais que atuam no Programa Ensino Integral e também ao apoio dos adolescentes e jovens em busca de uma aprendizagem bem-sucedida. Os cadernos ora apresentados têm um duplo objetivo: por um lado, oferecer subsídios para otimizar o uso dos laboratórios, com base nas diretrizes que fundamentam este Programa; por outro, destacar estratégias metodológicas que, em todos os componentes curriculares, concorrem para que os estudantes possam ampliar suas competências na área de investigação e compreensão – para observar, descrever, analisar criticamente os diferentes fenômenos de cada área, levantar hipóteses que os expliquem e propor iniciativas para mudar a realidade observada. A série é composta pelas seguintes publicações:

- Biologia: atividades experimentais e investigativas
- Ciências físicas e biológicas: atividades experimentais e investigativas
- Física: atividades experimentais e investigativas
- Manejo e gestão de laboratório: guia de laboratório e de descarte
- Matemática – Ensino Fundamental – Anos Finais: atividades experimentais e investigativas
- Matemática – Ensino Médio: atividades experimentais e investigativas
- Química: atividades experimentais e investigativas
- Pré-iniciação científica: desenvolvimento de projeto de pesquisa
- Robótica – Ensino Fundamental – Anos Finais
- Robótica – Ensino Médio

Pretende-se, dessa maneira, contribuir para que as escolas desenvolvam atividades experimentais e investigativas nos laboratórios, nos segmentos a seguir:

- Ensino Fundamental – Anos Finais: nas aulas de Ciências Físicas e Biológicas e de Matemática; nas aulas de Práticas Experimentais e nas aulas de disciplinas eletivas, dependendo da especificidade dos temas e conteúdos selecionados.
- Ensino Médio: nas aulas de Biologia, Física e Química, da 1ª a 3ª séries; nas aulas de Prática de Ciências, na 1ª e 2ª séries; nas aulas de disciplinas eletivas, da 1ª a 3ª série, dependendo da especificidade dos temas e conteúdos selecionados; e em atividades para o desenvolvimento de Projetos de Pré-iniciação Científica dos alunos.

Bom trabalho!

Equipe do Programa Ensino Integral

SUMÁRIO

ORIENTAÇÕES SOBRE OS CONTEÚDOS DO CADERNO

O fascínio que os jovens demonstram pela tecnologia pode ser usado de maneira muito recompensadora em atividades práticas de diversas disciplinas. A robótica, em suas diversas formas e aplicações, é um dos instrumentos de tecnologia que permitem essas práticas.

A proposta do Caderno de Robótica é possibilitar aos estudantes exercitar os principais conceitos desse ramo da tecnologia do componente curricular Física, sem deixar de lado as interfaces que naturalmente ocorrem com todos os demais. A plataforma Arduino, por combinar simplicidade e versatilidade para a realização de experimentos, pode ser usada como instrumento para concretizar os objetivos da proposta. Com as atividades descritas e todas as que podem ser desenvolvidas a partir delas, o professor tem a possibilidade de trabalhar aspectos do conteúdo programático de Física, de maneira a atrair a atenção dos estudantes e motivá-los para o estudo das diversas teorias.

Os fenômenos físicos envolvidos nos projetos visaram contemplar as três séries do Ensino Médio, explorando a montagem de diferentes projetos eletrônicos. Os projetos selecionados funcionam como uma base sólida para o conhecimento dos fundamentos da robótica.

Para a 1ª e a 2ª série, o foco dos projetos está nos resultados alcançados pela montagem dos protótipos eletrônicos. Para a 3ª série, todos os projetos podem ser trabalhados sob a ótica do funcionamento dos circuitos montados. Em todas as séries, a lógica de programação é trabalhada. Assim, os estudantes podem explorar o funcionamento de cada projeto com diferentes olhares, de acordo com os assuntos estudados na série que estão cursando.

As atividades investigativas proporcionadas pelo uso do *kit* favorecem a aplicação imediata de conceitos estudados em sala de aula. Elas permitem, ao mesmo tempo, que o professor enriqueça o seu repertório de atividades a ser abordadas com os estudantes e que cada um deles analise e compreenda os fenômenos no seu próprio ritmo. Além disso, a realização de projetos simples desperta a curiosidade para a construção de projetos diferentes e mais complexos. Com isso, são oferecidas aos professores e aos estudantes ferramentas para exercer a sua criatividade.

A plataforma Arduino apresenta código aberto e conta com uma grande comunidade de usuários organizada em *sites* e fóruns na internet. O resultado disso é uma farta quantidade de referências e tutoriais, disponíveis gratuitamente. Dessa maneira, há o incentivo adicional de, depois de ter conseguido compreender a base de funcionamento do sistema, acessar um grande número de projetos de referência, que podem ser usados como exercícios, ser melhorados ou usados como fonte de inspiração.

A disposição de fazer os projetos funcionarem como o planejado virá em grande parte de forma espontânea, dos próprios estudantes. Assim, a avaliação do aprendizado pode ser realizada com base no resultado imediato do projeto, ou seja, se ele foi bem-sucedido ou não.

Nas atividades de experimentação em laboratório, entretanto, a verificação da aprendizagem não necessariamente está atrelada ao sucesso na realização do projeto. Na prática, aprende-se

tanto com os sucessos quanto com os fracassos. O importante nas atividades de projeto é que todos os estudantes consigam realizá-las e aprendam com os erros e os acertos. As formas que você pode usar para avaliar o desempenho dos estudantes são muitas. Por exemplo: pedir a eles que expliquem como funciona o circuito, as características de cada componente e as aplicações da tecnologia. Normalmente, ao finalizar a atividade, eles estarão ansiosos para compartilhar com você, professor, e com os colegas os resultados alcançados. A empolgação desse momento pode ser aproveitada para começar o processo de avaliação. Você, professor, terá a oportunidade de verificar o poder que o simples acendimento de um LED, a partir de um conjunto desconexo de componentes e fios, possui sobre a motivação dos estudantes.

Este Caderno envolve atividades que não exigem conhecimento anterior da plataforma Arduino, nem conhecimento prévio de programação de computadores. O objetivo é colocar qualquer novo usuário da plataforma à vontade com o seu uso.

Ao final da descrição de cada projeto, há uma seção para a discussão dos resultados, que faz uma ligação direta com os conteúdos abordados nos Cadernos do Professor de apoio ao Currículo do Estado de São Paulo (São Paulo faz escola). Essas discussões podem ser feitas em sala de aula, após a realização das experiências.

Recomenda-se que você, professor, se concentre em duas frentes para poder fazer projetos cada vez mais interessantes para os estudantes: programação em linguagem C e interação do Arduino com outros componentes. Lembre-se de que a experimentação fundamentada na teoria é a base do aprendizado. Combinando os seus conhecimentos de Física com a eletrônica, você pode alcançar resultados muito interessantes e motivar os estudantes. Dessa maneira, eles estarão aptos para que, em seguida, possam seguir com seus próprios projetos.

AS POSSIBILIDADES DE USO DA ROBÓTICA NO ENSINO MÉDIO

Nos últimos anos, em todo o país, têm sido realizadas algumas iniciativas para o uso da robótica como ferramenta de auxílio ao aprendizado no Ensino Médio. Essas iniciativas procuram associar as habilidades adquiridas pelo trabalho dos conteúdos vistos em sala de aula com as atividades relacionadas ao funcionamento de robôs e de sistemas automatizados.

O QUE É UM ROBÔ?

De uma maneira objetiva, o que vem a ser exatamente um robô? Um robô é definido de uma forma bastante geral pela Figura 1. Os sensores de diversas naturezas enviam sinais para uma unidade de processamento, que trata esses sinais e realiza atividades a partir deles, por meio de atuadores.

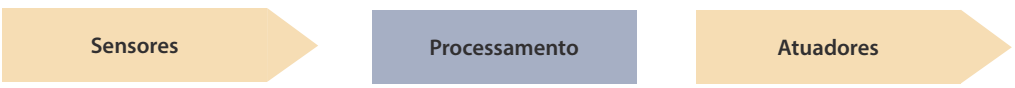


Figura 1 – Componentes de um robô.

Apesar da simplicidade mostrada na Figura 1, é possível, em virtude da grande variedade de sensores e atuadores, associada à liberdade de fazer o processamento de diferentes formas, empregar a robótica em uma quantidade muito grande de aplicações.

Apesar de estar fortemente associada à ficção científica, a robótica está cada vez mais próxima do nosso dia a dia. Um dos exemplos disso são os aspiradores robôs comercializados por algumas empresas de eletrodomésticos no mercado brasileiro. Na Figura 2, é possível ver a aparência que esses equipamentos possuem.

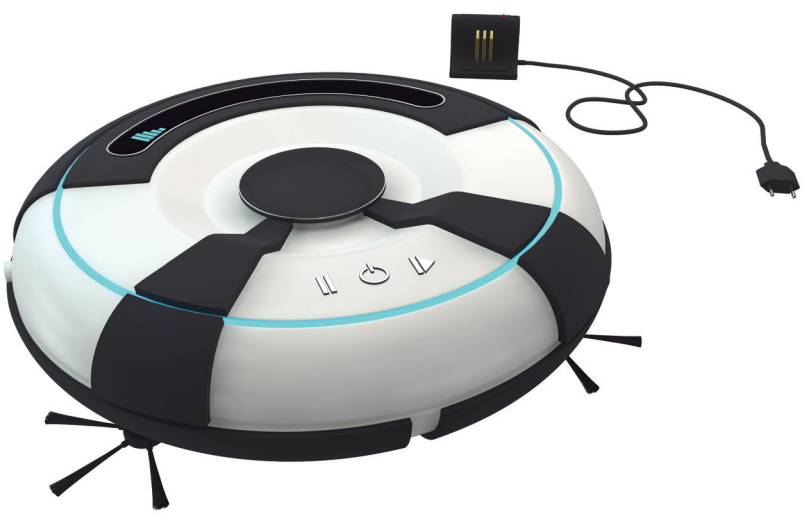


Figura 2 – Robô aspirador.

Eles são construídos basicamente na forma de disco e leem a geometria do ambiente no qual estão localizados, fazendo trajetórias de idas e vindas que buscam cobrir toda a área a ser limpa. Podem ser programados para fazer a limpeza em horas específicas e dirigem-se para o terminal de recarga quando o seu nível de energia está baixo. Eles ainda se protegem de acidentes, evitando degraus e obstáculos.

Nas indústrias, os robôs já são uma realidade há muitos anos, executando operações perigosas para trabalhadores ou que não podem ser feitas com a precisão necessária pelo trabalho humano (por exemplo, nas operações de soldagem de carrocerias de automóveis produzidos em série).

BENEFÍCIOS DA ROBÓTICA NA FORMAÇÃO DOS JOVENS

Os benefícios do desenvolvimento de atividades de robótica na formação dos jovens acontecem de diversas maneiras. Entre elas, podem ser destacadas:

- O desenvolvimento da interdisciplinaridade, já que o conjunto de componentes mostrado na Figura 1, em geral, mistura assuntos que são estudados, tradicionalmente, de maneira isolada durante o ciclo formal. Por exemplo, para fazer um motor ser acionado pela mudança de temperatura, os assuntos de Cinemática, Termologia e Eletricidade são, necessariamente, considerados em conjunto.
- O desenvolvimento de projetos ajuda a criar nos estudantes o senso de trabalho em equipe e possibilita a apreensão muito clara de noções de relações de causa e efeito.
- O princípio básico de solução de problemas é também amplamente trabalhado, pois a percepção do que são dados, processamento e resultados fica bastante evidente. Além disso, as soluções de compromisso, que são a premissa básica de grande parte das decisões profissionais que os estudantes terão de tomar no futuro, são também adotadas naturalmente.
- O procedimento de teste de hipóteses, uma das bases do método científico, também é exercitado. Por exemplo, se por algum motivo o projeto não funcionar, há normalmente várias causas possíveis. Os estudantes, então, intuitivamente, elaboram testes que procuram verificar cada uma delas, começando com as mais prováveis, até que o projeto seja bem-sucedido.

Quando são confrontadas atividades experimentais de robótica com os conhecimentos trabalhados no Ensino Médio, é possível planejar práticas envolvendo grande parte do conteúdo estudado. Após a fase em que o Arduino e os componentes do *kit* forem apresentados, essas possibilidades serão discutidas a partir dos projetos de aplicação.

Este Caderno pode, então, servir como uma base para o alcance de voos mais elevados na área de robótica.

A PLATAFORMA ARDUINO

O QUE É O ARDUINO?

O Arduino é uma plataforma de código aberto para a construção de protótipos eletrônicos. Ele possui como principal característica a facilidade de uso. A plataforma foi criada por um grupo de professores e estudantes do Instituto de Design Interativo de Ivrea, na Itália, em 2005, com o objetivo de conceber uma opção moderna e barata para os seus estudantes desenvolverem projetos envolvendo eletrônica. O nome Arduino foi escolhido em homenagem ao rei Arduino (955–1014), que nasceu na região de Ivrea, a mesma do instituto no qual a plataforma foi desenvolvida, e teve um papel histórico importante.

O projeto Arduino possui um *site* oficial, disponível em: <http://arduino.cc> (acesso em: 18 jul. 2014). Nele, há muitas informações úteis sobre a plataforma e as suas características.

A plataforma permite que sejam feitas atividades como:

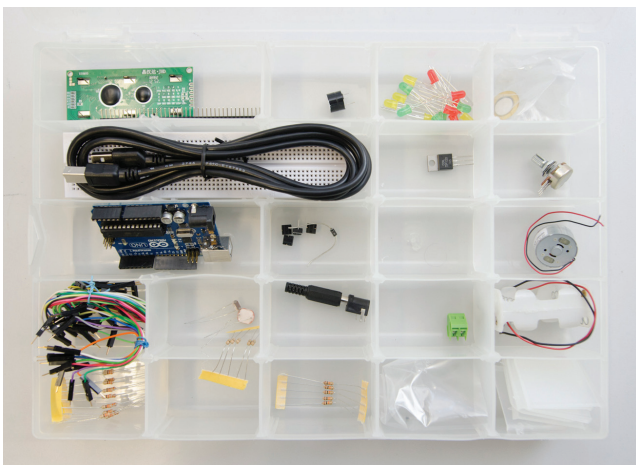
- entrada de dados por meio de sensores de diversas naturezas;
- processamento dos dados vindos dos sensores, por equações;
- envio de sinais a dispositivos diversos, a partir dos resultados do processamento.

O processamento das informações é feito por um ambiente de desenvolvimento que pode ser baixado na própria página do projeto Arduino. A linguagem de programação usada possui uma sintaxe muito próxima à linguagem C/C++.

OS COMPONENTES DO *KIT*

O *kit* utilizado neste Caderno é composto por uma série de itens que serão detalhados a seguir. Uma visão geral, com a caixa plástica e todos os componentes organizados, é mostrada na Figura 3.

Durante a apresentação do *kit*, cabe uma observação importante. Muitos dos componentes são pequenos e frágeis. Eles podem ser facilmente perdidos ou danificados se forem esquecidos em determinados locais, por exemplo. Desenvolva nos estudantes, desde a primeira atividade no laboratório, o hábito de manter os componentes organizados e separados por tipo, depois de cumpridas as tarefas. Isso dá trabalho e exige disciplina, mas proporciona resultados imediatos e, uma vez que se tenha adquirido o hábito, não se trabalha mais de outra forma.



© Fernando Genaro/Fotoarena

Figura 3 – *Kit* de Robótica.

Arduino Uno

Existem vários tipos de placa Arduino. A mais comum é a Arduino Uno. Ela é muito versátil e funciona com um grande número de sistemas operacionais, além de poder ser conectada a uma grande variedade de sensores e atuadores, alguns dos quais fazem parte do *kit*. Para ser programada, a placa deve ser acoplada a um computador, que normalmente também é sua fonte de energia elétrica. Pela conexão de sistemas de baterias à placa, seu funcionamento pode ser independente do computador. A Figura 4 mostra a placa original Arduino Uno.

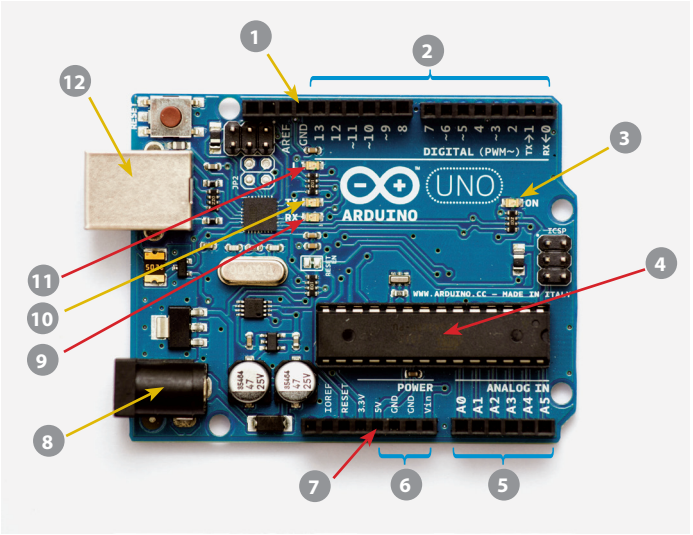


Figura 4 – Placa original Arduino Uno e seus principais componentes.

De acordo com a numeração da Figura 4, existem na placa os seguintes componentes:

1 GND (*ground*) ou pino terra – referência com voltagem 0 V para os circuitos, de tal forma que a corrente possa passar pelos elementos dos circuitos e retornar a eles, passando por esse pino.

2 Entradas/saídas digitais, numeradas de 0 a 13 – servem como entrada ou saída de sinais digitais para os sensores ou atuadores. As com o sinal "~" ao lado são do tipo PWM (*pulse witdh modulation* ou modulação por largura de pulso). A técnica PWM consiste em simular um sinal analógico por meio de sinais digitais. Isso permite usar o Arduino para acionar e/ou controlar dispositivos que requerem entrada analógica, utilizando as portas digitais.

3 LED indicador de estado ligado/desligado do Arduino – fica aceso sempre que o Arduino está conectado a uma fonte de energia.

4 Microprocessador – realiza o processamento dos sinais que entram e saem do Arduino. É o elemento programável da plataforma.

5 Entradas analógicas, numeradas de 0 a 5 – servem como entrada de dados vindos de sensores analógicos.

6 GND (*ground*) ou pinos terra – referência com voltagem 0 V para os circuitos, de tal forma que a corrente possa passar pelos elementos dos circuitos e retornar a eles, passando por esses pinos.

7 Pino de alimentação de 5 V para os elementos dos circuitos – alimenta com corrente os sensores e/ou atuadores dos circuitos.

8 Conector para alimentação externa – permite que o Arduino, após ser programado, funcione independentemente do computador. Pode usar qualquer fonte externa com tensões entre 6 V e 20 V. Por exemplo, uma fonte de *notebook*.

9 LED RX, de recepção de dados – acende todas as vezes que o Arduino recebe dados do computador.

10 LED TX, de transmissão de dados – acende todas as vezes que o Arduino transmite dados para o computador.

11 LED associado ao pino 13.

12 Porta USB – faz a recepção e a transmissão de dados com o computador. Também é responsável por alimentar de energia o Arduino.

A Tabela 1 mostra as características técnicas da placa Arduino Uno.

Características Técnicas da placa Arduino Uno (rev. 3)	
MICROCONTROLADOR	ATmega328
VOLTAGEM DE OPERAÇÃO (V)	5
VOLTAGEM DE ENTRADA RECOMENDADA (V)	7 a 12
LIMITES DA VOLTAGEM DE ENTRADA (V)	6 a 20
PINOS DE ENTRADA E SAÍDA DIGITAIS	14 (dos quais 6 produzem saída PWM)
PINOS DE ENTRADA ANALÓGICOS	6
CORRENTE CONTÍNUA POR PINO DE ENTRADA/SAÍDA (mA)	40
CORRENTE CONTÍNUA PARA O PINO DE 3,3 V (mA)	50
MEMÓRIA *FLASH* (kB)	32, dos quais 0,5 usado pelo *bootloader*
SRAM (kB)	2
EEPROM (kB)	1
CLOCK (MHz)	16

Tabela 1 – Características técnicas da placa Arduino Uno (rev. 3).

Há no mercado placas similares ao Arduino, chamadas de clones do Arduino[1]. Isso acontece pelo fato de ele ser uma plataforma de código aberto tanto para *software* quanto para *hardware*.

[1] "Arduino" é uma logomarca registrada e não pode ser utilizada sem autorização.

Protoboard

Os projetos eletrônicos materializam-se pela utilização de placas de circuito impressas, nas quais os componentes são conectados permanentemente entre si, por solda ou outros tipos de conexão. Entretanto, durante a fase de desenvolvimento dos sistemas, é mais prático construir os circuitos na forma de protótipos. Para isso, são usadas as *protoboards*, também chamadas de *breadboards*. A *protoboard* é uma placa composta por vários orifícios conectados entre si, organizados em uma camada plástica, como mostrado na Figura 5. Existem *protoboards* de diversos tamanhos. O que diferencia uma da outra e define o tamanho é o número de pinos de conexão. No *kit* utilizado como referência neste Caderno, é usada uma *protoboard* de 840 pinos.

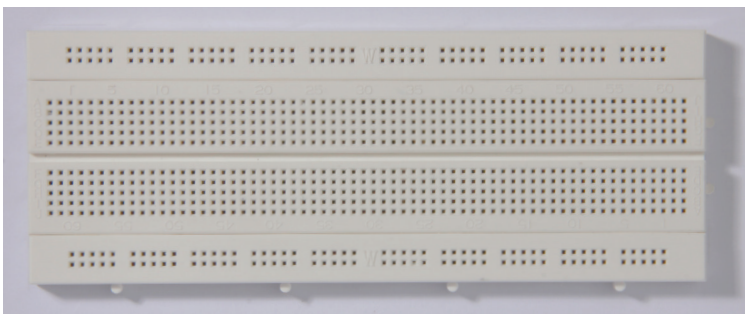


Figura 5 – *Protoboard*.

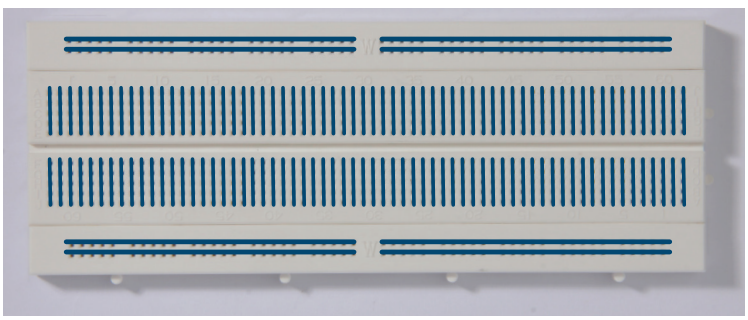


Figura 6 – Representação esquemática dos contatos internos de uma *protoboard* de 840 pinos.

Para que os componentes possam estar em contato uns com os outros, o arranjo físico interno da *protoboard* é tal que a camada abaixo da que contém os furos é formada por um conjunto de placas metálicas que farão os contatos elétricos entre os componentes, conforme mostrado na Figura 6. De acordo com as características do projeto a ser montado, esse arranjo é usado de maneira a conectar os componentes. Em geral, as duas linhas paralelas maiores e horizontais, na Figura 6, tanto na extremidade superior quanto na inferior, são usadas para conectar os componentes às fontes de alimentação do circuito ou ao pino terra. As partes menores, verticais na Figura 6, são usadas para ligar os componentes entre si e estes com a fonte de alimentação ou ao pino terra. Entretanto, você tem liberdade para montar os circuitos como quiser, desde que as conexões sejam as mesmas especificadas nos projetos. Em cada projeto a ser trabalhado, será mostrada a montagem na *protoboard* e o desenho esquemático do circuito.

Display LCD

O *display* LCD (*liquid crystal display*) é usado para mostrar informações na forma de sequências de caracteres alfanuméricos. Por exemplo, ele pode informar, para o usuário, valores de temperatura, de velocidade, de intensidade luminosa, de valores processados, ou mensagens de qualquer natureza. O *display* LCD possui pinos, um para cada coluna, pelos quais são enviadas as informações a ser mostradas. O *display* do *kit* é mostrado na Figura 7. Ele pode escrever mensagens em duas linhas de 16 colunas, sendo, por isso, chamado de 16 × 2. No *kit*, um pente de contatos para circuitos integrados é fornecido. Ele deve ser soldado no *display* LCD, conforme a Figura 7. Para fazer essa tarefa, é necessário contar com um técnico em soldagem, de preferência da própria escola. Não tente fazer isso, a não ser que conheça os procedimentos, pois o *display* pode ser danificado se essa operação não for feita corretamente. Como o pente possui mais de 16 pinos, sobrarão alguns, que você pode usar para soldar nos terminais do motor de corrente contínua, como será mostrado depois.

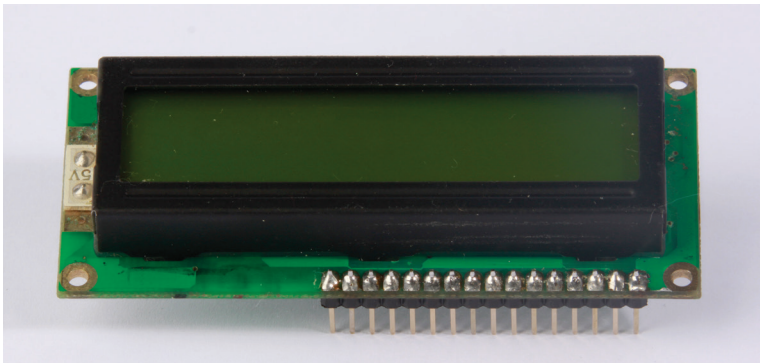


Figura 7 – *Display* LCD 16 × 2. Note os 16 pinos do pente soldados no *display*.

Cabo USB A/B

O cabo USB A/B faz a conexão entre a placa Arduino e o seu computador. Ele é mostrado na Figura 3, com os outros componentes do *kit*, sobre a *protoboard*.

LEDs

O LED é uma sigla referente a *light emitting diode* (díodo emissor de luz). Esse componente emite luz a partir da passagem de corrente elétrica. A luz é gerada pelo movimento dos elétrons em um material semicondutor. Os LEDs possuem muitas aplicações, em virtude, principalmente, do fato de consumirem pouca energia e serem muito duráveis. Exemplos de aplicação são os mostradores digitais, os televisores, os controles remotos e os dispositivos de sinalização. Eles são elementos polarizados, o que quer dizer que sua orientação no circuito é fundamental. O terminal positivo, que é caracterizado por ser o terminal mais longo, é o ânodo, que deve estar conectado na fonte de energia do circuito. Já o cátodo, que é o terminal mais curto,

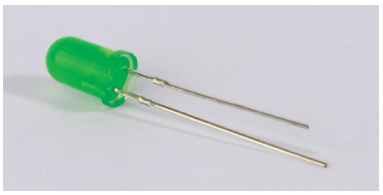


Figura 8 – LED.

deve ser ligado ao pino terra do circuito. A cor da luz do LED é definida pela frequência emitida pelo material semicondutor ao receber a passagem de corrente elétrica e não pela cor da capa plástica que o reveste, apesar de que efeitos de diferentes cores também podem ser conseguidos alterando a cor das capas plásticas. A Figura 8 mostra um dos LEDs do *kit*.

Sonorizador piezo e *buzzer*

O sonorizador piezo é uma pequena placa que vibra produzindo som, desde que uma corrente elétrica esteja passando por ela. Um material piezoelétrico reage a uma passagem de corrente pela vibração. A vibração da placa desse material é que causa o som. Seu funcionamento é semelhante ao do *buzzer*. Entretanto, este último é um pouco mais robusto e possui uma proteção, em geral plástica, que também serve como meio para controlar a propagação do som. As Figuras 9 e 10 mostram o *buzzer* e o sonorizador piezo que fazem parte do *kit*. Os sonorizadores e os *buzzers* são muito usados em dispositivos de sinalização, como alarmes e buzinas. Os *buzzers* e os sonorizadores são polarizados. Isso quer dizer que eles possuem um terminal positivo e um terminal negativo que devem ser conectados de maneira adequada no circuito.

Fotos: © Manoel José dos Santos Sena

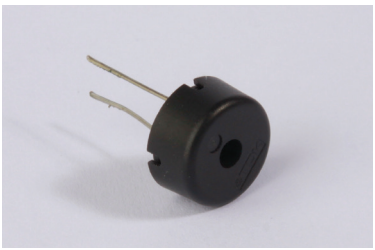


Figura 9 – *Buzzer*.

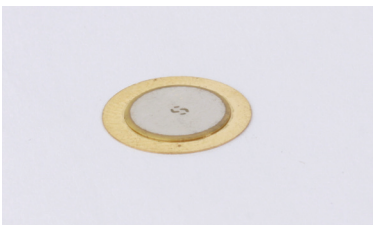


Figura 10 – Sonorizador piezo.

Transistor

O transistor é basicamente uma chave eletrônica, com um estado aberto ou fechado. Ele pode ser usado para uma infinidade de aplicações. No caso deste Caderno, um transistor será usado para ligar e desligar o fluxo de energia elétrica externa para o Arduino, se for necessário que ele acione elementos que exijam maior potência. A Figura 11 mostra o transistor que faz parte do *kit*. O transistor possui três terminais, um que é coletor, um emissor e um de base, que fica localizado no meio. Normalmente, uma corrente passa do terminal receptor para o terminal emissor. Por exemplo, pode-se controlar, interrompendo ou não, essa corrente, injetando uma corrente adicional pelo terminal de base.

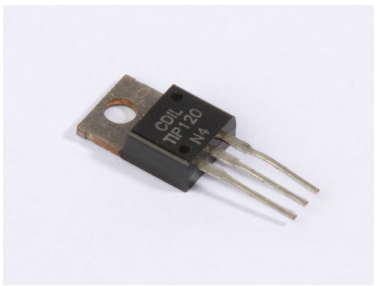


Figura 11 – Transistor.

Potenciômetro

O potenciômetro é um resistor que tem sua resistência elétrica ajustável. Esse ajuste é normalmente realizado por uma chave giratória. A variação da resistência em um circuito é

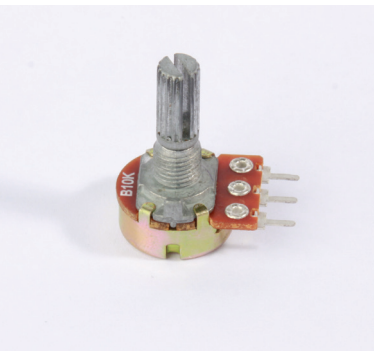


Figura 12 – Potenciômetro.

uma maneira indireta de fazer variar a tensão e a corrente entre pontos. Esse aspecto tem muita utilidade, por exemplo, quando há necessidade de controlar a velocidade de motores de corrente contínua, já que ela depende da tensão aplicada. A Figura 12 mostra o potenciômetro do *kit*. O potenciômetro não possui polaridade e tem três terminais. A corrente entra por um dos terminais da extremidade e flui, dividindo-se, proporcionalmente, pelo terminal do meio e pelo terminal da outra extremidade, que está ligado ao terra do circuito.

Chaves *pushbutton*

As chaves *pushbutton* servem para fechar ou abrir um circuito, ligando ou desligando, por exemplo, seus componentes. Os *pushbuttons* são usados como botões de controles de *video games*, controles remotos em geral e botões para entrada de dados. A Figura 13 mostra um dos quatro *pushbuttons* do *kit*.

Fotos © Manoel José dos Santos Sena

Figura 13 – Chave *pushbutton*.

Díodo

Os díodos são elementos semicondutores que permitem a passagem de corrente apenas em um sentido. Esse dispositivo pode funcionar como uma chave. O díodo tem um ânodo e um cátodo. O cátodo é representado no corpo do díodo por uma listra de cor clara. Se a tensão no ânodo for maior do que no cátodo, a corrente flui no sentido do ânodo para o cátodo. Entretanto, se a tensão no cátodo for maior do que no ânodo, a corrente não passa do cátodo para o ânodo.

Figura 14 – Díodo.

Os díodos são muito usados como elementos de proteção aos circuitos. Às vezes, altas intensidades de corrente podem ser geradas e fluir em sentido inverso no circuito, o que pode danificar componentes. Assim, para proteger o sistema, basta montar um díodo que impeça que a corrente flua no sentido inverso ao normal. Neste Caderno, será vista uma aplicação dos díodos no caso dos circuitos de acionamento de um motor de corrente contínua. A Figura 14 mostra o díodo do *kit*.

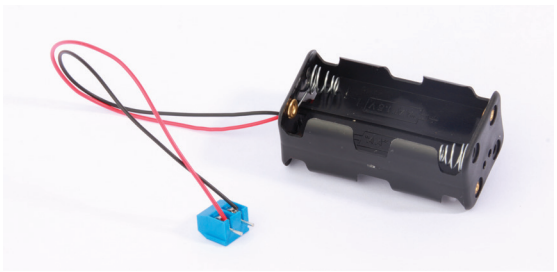
Motor de corrente contínua

Os motores de corrente contínua são motores muito simples, que funcionam por meio da passagem de corrente contínua e giram com uma velocidade que é proporcional à tensão aplicada, dentro de seus limites construtivos. Assim, para controlar a velocidade de um motor de corrente contínua, basta atuar na tensão aplicada entre os polos dele. A Figura 15 mostra o motor de corrente contínua que faz parte do *kit*.

Figura 15 – Motor de corrente contínua.

Porta-pilhas

O porta-pilhas é um dispositivo capaz de abrigar um conjunto de pilhas quando há a necessidade de fornecer mais tensão, a determinado componente, do que o Arduino é capaz de oferecer, ou então quando se quer alimentar o Arduino com uma fonte de energia que não seja o computador, para que ele possa trabalhar embarcado em outro equipamento, por exemplo. A Figura 16 mostra o porta-pilhas do *kit*.



© Manoel José dos Santos Sena

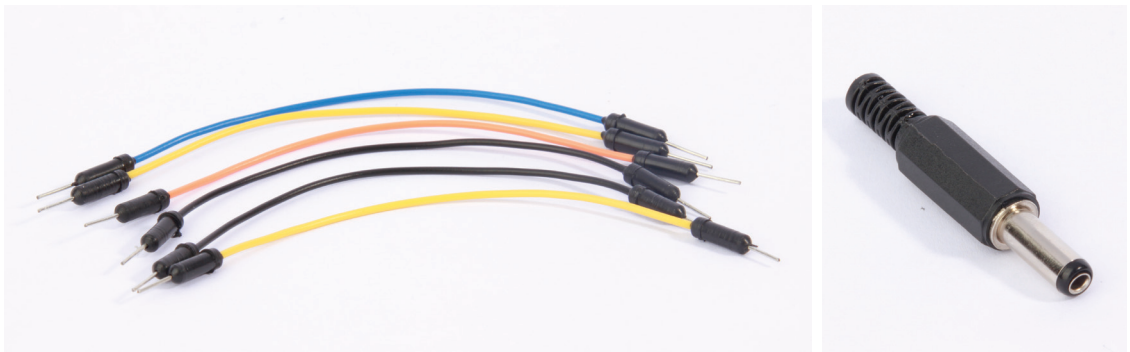
Figura 16 – Porta-pilhas.

Terminal de parafuso

Os conectores de alguns componentes, às vezes, são compostos por cabos elétricos e não por fios elétricos. Os motores e os porta-pilhas são exemplos de componentes desse tipo. Nesse caso, não é tão fácil conectá-los à *protoboard*. Uma saída é conectá-los a um terminal de parafusos e este à *protoboard*. Na Figura 16, o terminal de parafusos aparece conectado aos fios do porta-pilhas.

Fios *jumpers* e plugues

As conexões entre os componentes na *protoboard* e destes com o Arduino é feita por meio de fios *jumpers*, que nada mais são do que fios elétricos com um diâmetro tal que se adequem aos orifícios da *protoboard*. Já os plugues são conectores usados para proporcionar a conexão de elementos do circuito com fontes externas de energia. Uma possível aplicação no *kit* pode ser a de alimentar o Arduino com pilhas, e não pelo computador. Nesse caso, basta desmontar o plugue, puxando a sua capa plástica, e conectar os fios do porta-pilhas aos terminais dele. As Figuras 17 e 18 mostram alguns dos fios *jumpers* e o plugue que fazem parte do *kit*.



Fotos: © Manoel José dos Santos Sena

Figuras 17 e 18 – 17) Fios *jumpers*; 18) Plugue.

Resistores

© Manoel José dos Santos Sena

Figura 19 – Resistor.

Alguns elementos dos circuitos trabalham com valores de intensidade de corrente elétrica que não podem ultrapassar determinado limite ou com valores de tensão mais baixos do que os fornecidos pelas fontes, seja o Arduino ou uma fonte externa. Os resistores são elementos usados no circuito para limitar a corrente elétrica ou baixar os valores de tensão, com dissipação pura e simples de energia pelo calor. Assim, eles permitem o uso seguro de elementos de baixa tensão nos circuitos. A Figura 19 mostra um resistor. Para cada situação em um circuito, pode ser calculado um resistor adequado. Na seção de projetos do Caderno, serão tratadas as formas de cálculo do resistor necessário.

Em razão da necessidade de ter resistores com diferentes valores, dependendo das exigências do circuito, eles são identificados por anéis coloridos dispostos lado a lado. Os códigos de identificação dos resistores estão listados na Figura 20. As duas primeiras faixas formam o primeiro e o segundo dígitos de um número de base. A terceira faixa é um multiplicador para o número de base. A quarta faixa é a tolerância do valor de resistência, que varia de acordo com a qualidade do processo de fabricação do resistor. Além do resistor de 4 linhas, a Figura 20 mostra outros possíveis modelos.

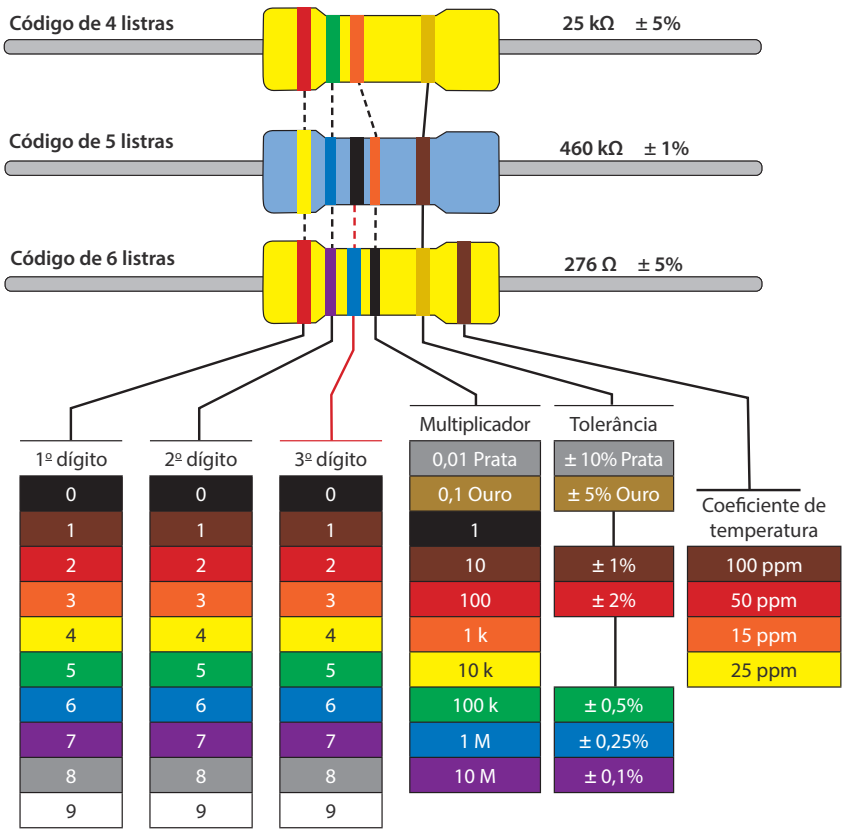


Figura 20 – Exemplos de valores de resistores e seus respectivos códigos de cores. Fonte de dados: International Electrotechnical Commission norma 60062.

O resistor da Figura 19, por exemplo, tem as faixas nas cores marrom, preto, laranja e dourado. De acordo com a Figura 20, o valor da sua resistência é $10 \times 10^3 = 10$ kOhm ±5%.

Para calcular o valor necessário de um resistor que limitará a corrente ou diminuirá a tensão para o uso de um componente, é necessário conhecer a voltagem de entrada do resistor (Ve), a voltagem de que o componente precisa (Vc) e a corrente limite do componente (Ic). Calcula-se, então, a resistência usando a seguinte equação:

$$R = (Ve - Vc)/Ic$$

Por exemplo, se um componente como um LED precisar de 2 V a 35 mA, e o circuito estiver recebendo 5 V do Arduino, será necessário um resistor de $R = (5 - 2)/0{,}035 = 85{,}71\ \Omega$. Deve-se, então, selecionar um resistor de resistência mais próxima a essa, mas com um valor maior, para não haver possibilidade de danificar o LED. Nesse caso, o resistor mais próximo seria o de 100 Ω. No caso do *kit* utilizado como referência neste Caderno, o menor resistor que ultrapassa o valor de 85,71 Ω é o de 330 Ω.

Termistor

O termistor é um resistor cuja resistência elétrica varia com a temperatura. Os termistores são muito usados para monitorar a temperatura de determinados ambientes, servindo para disparar alarmes ou eventos específicos dependentes da variação da temperatura. A Figura 21 mostra o termistor que faz parte do *kit*.

Fotos © Manoel José dos Santos Sena

Figura 21 – Termistor.

Sensor LDR

LDR é a sigla para *light dependent resistor* (resistor dependente de luz). Os LDRs são resistores cuja resistência é alterada se eles estão mais ou menos iluminados. Os LDRs são construídos com materiais semicondutores que podem ter altos valores de resistência se não estiverem na presença da luz. Quanto maior a luminosidade, menor o valor da resistência. Dessa forma, são usados como sensores de luminosidade. Os LDRs podem ser encontrados em diversos tamanhos. A Figura 22 mostra um sensor LDR. Eles são elementos não polarizados, ou seja, não importa o sentido com o qual os seus terminais são conectados no circuito.

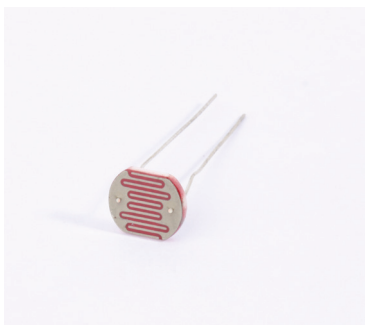


Figura 22 – Sensor LDR.

INSTALAÇÃO DO ARDUINO

O Arduino pode ser programado para receber sinais dos sensores e enviar sinais para elementos do circuito pela interface de *software* chamada Arduino IDE (*integrated development environment*).

Além de ter o IDE do Arduino instalado no computador, também é necessária a instalação do *driver*, para que o sistema o reconheça como dispositivo e seja capaz de se comunicar com ele. Todos esses programas podem ser baixados do *site* oficial do Arduino.

Convém fazer uma observação neste momento. Frequentemente, ocorrem atualizações nos programas de instalação de *drivers* e IDEs. As instruções contidas nesta seção referem-se ao momento no qual este Caderno foi desenvolvido. O processo pode ser alterado por modificações no *software* da página do projeto Arduino. Entretanto, o *site* oficial sempre tem informações detalhadas e atualizadas sobre a instalação.

Instalação do IDE e do *driver* do Arduino para Windows, Mac OS X ou Linux

Baixe na seção "Downloads" do *site* do projeto Arduino a última versão do Windows Installer do IDE. Após fazer o *download*, execute o arquivo e siga as instruções de instalação. De preferência, aceite todas as opções padrão sugeridas. Após o processo, será adicionado à sua área de trabalho o ícone associado ao IDE. A página de *download* do projeto Arduino contém instruções adicionais para instalação.

No caso de usar o sistema Mac OS X, copie o arquivo que você baixou do *site* para o diretório "Applications".

No caso de usar o sistema Linux, o processo varia de acordo com a distribuição usada. Procure por instruções específicas no *site* do projeto Arduino.

Instalação do *driver* do Arduino

Os processos podem ser um pouco diferentes, dependendo do sistema operacional que você estiver utilizando. No caso do Windows, conecte a placa Arduino ao seu computador por meio do cabo USB A/B. Normalmente, o LED verde marcado PWR acenderá, indicando que a placa está recebendo energia. Também será aceso o LED próximo ao pino 13. Em seguida, o Windows tentará instalar os *drivers*. Ele falhará nesse processo, se você estiver usando o Windows 7, Windows Vista ou Windows XP. Abra, então, o Painel de Controle do Windows e escolha a seção "Hardware e Sons". Depois, selecione "Dispositivos e Impressoras". Agora, selecione "Gerenciador de Dispositivos". Você verá então o Arduino na lista de dispositivos. Clique com o botão direito do *mouse* sobre ele e escolha a opção "Atualizar driver". Selecione a opção "Procurar software de driver no computador". Localize no diretório "Drivers", situado no diretório de instalação do Arduino em seu computador, o arquivo "arduino.inf". Avance e termine a instalação. O Windows fará, então, a instalação correta do *driver*. Ao terminar o processo, verifique na lista de dispositivos o número da porta COM que foi associada ao Arduino. Pode ser que seja necessário configurar esse número ao utilizar o IDE pela primeira vez.

No caso de usar o sistema Mac OS X, não é necessário instalar *drivers*. Basta usar a aplicação.

Para o sistema Linux, siga as instruções apresentadas anteriormente.

EXECUTANDO O IDE E TESTANDO O ARDUINO

A partir deste momento, o sistema usado será o Windows, mas os passos para os outros sistemas são semelhantes.

Selecionando a placa e a porta de comunicação

Para testar o Arduino, execute o IDE. Você terá uma tela semelhante à mostrada na Figura 23.

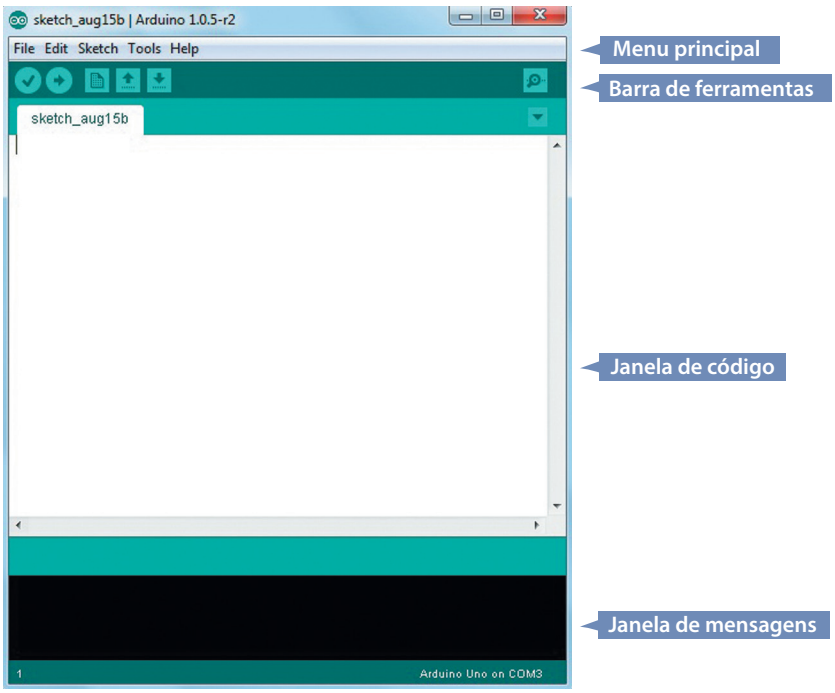


Figura 23 – Interface do IDE do Arduino (http://arduino.cc).

- Para selecionar a placa, clique no item de menu "Tools → Board" e escolha a placa Uno.
- Para definir a porta de comunicação, clique no item de menu "Tools → Serial Port" e selecione a porta que você anotou durante a instalação do *driver*.

Análise da interface

O menu principal contém as opções "File", para gerenciamento de arquivos, "Edit", para a edição de arquivos, "Sketch", com ferramentas de auxílio à programação, "Tools", para configurações do IDE, e "Help", para ajuda.

A barra de ferramentas será muito usada para fazer a interface com o Arduino. A Tabela 2 descreve seus ícones.

Descrição dos ícones da Barra de Ferramentas	
 *VERIFY*	Verificar se o código não tem erros.
 *LOAD*	Carregar o código para o Arduino.
 *NEW*	Criar um novo arquivo de código.
 *OPEN*	Abrir um arquivo de código.
 *SAVE*	Salvar o arquivo de código da janela.
 *SERIAL MONITOR*	Monitorar a comunicação serial entre o Arduino e o computador.

Tabela 2 – Descrição dos ícones da barra de ferramentas.

Testando o Arduino

Para fazer o Arduino executar tarefas, deve ser carregado um código para ele, pelo IDE. O código é formado por instruções que compõem um programa de computador que o IDE criará e transferirá para o Arduino. O sistema possui uma série de exemplos de código para ajudar os estudantes nos projetos iniciais. Abra um deles. No menu principal, selecione "File → Examples → 01.Basics → Blink". Será aberta, então, uma nova janela com um código para o exemplo. Clique no ícone *Load* da barra de ferramentas. O código será carregado para o Arduino. Durante o processo de transferência, os LEDs LX e TX piscarão, mostrando que o processo está acontecendo. Assim que a transferência for concluída, o Arduino passará a agir conforme determina o código. Nesse caso, ele piscará o LED perto do pino 13 em intervalos de 1 segundo. É importante ressaltar que, a partir do momento em que o programa foi carregado para o Arduino, ele passa a executar as tarefas "por conta própria", precisando da conexão com o computador apenas para a alimentação de energia.

Compreensão do código

A linguagem de programação do Arduino é a baseada nas linguagens C e C++. Para os projetos deste Caderno, apenas um pequeno conjunto de comandos será usado, de tal forma que uma pessoa sem experiência em programação possa compreender e fazer os exemplos. Entretanto, existe um grande número de livros e *sites* que têm muitas informações sobre essas linguagens.

Quando um programa é escrito em uma linguagem de programação, o nome para o arquivo que contém as instruções é chamado de código-fonte. O código-fonte do exemplo *Blink* é mostrado na Listagem 1. Nessa listagem, para facilidade de visualização, foram retirados os comentários, que são delimitados pelos grupos de caracteres `/*` e `*/`. Os comentários são usados no código como documentação, para explicar no próprio código o significado de algumas instruções.

Atenção: as listagens dos códigos deste Caderno terão as linhas numeradas. Isso não acontece no IDE do Arduino, e foi feito aqui apenas para poder referenciar melhor o código durante a discussão. Ao digitar o código na IDE do Arduino, os números dentro dos círculos não devem ser colocados.

Listagem 1 – Código-fonte do exemplo *Blink*

```
1  int led = 13;
2  void setup() {
3  pinMode(led, OUTPUT);
4  }
5  void loop() {
6  digitalWrite(led, HIGH);
7  delay(1000);
8  digitalWrite(led, LOW);
9  delay(1000);
10 }
```

Na primeira linha do código aparece uma variável chamada "led", do tipo inteiro, daí o código "int". Uma variável "int" pode armazenar valores inteiros em uma faixa entre –32 768 e 32 767. Nesse código, a variável "led" vai armazenar o valor 13, que vai servir para identificar o "led" mais à frente, no código.

Existem partes do código que são executadas repetidas vezes. A elas é dado o nome de funções. O código que é executado nas funções é delimitado por chaves que são abertas logo após o nome da função e são fechadas após o término do código. Há duas funções básicas para o funcionamento do Arduino: a função "setup" e a função "loop". Elas serão descritas a seguir.

- **Função "setup":** esta função é chamada quando o Arduino começa a executar o programa. Ela é acionada apenas uma vez durante a execução. Normalmente, ela serve para fazer a inicialização de variáveis do Arduino. No exemplo *Blink*, ela é definida da linha 2 até a linha 4. A função "pinMode" serve para especificar que o "led 13" será usado como saída (*output*). Na seção "Learning" do *site* do projeto Arduino, há uma descrição de cada função que faz parte da linguagem. Existem também conjuntos de funções que são reunidas em bibliotecas (*libraries*), que podem ser incorporadas ao código quando necessário. Elas também são descritas na seção "Reference" do *site*.
- **Função "loop":** esta função é chamada repetidas vezes pelo Arduino, durante todo o tempo em que ele estiver ligado, após o programa ter sido carregado. Note que neste exemplo ela é delimitada pela chave aberta na linha 5 e pela chave fechada na linha 10. Na linha 6, a função "digitalWrite" especifica um sinal de saída digital para o pino 13 (número que está armazenado na variável "led"), com intensidade "HIGH", o que fará o LED acender. A função "delay", na linha 7, recebe como parâmetro um valor em milissegundos. Sua função é interromper o processamento durante esse tempo. Assim, o LED continuará aceso durante 1 000 milissegundos, ou seja, durante 1 segundo. Na linha 8, a função "digitalWrite" é novamente chamada, mas com o valor de intensidade "LOW". Isto fará o LED apagar. Como a função "delay" é chamada novamente na linha 9, o processamento será suspenso novamente por 1 segundo. Como o estado do LED agora é apagado, ele permanecerá assim durante esse período de tempo.

AS POSSIBILIDADES DE TRABALHO COM O ARDUINO

Nesta seção, serão descritas sugestões de projetos de trabalho para o Arduino e como eles podem ser usados no escopo do Ensino Médio. A descrição de cada projeto começará com os componentes necessários. Em seguida, será mostrada a configuração do circuito, seguida de uma análise do código-fonte. A partir do momento que o projeto tiver sido bem descrito e organizado pelo professor, serão, então, discutidas as sugestões de tratamento do conteúdo do Ensino Médio. É fortemente recomendado que cada projeto seja preparado antes da leitura dessas sugestões, pois o professor terá uma visão bem mais clara do processo.

ACENDIMENTO DE LED

Problema a ser resolvido

Este projeto tem o objetivo de trabalhar o acendimento de um LED. Ele, apesar de simples, permitirá a compreensão de aspectos importantes do funcionamento do Arduino, bem como de aplicações práticas.

Habilidades que se pretende que os estudantes desenvolvam

Neste projeto, os estudantes devem desenvolver a habilidade de interpretar um esquema de montagem de circuito, além de compreender o que são a lógica de programação envolvendo temporização e repetição, o comportamento de um LED, a corrente elétrica, funções do tempo, frequência e período.

Muita criatividade pode ser usada no desenvolvimento de lógicas para o acendimento do LED. As iniciativas individuais podem ser trabalhadas. Por exemplo, cada estudante pode definir a sua maneira de fazer o LED funcionar.

Tempo previsto

O tempo previsto para fazer a atividade é em torno de 50 minutos. É interessante separar antes das aulas os componentes e deixá-los organizados sobre as bancadas. Com isso, evita-se o uso de componentes inadequados.

Como resolver o problema?

Material necessário

Resistor de 330 Ω, *protoboard*, LED vermelho de 5 mm e 2 fios *jumpers*.

Peça aos estudantes que descrevam o princípio de funcionamento de cada um dos componentes. Se eles ainda não tiverem estudado a área da Física chamada *eletricidade*, faça uma pequena

introdução a respeito do fluxo de corrente elétrica em um circuito e os papéis das diferenças de potencial e do resistor no circuito.

Configuração do circuito

A configuração do circuito é mostrada na Figura 24. O tamanho dos cabos ou sua disposição não são relevantes. O que importa são os pontos onde eles são conectados, tanto na *protoboard* quanto no Arduino. Ao conectar os componentes, procure fazer que os terminais entrem em contato efetivamente com as placas metálicas da *protoboard*, encaixando-os bem. No início, essa atividade exige um pouco mais de trabalho, mas logo adquire-se prática.

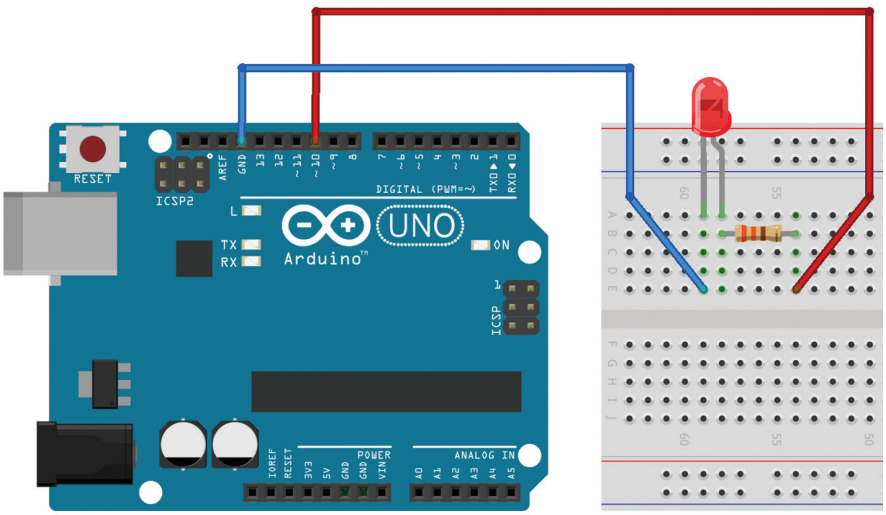


Figura 24 – Montagem esquemática para o projeto *Acendimento de LED* (*software* Fritzing – uso livre).

A Figura 25 mostra como ocorre o contato entre os componentes do circuito no interior da *protoboard*. Nessa figura, é possível notar que se tem certa liberdade para posicionar os terminais dos componentes e dos fios, desde que o arranjo físico seja equivalente.

Analise a Figura 6 para compreender como as conexões da Figura 25 são fornecidas pela *protoboard*.

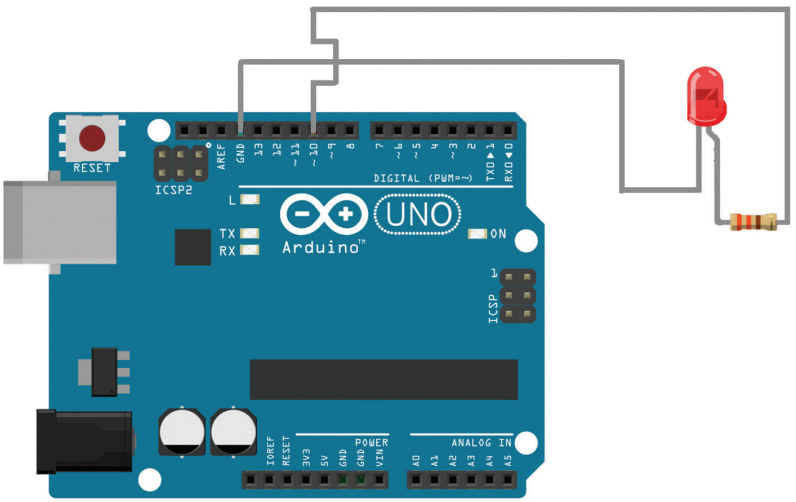


Figura 25 – Arranjo esquemático do circuito com as conexões providas pela *protoboard*.

Não esqueça que o LED é um elemento polarizado. Assim, o polo positivo dele, que é o terminal mais longo, deve ser conectado ao terminal de saída digital de 5 V do Arduino, e seu polo negativo, que é o terminal mais curto, deve ser ligado ao terra do circuito, que é o terminal GND do Arduino.

A Figura 26 mostra o projeto montado, para que você possa notar como os componentes devem ser conectados.

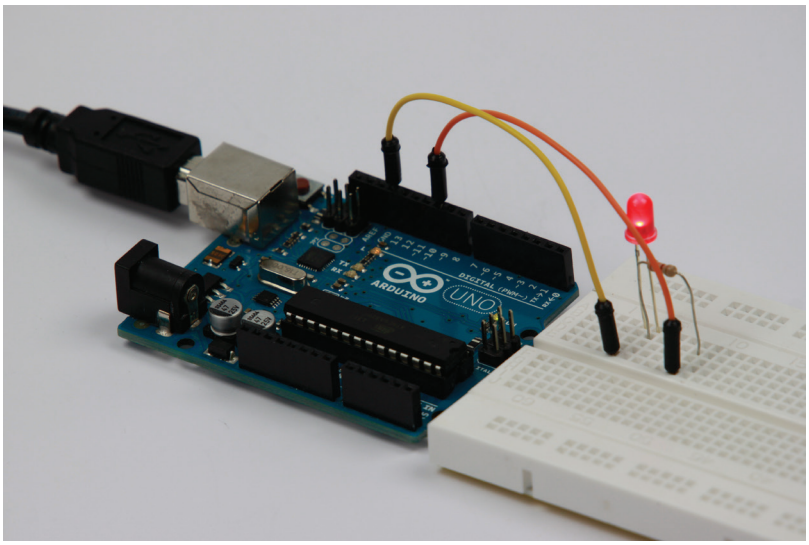


Figura 26 – Projeto *Acendimento de LED* montado.

Observação: pode acontecer de o projeto não funcionar como previsto, pelos mais diferentes motivos. Se isso acontecer, e acredite, é muito comum, não perca a calma. Lembre-se de que sempre é possível descobrir as causas do problema. Comece pelos pontos mais prováveis e vá checando o que pode estar causando o contratempo. Primeiro, cheque todas as conexões dos componentes. Acontece de, frequentemente, elas não estarem corretamente feitas. Às vezes, mesmo com as conexões corretas, um mau contato pode ser gerado pela montagem em sequência. Por exemplo, é possível esbarrar em um fio sem querer e não perceber que ele perdeu o contato com a *protoboard*. Se você tiver certeza de que está tudo certo com as conexões, cheque o código-fonte. A quase totalidade dos problemas que podem acontecer concentra-se nas conexões e no código-fonte. Se ainda assim não funcionar, e você tiver certeza de que está tudo certo com as conexões e o código-fonte, substitua os componentes, um a um, pois eles podem ser a causa do problema.

Código-fonte do projeto

O código-fonte do exemplo é mostrado na Listagem 2. Abra o IDE do Arduino, digite o código-fonte e salve-o com o nome Projeto 1. O Arduino criará um diretório e salvará esse arquivo nele, no local em que você especificar na caixa de diálogo que aparecerá.

Listagem 2 – Código-fonte do projeto *Acendimento de LED*

```
1  //Projeto 1 – Acendimento de LED
2  int pinoLED = 10;      //Identificador para o pino 10 do Arduino
3  void setup() {
4  pinMode(pinoLED, OUTPUT);   //Especifica o pino 10 como saída de sinal
5  }
6  void loop() {
7  digitalWrite(pinoLED, HIGH);   //Escreve uma saída de energia de 5 V para o pino 10
8  delay(1000);   //Espera 1 segundo
9  digitalWrite(pinoLED, LOW);   //Escreve uma saída de energia de 0 V para o pino 10
10 delay(1000);   //Espera 1 segundo
11 }
```

Em seguida, verifique o código, clicando no ícone *Verify*. Se houver algum problema, o IDE o indicará. Por exemplo, se você esquecer o ponto e vírgula no final da primeira instrução "delay(1000)" e verificar o código, o IDE dará uma mensagem de erro, que será como a mostrada na Figura 27. Normalmente, as mensagens de erro dão uma boa indicação do problema, mas nem sempre é uma indicação exata. No caso da Figura 27, a mensagem diz que falta um ponto e vírgula antes da instrução que está na linha seguinte àquela na qual está faltando o ponto e vírgula. Corrija, então, o código. Ao fazer a verificação, você verá que essa mensagem de erro não aparecerá.

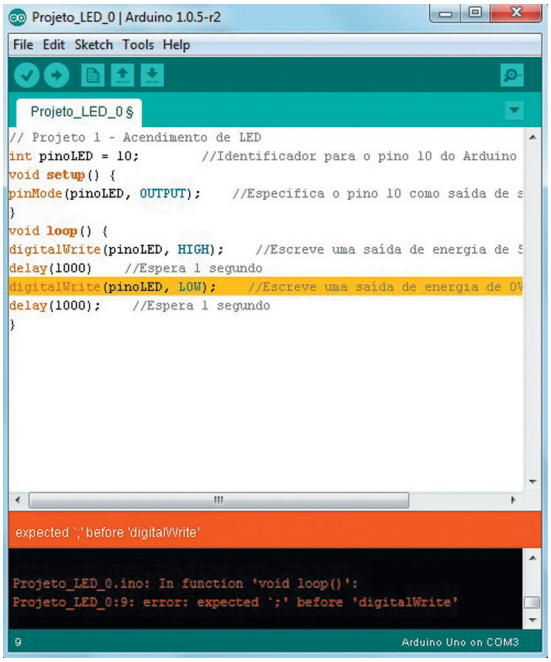


Figura 27 – Mensagem de erro de compilação (http://arduino.cc).

Após a verificação, você fará o carregamento do programa para o Arduino. Nesse momento, o código-fonte que você escreveu é transformado em um programa de computador, em código binário. Esse processo é chamado de compilação. Esse código binário é o programa de computador que será transferido para o Arduino.

É importante notar que não é obrigatório realizar a verificação explicitamente. Se você tentar carregar o programa para o Arduino sem a verificação, o IDE o fará automaticamente e não compilará o programa, caso o código-fonte tenha algum erro.

Conecte o Arduino ao seu computador e clique no botão "Upload". O processo de envio do programa ao Arduino terá início. O LED do circuito começará a piscar em intervalos de 1 segundo.

As funções envolvidas no código-fonte da Listagem 2 são as mesmas do código-fonte da Listagem 1. Assim sendo, a análise do código é similar nas duas situações. Foram mudados apenas os nomes das variáveis, para torná-los mais legíveis em língua portuguesa.

Um aspecto bastante interessante deste projeto é a possibilidade de alterar a forma como o LED se comporta, de maneira muito simples. Para isso, basta alterar o tempo de *delay* e as sequências de acendimento e apagamento. Por exemplo, é possível alterar o código de tal forma que o LED acenda em intervalos de 10 segundos, inicialmente, e, após cada acendimento, ir diminuindo de 1 em 1 segundo, até chegar ao valor de 1 segundo, quando reiniciará o ciclo. Para isso, o código-fonte deve estar como na Listagem 3.

Listagem 3 – Alteração do código-fonte do projeto *Acendimento de LED*

```
1  //Projeto 1 – Acendimento de LED – Alteração
2  int pinoLED = 10;      //Identificador para o pino 10 do Arduino
3  int intervalo = 10;
4  void setup() {
5  pinMode(pinoLED, OUTPUT);   //Especifica o pino 10 como saída de sinal
6  }
7  void loop() {
8  digitalWrite(pinoLED, HIGH);   //Escreve uma saída de energia de 5 V para o pino 10
9  delay(intervalo*1000);   //Espera intervalo vezes 1 segundo
10 digitalWrite(pinoLED, LOW);   //Escreve uma saída de energia de 0 V para o pino 10
11 delay(intervalo*1000);   //Espera 1 segundo
12 intervalo = intervalo - 1;   //Diminui o intervalo
13 if (intervalo == 0) intervalo = 10;   //Se o intervalo diminuir do valor 1, reiniciá-lo para 10
14 }
```

Altere o código e refaça o processo de verificação e de carregamento dele para o Arduino e verifique as mudanças no comportamento do sistema.

Você verá que o sistema fará que o LED fique aceso e apagado em intervalos cada vez menores, começando com um intervalo de 10 segundos e diminuindo a cada vez que o LED é aceso.

Analise um pouco mais a fundo a lógica do programa da Listagem 2 para compreender melhor o processo. Uma nova variável foi incluída no código da Listagem 3: a variável "intervalo". Essa variável recebe, no momento da sua criação, o valor 10.

Na primeira execução da função "loop", o LED é então aceso, com a instrução "digitalWrite (pinoLED, HIGH)". O programa aguarda um total de intervalo · 1 000 milissegundos, ou seja, 10 · 1 000 milissegundos = 10 segundos, por meio da função "delay". Depois, o LED é apagado com a instrução "digitalWrite(pinoLED, LOW)". O programa, então, aguarda por mais 10 segundos, com mais uma execução da função "delay". Em seguida, a variável "intervalo" é decrementada de 1 em seu valor, passando a ter o valor 9. É feita então, nesse momento, uma checagem do valor, pela instrução "if", que será discutida no próximo parágrafo. Na próxima execução da função "loop", o processo se repetirá, mas com a variável "intervalo" tendo o valor 9, o que fará que o LED fique aceso e apagado por 9 segundos. Na execução seguinte da função "loop", o valor da variável "intervalo" será 8, e assim por diante.

Decrescendo a cada execução da função "loop" de 1, algum tempo depois, a variável "intervalo" receberá o valor 0. A instrução "if(intervalo == 0)" tem a função de checar se o valor da variável "intervalo" é igual a zero. Quando isso acontecer, a variável "intervalo" recebe novamente o valor 10, e o ciclo reinicia.

Discussão dos resultados

- Física 1ª série – volume 2. Situação de Aprendizagem 7. **Pergunta sugerida**: como o LED é capaz de emitir luz?
  O fenômeno físico que torna possível ao LED gerar luz quando da passagem de corrente está associado ao movimento dos elétrons no material semicondutor do qual o LED é feito. Os resultados desse projeto podem ser discutidos sob o prisma de como é possível explicar a luminosidade emitida pelos LEDs a partir do conhecimento do modelo atômico da matéria, em especial o comportamento dos elétrons em movimentação.
  Sugere-se ainda que seja solicitado aos estudantes que elaborem uma explicação de como os modelos atômicos, que permitem a compreensão do que ocorre em um LED, evoluíram ao longo do tempo, usando os seguintes temas de pesquisa da tabela da página 68: *Os átomos, o movimento e a matéria; O vazio; A matéria e os átomos* e *O modelo do átomo*.
- Física 2ª série – volume 1. Situação de Aprendizagem 6. **Perguntas sugeridas**: no projeto, quais são os elementos do circuito onde ocorrem os processos de transferência de calor? Quais são os principais processos que você pode identificar?
  Na página 39, são descritos os processos de transferência de calor. Esses processos são a condução, a convecção e a irradiação. No circuito usado para realizar o projeto, há um elemento que foi colocado com o principal objetivo de dissipar a energia, para que o LED não seja danificado. Esse elemento é o resistor. A passagem da corrente elétrica pelo resistor gera calor, que é transmitido por ele ao ar que está ao seu redor. Assim sendo, ocorre uma transferência por convecção para o ar ao redor do resistor. Em menor grau, também ocorre transferência de calor por irradiação.
  De uma maneira geral, esses processos ocorrem, com menor intensidade, em todos os elementos do circuito, pois, durante a passagem da corrente elétrica, ocorre sempre a dissipação de parte da energia na forma de calor.
- Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida 1**: quais são os elementos que compõem o circuito elétrico e como a corrente elétrica flui de um elemento para o outro?

Na página 19 do Caderno, no roteiro 3, é sugerida a montagem de um circuito composto por pilhas e lâmpadas. Um paralelo imediato com o circuito composto pelo LED pode ser feito. As semelhanças e as diferenças entre as duas montagens podem, então, ser exploradas. Por exemplo, os circuitos possuem voltagens semelhantes, mas componentes diferentes para executar uma tarefa parecida.

Pergunta sugerida 2: Quais são as diferenças entre o circuito da página 19 e o circuito do projeto?

A maior diferença está no controle do circuito que é conseguido pelo uso do Arduino. Isso não é possível de ser obtido com o arranjo do circuito da página 19 do Caderno. Existem outras diferenças marcantes entre os dois, entretanto. Por exemplo, a alimentação do circuito convencional é feita diretamente por meio das pilhas, enquanto no caso do projeto é realizada pelo computador. No caso do projeto com o Arduino, os contatos são feitos pela *protoboard* e os fios *jumpers*, enquanto no circuito da página 19 do Caderno é feita por fios.

CONSTRUÇÃO DE UM SEMÁFORO

Problema a ser resolvido

Construir a estrutura física necessária para o funcionamento do semáforo, bem como conceber sua lógica completa de funcionamento, incluindo a sequência necessária de acendimento das luzes e temporização.

Habilidades que se pretende que os estudantes desenvolvam

Neste projeto os estudantes terão a oportunidade de fazer que mais de um componente eletrônico funcione de forma independente no circuito.

Tempo previsto

O tempo previsto para esta atividade é de cerca de 100 minutos.

Como resolver o problema?

Material necessário

Três resistores de 330 Ω, *protoboard*, LED vermelho de 5 mm, LED amarelo de 5 mm, LED verde de 5 mm e 7 fios *jumpers*.

Configuração do circuito

O circuito para este projeto é mostrado na Figura 28.

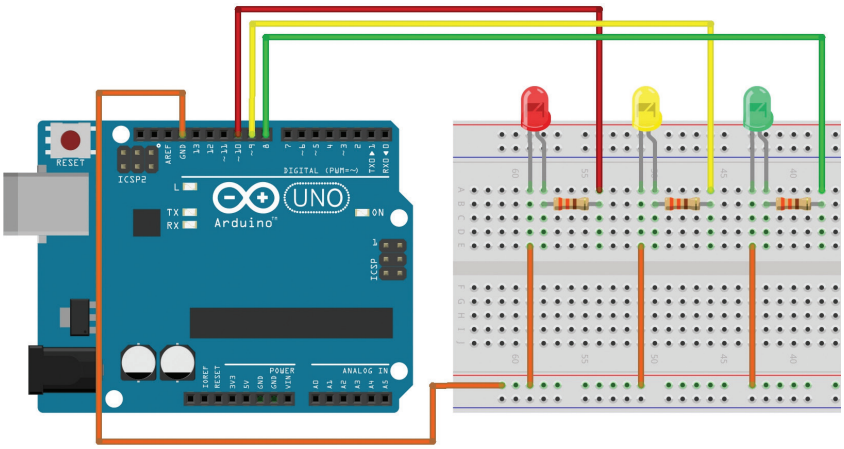


Figura 28 – Montagem esquemática para o projeto *Construção de um semáforo*.

A Figura 29 mostra o projeto montado.

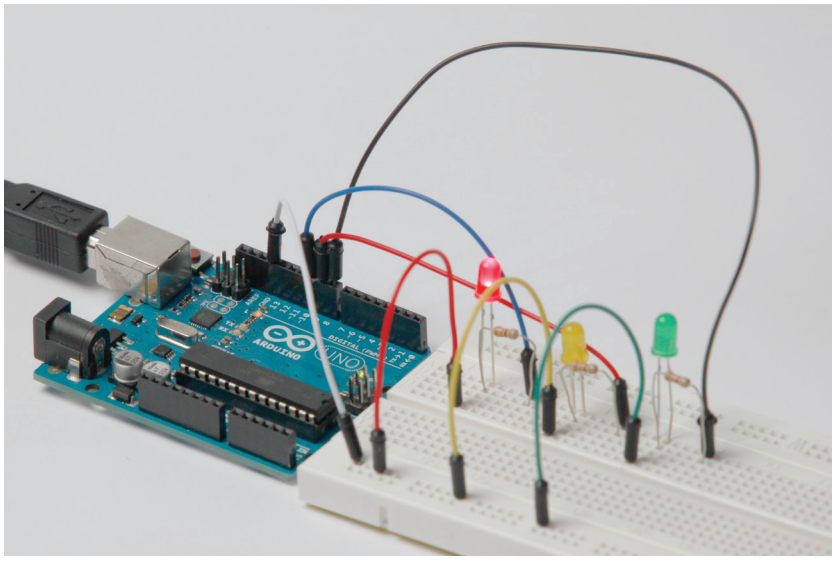


Figura 29 – Projeto *Construção de um semáforo* montado.

Código-fonte do projeto

A Listagem 4 mostra a alteração do código para o funcionamento do semáforo.

Listagem 4 – Código-fonte do projeto *Construção de um semáforo*

```
1 int pinoVermelho = 10;
2 int pinoAmarelo = 9;
```

```
3  int pinoVerde = 8;
4  void setup() {
5  pinMode(pinoVermelho, OUTPUT);
6  pinMode(pinoAmarelo, OUTPUT);
7  pinMode(pinoVerde, OUTPUT);
8  }
9  void loop() {
10 digitalWrite(pinoVermelho, HIGH); //Acende o LED vermelho
11 delay(10000); //Espera 10 segundos
12 digitalWrite(pinoVermelho, LOW); //Apaga o LED vermelho
13 digitalWrite(pinoVerde, HIGH); //Acende o LED verde
14 delay(10000); //Espera 10 segundos
15 digitalWrite(pinoVerde, LOW); //Apaga o LED verde
16 digitalWrite(pinoAmarelo, HIGH); //Acende o LED amarelo
17 delay(2000); //Espera 2 segundos
18 digitalWrite(pinoAmarelo, LOW); //Apaga o LED amarelo
19 }
```

Ao analisar o código-fonte, nota-se que ele usa as mesmas funções dos códigos-fonte das Listagens 1 e 2, ou seja, as funções "pinMode", "digitalWrite" e "delay". Entretanto, elas são usadas de maneira a provocar um comportamento específico para cada LED.

Primeiro, na função "setup", é atribuída para cada LED uma saída digital do Arduino. Elas são as saídas 8, 9 e 10, respectivamente, para os pinos verde, amarelo e vermelho.

Na função "loop", a função "digitalWrite", com o parâmetro "HIGH", é usada para acender o LED vermelho por 10 segundos. A execução da função "delay" faz o sistema aguardar 10 segundos com o LED aceso. Após esse período, uma nova chamada à função "digitalWrite", mas com o parâmetro "LOW", é usada para apagar o LED vermelho.

Logo em seguida, o LED verde é aceso, e o sistema espera por mais 10 segundos para apagá-lo.

O LED amarelo é, então, aceso, durante 2 segundos. Em seguida, ele é apagado e o ciclo recomeça.

Discussão dos resultados

- Física 1ª série – volume 1. Situação de Aprendizagem 4. **Pergunta sugerida:** qual é o efeito da temporização da mudança de verde para vermelho, passando pelo amarelo, sobre a força que terá de ser exercida pelo sistema de freios para parar o veículo?
  Pode-se analisar essa pergunta com base na discussão que é proposta pelo roteiro 4, da página 28 do Caderno, que é chamado *Descobrindo o que produz a mudança no movimento*. Levando

em consideração que quanto maior for o tempo em que o semáforo ficar amarelo, maior será o tempo necessário para um carro desacelerar sua velocidade da de cruzeiro para zero, a temporização tem um efeito muito forte na facilidade para a realização da tarefa. Isso é explicado porque a mudança de velocidade implica uma desaceleração. Quanto maior o tempo que o semáforo fica amarelo, menor a desaceleração necessária e, pela aplicação da 2ª Lei de Newton, menor a força necessária também.

- Física 2ª série – volume 2. Situação de Aprendizagem 11. **Pergunta sugerida:** como os LEDs conseguem gerar as três diferentes cores da experiência?
  Na página 81 do Caderno, existe uma tabela relacionando as cores com a frequência da onda eletromagnética correspondente. Os LEDs geram as diferentes cores dependendo do material do qual são feitos. Os materiais semicondutores dos quais são feitos os LEDs possuem uma variação da frequência na qual vibram ao sofrerem a ação da corrente elétrica. Isso é a causa das diferentes cores geradas. No início do desenvolvimento das tecnologias de LED, eles eram monocromáticos, entretanto, com o passar do tempo, foram desenvolvidos novos materiais capazes de gerar padrões de cores diferentes. A relação desse aspecto de funcionamento com a tabela de frequências é muito forte.
- Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida:** os circuitos do projeto, para cada LED, são independentes entre si?
  Sim, eles são independentes entre si. Isto fica claro ao analisar o fato de que o Arduino disponibiliza uma saída de dados digital diferente para cada LED. Assim sendo, eles podem ser controlados independentemente. É possível, por exemplo, criar qualquer padrão de acendimento dos LEDs pelas alterações no código-fonte do projeto. Os três circuitos compartilham, entretanto, o mesmo terra do circuito.

GERAÇÃO DE SONS COM O *BUZZER*

Problema a ser resolvido

Este projeto tem o objetivo de gerar sons por intermédio de um *buzzer*, usando uma função seno para controlar a frequência, e prover base para os estudantes gerarem sons diferentes, empregando outras funções.

Habilidades que se pretende que os estudantes desenvolvam

Neste projeto, os estudantes desenvolverão habilidades ligadas às diferentes características de uma onda sonora e compreenderão os fundamentos da piezoeletricidade.

Os estudantes podem desenvolver diversas maneiras criativas de gerar sons com diferentes características a partir do modelo de base, alterando principalmente a frequência da onda sonora gerada. Eles aplicarão também a lógica de programação em estruturas de laço do tipo *for*, que são parte integrante da imensa maioria dos programas de computador.

Tempo previsto

O tempo necessário para fazer a atividade é em torno de 50 minutos. A maior parte do tempo será dedicada ao trabalho no código-fonte. O arranjo físico da experiência é mais simples até do que no caso do primeiro projeto, que usava LEDs.

Como resolver o problema?

Material necessário

Protoboard, *buzzer* e 2 fios *jumpers*.

Peça aos estudantes que descrevam o princípio de funcionamento de cada um dos componentes. Dê especial atenção ao funcionamento do *buzzer*.

Configuração do circuito

A configuração é mostrada na Figura 30.

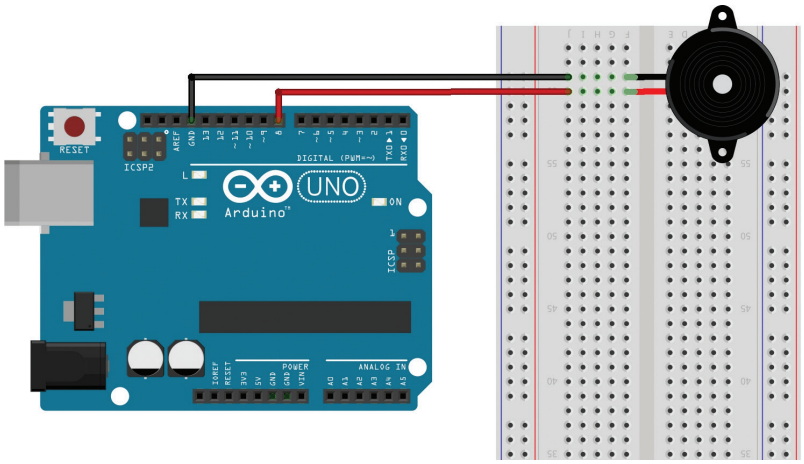
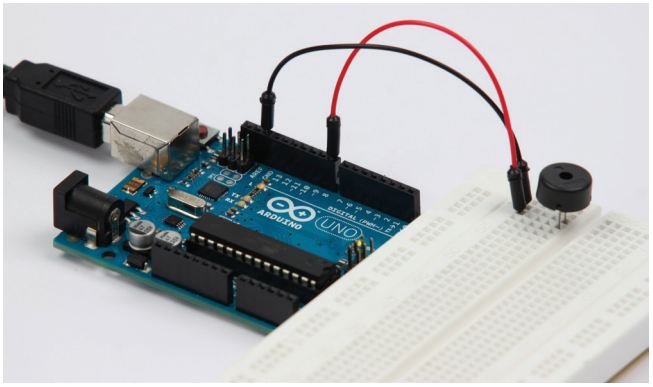


Figura 30 – Montagem esquemática para o projeto *Geração de sons com o* buzzer.

A Figura 31 mostra o projeto montado.

O *buzzer* é um elemento polarizado. Assim, o polo positivo dele, que é o terminal mais longo, deve ser conectado ao terminal de saída digital de 5 V do Arduino, e seu polo negativo, que é o terminal mais curto, deve ser ligado ao terra do circuito, que é o terminal GND do Arduino.



© Manoel José dos Santos Sena

Figura 31 – Projeto *Geração de sons com o* buzzer montado.

Código-fonte do projeto

O código-fonte do exemplo é mostrado na Listagem 5. Abra o IDE do Arduino, digite o código-fonte e salve-o com o nome Projeto 2.

Listagem 5 – Código-fonte do projeto *Geração de sons com o* buzzer

```
1  //Projeto 2 – Geração de sons com o buzzer
2  float valorSeno;
3  int valorFreq;
4  int base = 2000;
5  void setup() {
6  pinMode(8, OUTPUT);
7  }
8  void loop() {
9  for (int x=0; x<180; x++) {
10    valorSeno = (sin(x*(3.1416/180)));   //Calcula o seno de x
11    valorFreq = base+(int(valorSeno*1000));   //Calcula frequência
12   tone(8, valorFreq);
13   delay(2);
14  }
15  }
```

Efetuada a verificação do código, você fará o carregamento do programa para o Arduino. O *buzzer* emitirá um som oscilante. Você pode alterar o valor da variável "base", para mais ou para menos, para ver o efeito que isso causa no som. Altere a linha "int base = 2000;" para "int base = 1000;", faça o *upload* para o Arduino e veja a diferença. Depois, altere o valor da variável "base" para 4000 e teste novamente.

Deve-se analisar o código-fonte desde o começo. Inicialmente, são declaradas as variáveis "valorSeno", "valorFreq" e "base". Note que as variáveis "valorFreq" e "base" são do tipo "int", já discutida. A variável "valorSeno" é do tipo "float". Esse tipo de variável admite números em ponto flutuante (que podem ter casas decimais), no limite de -3,4028235E+38 a 3,4028235E+38. O código enviará um sinal para a saída número 10 do Arduino, que é variante no tempo. Essa variação ocorre na frequência do sinal e é na forma senoidal. Veja como isso acontece.

O código usa o conceito de laço de repetição. O laço de repetição está definido entre as linhas 8 e 14.

O comando "for" possui três parâmetros, separados por ";". Todos eles são ligados à variável **x**. O primeiro parâmetro declara a variável **x**. O segundo é o critério de parada do processo de repetição. O terceiro é o incremento que a variável **x** terá todas as vezes que o laço for executado. Assim sendo, esse laço é executado um total de 180 vezes para cada chamada da função "loop". Em cada repetição do laço, a variável **x** tem um valor diferente. Ela começa com 0 na

primeira execução do laço, tem o valor 1 na segunda repetição, o valor 2 na terceira repetição, e assim por diante. Quando chega ao valor de 180, o laço é interrompido, e a execução continua na linha 15, já fora do laço. Como essa linha já delimita o final da função "loop", isso significa que ela será executada novamente.

Nesse código, a função que define o som que será enviado ao *buzzer* é a função "tone". Essa função recebe como parâmetros o pino do Arduino, para onde o sinal será enviado, e o valor da frequência do som, em Hertz, que será emitido pelo *buzzer*. Esse código faz que sejam emitidos diferentes valores de frequência pelo *buzzer*, em uma variação na forma de uma função seno. Quando cada repetição do laço começa a ser realizada, o valor da função seno é calculado na linha 10, com a ajuda da função "sin". Essa função recebe um parâmetro em radianos, que é o ângulo, e retorna o valor do seno desse ângulo. Como **x** variará de 0 a 180, o seno (parâmetro da função "sin"), como pode ser notado na linha 10, oscilará de 0 até 3,1416, ao longo das repetições do laço.

Na linha 11 é que a frequência é calculada. Note que existe um valor de base em torno do qual ela será calculada. Esse valor é definido pela variável "base", na linha 4, como sendo 2 000. Na própria linha 11, o valor do seno é multiplicado por 1 000. Isso quer dizer que, a cada execução do laço, o valor da frequência oscilará de acordo com uma função senoidal de amplitude 1 000, com variação de 0 a 3,1416, conforme mostrado na Figura 32.

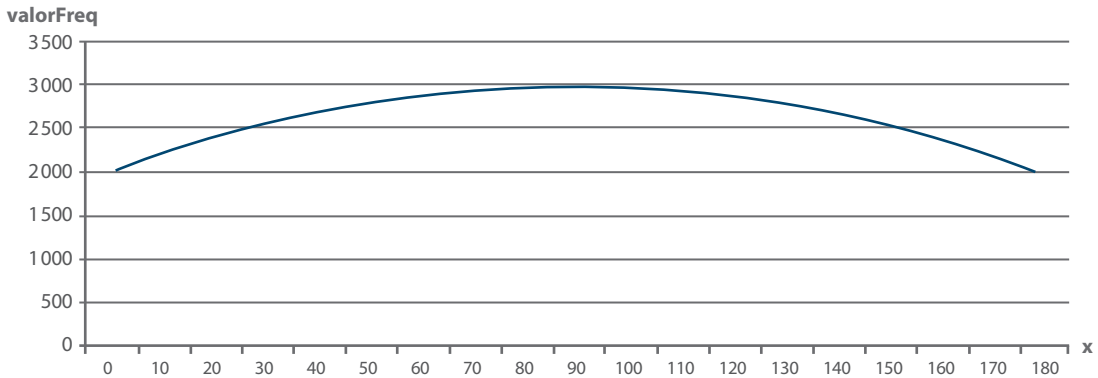


Figura 32 – Variável "valorFreq" *versus* variável x, para x de 0 a 180.

Na linha 12, o valor da frequência é enviado ao pino 10, com a função "tone". Na linha 13, há uma instrução para o código esperar por 2 milissegundos, e a função "loop" termina e pode ser executada novamente.

Discussão dos resultados

- Física 2ª série – volume 2. Situação de Aprendizagem 2. **Pergunta sugerida**: qual o princípio de funcionamento do *buzzer*, ou seja, como ele gera o som?

O *buzzer* é feito com um material piezoelétrico, que vibra ao receber uma corrente elétrica. A vibração desse material é que causa variações de pressão na atmosfera, exatamente como as mostradas na Figura 1 da página 15 do Caderno de Física.

- Física 2ª série – volume 2. Situação de Aprendizagem 3. **Pergunta sugerida**: qual é o papel da frequência de controle no som gerado?
  Nesse caso, a discussão do roteiro 3, na página 19 do Caderno, adquire uma utilidade muito grande. A frequência é definida e é apresentada a diferença entre sons graves e agudos. Isso é interessante, porque a frequência é exatamente o parâmetro usado para controlar o som emitido pelo *buzzer*.
- Física 2ª série – volume 2. Situação de Aprendizagem 4. **Pergunta sugerida**: a vibração do *buzzer* é natural ou forçada?
  A discussão da página 28 pode ser retomada aqui. No caso do *buzzer*, a vibração é forçada, pois a passagem da corrente elétrica faz que ele vibre. Cessada a corrente elétrica, ele para de vibrar.

PERCEBENDO A LUMINOSIDADE DO AMBIENTE

Problema a ser resolvido

É possível construir um dispositivo que permita traduzir a luminosidade em sinais sonoros? Para resolver esse problema, será usado um sensor de luminosidade e um *buzzer*, que gerará uma série de bips, cujo intervalo variará de acordo com a intensidade luminosa captada pelo sensor.

Habilidades que se pretende que os estudantes desenvolvam

Neste projeto, os estudantes perceberão como funciona um sensor de luminosidade e como ele pode interagir com outros elementos em um sistema automatizado. Como será usada parte dos componentes da experiência anterior, os estudantes verão que é possível empregar diferentes componentes combinados para gerar comportamentos bastante diferentes.

Tempo previsto

O tempo necessário para a execução desta atividade é de cerca de 50 minutos.

Como resolver o problema?

Material necessário

Protoboard, *buzzer*, resistor de 10 kΩ, sensor LDR e 7 fios *jumpers*.

Configuração do circuito

Os componentes devem ser montados como mostrado na Figura 33.

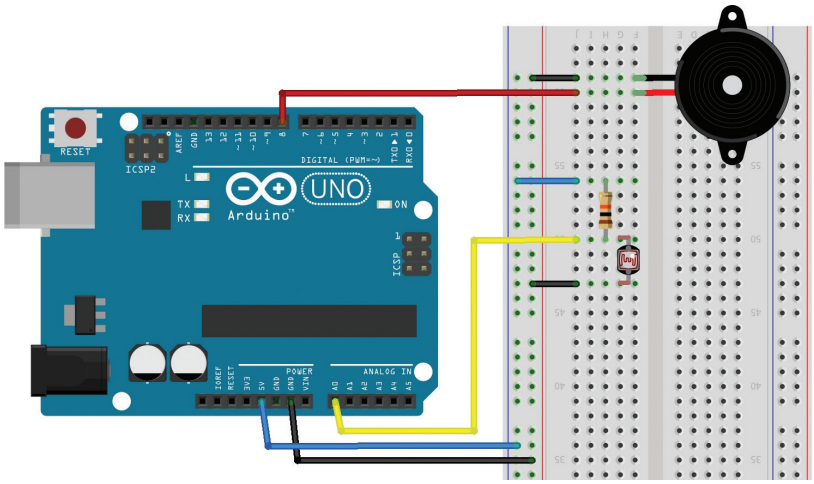


Figura 33 – Montagem esquemática do projeto *Percebendo a luminosidade do ambiente*.

A Figura 34 mostra o projeto montado.

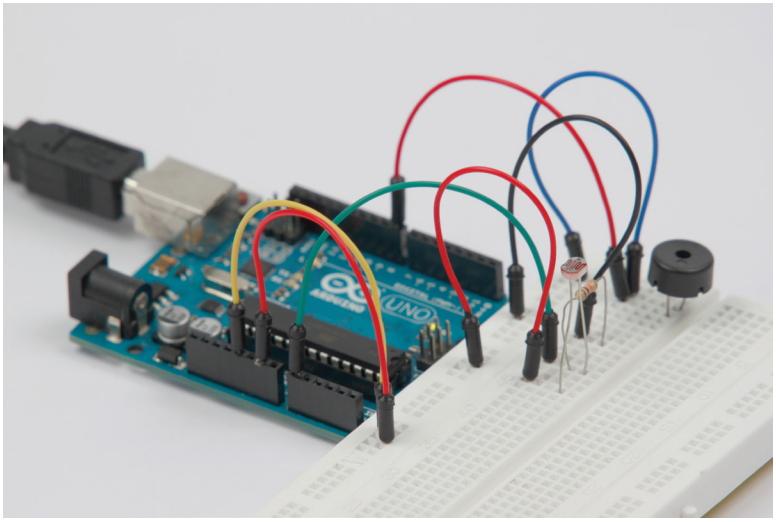


Figura 34 – Projeto *Percebendo a luminosidade do ambiente* montado.

Código-fonte do projeto

Listagem 6 – Código-fonte do projeto *Percebendo a luminosidade do ambiente*

```
1 //Projeto 3 – Percebendo a luminosidade do ambiente
2 int pinoBuzzer = 8; //Piezo no pino 8
3 int pinoLDR = 0; //LDR no pino analógico 0
4  int valorLDR = 0; //Valor lido do LDR
5 void setup() {
```

```
6  }
7  void loop() {
8   valorLDR = analogRead(pinoLDR); //Leitura do valor do LDR
9   tone(pinoBuzzer, 1000); //Toca um som de 1 000 Hz no buzzer
10   delay(10); //Espera 10 milissegundos
11   noTone(pinoBuzzer); //Suspende o som
12   delay(valorLDR); //Espera o valor do LDR milissegundos
13 }
```

O código-fonte para este projeto é iniciado com a declaração e inicialização de três variáveis, "pinoBuzzer", que contém o número do pino de saída digital que será usado para acionamento do *buzzer*, "pinoLDR", que contém o número do pino de entrada analógica que receberá os sinais do sensor LDR, e a variável "valorLDR", que armazenará o valor enviado pelo sensor LDR.

A função "setup" não está sendo usada para nenhum comando explícito, mas ela tem de ser executada, mesmo com o conteúdo vazio.

A função "loop" usa uma sequência de atividades, que são discutidas a seguir.

Na linha 8, ocorre a leitura do valor vindo do sensor LDR. É usada a função "analogRead", que precisa receber como parâmetro o número do pino do qual será feita a leitura. Esse número está armazenado na variável "pinoLDR", que é, então, passada como parâmetro. A função "analogRead" será usada também nos próximos projetos, e pode ser empregada sempre que for necessário ler a entrada de dados de um sensor analógico conectado a um dos pinos de leitura analógica do Arduino.

Na linha 9, é enviado um sinal para que o *buzzer*, que está conectado no pino cujo número está armazenado na variável "pinobuzzer", emita um som de 1 000 Hz. A linha 10, então, faz o sistema aguardar 10 milissegundos, com o *buzzer* gerando o som. Na linha 11, a função "noTone" interrompe o som do *buzzer*. Ela precisa como parâmetro o número do pino no qual o *buzzer* está conectado.

Na linha 12, o sistema, então, aguarda uma duração em milissegundos que é exatamente igual ao valor enviado pelo sensor LDR. Durante o funcionamento do sistema, você perceberá que um som composto por pequenos bips será gerado. Se o sensor LDR receber mais luz, ele enviará um valor menor para o Arduino. Nesse caso, o tempo de espera especificado na linha 12 será tanto menor quanto mais luz o sensor receber. É possível verificar o efeito disso cobrindo o sensor com a mão ou iluminando-o com uma lanterna.

Discussão dos resultados

- Física 2ª série – volume 2. Situação de Aprendizagem 7. **Pergunta sugerida:** o sensor LDR é sensível à luz, assim como os nossos olhos. Como o sensor LDR é capaz de perceber as diferenças de luminosidade?

Os estudantes pesquisarão então sobre o funcionamento do sensor LDR. Esse sensor é feito com um material que tem sua resistência variante com a temperatura. Assim, ao receber mais ou menos luz, variará a diferença de potencial entre seus terminais, se por ele estiver passando uma corrente elétrica. Os princípios de funcionamento do olho humano são os mesmos. Entretanto, os sensores são os fotorreceptores presentes em nossa retina, conforme explicado no texto *A câmara escura*, que se inicia na página 44 do Caderno.

- Física 3ª série – volume 1. Situação de Aprendizagem 1. **Pergunta sugerida:** o sistema de visão do olho humano tem algo a ver com o circuito montado neste projeto?
  A resposta é sim. Conforme pode ser verificado no quadro *Eletricidade no Corpo Humano: impulsos elétricos do olho para o cérebro*, parte da estrutura do olho humano envia sinais elétricos para o cérebro, onde são processados. Isso equivale aos sinais analógicos enviados do sensor LDR para o Arduino processar.
  Outro aspecto interessante que pode ser discutido é o fato de as pessoas com deficiência visual não poderem ver e, portanto, não perceberem a intensidade luminosa. No entanto, elas podem ouvi-la. Este dispositivo poderia trazer essa informação a esse grupo de pessoas. Talvez essa seja uma maneira de os deficientes visuais apreciarem a luminosidade de diferentes pontos em um ambiente e poderem construir uma representação sobre ela.
- Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida:** como acontecem as conexões de circuito entre o Arduino e o *buzzer* e o Arduino e o sensor LDR, com o resistor?
  As conexões acontecem de forma a atuar em circuitos conectados de maneira lógica pelo Arduino. Existe uma malha que é responsável por acionar o *buzzer*, enquanto outra alimenta e recebe sinais do sensor LDR. Como no caso do LED, o resistor existe no circuito para proteger o sensor LDR de altos valores de tensão.

ACIONAMENTO DE UM MOTOR DE CORRENTE CONTÍNUA

Problema a ser resolvido

É possível controlar a velocidade de grandes e potentes motores usando apenas dois dedos? Neste projeto, será verificado que pequenas variações de corrente podem ser usadas para variar a potência enviada para sistemas maiores, como é o caso de motores.

Habilidades que se pretende que os estudantes desenvolvam

Neste projeto, os estudantes terão a oportunidade de reconhecer e compreender o funcionamento de motores de corrente contínua e as aplicações de resistências variáveis e fontes externas para os componentes.

Tempo previsto

O tempo previsto para esta atividade é de cerca de 110 minutos.

Como resolver o problema?

Primeiro, será trabalhado um modelo simplificado, apenas para compreender o funcionamento de um motor de corrente contínua. Depois dessa primeira montagem, será feita outra montagem, que utilizará um potenciômetro, que, atuando com outros componentes, será responsável por controlar a velocidade do motor.

Material necessário

Motor de corrente contínua, porta-pilhas, terminal de dois parafusos, díodo, transistor TIP 120, potenciômetro e 11 fios *jumpers*.

Atenção: são quatro pilhas no porta-pilhas do *kit*. Como elas estão ligadas em série, o circuito não será alimentado se uma delas não estiver presente ou estiver descarregada.

Observação: é conveniente, neste momento, realizar três ações para facilitar o uso do motor e do porta-pilhas. Essas atividades não são obrigatórias para o funcionamento dos projetos, mas facilitam bastante a manipulação desses dois componentes. Primeiro, parafuse os terminais do porta-pilhas no terminal de parafusos. Depois, solde cada terminal do motor a um pino do pente de pinos que veio com o *kit*. Além disso, corte uma tira de papel e coloque no eixo do motor para facilitar a visualização de sua rotação. Essas modificações são mostradas na Figura 35.

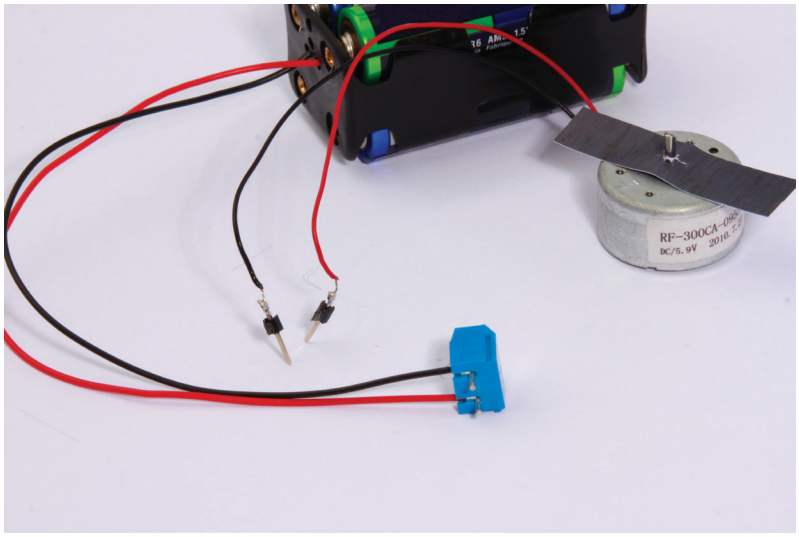


Figura 35 – Terminais do porta-pilhas, terminal de parafusos e terminais do motor com pinos soldados, bem como a tira de papel no eixo do motor de corrente contínua.

Faça a montagem como mostrado na Figura 35. Efetue primeiro a configuração sem colocar as pilhas – coloque-as depois de ter montado o circuito. A Figura 36 mostra a configuração esquemática do projeto.

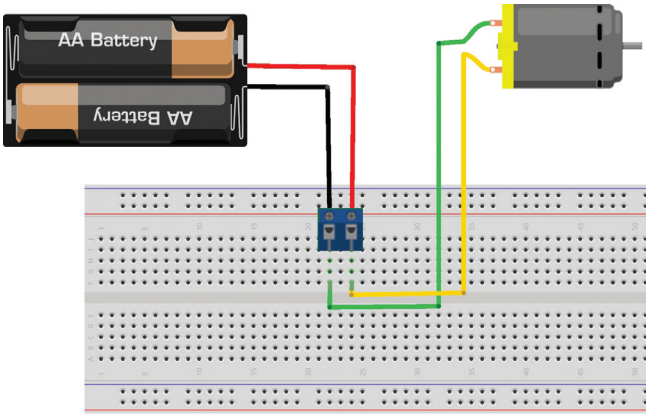


Figura 36 – Montagem esquemática para funcionamento do motor sem controle de velocidade.

A Figura 37 mostra a configuração montada.

Você verá que o motor girará assim que as pilhas estiverem todas colocadas.

Nesta configuração, a energia elétrica das pilhas é transmitida diretamente ao motor, sem haver nenhum intermediário. Se você inverter a polaridade do porta-pilhas, apenas conectando-o no sentido inverso, o motor girará no sentido contrário ao que girava antes. Isso é uma característica dos motores de corrente contínua.

Ao controlar a velocidade de rotação do motor, por intermédio do Arduino, não é aconselhável usar diretamente a fonte de energia dele, mas sim uma fonte de energia externa, porque em muitos casos a corrente ou a voltagem necessárias para acionar o motor são maiores do que aquela que ele pode fornecer.

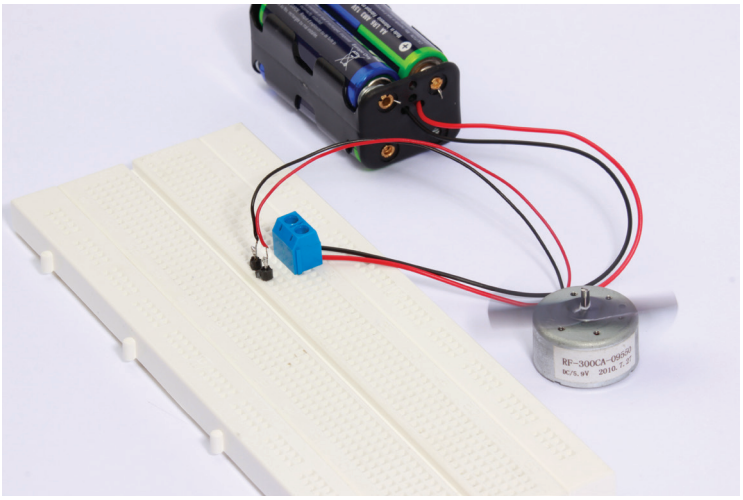


Figura 37 – Configuração da Figura 36 montada.

Para controlar o fornecimento dessa energia externa ao motor, fazendo que ele gire com mais ou menos velocidade, será usado um potenciômetro, que enviará um sinal analógico ao Arduino. O Arduino, com base nesse sinal, enviará um aviso a um transistor, que, por sua vez, controlará a quantidade de energia que será transmitida do sistema de pilhas para o motor.

Configuração do circuito

A configuração é mostrada na Figura 38.

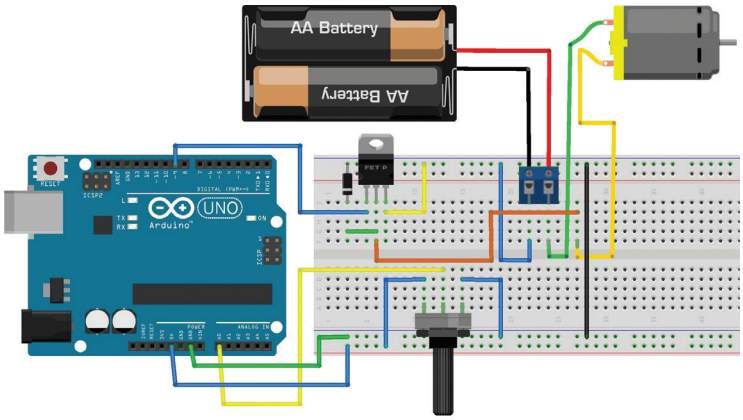


Figura 38 – Montagem esquemática do projeto *Acionamento de um motor de corrente contínua*.

A Figura 39 mostra o projeto montado.

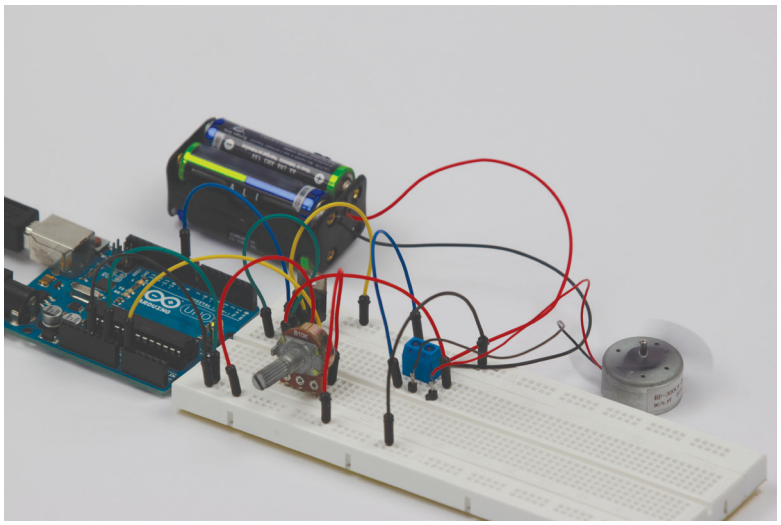


Figura 39 – Projeto *Acionamento de um motor de corrente contínua* montado.

Código-fonte do projeto

Listagem 7 – Código-fonte do projeto *Acionamento de um motor de corrente contínua*

```
1 //Projeto 4 – Acionamento de um motor de corrente contínua
2 int pinoPotenciometro = 0; //Entrada Analógica A0, para o potenciômetro
3 int pinoTransistor = 9; //Pino PWM para o transistor
4 int valorPotenciometro = 0; //Valor retornado pelo potenciômetro
5 void setup() {
6 pinMode(pinoTransistor, OUTPUT);  //Definição da saída para o pino ligado ao transistor
7 }
```

```
8  void loop() {
9  valorPotenciometro = analogRead(pinoPotenciometro) / 4;  //Leitura do potenciômetro
10  analogWrite(pinoTransistor, valorPotenciometro);  //Envio do sinal para o transistor
11 }
```

O código-fonte do projeto inicia-se com a declaração das variáveis que conterão os valores dos pinos do potenciômetro e do transistor, respectivamente, "pinoPotenciometro" e "pinoTransistor". Note que o pino para o transistor é o número 9. Isso é necessário porque esse pino é do tipo PWM. O transistor precisa de um sinal analógico para poder controlar a quantidade de energia enviada ao motor.

Em seguida, é criada e iniciada com o valor 0 a variável "valorPotenciometro", que estará associada ao valor lido pelo Arduino a partir do potenciômetro.

Na função "setup", na linha 6, é especificado o pino conectado ao transistor como de saída. Isso é necessário porque, não esqueça, os pinos digitais podem funcionar tanto como entrada quanto como saída de dados.

Na função "loop", a variável "valorPotenciometro" recebe o valor analógico lido pelo pino ligado ao potenciômetro. Esse valor é dividido por 4. Essa divisão deve ocorrer porque o sinal vindo do potenciômetro varia de 0 até 1023, respectivamente, quando ele estiver enviando 0 V e 5 V. Enquanto isso, o valor que pode ser enviado ao pino do transistor varia de 0 a 255. Assim sendo, o valor da variável "valorPotenciometro", como é um inteiro, estará sempre entre 0 e 255.

Em seguida, na linha 10, o "valorPotenciometro" é enviado ao pino conectado ao transistor e permitirá o controle do motor, passando a ele ou não toda a energia do sistema de pilhas.

Discussão dos resultados

- Física 1ª série – volume 1. Situação de Aprendizagem 1. **Pergunta sugerida**: quais são os tipos de movimento da experiência e como eles podem ser quantificados?
  Na experiência, ocorrem dois tipos de movimentos de rotação: no potenciômetro e no motor. Enquanto o movimento do potenciômetro é limitado a um valor menor do que uma volta, o do motor não tem limite com relação ao *deslocamento angular*, que é contínuo.
  Por causa da função exercida pelo potenciômetro, que está diretamente ligada a sua posição angular, é mais conveniente medir esse movimento pelo parâmetro deslocamento angular, que pode ser expresso tanto em graus quanto em radianos. Por outro lado, no caso do motor, a *velocidade angular* é o parâmetro mais adequado para medir o seu movimento, na maior parte das ocasiões. Isso acontece porque os processos nos quais o motor está envolvido, normalmente, têm a *velocidade de rotação* como o parâmetro a ser monitorado. Esse é o caso do motor que está sendo usado para acionar um ventilador, por exemplo.

- Física 1ª série – volume 1. Situação de Aprendizagem 11. **Pergunta sugerida**: quais são as fontes de energia envolvidas no funcionamento do projeto?
  As fontes de energia envolvidas são de duas origens. Para alimentar o Arduino, a energia vem do computador, por meio da porta USB. Para alimentar o motor, por outro lado, a energia vem das pilhas que estão conectadas em série, no porta-pilhas. Nesse momento, podem ser discutidas as fontes de energia de vários equipamentos que fazem parte do dia a dia do estudante.
- Física 3ª série – volume 1. Situação de Aprendizagem 12. **Pergunta sugerida**: qual é o princípio físico que rege o funcionamento dos motores de corrente contínua?
  Ao realizar a atividade associada à Situação de Aprendizagem 12, o estudante será capaz de compreender o funcionamento de motores de corrente contínua. Essa Situação de Aprendizagem discorre sobre muitos aspectos que podem ser verificados pelos estudantes ao montar o projeto. Por exemplo, a dependência que a velocidade de rotação tem da diferença de potencial aplicada ou a inversão do sentido de rotação que ocorre ao trocar os terminais que conectam o motor à *protoboard*.
- Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida**: qual é a relação entre o transistor e o potenciômetro no projeto montado?
  O potenciômetro envia um sinal analógico ao Arduino, que é proporcional ao giro do botão. Esse movimento altera a resistência efetiva do potenciômetro. É essa variação de resistência que é sentida pelo Arduino, pois há uma queda de tensão em seu pino de entrada analógico. O Arduino, por sua vez, envia um sinal ao transistor, que depende da leitura dessa entrada analógica. O transistor, então, permite que flua uma parte da energia das pilhas ao motor.

USANDO O SENSOR DE TEMPERATURA

Problema a ser resolvido

Como medir a temperatura de um *freezer* sem passar frio? Será visto neste projeto que é possível instalar um termistor para ler os valores de temperatura e realizar a análise dela no computador.

Habilidades que se pretende que os estudantes desenvolvam

Neste projeto, os estudantes terão a oportunidade de reconhecer e compreender o funcionamento do termistor e a aplicação dos conhecimentos de transferência de calor por condução, convecção e radiação.

Tempo previsto

O tempo previsto para esta atividade é de aproximadamente 50 minutos.

Como resolver o problema?

Material necessário

Resistor de 10 kΩ, termistor de 10 kΩ, *protoboard* e 3 fios *jumpers*.

Configuração do circuito

A configuração é mostrada na Figura 40.

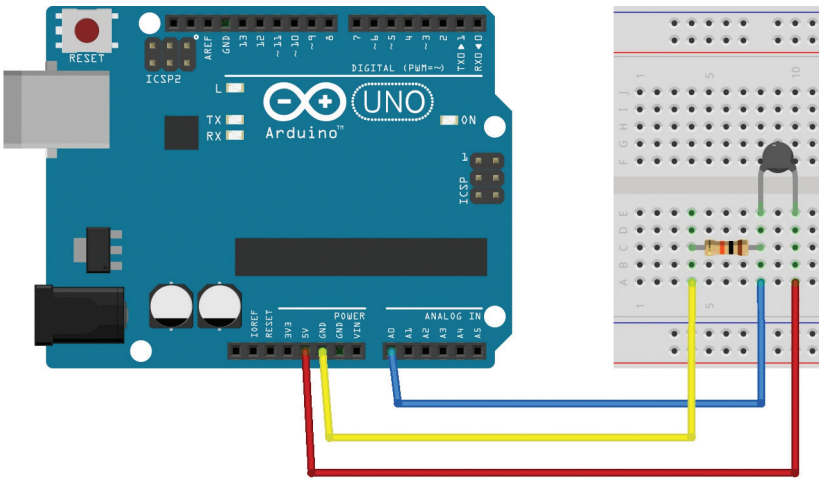


Figura 40 – Configuração esquemática do projeto *Usando o sensor de temperatura*.

A Figura 41 mostra o projeto montado.

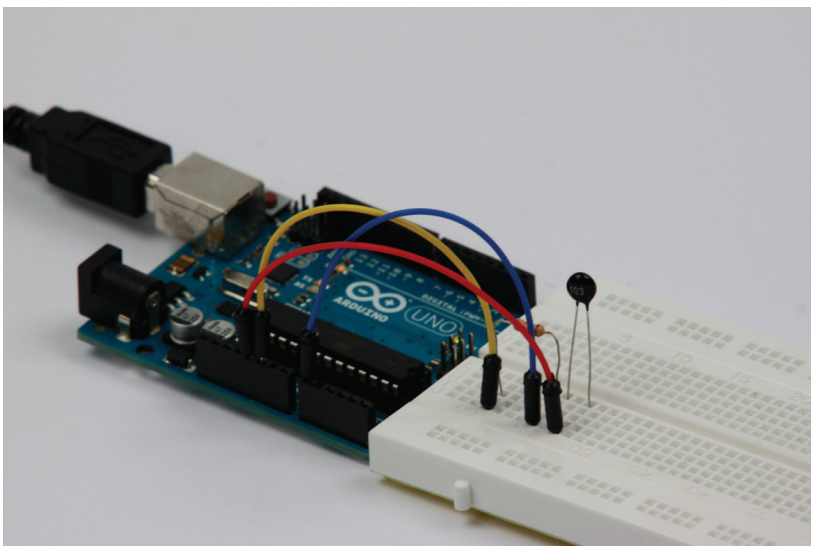


Figura 41 – Projeto *Usando o sensor de temperatura* montado.

Código-fonte do projeto

Listagem 8 – Código-fonte do projeto *Usando o sensor de temperatura*

```
1  //Projeto 5 – Usando o sensor de temperatura
2  #include <math.h>
3  double Termistor(int adc) {
4  double Temp;
5   //Processamento do valor lido para o Termistor, para mostrar a temperatura
6   //Temperatura em Kelvin:
7   Temp = log(10000.0*((1024.0/adc -1)));
8  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
9   //Conversão para graus Celsius
10   Temp = Temp - 273.15;
11  return Temp;
12  }
13 void setup() {
14  Serial.begin(9600);
15 }
16 void loop() {
17   Serial.print("Temperatura: ");
18  Serial.print(int(Termistor(analogRead(0))));  //Impressão da temperatura
19  Serial.println(" oC");
20  delay(100);
21 }
```

O código-fonte é iniciado com a instrução na linha 2, para a inclusão das funções matemáticas no programa. Isso se faz necessário porque é preciso usar a função logarítmica da linguagem C, que está definida no arquivo de inclusão *math.h*.

As linhas de 3 até 9 contêm a definição de uma função, a "Termistor", que é capaz de calcular o valor da temperatura a partir do valor lido do termistor. Na linha 3, tem-se a declaração do nome da função, "Termistor". A palavra "double" antes do nome significa que a função retornará a um valor do tipo "double". A informação que vem em seguida, na mesma linha, é a lista de parâmetros que a função precisará receber para funcionar. Essa lista de parâmetros contém apenas uma variável, chamada de "ADC". Esse nome foi escolhido para fazer referência direta à operação de conversão (*analog-digital conversion*) de analógico, vindo do termistor, para digital, que será o valor lido.

Entre as linhas 4 e 11, é aplicado o método de cálculo da temperatura a partir do valor lido pelo termistor, usando a modelagem de *Steinhart-Hart*, que emprega uma equação cujos coeficientes numéricos dependem do tipo de termistor. Os valores utilizados nas equações desse código-fonte são específicos para os resistores de 10 kΩ.

O valor da temperatura é retornado pela função na linha 11.

A função "setup" contém a definição da taxa de transmissão de dados em bits por segundo. No caso, 9 600 bits por segundo. Outras taxas podem ser especificadas, mas o valor de 9 600 é normalmente o que é configurado pela porta dos computadores. Neste projeto, essa definição é necessária porque os valores de temperatura serão enviados para o computador a partir do Arduino, para que você possa efetuar sua leitura. Essa leitura é feita na linha 18, já dentro da função "loop". Note que a função "analogRead" é chamada já como parâmetro da função "Termistor". Por sua vez, a função "Termistor" é colocada como parâmetro para a função "Serial.print", que envia seu parâmetro para o computador, e o resultado, como será visto adiante, pode ser mostrado na janela *Serial monitor*. O resultado desse processo é que o valor recebido do termistor é repassado imediatamente para a função "Termistor", que retorna o valor da temperatura. Esse valor, por sua vez, é enviado ao computador, pois está como parâmetro da função "Serial.print".

Note que há uma chamada anterior à função "Serial.print" na linha 17, para imprimir o texto "Temperatura" na janela *Serial monitor*. Na linha 19, é acionada a função "Serial.println", para imprimir o texto com a unidade de graus Celsius. A diferença entre as funções "Serial.print" e "Serial.println" é que a primeira imprime e continua na mesma linha e a segunda pula uma linha após a impressão na janela *Serial monitor*.

Para ver o valor de temperatura lido pelo Arduino, acesse o "Menu Tools → Serial Monitor". Será aberta então uma janela, a *Serial monitor*, onde os dados enviados pelo Arduino poderão ser vistos. Isso é mostrado na Figura 42. Note que, na figura, a porta do Arduino no computador é a de número 12. No seu computador pode ser uma porta diferente.

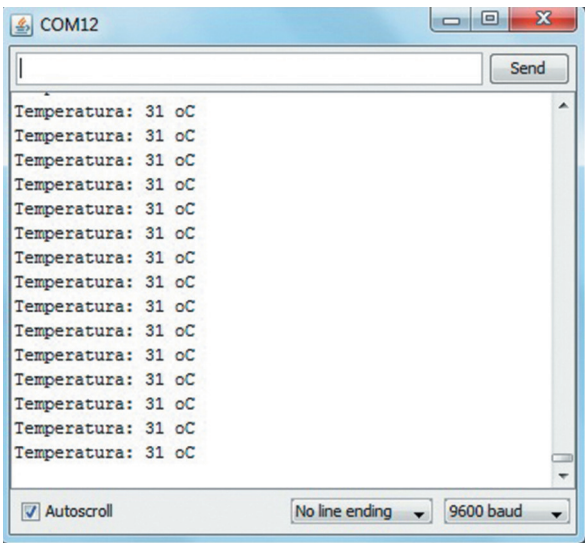


Figura 42 – Janela *Serial monitor* mostrando o valor da temperatura (http://arduino.cc).

Observação: Se você, porventura, alterar algum componente na *protoboard*, pode ser que seja necessário reiniciar o Arduino. Isso pode ser feito desconectando e conectando o cabo USB.

Discussão dos resultados

- Física 2ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida**: quais são as diferenças entre a medição de temperatura com o termômetro construído, usando o termistor, e o termômetro, cujas instruções de montagem estão na página 17 do Caderno? Quais são as vantagens de um e de outro?

Sem dúvida, os dois tipos de termômetro possuem vantagens e desvantagens em relação um ao outro. O termômetro que o Caderno sugere é bem mais simples e barato, por exemplo. Por outro lado, o termômetro construído por intermédio do termistor e do Arduino é bem mais sensível. Com certeza os estudantes perceberão outras vantagens e desvantagens. Além disso, com o uso do Arduino e do termistor, a leitura da temperatura pode ser feita em um ambiente e a sua visualização realizada em outro. Basta conectar o termistor ao circuito por meio de fios elétricos longos. Peça que sejam precisos nessa análise.

Segundo os fabricantes, um termistor em geral funciona em um intervalo de temperaturas que vai de –40 °C a 125 °C. Assim, o termistor pode ser usado em diversas situações e ambientes, inclusive aqueles que não são adequados aos seres humanos, no que se refere à temperatura.

- Física 2ª série – volume 1. Situação de Aprendizagem 15. **Pergunta sugerida**: existem equipamentos embutidos no motor para monitorar a temperatura dos sistemas de um carro? Onde eles estão instalados?

  Como o motor de um automóvel é uma máquina térmica baseada na queima de uma mistura de ar e combustível, ele trabalha com temperaturas elevadas. Por isso, se não for refrigerado continuamente, em pouco tempo essas temperaturas fundem as peças do motor, fazendo-o travar e parar de funcionar, danificando vários sistemas do carro, como consequência. Para que o sistema de refrigeração funcione, há a necessidade de uma medição muito precisa e robusta da temperatura. Dependendo do modelo do motor, a posição muda, mas normalmente ele se encontra na parte mais quente, que é o cabeçote. O sensor atua também com base na variação da resistência à passagem de corrente elétrica com a temperatura.

  Além da medição de temperatura do motor, também é necessário monitorar a temperatura no interior do veículo, para o sistema de condicionamento de ar, pelo qual é possível ajustá-la.
- Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida**: qual é o papel de cada um dos componentes do circuito trabalhado neste projeto?

  Os estudantes devem descrever a função de cada um dos elementos do circuito e como a corrente elétrica flui de um local para o outro. Especial atenção deve ser dada à necessidade de utilização do resistor, para proteger o termistor, baixando a tensão que chega nele.

USANDO O *DISPLAY* LCD

Problema a ser resolvido

Como receber informações de um sistema eletrônico sem ter de utilizar o monitor de um computador?

Habilidades que se pretende que os estudantes desenvolvam

Enviar mensagens de texto para mostradores do tipo LCD.

Tempo previsto

O tempo previsto para esta atividade é de cerca de 100 minutos.

Como resolver o problema?

Material necessário

Protoboard, *display* LCD e 10 fios *jumpers*.

Configuração do circuito

O circuito na primeira fase do projeto é mostrado na Figura 43.

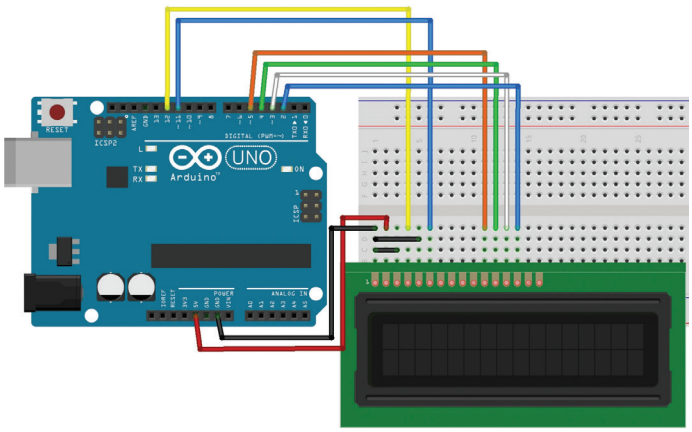


Figura 43 – Configuração esquemática do projeto *Usando o* display *LCD*.

A Figura 44 mostra o projeto montado.

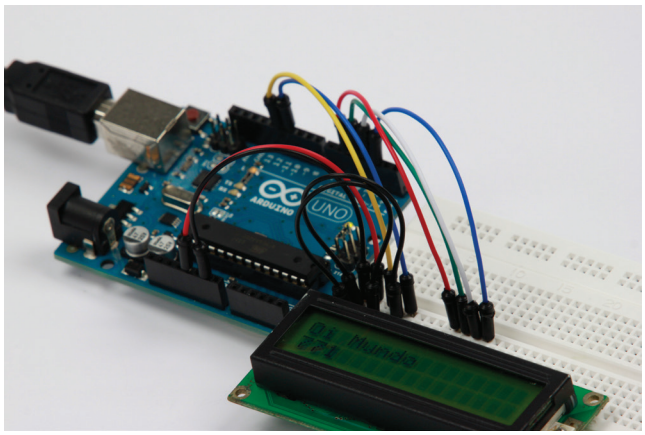


Figura 44 – Projeto *Usando o* display *LCD* montado.

Código-fonte do projeto

A listagem do código-fonte para a primeira fase do projeto é mostrada a seguir.

Listagem 9 – Código-fonte da primeira parte do projeto *Usando o* display *LCD*

```
1 //Projeto 6 – Usando o display LCD
2 #include <LiquidCrystal.h>
3 LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //Inicialização do display com os pinos da interface
```

```
4  void setup() {
5  lcd.begin(16, 2);   //Especifica número de colunas e linhas do LCD
6  lcd.print("Oi Mundo");   //Texto a ser exibido
7  }
8   void loop() {
9    lcd.setCursor(0, 1); //Posiciona o cursor na coluna 0 e linha 1
10   //Imprime o número de segundos desde o reinício
11   lcd.print(millis()/1000);
12  }
```

O Arduino possui uma biblioteca própria para a manipulação do *display* LCD. Essa biblioteca é chamada *LiquidCrystal*, e é incluída no programa pela instrução da linha 2.

A configuração inicial do LCD é feita na linha 3. É criado um objeto no código chamado "lcd". Esse objeto tem as funções que serão usadas para configurar e enviar informações para o LCD. Nesse momento, são definidos os pinos da interface do Arduino que serão empregados para a comunicação. Esses pinos serão 12, 11, 5, 4, 3 e 2.

A função "setup", que vai da linha 4 à linha 7, especifica o número de colunas e linhas do *display*, respectivamente, 16 e 2, por intermédio da função "lcd.begin".

Na linha 5, a função "lcd.print" envia uma informação na forma de texto para o *display*. No caso, a mensagem é "Oi Mundo".

A função "loop" vai das linhas 8 a 12. Na linha 9, a função "lcd.setCursor" permite a especificação da coluna e da linha, nessa ordem, para que a mensagem comece a ser escrita. A numeração das colunas obedece à seguinte lógica: na horizontal, a primeira linha de cima para baixo é a linha 0, a segunda linha é a linha 1; na vertical, as colunas são numeradas de 0 a 15, da esquerda para a direita.

Então, a função "lcd.print" é novamente usada para imprimir o número de segundos desde o início da execução do código. Isso é conseguido pela utilização da função "millis()", que retorna exatamente o número de milissegundos desde o início da execução.

Discussão dos resultados

- Física 3ª série – volume 2. Situação de Aprendizagem 17. **Pergunta sugerida**: como funciona um *display* do tipo LCD? Cite equipamentos nos quais os *displays* LCD são usados em sua casa.
  Os cristais líquidos são materiais que possuem, ao mesmo tempo, características associadas aos sólidos, como a orientação das suas moléculas em direções específicas, daí o nome cristal, bem como aos líquidos, como o fato de poderem fluir de um local para outro. Eles são muito sensíveis a mudanças de temperatura. Se uma corrente elétrica passar pelos cristais líquidos, por exemplo, aumentando a sua temperatura, podem mudar do estado líquido para o sólido – e, com isso, suas propriedades – deixando passar mais ou menos luz. Intercalando camadas de cristal

e grades que definem *pixels* ou formas maiores para gerar letras e números, e materiais condutores de eletricidade, é possível obter telas que reagem de uma maneira específica quando a eletricidade passa pelos seus terminais.
Em sua casa, o estudante encontrará telas de LCD em equipamentos como relógios, televisores, termômetros ou *displays* de eletrodomésticos.

Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida**: o *display* de LCD do *kit* funciona como um pequeno processador separado do Arduino?
Sim, é exatamente esse o caso. Ao virar o *display* LCD do *kit* para olhar o lado contrário à tela, por exemplo, podem ser vistos os circuitos integrados que são necessários para fazê-lo funcionar.

USANDO O *DISPLAY* LCD PARA MOSTRAR VALORES DE TEMPERATURA

Problema a ser resolvido

Como transferir o valor de temperatura medido pelo termistor para a tela de LCD?

Habilidades que se pretende que os estudantes desenvolvam

Combinar diferentes componentes de circuitos já estudados para criar uma nova solução. Esse é um aspecto que dará mais segurança para o desenvolvimento de projetos futuros.

Tempo previsto

O tempo previsto para esta atividade é em torno de 100 minutos.

Como resolver o problema?

Material necessário

Protoboard, 13 fios *jumpers*, *display* LCD, resistor de 10 kΩ e termistor de 10 kΩ.

Configuração do circuito

A configuração é mostrada na Figura 45.

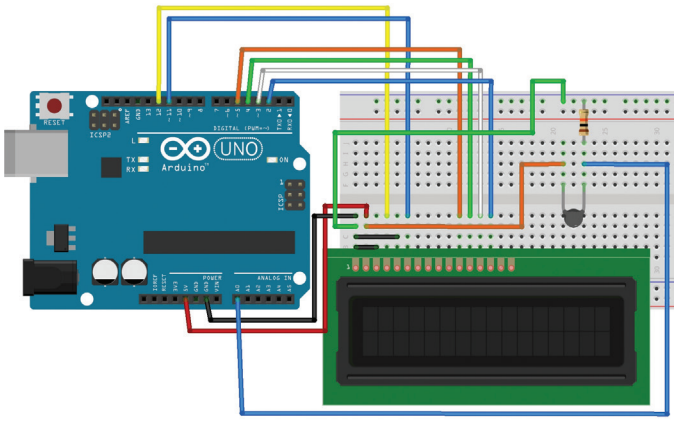


Figura 45 – Configuração esquemática para o projeto *Usando o* display *LCD para mostrar valores de temperatura*.

A Figura 46 mostra o projeto montado.

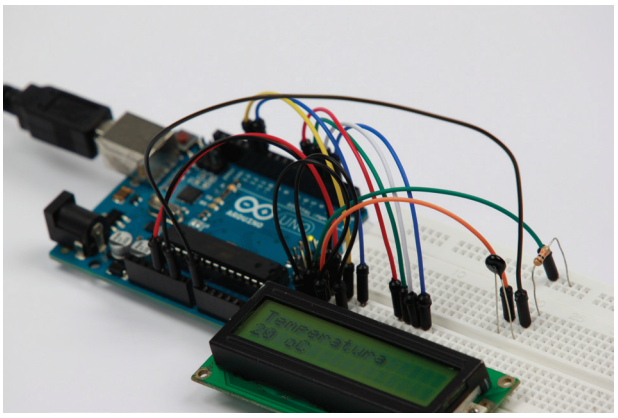


Figura 46 – Projeto *Usando o* display *LCD para mostrar valores de temperatura* montado.

Código-fonte do projeto

Na segunda fase do projeto, o código-fonte é mostrado na listagem a seguir.

Listagem 10 – Código-fonte do projeto *Usando o* display *LCD para mostrar valores de temperatura*

```
1  //Projeto 7 – Usando o display LCD para mostrar valores de temperatura
2  #include <LiquidCrystal.h>
3  #include <math.h>
4  LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //Inicialização do display com os pinos da interface
5  void setup() {
6   lcd.begin(16, 2);   //Especifica número de colunas e linhas do LCD
7   lcd.print("Temperatura");   //Texto auxiliar da temperatura
8  }
9  double Termistor(int adc) {
10   double Temp;
11   //Processamento do valor lido para o Termistor, para mostrar a temperatura
12   //Temperatura em Kelvin:
13   Temp = log(10000.0*((1024.0/adc -1)));
14  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
15  //Conversão para graus Celsius
16  Temp = Temp - 273.15;
17  return Temp;
18 }
19 void loop() {
20  lcd.setCursor(0, 1); //Posiciona o cursor na coluna 0 e linha 1
21   //Imprime a temperatura
22  lcd.print(int(Termistor(analogRead(0))));
23   lcd.print(" oC");
24 }
```

Esse código é uma mescla dos códigos dos projetos *Usando o sensor de temperatura* e *Usando o* display *LCD*. Assim sendo, serão apontadas as diferenças que aparecem devido, exatamente, a essa combinação.

Essas diferenças acontecem na função "loop".

A linha 20 é muito importante. A cada execução da função "loop", o cursor é reposicionado na localização (0, 1). Na linha 22, a função "lcd.print" é executada, com o parâmetro não sendo mais uma mensagem de texto, mas sim o valor de temperatura lido pelo sensor de temperatura e tratado pela função "Termistor". Na linha 23, a unidade, °C, é também enviada para o *display* LCD.

Na próxima execução da função "loop", quando for novamente executada a linha 20, tudo o que for escrito o será sobre o que havia antes. Isso permite que eventuais novos valores de temperatura sejam escritos no *display* LCD.

Discussão dos resultados

- Física 3ª série – volume 2. Situação de Aprendizagem 19. **Pergunta sugerida:** a temperatura medida pelo termistor e processada pelo Arduino pode ser usada para leitura remota, ou seja, em outro local, distante da medição?
  A resposta é sim. A partir do momento em que o dado de temperatura é enviado do Arduino ao computador, ele é digitalizado e, assim, pode ser enviado para qualquer local do mundo conectado à internet. Mas também é possível processá-lo das mais diversas maneiras. Por exemplo: com base em valores lidos periodicamente, podem ser feitas análises estatísticas para calcular valores sazonais médios de temperatura e para evidenciar tendências de aumento ou de diminuição desses valores.
- Física 3ª série – volume 1. Situação de Aprendizagem 3. **Pergunta sugerida:** qual é o caminho percorrido pelo sinal de temperatura até chegar ao *display* LCD?
  O caminho percorrido pelo sinal de temperatura começa pelo sinal enviado à porta analógica A0 contendo a influência do termistor. O Arduino processa, então, esse sinal, faz os cálculos da temperatura de acordo com o modelo de *Steinhart-Hart* e envia o sinal processado para as saídas digitais conectadas ao *display* LCD. Em termos de programação, a complexidade desse processo de envio ao LCD está contida na biblioteca *liquidCrystal*, da qual não é preciso conhecer todos os detalhes para usá-la.

PARA SABER MAIS

Este Caderno é apenas um ponto de partida para tudo o que se pode fazer com a plataforma Arduino. Existe uma grande quantidade de informações disponível na internet, o que permite desenvolver projetos com diferentes níveis de sofisticação. A seguir, serão comentados alguns livros e *sites* sugeridos, em que informações adicionais podem ser obtidas.

Tenha em mente que os conteúdos, bem como os endereços dos *sites* indicados, podem mudar com o passar do tempo. Além disso, todos os dias surgem novas fontes na internet. Fique sempre atualizado.

- ***Site* oficial do projeto Arduino**. Contém uma grande quantidade de informações sobre diversos aspectos da plataforma. A seção "Learning" é dedicada especificamente a isso. Essa seção é subdividida em três áreas. A primeira, "Getting Started", tem informações básicas sobre como iniciar-se na plataforma. A segunda, "Examples", permite que você tenha contato com exemplos simples. E a terceira, "Playground", permite que você navegue por projetos e diversas sugestões de projetos. O *site* está em inglês, mas a seção "Playground" está disponível em português. Disponível em: <http://arduino.cc>. Acesso em: 18 jul. 2014.

- **Blog oficial do projeto**. É uma grande fonte de inspiração e permite que você fique convenientemente informado sobre as últimas notícias envolvendo a plataforma. Verifique periodicamente este *link* e peça que os estudantes também o façam. *Site* em inglês. Disponível em: <http://blog.arduino.cc>. Acesso em: 18 jul. 2014.

- ***Site* do programa Fritzing**. Todas as representações esquemáticas deste Caderno foram feitas usando esse programa, que é gratuito e dá a possibilidade de representar todos os seus projetos de maneira rápida e fácil. Além disso, o *site* também traz muitas informações técnicas. *Site* em inglês. Disponível em: <http://fritzing.org>. Acesso em: 18 jul. 2014.

- ***Site* Robótica Simples**. Empresa que disponibiliza projetos feitos com Arduino e com eletrônica de uma maneira geral. *Site* em português. Disponível em: <http://roboticasimples.com>. Acesso em: 18 jul. 2014.

- ***Site* Brasil Robotics**. Tem muita informação sobre o Arduino e projetos simples, bem como muitas explicações sobre diversos componentes eletrônicos. *Site* em português. Disponível em: <http://brasilrobotics.blogspot.com.br>. Acesso em: 18 jul. 2014.

- MCROBERTS, Michael. *Arduino básico*. São Paulo: Novatec, 2011. Este livro é um ótimo ponto de partida para estudar a plataforma de uma maneira bastante global. É abordada, por capítulo, uma grande variedade de sensores e de projetos.

- MONK, Simon. *30 projetos com Arduino*. Porto Alegre: Bookman, 2014. Este livro possui uma descrição muito concisa da plataforma Arduino, bem como projetos que vão dos mais simples aos mais sofisticados.

WARREN, John-David; ADAMS, Josh; MOLLE, Harald. *Arduino robotics*. New York: Apress, 2011. Este livro é uma fonte de referência indispensável para conhecer a fundo a robótica e como ela pode ser abordada com o Arduino. O livro é composto por capítulos que descrevem os detalhes construtivos de nove tipos diferentes de robôs, indo desde seguidores de linha até robôs que simulam o movimento de insetos, passando por bardos robotizados.

CONCEPÇÃO E COORDENAÇÃO GERAL
PRIMEIRA EDIÇÃO 2014

COORDENADORIA DE GESTÃO DA EDUCAÇÃO BÁSICA (CGEB)

Coordenadora
Maria Elizabete da Costa

Diretor do Departamento de Desenvolvimento Curricular de Gestão da Educação Básica
João Freitas da Silva

Diretora do Centro de Ensino Fundamental dos Anos Finais, Ensino Médio e Educação Profissional – CEFAF
Valéria Tarantello de Georgel

Coordenação Técnica
Roberto Canossa
Roberto Liberato
Suely Cristina de Albuquerque Bomfim

PROGRAMA ENSINO INTEGRAL

Coordenação da elaboração dos materiais de apoio ao Programa Ensino Integral
Valéria de Souza

Apoio técnico e pedagógico
Marilena Rissutto Malvezzi

Equipe Técnica
Maria Silvia Sanchez Bortolozzo (coordenação), Carlos Sidiomar Menoli, Dayse Pereira da Silva, Elaine Aparecida Barbiero, Helena Cláudia Soares Achilles, João Torquato Junior, Kátia Vitorian Gellers, Maria Camila Mourão Mendonça de Barros, Maria Cecília Travain Camargo, Maria do Carmo Rodrigues Lurial Gomes, Maúna Soares de Baldini Rocha, Pepita de Souza Figueredo, Sandra Maria Fodra, Tomás Gustavo Pedro, Vera Lucia Martins Sette, Cleuza Silva Pulice (colabor.) e Wilma Delboni (colabor.)

GESTÃO DO PROCESSO DE PRODUÇÃO EDITORIAL 2014

FUNDAÇÃO CARLOS ALBERTO VANZOLINI
Presidente da Diretoria Executiva
Mauro de Mesquita Spínola

Vice-Presidente da Diretoria Executiva
José Joaquim do Amaral Ferreira

GESTÃO DE TECNOLOGIAS EM EDUCAÇÃO

Direção da Área
Guilherme Ary Plonski

Coordenação Executiva do Projeto
Angela Sprenger e Beatriz Scavazza

Gestão da Produção Editorial
Luis Marcio Barbosa e Renata Simões

Equipe de Produção

Editorial: Guiomar Milan (coordenação), Bruno Reis, Carina Carvalho, Karina Kempter, Karinna A. C. Taddeo, Letícia Maria Delamare Cardoso, Marina Murphy e Natália Pereira Leal

Direitos autorais e iconografia: Denise Blanes (coordenação), Beatriz Fonseca Micsik, Érica Marques, José Carlos Augusto, Marcus Ecclissi e Vanessa Leite Rios

Produção editorial: Adesign (projeto gráfico) e Casa de Ideias (diagramação e ilustrações não creditadas)

ELABORAÇÃO DOS CONTEÚDOS ORIGINAIS

Coordenação do desenvolvimento dos conteúdos dos volumes de apoio ao Programa Ensino Integral
Ghisleine Trigo Silveira

Cadernos do Gestor

Avaliação da aprendizagem e nivelamento
Zuleika de Felice Murrie

Diretrizes do Programa Ensino Integral
Valéria de Souza (coord.), Carlos Sidiomar Menoli, Dayse Pereira da Silva, Elaine Aparecida Barbiero, Helena Cláudia Soares Achilles, João Torquato Junior, Kátia Vitorian Gellers, Maria Camila Mourão Mendonça de Barros, Maria Cecília Travain Camargo, Maria do Carmo Rodrigues Lurial Gomes, Maria Silvia Sanchez Bortolozzo, Maúna Soares de Baldini Rocha, Pepita de Souza Figueredo, Sandra Maria Fodra, Tomás Gustavo Pedro, Vera Lucia Martins Sette, Cleuza Silva Pulice (colabor.) e Wilma Delboni (colabor.)

Formação das equipes do Programa Ensino Integral – Vol. 1
Beatriz Garcia Sanchez, Cecília Dodorico Raposo Batista, Maristela Gallo Romanini e Thais Lanza Brandão Pinto

Formação das equipes do Programa Ensino Integral – Vol. 2
Beatriz Garcia Sanchez, Cecília Dodorico Raposo Batista, Maristela Gallo Romanini e Thais Lanza Brandão Pinto

Modelo de gestão do Programa Ensino Integral
Maria Camila Mourão Mendonça de Barros

Modelo de gestão de desempenho das equipes escolares
Ana Carolina Messias Shinoda e Maúna Soares de Baldini Rocha

Cadernos do Professor

Biologia: atividades experimentais e investigativas
Maria Augusta Querubim e Tatiana Nahas

Ciências Físicas e Biológicas: atividades experimentais e investigativas
Eugênio Maria de França Ramos, João Carlos Miguel Tomaz Micheletti Neto, Maíra Batistoni e Silva, Maria Augusta Querubim, Maria Fernanda Penteado Lamas e Yassuko Hosoume

Física: atividades experimentais e investigativas
Eugênio Maria de França Ramos, Marcelo Eduardo Fonseca Teixeira, Ricardo Rechi Aguiar e Yassuko Hosoume

Manejo e gestão de laboratório: guia de laboratório e de descarte
Solange Wagner Locatelli

Matemática: atividades experimentais e investigativas – Ensino Fundamental – Anos Finais
Maria Silvia Brumatti Sentelhas

Matemática: atividades experimentais e investigativas – Ensino Médio
Ruy César Pietropaolo

Pré-iniciação Científica: desenvolvimento de projeto de pesquisa
Dayse Pereira da Silva e Sandra M. Rudella Tonidandel

Preparação Acadêmica
Marcelo Camargo Nonato

Projeto de Vida – Ensino Fundamental – Anos Finais
Isa Maria Ferreira da Rosa Guará e Maria Elizabeth Seidl Machado

Projeto de Vida – Ensino Médio
Isa Maria Ferreira da Rosa Guará e Maria Elizabeth Seidl Machado

Protagonismo Juvenil
Daniele Próspero e Rayssa Winnie da Silva Aguiar

Química: atividades experimentais e investigativas
Hebe Ribeiro da Cruz Peixoto e Maria Fernanda Penteado Lamas

Robótica – Ensino Fundamental – Anos Finais
Alex de Lima Barros

Robótica – Ensino Médio
Manoel José dos Santos Sena

Tutoria e Orientação de estudos
Cristiane Cagnoto Mori, Jacqueline Peixoto Barbosa e Sandra Maria Fodra

Cadernos do Aluno

Projeto de Vida – Ensino Fundamental – Anos Finais
Pepita de Souza Figueredo e Tomás Gustavo Pedro

Projeto de Vida – Ensino Médio
Pepita de Souza Figueredo e Tomás Gustavo Pedro

Apoio
Fundação para o Desenvolvimento da Educação – FDE

Catalogação na Fonte: Centro de Referência em Educação Mario Covas

• Nos cadernos de apoio ao Programa Ensino Integral são indicados *sites* para o aprofundamento de conhecimentos, como fonte de consulta dos conteúdos apresentados e como referências bibliográficas. Todos esses endereços eletrônicos foram checados. No entanto, como a internet é um meio dinâmico e sujeito a mudanças, a Secretaria da Educação do Estado de São Paulo não garante que os *sites* indicados permaneçam acessíveis ou inalterados.

• Os mapas reproduzidos no material são de autoria de terceiros e mantêm as características dos originais no que diz respeito à grafia adotada e à inclusão e composição dos elementos cartográficos (escala, legenda e rosa dos ventos).

S239r São Paulo (Estado) Secretaria da Educação.

Robótica: Ensino Médio; Caderno do Professor / Secretaria da Educação; coordenação, Valéria de Souza; textos, Manoel José dos Santos Sena. - São Paulo : SE, 2014.

56 p.

Material de apoio ao Programa Ensino Integral do Estado de São Paulo.

ISBN 978-85-7849-702-6

1. Robótica 2. Atividade prática 3. Tecnologia 4. Ensino Médio 5. Programa Ensino Integral 6. São Paulo I. Souza, Valéria de. II. Sena, Manoel José dos Santos. III. Título.

CDU: 371.314:681.5(815.6)

GOVERNO DO ESTADO
DE SÃO PAULO