UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Instituto de Física
Programa de Pós-Graduação em Ensino de Física
Mestrado Profissional em Ensino de Física

UMA PROPOSTA DE ENSINO DE FÍSICA PARA TURMAS NOTURNAS

Material Didático para o Professor

Marcelo Elias da Silva

Material institucional associado à Dissertação de Mestrado de Marcelo Elias da Silva, apresentada ao Programa de Pós-Graduação em Ensino de Física, Instituto de Física, da Universidade Federal do Rio de Janeiro.

Orientador: Helio Salim de Amorim

Rio de Janeiro
Fevereiro de 2014

Sumário

CADERNO DO PROFESSOR

Caro professor, este material possui atividades (Atividades I, II e III) sugeridas para serem aplicadas na sua aula. É importante ressaltar que as atividades a serem apresentadas não são sequenciais, ou seja, estão ordenadas numa ordem lógica, didática, mas não numa ordem temporal em que uma segue a outra imediatamente. São propostas que devem se inserir num contexto maior de planejamento particularmente quanto à estrutura dos seus pré-requisitos próprios.

As atividades práticas propostas não requerem um espaço especializado, como um laboratório, devendo ser desenvolvidas na própria sala de aula tendo a mesa do professor como suporte ou bancada. Para as experiências será necessário um conjunto básico de materiais e ferramentas que relacionamos na Tabela I.1 do Anexo I. Esse material é acondicionado numa maleta que visa facilitar o transporte, a organização e o controle de estoque de materiais e do acervo de experiências, e que denominamos doravante Maleta do Professor. Em cada atividade particularizamos os materiais necessários.

Na apresentação das atividades seguimos um roteiro que estabelece uma sequência temporal de eventos em sala de aula que nos parece à forma mais interessante para a aplicação da atividade. Não se trata de um plano de aula, mas uma exposição de ideias que poderão servir como base para a elaboração do seu plano de aula. Em cada atividade temos uma seção **Comentários** onde fazemos um resumo da exposição da matéria naquilo que é pertinente à atividade, estabelecendo o que consideramos o nível de aprofundamento adequado e privilegiando a argumentação. Os comentários assumem um formato coloquial já visando à sala de aula. Um dos pontos que requer maior atenção é a aplicação da placa Arduino nas atividades. Como se trata sempre de um recorte das propriedades técnicas gerais de funcionamento da placa Arduino, direcionada a um público iniciante, isso vai exigir uma estratégia didática bem cuidada. Procuramos introduzir a placa Arduino gradualmente, num processo em espiral, onde cada atividade nova repassa os assuntos já aprendidos acrescentando novas propriedades e, por outro lado, partimos sempre da necessidade prática privilegiando a questão do *Como se faz? Como se usa?* Nos comentários voltamos a apresentar elementos da placa Arduino com a finalidade de exemplificar essa abordagem.

A aplicação dessas atividades vai exigir do professor certa familiaridade com a plataforma Arduino. Assumimos também, no que se segue que o professor tenha um microcomputador (PC) pronto para a aplicação das atividades com a placa Arduino, com a última versão da IDE instalada, com as definições de *setup* para a placa Arduino e a porta USB em uso e contendo todos os arquivos (*files*) pertinentes aos projetos que vamos descrever. O equipamento que melhor se adapta é o Notebook como esse distribuído pelo governo do Estado do Rio de Janeiro aos professores da rede pública. O Notebook é material do professor, que pode ser usado em casa para realizar todas as instalações e teste prévios com o material usado nas atividades. Gostaria de ressaltar que essas atividades são exemplificações, a placa Arduino pode ser aplicada em diversas áreas da Física no Ensino Médio, o professor pode desenvolver projetos utilizando esse material como base.

As atividades estão divididas em seções. As seções foram pensadas para que correspondam a uma aula, com uma estrutura própria e formando uma sequência lógica e didática. Essa divisão, no entanto, não é rígida podendo ser reformulada a critério de cada professor. Os exemplos de aplicações do método POE são apenas ilustrativos e podem, também, ser modificados conforme as circunstâncias enfrentadas por cada professor.

Apresentaremos a seguir modelo do plano de aula, porém deixamos o professor a vontade para confeccioná-lo de forma que melhor possa atender as suas aulas e necessidades dentro do processo de ensino. Ressaltamos a necessidade de utilização, nas aulas expositivas, de recursos modernos, com sua eficácia já testada e comprovada, para uma contextualização dos temas da Física a serem trabalhados e desenvolvidos durante as aulas. Os conhecimentos prévios e a realidade dos alunos devem ser aproveitados na associação da Física com o cotidiano dos estudantes. O planejamento de todo processo de ensino é fundamental para o seu melhor aproveitamento, pois o professor não deve esquecer que na prática, no ensino médio noturno, os tempos de aulas são menores que na teoria, sendo necessária uma otimização desse tempo através de recursos que facilitem o professor, como a utilização de slides nas aulas expositivas e práticas.

Plano de Aula

I. *Tema:*
O professor deverá relacionar o tema da Física que será abordado em sala de aula. Nas atividades propostas os temas são eletricidade e magnetismo.

II. *Metas da Atividade:*
Nessa seção o professor deve descrever as competências e habilidades que os alunos deverão desenvolver e os métodos utilizados para concretizar os objetivos. Nas atividades apresentadas, as metas são descritas de acordo com o tema e os objetivos.

III. *Objetivos:* O professor deve descrever os objetivos a serem alcançados pelos alunos, de forma clara e concisa, projetando o resultado geral relativo à execução de conteúdos e procedimentos, especificando os resultados esperados. Em cada uma das atividades propostas será especificada os objetivos de acordo com o tema.

IV. *Conceitos:* O professor deve descrever os conceitos planejados para serem desenvolvidos durante a aula. Nas atividades os conceitos trabalhados estão relacionados com a eletricidade e o magnetismo.

V. *Pré-requisito:* Antes da aplicação das atividades, os alunos precisam de aulas expositivas sobre os conceitos trabalhados, com as finalidades do desenvolvimento e assimilação do conteúdo. Algumas atividades também necessitam de conceitos assimilados em atividades anteriores, devendo o professor estar atento sobre essa possibilidade no seu planejamento. Nas atividades sugeridas, a atividade III tem como pré-requisito a atividade I, além da aula expositiva sobre os conceitos.

VI. *Material Utilizado:* O professor deve descrever todo o material que será utilizado, assim como foi apresentado nas atividades sugeridas através de tabelas.

VII. *Avaliação:* Ao final de cada atividade o professor deve passar exercícios e problemas para os alunos, onde uma parte deve ser apresentada e corrigida, como forma de exemplo, na própria aula; e a outra parte para o aluno desenvolver em casa, devendo o professor corrigir na aula seguinte. Para o aluno que se interessar, o professor deve propor um problema extra, como forma de desafio. Nas atividades encontram-se exemplos de exercícios e problemas para serem desenvolvidos com os estudantes.

VIII. *Tempo de Aula:* O professor deve usar 2 tempos de aula para a parte expositiva, mais 2 tempos de aula para a aplicação da prática e 1 tempo de aula para correção dos exercícios e problemas propostos. Caso o professor perceba a necessidade de mudança, os tempos de aulas podem variar de acordo com as características de cada turma e dos conceitos trabalhados.

IX. Observações Finais: Ao final de cada aplicação da atividade, o professor deve fazer uma avaliação do seu trabalho, dos resultados dos estudantes e do andamento da aula de uma forma geral, com o objetivo de sempre estar aprimorando a sua aula de acordo com esses resultados. Durante as aulas, se possível, o professor deve fazer uso do projetor, através do power point, nas aulas expositivas e práticas, com o objetivo de uma otimização do tempo.

Atividade I – Fazendo o LED piscar com a Arduino.

Metas da atividade: nesta atividade queremos desenvolver competências e habilidades dos conceitos de corrente elétrica, diferença de potencial, resistência elétrica, efeito Joule e da lei de Ohm. Dentre essas competências e habilidades que se encontram no currículo mínimo, destacamos:

- Reconhecer, utilizar, interpretar e propor modelos explicativos para fenômenos naturais ou sistemas tecnológicos (Nova EJA e Ensino Médio Noturno).
- Dimensionar circuitos ou dispositivos elétricos de uso cotidiano (Ensino Médio Noturno).
- Compreender os conceitos de corrente, resistência e diferença de potencial elétrico (Ensino Médio Noturno).

Montaremos um circuito elétrico para acender uma lâmpada, que depois trocaremos por um LED. Utilizaremos a placa Arduino para acender um LED e introduzir conceitos da linguagem de programação.

Objetivos da Atividade: construir o circuito elétrico físico a partir de sua representação esquemática; aprender a utilizar o *protoboard*; reconhecer os componentes elétricos comerciais (resistor, pilha, chave, lâmpada, LED e fios) e suas propriedades elétricas; aplicar a lei de Ohm e efeito Joule; introdução à placa Arduino.

Conceitos trabalhados: Diferença de potencial elétrico (DDP ou tensão); corrente e resistência elétrica (V, I e R); lei de Ohm; efeito Joule.

Pré-requisito: Os alunos devem ter os conceitos de corrente elétrica, resistência elétrica, diferença de potencial, Lei de Ohm e efeito Joule bem desenvolvido em aulas anteriores, pois essas atividades propostas não devem ser utilizadas para adquirir o conceito ou teoria.

Material utilizado: para a realização desta atividade, dividida em três seções, serão necessários os itens relacionados na Tabela 1.1.

Tabela 1.1 - Relação de materiais para a Atividade I

	Item:	Quant.	Observação
1	*Protoboard*	01	
2	Soquete e lâmpada (3V)	01	
3	LED de 5mm	03	Cores: amarelo, verde, vermelho;
4	Porta pilha para duas pilhas pequenas (AA)	01	
5	Pilha AA	02	Dê preferencia a pilhas recarregáveis. São mais ecológicas e fornecem mais corrente;
6	Fios encapados ou esmaltados	QN*	Para pequenas conexões;
7	Chave tipo *push-button*	01	
8	Resistores 100 ohm (1/8W)	03	
9	Arduino Uno (REV3)	01	
10	Cabos para conexão (Jumpers M/M)	04	Usados na conexão do *protoboard* e portas da Arduino.

(*) QN – quantidade necessária

1.1 – Descrição da Atividade I.

1.1.1 - Primeira seção.

Iniciamos apresentando o circuito esquemático indicado na Figura 1.1. Neste circuito temos as propriedades fundamentais de um circuito elétrico, isto é, uma fonte de tensão, um circuito elétrico fechado constituído de fios condutores e uma carga passiva que se caracteriza por uma resistência elétrica.

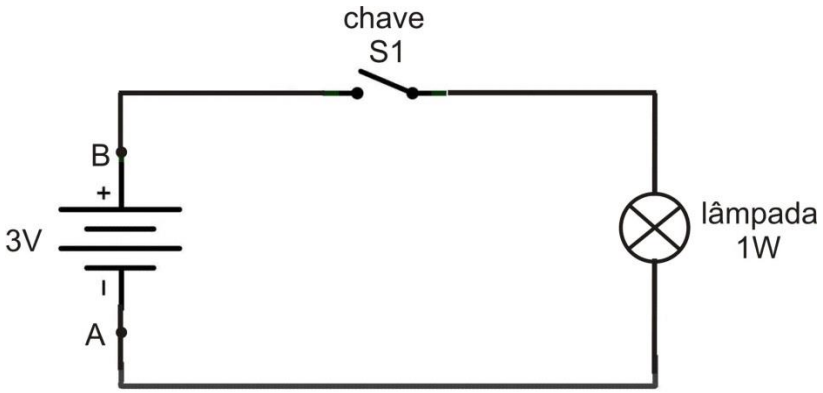


Figura 1.1 – Circuito elétrico esquemático que estabelece a base conceitual da atividade. No circuito identificamos uma fonte de tensão elétrica (pilhas) que irá produzir uma diferença de potencial elétrico entre os pontos A e B, um circuito de fios condutores que se fecha na chave S1 e uma carga passiva (lâmpada) que representa a resistência à passagem das cargas elétricas (corrente elétrica).

Na sequência os alunos são convidados a montar, junto com o professor, o circuito físico utilizando um *protoboard* e os componentes elétricos disponibilizados pelo professor. A Figura 1.2 é uma ilustração do circuito físico.

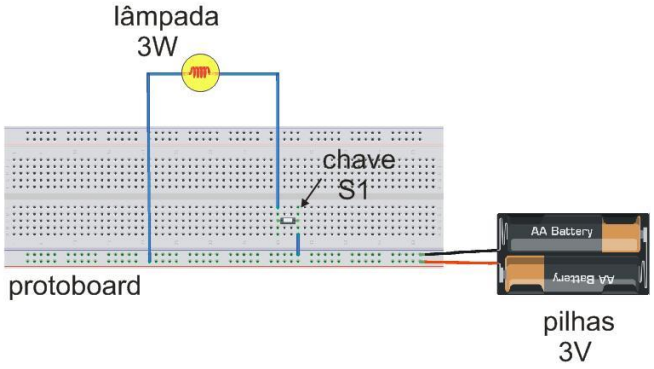


Figura 1.2 - O circuito físico do esquema da Figura 1.1. Este desenho foi feito com auxílio do programa Fritzing, de acesso livre (www.fritzing.org).

Ação: fechamos a chave e acendemos a lâmpada.

Comentário: os alunos são estimulados a pensar no que está acontecendo identificando o papel de cada elemento do circuito. Ao fechar a chave permitimos que as cargas elétricas livres (elétrons) experimentem uma força que as empurram pelo circuito agora fechado, dando origem a uma corrente elétrica. Essas cargas ganham energia pela ação dessa força e, por outro lado, perdem energia pela colisão com os átomos que compõem os condutos por onde passam. Essas colisões que impedem a livre circulação dos elétrons opõem uma resistência natural à passagem dessas cargas elétricas. Esses três fatores se conjugam na lei de Ohm,

V = R.i

Nas colisões com os elétrons os átomos que constituem o material condutor passam a oscilar mais fortemente. Observamos isso através do aquecimento do circuito. Na lâmpada, temos o segmento do circuito onde há maior resistência à passagem dos elétrons e é ali onde se dá o maior efeito de aquecimento. O aquecimento é tão forte que o filamento emite luz. Com isso verificamos e demonstramos a transformação de parte da energia elétrica em energia térmica (calor), chamado de efeito Joule. O sentido convencional de circulação da corrente é dado assumindo-se que os portadores de carga são positivos ao contrário dos elétrons que são portadores de carga negativa, e assim a corrente circula do terminal positivo para o terminal negativo.

Evento POE - A pilha tem uma polaridade. Se trocarmos a polaridade o que acontece? A lâmpada acende?

1.1.2 - Segunda seção.

Vamos refazer o circuito da Figura 1.1 trocando a lâmpada por um LED (***L**ight **E**mitting **D**iode*, ou diodo emissor de luz). Os LED's são muito utilizados nos utensílios eletroeletrônicos modernos. Nossos alunos conhecem bem seus usos, mas não estão familiarizados com suas características e propriedades. Hoje em dia estamos presenciando uma revolução na tecnologia de iluminação com o uso de LED's de alto poder de iluminação. Sua grande vantagem está no baixíssimo consumo de energia e a alta durabilidade quando comparado com as lâmpadas de filamentos ou mesmo as modernas lâmpadas fluorescentes. Diminuir o custo dessas lâmpadas é atualmente um dos desafios da indústria. Nas épocas festivas, particularmente no Natal, os cordões elétricos com dezenas de LED's, os conhecidos "pisca–pisca", são muito usados para enfeitas de casas e lojas. O LED é um diodo que se caracteriza por conduzir a corrente elétrica apenas em um sentido, ou seja, é um componente elétrico polarizado. Nesse momento, uma analogia com a válvula hidráulica unidirecional pode ser muito útil. A Figura 1.3 ilustra esquematicamente o funcionamento desta válvula.

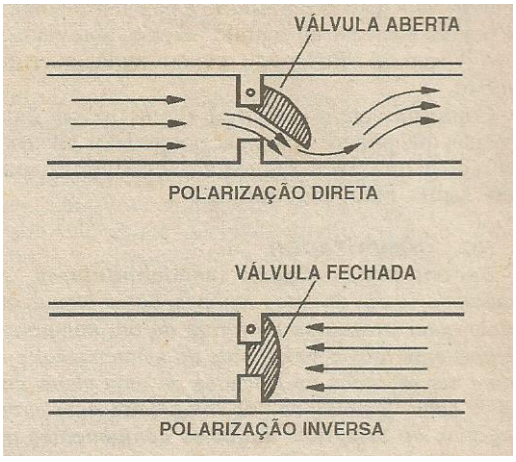


Figura 1.3 – Esquema de funcionamento de uma válvula hidráulica unidirecional (Braga 2001, p. 55).

Com o LED temos um comportamento similar, ilustrado na Figura 1.4.

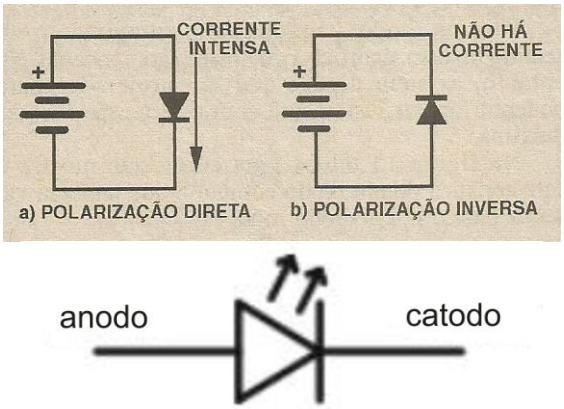


Figura 1.4 – Polarização de um diodo (Braga 2001, p. 55) e símbolo eletrônico de um LED.

A maneira como o LED emite luz é diferente da lâmpada de filamento de tungstênio. A lâmpada, como vimos, emite luz por aquecimento (radiação térmica) e por isso precisa de muita energia para funcionar. Já o LED não emite luz por aquecimento e, portanto, consome muito menos energia. Na Figura 1.5 podemos ver a sua curiosa estrutura interna.

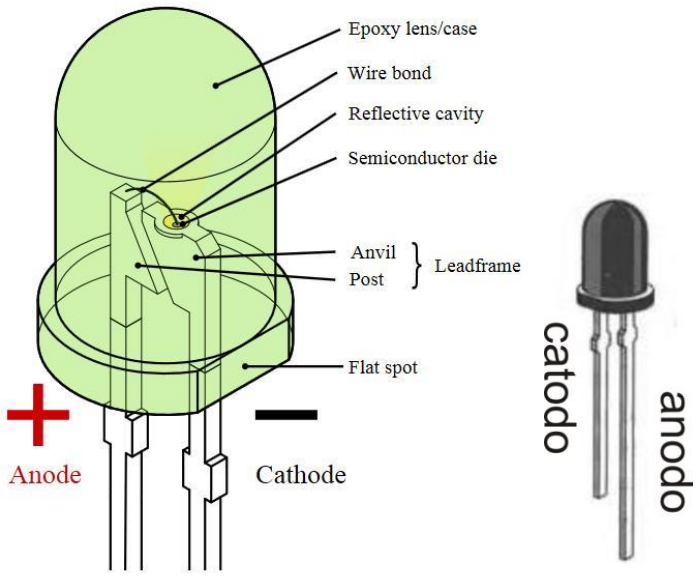


Figura 1.5 – Estrutura típica de um LED de 5 mm (http://en.wikipedia.org/wiki/Light-emitting_diode, em 11/2013). No involucro comercial mais comum a perna maior corresponde ao anodo e a menor ao catodo.

Para substituirmos a lâmpada pelo LED em nosso circuito temos que ter dois cuidados: precisamos respeitar a sua polaridade e temos que considerar sua limitação em corrente. O LED é um componente frágil e possui um limite máximo de corrente que pode suportar. Esse limite de corrente é informado pelo fabricante, mas tipicamente

não deve ser maior do que 50 mA. Quando polarizado diretamente e em funcionamento sua resistência elétrica é muito pequena. Se tentarmos ligar diretamente um LED a uma fonte de tensão ele pode se danificar irreversivelmente. Levando em consideração esses fatos perguntamos a turma, o que devemos fazer para ligar o LED ao nosso circuito? A Figura 1.6 responde essa questão: temos que usar um resistor para limitar a corrente no circuito.

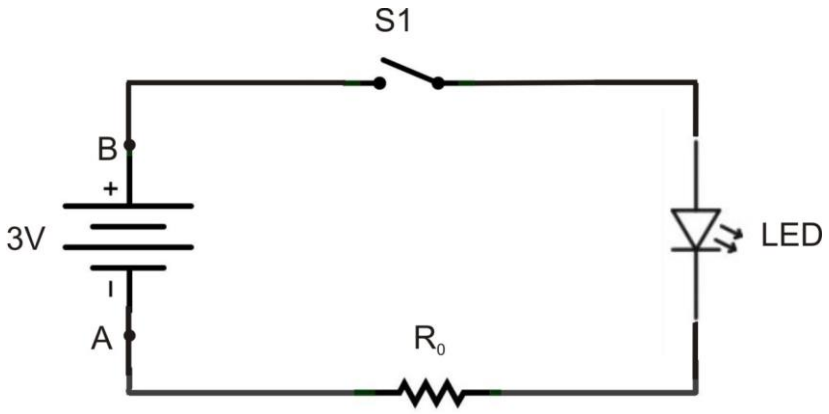


Figura 1.6 – Circuito elétrico esquemático em que substituímos a lâmpada do circuito original por um LED e um resistor limitador de corrente. O LED é um componente polarizado e tem que ser colocado no circuito na orientação correta. Na orientação inversa ao indicado na figura não há condução de corrente e o LED não acende.

Como calcular essa resistência? Aplicamos a lei de Ohm: se temos uma fonte de 3 V e se queremos que a corrente não exceda 50 mA podemos calcular,

$$R = \frac{V}{i} = \frac{3}{0{,}05} = 60\Omega = R_0$$

Como o LED tem uma resistência muito pequena podemos dizer que esse é o valor da resistência R_0 que devemos colocar no circuito. Ainda mais, como esse valor não é comum podemos adotar o valor mais próximo acima deste que é 100 Ω. Temos na Maleta do Professor um bom sortimento de resistores comerciais. Como selecionar o valor correto? Na Figura 1.7 temos uma fotografia de um resistor comercial desses tipicamente usados em circuitos eletrônicos. Muito de nossos alunos já viram esse componente em equipamentos domésticos ou profissionais encontrados no trabalho. Os anéis coloridos indicam o valor da resistência. No Anexo II apresentamos o código de cores e indicamos como usá-lo. Essa convenção deve ser apresentada aos alunos como parte da atividade e como parte de sua formação.

Figura 1.7 – Um típico de resistor de carvão usado em aplicações de eletrônica. As barras coloridas servem para indicar o valor da resistência através de uma convenção apresentada no Anexo II.

Na sequência, os alunos são convidados a montar, junto com o professor, o circuito físico utilizando um *protoboard* e os componentes elétricos disponibilizados pelo professor, exatamente como procedemos na montagem anterior. A Figura 1.8 é uma ilustração do circuito físico.

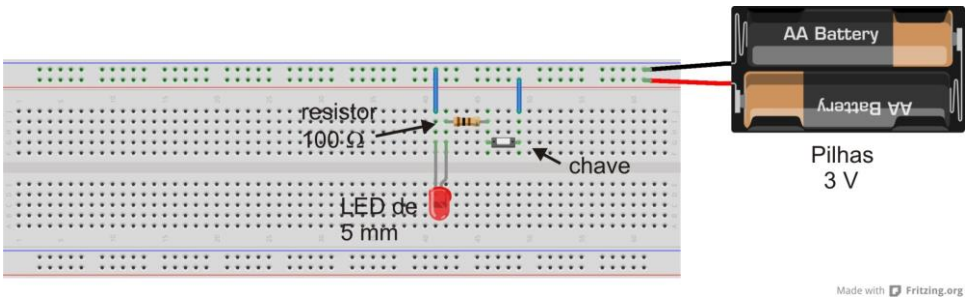


Figura 1.8 - O circuito físico do esquema da Figura 1.6. Este desenho foi feito com auxílio do programa *Fritzing*, de acesso livre (www.fritzing.org).

Ação: fechamos a chave e acendemos o LED. Apertamos a chave (*push-button*) várias vezes e vemos o LED piscar.

Evento POE: (1) se invertermos a polaridade do LED e fecharmos a chave: o LED acende?

(2) Se mantivermos o LED como está e invertemos os terminais da bateria: o LED acende?

Comentário: os alunos devem ser estimulados a pensar no que está acontecendo identificando o papel de cada elemento do circuito. No exemplo em questão o LED, como um diodo semicondutor que emite luz, representa uma carga com características claramente não ôhmicas. Queremos ressaltar suas diferenças com uma lâmpada incandescente comum, mas nessa fase não queremos explicar os princípios de funcionamento do diodo e particularmente do LED. Esse tipo de abordagem prioriza as propriedades operacionais, mas prepara os alunos para outras atividades que se dedique

a discutir os dispositivos semicondutores estimulando a sua curiosidade. Embora os dispositivos semicondutores, tipo diodo e transistores, não façam parte da ementa do sistema EJA, nada impede que possam ser abordados através de uma linguagem adaptada, mais fenomenológica, como aquela dos manuais de eletrônica direcionados para a formação de técnicos em eletrônica voltados para a área de manutenção de equipamentos. É muito interessante, a título de exemplo, que professores que desejem enveredar por estas aplicações extras conheçam o livro *Curso Básico de Eletrônica* do professor e engenheiro Newton C. Braga (ver Referências).

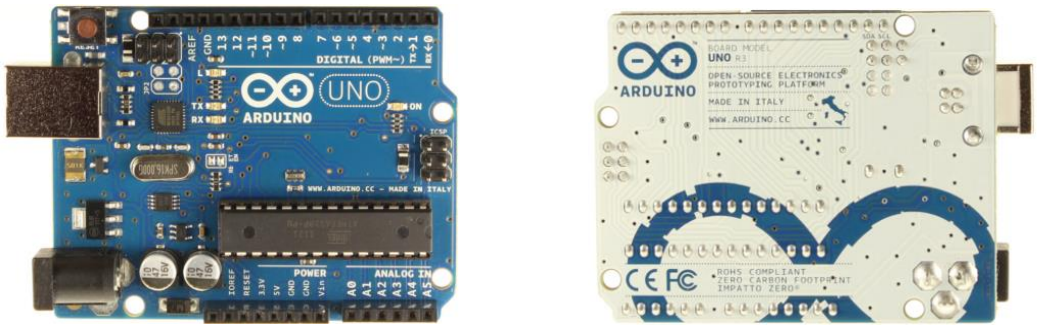
Temos agora um problema prático que queremos resolver: como podemos construir um pisca-pisca com LED's para enfeitar a nossa casa no próximo Natal? Será que conseguimos fazer alguma coisa mais barata do que aquele que encontro na loja do Chinês? Talvez seja possível acender vários LED's ao mesmo tempo, mas como faço para eles piscarem? Será que tenho que ficar apertando a chave? Na próxima seção vamos ver uma forma de piscar o LED.

1.1.3 - Terceira seção

Nesta seção vamos dar um grande salto tecnológico! Vamos usar um microcontrolador para fazer o LED piscar. No momento, podemos entender o microcontrolador como um dispositivo eletrônico que pode fazer o papel de um dedo apertando a nossa chave S1, do circuito anterior, fazendo o LED piscar. Mas, não se engane: um microcontrolador é muito mais do que isso, do que uma simples chave liga-desliga automática.

Um microcontrolador é uma coisa pequena, um dispositivo eletrônico pequeno, daí o prefixo micro, do grego *mikros*, que significa pequeno, ou seja, um dispositivo eletrônico controlador muito pequeno. Quase tudo em matéria de equipamentos eletrônicos que nos cerca hoje em dia tem um microcontrolador comandando. Pense por exemplo na enorme quantidade de funções em sua televisão, no seu micro-ondas, no seu celular, na câmara fotográfica: lá está o microcontrolador fazendo tudo. Podemos dizer que o microcontrolador é um pequeno computador com todas as funcionalidades do microcomputador em um único *chip*. Esse dispositivo eletrônico para funcionar, ou seja, para ser usado num circuito elétrico, precisa ser programado. Precisamos dizer ao microcontrolador o que queremos que ele faça. Um microcontrolador tem memória, onde armazenamos as instruções que queremos que ele execute. Teríamos muito que dizer sobre microcontroladores, e temos muitos estágios para cumprir até podermos dizer que conhecemos bem o assunto. Então vamos por partes, e veremos que é possível fazer uso dessa tecnologia sem sermos especialistas.

Como exemplo de microcontrolador faremos uso de um projeto muito legal: a placa Arduino, mais especificamente, Arduino Uno. É importante dizer isso, pois existem vários modelos de placa Arduino. Na placa Arduino temos um microcontrolador conhecido como ATmega 386 e um canal, ou porta, de comunicação com um microcomputador (PC), uma porta USB. Na Figura 1.9 vemos a Arduino Uno.



(Frente) (Verso)

Figura 1.9 – Placa Arduino Uno. Dimensões aproximadas: 7 cm x 5 cm.

Nas laterais temos uma série de conectores onde podemos ligar várias coisas. São como portas onde podemos entrar com um sinal ou obter um sinal. Algumas dessas portas são digitais (portas enumeradas de 0 a 13) e outra analógicas (A0, A1, ... A5). Mais a frente veremos o que é isso. Além dessas portas, a placa Arduino tem duas fontes de tensão uma de 5,0 V e outra de 3,3V. Isso é muito bom, pois podemos usar essas fontes em lugar de pilhas.

Vamos direto ao ponto: como podemos usar essa placa Arduino, com seu poderoso microcontrolador Atmega328, para fazer o LED piscar?

Vamos usar uma das portas digitais. Vamos tomar por base a porta 5. Como se trata de uma porta digital ela só pode estar em dois estado, ligado ou desligado. A palavra digital tem a ver com isso. Se esta ligada temos 5 V, se está desligada temos 0 V. Não tem meio termo!

Colocamos então um perna do LED, o anodo, na porta 5 e a outra perna, o catodo, no conector da placa que corresponde ao terminal negativo, como se fosse uma pilha. Não usamos aqui o termo “terminal negativo”, falamos *terminal terra* ou simplesmente *terra*.

Agora bem, se mandamos o microcontrolador ligar e desligar a porta 5 faremos o LED piscar. Não podemos esquecer que se aplicarmos 5 V diretamente sobre o nosso LED vamos queimá-lo. Há outra limitação também: essas portas digitais da placa Arduino não foram construídas para fornecer correntes elétricas altas. Elas também suportam no máximo 40 mA. Se não nos lembrarmos disso na hora de usar a placa podemos queimá-la e com isso nosso prejuízo será muito maior do que somente um LED. Vamos ficar bem atentos.

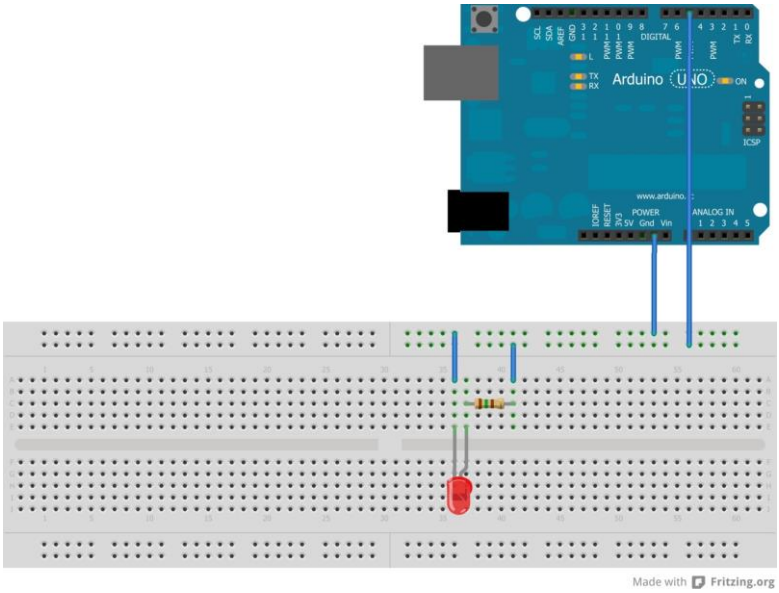


Figura 1.10 - O circuito físico é semelhante da Figura 1.8, onde substituímos a chave tipo *push-button*, o porta pilha e as duas pilhas pela placa Arduino Uno. Este desenho foi feito com auxílio do programa Fritzing, de acesso livre (www.fritzing.org).

Vamos usar um resistor limitador de corrente como fizemos antes. Se temos 5 V, aplicando a lei de Ohm e obtemos,

$$R_0 = \frac{5\,\text{V}}{0{,}04\,\text{A}} = 125\Omega$$

Se tomarmos um valor um pouco acima já estará bom. Podemos usar 150 Ω que é o mais próximo valor comercial que temos em nossa maleta. Quais são as cores desse resistor?

Agora vamos montar o circuito. Veja a Figura 1.10. Com o circuito montado como o fazemos funcionar? Bem, agora temos que instruir o nosso microcontrolador para ligar a porta digital 5 e é ai que começa a segunda parte de nosso trabalho!

Para instruir a Arduino temos que nos comunicar com ela e usar uma linguagem própria. Vamos usar o nosso PC para enviar instruções para a Arduino. No PC vamos usar um programa próprio para isso, uma espécie de interlocutor entre nós e a placa. Esse programa também se chama Arduino e sua função é conhecida genericamente pela sigla IDE, do inglês *Integrated Development Environment* ou simplesmente Ambiente de Desenvolvimento Integrado. Esse programa é integralmente gratuito e podemos fazer uma cópia a partir do site[1] oficial da Arduino. Uma vez aberto em nosso PC sua janela tem o aspecto mostrado na Figura 1.11.

[1] www.arduino.cc

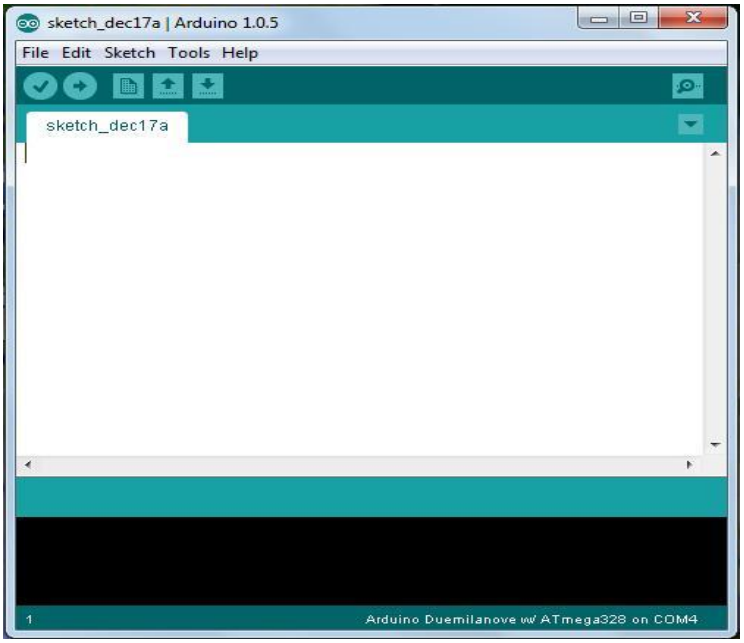


Figura 1.11 – Janela de acesso da IDE Arduino, versão 1.0.5.

Vamos escrever as nossas instruções no espaço em branco reservado na janela. Vamos escrever essas instruções usando uma regra, uma linguagem própria que é derivada da linguagem C. Essas instruções quando completas formam o que no jargão técnico dos usuários ou aplicadores da Arduino é conhecido como um *sketch*, em português dizemos esquete, um esquema simplificado.

Para piscarmos o LED ligado na porta digital 5 vamos então usar o nosso primeiro *sketch*:

```
//Atividade 1a: LED piscante

int ledPin = 10;

void setup() {pinMode(ledPin,
OUTPUT);}

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

Antes de entender o significado dessas instruções, vamos escrever essas instruções na IDE tal como está.

Ação: o professor digita o esquete na frente dos alunos.

Por fim, vamos ver na tela o esquete pronto, como indicado na Figura 1.12.

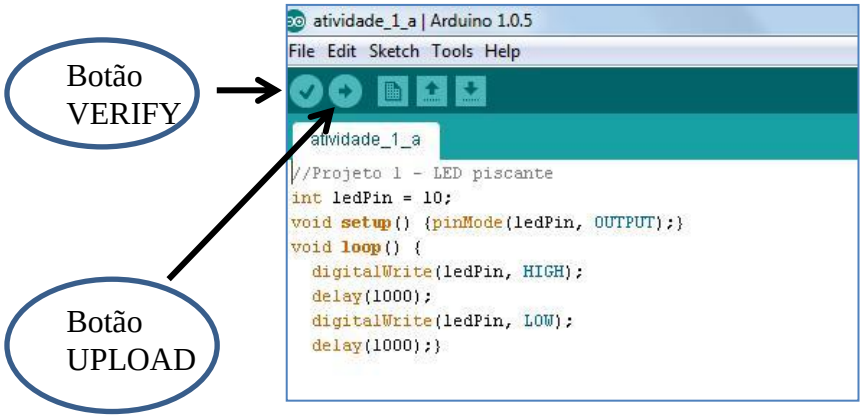


Figura 1.12 - Janela de acesso da IDE Arduino, versão 1.0.5, com o esquete (*sketch*) digitado (***atividade_1_a.ino***).

Para verificar se fizemos tudo certo, apertamos o botão ***VERIFY*** (ver Figura 1.12). O nome já indica a sua função. Com isso, o programa faz uma tradução do esquete em uma linguagem que nossa Arduino entenda. Se tudo estiver certo nenhum alerta de erros será emitido pela IDE. Isso é muito importante porque a linguagem usada é muito exigente. Por exemplo, a regra exige que toda instrução seja finalizada com um sinal de ponto e vírgula ";". Se esquecermos de colocar o ponto e vírgula no final de uma instrução receberemos uma mensagem de erro. O botão *VERIFY* é pois muito útil.

Ação: o professor clica no botão *VERIFY*.

Na sequência, salvamos o esquete num diretório apropriado e ligamos a Arduino a uma porta USB do PC através de um cabo próprio. Com a Arduino ligada ao PC vamos carregar (*upload*) o esquete: clicamos no botão *UPLOAD* (ver Figura 1.12). Antes porém, deligue o cabo de conexão ao pino digital 5. Como regra, faça o *upload* do esquete com os cabos de força desligados.

Ação 1: o professor clica no botão *UPLOAD*.

Ação 2: terminado o *upload*, o professor liga o cabo ao pino 5.

Vemos o LED piscando com uma frequência de 1/2 Hz.

Vamos agora procurar entender as instruções que compõem o nosso primeiro *sketch*. São as primeiras instruções e regras que vamos aprender.

//Projeto 1 - LED piscante

- Essa instrução começa com os símbolos "//". Nesse caso, tudo o que é escrito na linha após o símbolo é tomado como um comentário, não implicando em maiores

consequências. Colocar comentários no *sketch* é muito útil para nos lembrarmos depois o que queríamos fazer com essas ou aquelas instruções.

int ledPin = 5;

- Essa instrução define uma variável. Em Matemática, costumamos definir uma variável usando uma única letra como, por exemplo, **x** ou **y**. Aqui podemos dar nomes com várias letras. Chamamos a nossa variável de **ledPin** e o termo **int** é usado para designar a variável **ledPin** como um número inteiro, mais especificamente um número inteiro que varia de -32.768 a 32.767. Na Arduino temos diferentes tipos de variáveis que iremos conhecer na medida em que vamos avançando em nosso curso. Essa instrução é importante, pois diz para o nosso microcontrolador quanta memória ele tem que reservar para guardar o valor da variável. Se tivesse que guardar o número, por exemplo, 75.746 a Arduino precisaria de mais memória do que o número 23.448. O primeiro número não poderia ser considerado uma variável **int**.

Estamos atribuindo a variável **ledPin** o valor **5**. Sempre que possível escolhemos o nome de uma variável segundo a sua utilidade. Nesse caso ele indica o pino digital que escolhemos para ligar o LED, ou seja, **ledPin** = "pino do LED". Observe que a instrução termina, como já dissemos acima, com o ponto e virgula.

void setup() {pinMode(ledPin, OUTPUT);}

- Todo esquete tem duas coisas que estão invariavelmente presentes: as funções **setup()** e **loop()**. Não há esquete sem essas duas funções. A função **setup()** vem sempre antes da função **loop()** logo no início do esquete. Na função **setup()** não há nada entre os parêntesis e iniciamos com a palavra **void**, do inglês vazio. Esse **void** significa que a função **setup()** não é numérica. Podemos fazer uma analogia com uma função no senso matemático como, por exemplo, quando escrevemos,

$$F(x) = 2.x + 1,$$

Se atribuímos a variável x o valor 3, a função **F(x)** retorna o valor,

$$F(3) = 2.3 + 1 = 7.$$

No caso da função **setup()** não passamos nenhum valor, por isso os parêntesis estão vazios e a função não retorna, por sua vez, nenhum valor numérico por isso usamos o termos **void**. Mais tarde vamos aprender a usar outros tipos de função, mas

aqui já podemos adiantar uma propriedade notável de nosso microcontrolador: se necessário, ele pode fazer vários cálculos, coisas até bem complicadas! Por isso dizemos que o microcontrolador é também um microprocessador.

Toda função é definida entre os colchetes "{}". No caso da função **setup** que estamos definindo ela contém apenas instrução:

pinMode(ledPin,OUTPUT);

Essa instrução é usada para indicar se um pino digital (**pinMode** = modo do pino) é de entrada de um sinal (**INPUT**) ou de saída de um sinal (**OUTPUT**). Isso é um ponto importante: a Arduino não sabe se os seus pinos digitais serão usados para receber dados de fora ou se serão usados para enviar dados para fora. Nós temos que dizer isso a ela! No nosso caso, nós queremos ligar ou desligar o pino **5**, isso significa que vamos mandar (**OUTPUT**) uma instrução (liga / desliga) para esse pino. Dizemos também, que vamos escrever nessa porta e não ler o que é enviado para a porta. A maneira com escrevemos uma instrução **pinMode** é sempre assim:

pinMode(número da porta, OUTPUT ou INPUT);

No nosso caso o número da porta é dado pela variável **ledPin**, já no inicio do esquete. A função **setup()** tem portanto essa utilidade: é usada para definir o modo como as coisas vão funcionar na placa Arduino, é por isso que ela vem logo no início do esquete e é por isso que nada funciona na Arduino sem uma função **setup()**.

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

- Nesta instrução definimos a função **loop()**. Essa função tem as mesmas características da função **setup()**, ou seja, os parêntesis estão vazios e começamos com o termo **void** e isso pelas mesmas razões. Essa função tem por finalidade fazer uma atividade repetitiva (esse é o sentido da palavra inglesa *loop*...), isto é, sem fim. É aqui que vamos fazer o LED piscar. A primeira instrução na função **loop()** é,

digitalWrite (ledPin, HIGH);

Essa instrução escreve (*write*) na porta digital **ledPin** (porta 5) o estado **HIGH** (alto ou ligado). Em outra palavras, a porta 5 é ligada em 5 V. Nesse momento o LED é ligado, é aceso. Em seguida temos a instrução,

delay (1000);

A instrução **delay()** instrui a Arduino a esperar sem nada fazer durante um intervalo de tempo indicado dentro dos parêntesis. A unidade de tempo da Arduino é o milissegundo. Se escrevermos 1000 isso significa 1000 milissegundos, ou seja, 1000 ms = 1s. A próxima instrução é similar à primeira,

digitalWrite (ledPin, LOW);

Essa instrução escreve na porta digital **ledPin** (porta 5) o estado **LOW** (baixo ou desligado). Em outra palavras, a porta 5 é colocada em 0 V. Com isso desligamos o LED.

A maneira com escrevemos uma instrução **digitalWrite** é sempre assim:

digitalWrite (número da porta, HIGH ou LOW);

Em seguida repetimos a instrução,

delay (1000);

e mantemos assim o LED desligado por 1s. Terminado esse prazo começamos tudo novamente, retornando para a primeira instrução do *loop*, ligando o LED. E assim fazemos o LED piscar, a cada 1s. A única forma de parar será desligando a Arduino.

Podemos visualizar a operação completa do loop através de um diagrama circular, contendo as instruções (Figura 1.13).

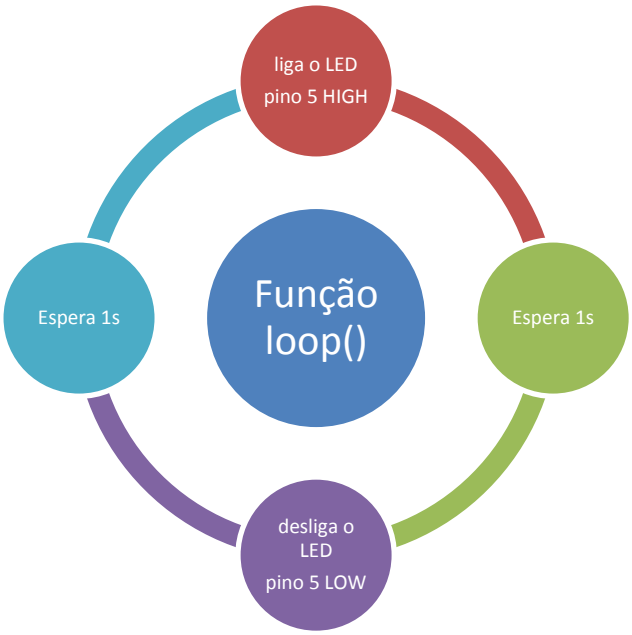


Figura 1.13 – Ilustração da operação de funcionamento da função **loop()** (ver texto).

1.2 – Conclusão da Atividade I

Como conclusão da terceira seção, planejamos um conjunto de exercícios de fixação e problemas de aprofundamento. A inserção dessas sugestões no plano de aula desta Atividade I deve ficar a cargo de cada professor.

Exercício I.1 (fixação) – (1) Como devemos modificar o esquete ***Atividade_1_a.pde*** para fazer com que o LED pisque com uma frequência de 0,10 Hz. Queremos que o LED fique três vezes mais tempo ligado do que desligado. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

Exercício I.2 (fixação) – Altere o esquete ***Atividade_1_a.pde*** substituindo a instrução ***delay (1000)*** pela instrução ***delay (random(100))***. A instrução ***randon(max)*** gera um número aleatório entre zero e o número entre parêntesis (excluído). Assim random(100) gera um número aleatório entre 0 e 99. Faça o *upload* desse novo esquete é comente o resultado observado.

Exercício I.3 (fixação) – (1) Como devemos modificar o esquete ***Atividade_1_a.pde*** para fazer piscar três LED's das cores amarelo, vermelho e verde em sequência e tempo de 1s. Usando os mesmos símbolos empregados na Figura 1.6 faça um desenho do circuito elétrico usado para acender o três LED's. (2) Monte o circuito com o material disponibilizado pelo professor e comprove o seu funcionamento.

Problema I.1 (aprofundamento) – Para situações de emergência no mar, barcos salva vidas recebem uma lâmpada de alerta que pisca de forma intermitente um sinal de SOS em código Morse. O código Morse foi criado por Samuel Morse, inventor do telégrafo, em 1835 para servir de base para comunicação à distância[2]. Através do telégrafo é possível mandar um conjunto de sinais elétricos curtos e longos que podem ser ouvidos. Morse estabeleceu um código que faz corresponder a cada letra, algarismos e símbolos de pontuação textual, um conjunto de pontos (sinal curto) e traços (sinal longo) e espaços. O sinal **SOS** é usado para comunicar uma emergência. No código Morse a letra **S** é representada por três sinais curtos (pontos) e o **O** por três sinais longos (traços) e assim o sinal de SOS é "• • • – – – • • •". Essa sequência pode ser comunicada com luz e é exatamente isso que faz a lâmpada do bote salva vidas. (1) Escreva um esquete para um *LED de Alerta* que emite o sinal de SOS. Para dar portabilidade ao seu projeto do LED de Alerta, alimente a Arduino com uma bateria de 9 V. O professor fornecerá as conexões necessárias.

[2] http://pt.wikipedia.org/wiki/C%C3%B3digo_Morse Acesso em 05/01/2014.

Com a conclusão desta atividade apresentamos um conjunto de elementos da linguagem de programação da Arduino. Em resumo, colocamos na Tabela 1.2 as instruções usadas. É importante não perder de vista a sequência de apresentação dos comandos e da sintaxe. Essa apresentação deve seguir uma ordem lógica e se desenvolver dentro de um esquema em espiral onde nas próximas atividades aplicamos os mesmo comandos introduzindo novos detalhes.

Tabela 1.2 – Comando da linguagem de programação praticada na Atividade I

1	//	Esta tabela possui os comandos usados na atividade I, porém o que foi explicado nessa atividade é apenas parte do que cada comando pode fazer. Para aprofundar mais o seu conhecimento sobre a estrutura completa de cada comando e obter maiores informações a respeito, o professor pode acessar o site da página oficial da arduino em http://arduino.cc/en/Reference/HomePage
2	int	
3	setup()	
4	loop()	
5	void	
6	pinMode()	
7	digitalWrite()	
8	delay()	
9	for	
10	random()	

Problema extra, um desafio para ir mais longe: As cores primárias são: azul, verde e vermelho. Através da combinação, ou seja, da intensidade dessas três cores podemos produzir diversas outras cores (ver Figura 1.14). As diferentes cores que observamos com a visão humana na natureza, podem ser decompostas em suas componentes, vermelha, azul e verde. Um exemplo é a luz branca que é uma composição de diferentes cores, quando decomposta obtemos o espectro de cores do arco-íris. Uma ***mood lamp*** é uma lâmpada formada de três LEDs, um vermelho, um verde e um azul, assim como o seu monitor de computador ou da televisão é formado de pequenos pontos vermelhos, verdes e azuis (RGB), onde uma cor é gerada apenas pelo ajuste do brilho de cada LED. **RGB** é a abreviatura do sistema de cores aditivas formado por vermelho (*Red*), verde (*Green*) e azul (*Blue*). Suas iniciais na língua inglesa forma a palavra RGB. O LED RGB é um componente único que vem com três LEDs dentro dele: vermelho, verde e azul, sendo bem mais caro que o LED comum, possui quatro pernas, um catodo comum e três anodos, um para o verde, um para o azul

e um para o vermelho. É possível então acender esses três LEDs no corpo de um único LED. Ele pode ficar com a cor que você desejar, dependendo apenas da intensidade de azul, verde e vermelho, como se fosse um pixel. Um pixel da televisão, ou do monitor de computador, é um ponto muito pequeno, ou seja, miniaturizado, composto de três LEDs, um vermelho, um azul e um verde (ver figura 1.15). Você pode controlar a ***mood lamp*** através de comandos do seu computador para a placa Arduino utilizando o *Serial Monitor* (comunicação serial), do IDE do seu Arduino. Monte um circuito com esse tipo de lâmpada e através de uma comunicação serial produza cores diversas. A ideia é utilizar os três LEDs e produzir cores como se fosse um pixel, assim como é feito no monitor da televisão ou do computador (SUGESTÃO: consulte o livro Arduino Básico de Michael McRoberts).

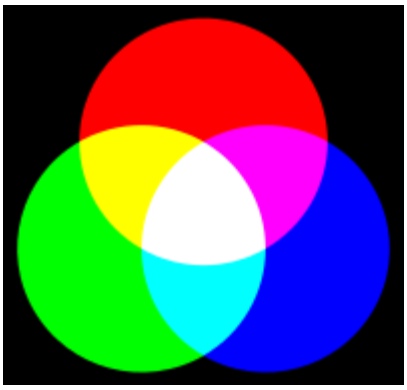
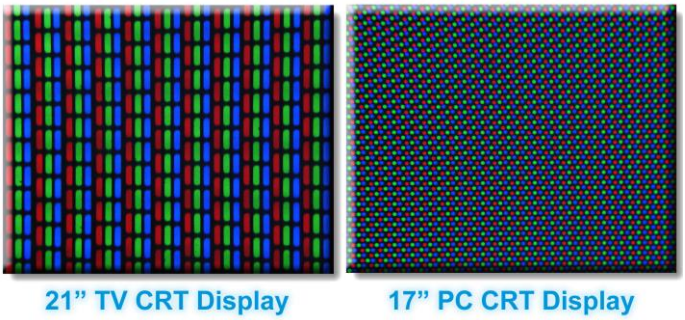


Figura 1.14 – A combinação das cores primárias formando novas cores.



(Televisão de 21”) (monitor de computador 17”)

Figura 1.15 – Imagem ampliada para a visualização dos pixels de uma televisão e de um monitor de computador.

1.3 - Solução dos Exercícios da Atividade I

Exercício I.1 (fixação) – (1) Como devemos modificar o esquete ***Atividade_1_a.pde*** para fazer com que o LED pisque com uma frequência de 0,10 Hz. Queremos que o LED fique três vezes mais tempo ligado do que desligado. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

(1) Se o tempo desligado é t então o tempo ligado tem que ser 3t. O ciclo completo é pois t + 3t = 4t. Se a frequência (f) é 0,10 Hz o período é T = 1/f = 1/0,10 = 10 s. Assim 4t = 10 e portanto t = 2,5 s. O LED deve ficar 2,5 s apagado e 7,5 s aceso. Para executar essa tarefa temos que alterar o esquete original ***Atividade_1_a.pde*** alterando o valor das duas instruções ***delay***, como se segue:

Esquete original		Esquete modificado (alterações em negrito)

//Atividade 1a: LED piscante
 	
int ledPin = 10;
 	
void setup() {pinMode(ledPin, OUTPUT);}
 	
void loop() {
digitalWrite (ledPin, HIGH);
delay (1000);
digitalWrite (ledPin, LOW);
delay (1000);}
		//Atividade **1b**: LED piscante
		
		int ledPin = 10;
		
		void setup() {pinMode(ledPin, OUTPUT);}
		
		void loop() {
		digitalWrite (ledPin, HIGH);
		delay (7500);
		digitalWrite (ledPin, LOW);
		delay (2500);}

(2) Não é necessário modificar o circuito. A modificação é apenas quanto ao controle do acionamento do LED e, portanto um problema de programação, de software.

Nota: procure tirar proveito dessa montagem. Peça a seus alunos que tentem cronometrar o tempo entre duas piscadas sucessivas. Eles podem usar o cronômetro existente nos celulares. Como esse tempo é pequeno procure cronometrar um número maior de piscadas e ao final divida o tempo pelo número de piscadas. Compare essas

medidas com os valores programados na placa Arduino. Com essa atividade procure exercitar conceitos de período e frequência.

Exercício I.2 (fixação) – Altere o esquete ***Atividade_1_a.pde*** substituindo a instrução ***delay (1000)*** pela instrução ***delay (random(100))***. A instrução ***randon(max)*** gera um número aleatório entre zero e o número entre parêntesis (excluído). Assim ***random(100)*** gera um número aleatório entre 0 e 99. Faça o *upload* desse novo esquete é comente o resultado observado.

Neste caso a modificação sugerida é apenas nas instruções ***delay***. O novo esquete passa a ser escrito como:

Esquete original		Esquete modificado (alterações em negrito)

//Atividade 1a: LED piscante
 	
int ledPin = 10;
 	
void setup() {pinMode(ledPin, OUTPUT);}
 	
void loop() {
digitalWrite (ledPin, HIGH);
delay (1000);
digitalWrite (ledPin, LOW);
delay (1000);}
		//Atividade **1c**: LED piscante
		
		int ledPin = 10;
		
		void setup() {pinMode(ledPin, OUTPUT);}
		
		void loop() {
		digitalWrite (ledPin, HIGH);
		delay (random(100));
		digitalWrite (ledPin, LOW);
		delay (random(100));}

Neste caso as piscadas ficarão muito mais rápidas, pois o tempo máximo em que o LED ficará ligado ou desligado é sempre menor ou igual 0,99 s. Por outro lado as piscadas não ficarão regulares, periódicas. Muitos efeitos luminosos especiais podem ser obtidos com a instrução ***random()***.

Exercício I.3 (fixação) – (1) Como devemos modificar o esquete ***Atividade_1_a.pde*** para fazer piscar três LED's das cores amarelo, vermelho e verde em sequência e tempo de 1s. Usando os mesmos símbolos empregados na Figura 1.6, faça

um desenho do circuito elétrico usado para acender o três LED's. (2) Monte o circuito com o material disponibilizado pelo professor e comprove o seu funcionamento.

A forma mais simples e direta para acionar os três LED's é usar três portas digitais. Vamos começar observado o circuito de acionamento (Figura 1.16). As chaves representam as portas digitais D1, D2 e D3 e sua função de ligar e desligar a fonte de tensão. O acionamento dos LED's é sequencial e não simultâneo o que não compromete a Arduino com alguma sobrecarga de corrente. Na Figura 1.17 mostramos uma representação do circuito físico feita com o programa **Fritzing**. Esse programa é gratuito e pode ser obtido a partir do site[3] oficial. É muito fácil de ser usado e muito útil ao professor de Física que queira usar a Arduino com seus alunos. Todos os circuitos envolvendo a Arduino, componentes diversos e sensores podem ser representados com essa ferramenta.

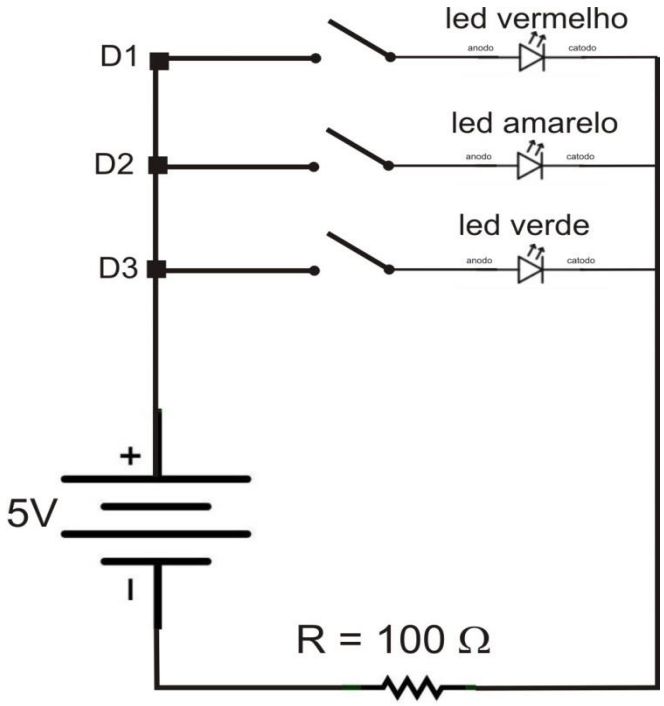


Figura 1.16 – Esquema do circuito elétrico para o acionamento de três LED's coloridos do Exercício I.3. As chaves representam os pinos digitais da Arduino. Usamos uma porta para cada LED. O acionamento é sempre sequencial e não simultâneo.

[3] www.fritzing.org

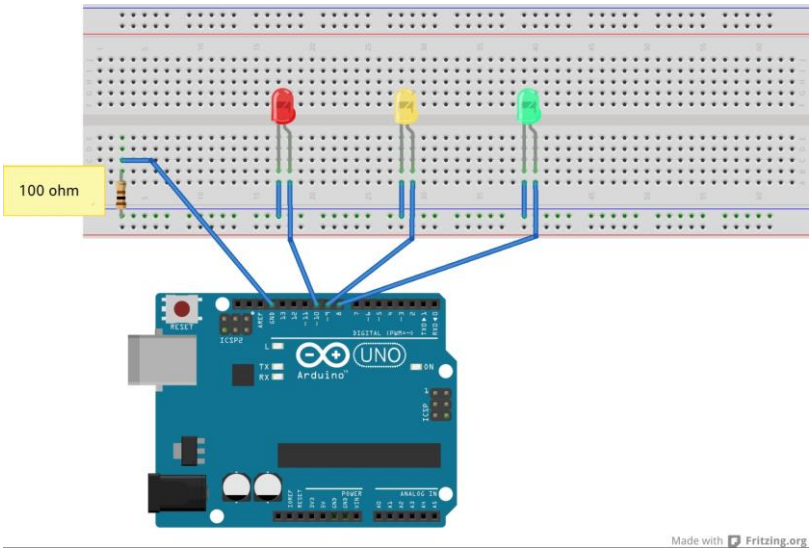


Figura 1.17 – Representação do circuito físico da Figura 1.16. Desenho feito com o auxílio do programa gratuito Fritzing.

O esquete é tão somente uma repetição do esquete original para um único LED e pode ser escrito como,

```
//Atividade Id - Acionamento sequencial de três LED's
//Exercício I.3

int ledvermPin = 10;
int ledamarPin = 9;
int ledverdPin = 8;

void setup() {
pinMode(ledvermPin, OUTPUT);
pinMode(ledamarPin, OUTPUT);
pinMode(ledverdPin, OUTPUT);}

void loop() {
  digitalWrite(ledvermPin, HIGH);
  delay(1000);
  digitalWrite(ledvermPin, LOW);
  delay(100);

  digitalWrite(ledamarPin, HIGH);
  delay(1000);
  digitalWrite(ledamarPin, LOW);
  delay(100);

  digitalWrite(ledverdPin, HIGH);
  delay(1000);
  digitalWrite(ledverdPin, LOW);

delay(100);}
```

Problema I.1 (aprofundamento) – Para situações de emergência no mar, barcos salva vidas recebem uma lâmpada de alerta que pisca de forma intermitente um sinal de SOS em código Morse. O código Morse foi criado por Samuel Morse, inventor do telégrafo, em 1835 para servir de base para comunicação à distância. Através do telégrafo é possível mandar um conjunto de sinais elétricos curtos e longos que podem ser ouvidos. Morse estabeleceu um código que faz corresponder a cada letra, algarismos e símbolos de pontuação textual, um conjunto de pontos (sinal curto) e traços (sinal longo) e espaços. O sinal **SOS** é usado para comunicar uma emergência. No código Morse a letra **S** é representada por três sinais curtos (pontos) e o **O** por três sinais longos (traços) e assim o sinal de SOS é "• • • – – – • • •". Essa sequência pode ser comunicada com luz e é exatamente isso que faz a lâmpada do bote salva vidas. (1) Escreva um esquete para um *LED de Alerta* que emite o sinal de SOS. Para dar portabilidade ao seu projeto do LED de Alerta, alimente a Arduino com uma bateria de 9 V. O professor fornecerá as conexões necessárias.

Neste problema não temos que fazer alterações na montagem usada na para piscar o LED. Mantemos exatamente o circuito e o que temos que fazer é programar a Arduino para controlar a sequência exata na qual o LED deve piscar o SOS.

```
//Atividade Ie  - LED pisca o SOS em código Morse
//Problema I.1

int ledPin=10;

void setup(){                //executa uma vez
quando o sketch inicia
 pinMode(ledPin, OUTPUT);          //define o pino
10 como saída
}

void loop(){          //executa repetidas vezes

  for (int x=0; x<3;x++) {          // 3 pontos
    digitalWrite(ledPin,HIGH);   //acende o LED
    delay(150);                  //espera 150ms
    digitalWrite(ledPin,LOW);    //apaga o Led
    delay(100);                  //espera 100ms
  }

   delay(100);          //espera 100ms para marcar
o intervalo entre as letras


   for (int x=0; x<3;x++) {           // 3 traços
         digitalWrite(ledPin,HIGH);   //acende o
LED
    delay(400);                       //espera
400ms
    digitalWrite(ledPin,LOW);         //apaga o Led
    delay(100);                       //espera
100ms
  }

    delay(100);  //espera 100ms para marcar o
intervalo entre as letras

  for (int x=0; x<3;x++) {           // 3 pontos
novamente
    digitalWrite(ledPin,HIGH);     //acende o LED
    delay(150);                    //espera 150ms
    digitalWrite(ledPin,LOW);      //apaga o Led
    delay(100);                    //espera 100ms
  }

delay(5000);    // espera 5 segundos antes de
repetir o sinal de SOS

}
```

O problema é então basicamente de software. No quadro, damos uma solução para este esquete. Neste esquete usamos uma instrução nova e muito poderosa: para repetir as três piscadas do LED usamos uma instrução ***for*** ou um laço ***for***. A instrução ***for*** constitui um laço no sentido de que uma tarefa é repetida um número definido de vezes muito parecido com o que temos com um laço ***loop***. No caso da instrução ***for*** o número de repetições é previamente definido. Vejamos a sua estrutura através do exemplo que estamos estudando,

```
for (int x=0; x<3;x++){....}
```

- Quando iniciamos o laço ***for*** começamos com o comando **int x = 0;** em que definimos a variável **x** como inteira (**int**) e atribuímos a ela o valor **0.** Essa instrução é executada apenas uma vez quando iniciamos o laço ***for***.
- Em seguida testamos se a variável **x** é menor do que **3** (x < 3). No caso da primeira passagem x é igual a zero e, portanto, menor do que **3**. Neste caso passamos para a etapa seguinte. A instrução **x++** é uma forma simplificada da operação *x(novo valor) = x(valor anterior) + 1*, ou seja, redefinimos o valor de **x** como sendo o valor anterior acrescido de **1**.
- Em seguida, executamos todas as instruções que estiverem entre os colchetes **{...}**. No nosso caso vamos acender e apagar o LED numa cadência adequada para simular os pontos e os traços. Para os pontos deixamos o LED aceso por 150 ms e para os traços 400 ms. Quando essa sequência é concluída retornamos para o inicio do laço ***for***. Como dissemos, a primeira instrução **int x = 0;** não é mais repetida e vamos direto para o teste: **x < 3;** como x = 1 essa condição é satisfeita e passamos para a seção seguinte onde incrementamos o valor de **x** mais uma vez e que passa agora a valer **2**. Em seguida as instruções entre colchetes são novamente executadas e o LED pisca mais uma vez. A sequência só será interrompida quando x = 3. Nesse caso a condição **x < 3;** não é mais satisfeita e assim o laço é interrompido.

Vemos assim a diferença entre o ***for*** e o ***loop***: enquanto o ***loop*** é repetido indefinidamente o ***for*** é repetido um número pré-selecionado de vezes.

Atividade II – Fazendo uma Lâmpada Piscar com a Arduino.

Metas da atividade: nesta atividade queremos mostrar alguns usos importantes da lei de Ampere. Usamos como relação com a tecnologia o relé, ou chave eletromagnética. No dicionário Houaiss, lemos que relé é um *aparelho graças ao qual uma energia relativamente pequena controla uma energia maior* e é nesse sentido que o introduzimos aqui. Na Atividade I usamos a Arduino para acionar um LED e vimos ali uma clara limitação: não podemos usar a placa Arduino diretamente para acionar circuitos de potência. Como usamos a microeletrônica para controlar os grandes equipamentos a nossa volta? Desenvolveremos competências e habilidades propostas no currículo mínimo como a compreensão da relação do avanço do eletromagnetismo e dos aparelhos eletrônicos (Nova EJA e Ensino Médio Noturno) e também interpretar e propor modelos explicativos para fenômenos eletromagnéticos (Ensino Médio Noturno).

Objetivos da atividade: construir o circuito elétrico físico a partir da sua representação esquemática; reconhecer os componentes elétricos comerciais (resistor, pilha, chave, lâmpada, relé, LED e fios) e suas propriedades elétricas; aplicar a lei de Ohm; introdução à placa Arduino.

Conceitos trabalhados: Lei de Ampere, diferença de potencial elétrico (DDP ou tensão), corrente e resistência elétrica (V, I e R); lei de Ohm.

Pré-requisito: Atividade I. Os alunos precisam ter participado da parte expositiva onde os conceitos pertinentes foram apresentados e discutidos.

Material utilizado: para a realização desta atividade, dividida em três seções, serão necessários os itens relacionados na Tabela 2.1.

Tabela 2.1 – Relação de materiais para a Atividade II

	Item:	Quant.	Observação
1	*Protoboard*	01	
2	Eletroímã	01	O eletroímã pode ser artesanal. Ver Anexo I para sugestões de montagens;
3	Bússola portátil	01	Modelo simples que pode ser comprada em papelarias;
4	Ímã permanente	01	
5	Lâmpada incandescente (127V / 15W)	01	
6	Bocal para lâmpada	01	
7	Fio paralelo com chave interruptora e *plug* macho.	01	Podem ser comprados já prontos como reparo para abajur;
8	Pilha recarregável (AA)	02	Para acionamento do eletroímã;
9	Pilha comum (AA)	02	Para acionamento do motor DC;
10	Porta pilha para duas pilhas pequenas (AA)	02	
11	Chave tipo *push-button*	01	
12	Arduino Uno (REV3)	01	
13	Módulo relé, 5 VDC / 10A	01	
14	Cabos para conexão (Jumpers M/M)	04	Usados na conexão do *protoboard* e portas da Arduino.

2.1 – Descrição da Atividade II.

2.1.1 - Primeira seção.

Iniciamos apresentando um eletroímã e o circuito elétrico de acionamento. Nesta montagem temos a base para discutir o fenômeno da produção de um campo magnético por uma distribuição de corrente elétrica no espaço. Como sabemos, o fenômeno foi originalmente descoberto em 1819, pelo físico dinamarquês Hans Christian Oersted. O

físico francês André Marie Ampère (1775 -1836), pouco depois da descoberta de Oersted, estabeleceu um relação quantitativa entre a intensidade do magnetismo (campo magnético) e a corrente elétrica em um fio, conhecida como lei de Ampere.

O circuito apresentado na Figura 2.1 permite o acionamento do eletroímã usando uma fonte de 3V e uma chave S1 (tipo *push-button*). Nesse desenho usamos uma representação informal para o eletroímã. Não se trata de um símbolo convencional uma vez que eletroímãs não são elementos de circuito eletrônicos comuns.

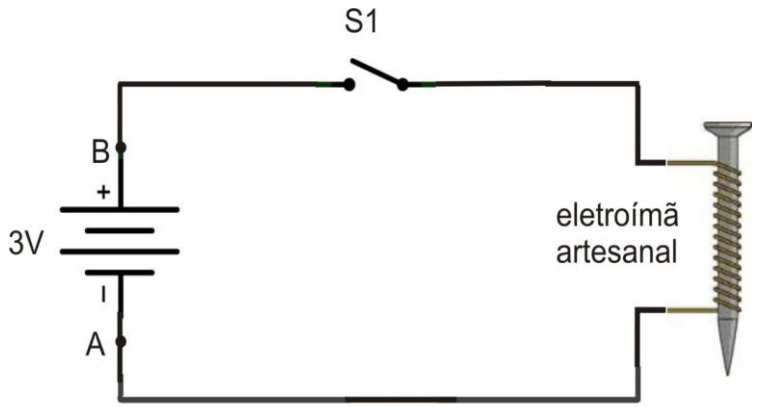


Figura 2.1 - Circuito esquemático para o acionamento do eletroímã.

Na sequência os alunos são convidados a montar, junto com o professor, o circuito físico utilizando um *protoboard*, o eletroímã e os componentes elétricos disponibilizados pelo professor. A Figura 2.2 é uma ilustração do circuito físico.

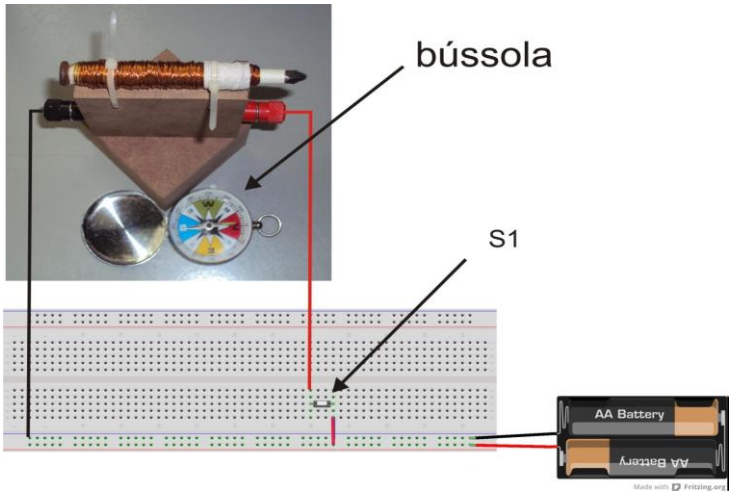


Figura 2.2 - Circuito físico para acionamento do eletroímã. Na figura mostramos uma bússola simples para indicar a presença do campo magnético gerado pelo eletroímã. O eletroímã mostrado foi construído artesanalmente e alguns detalhes são apresentados no Anexo I.

Ação: O professor usa um pequeno imã permanente para mostrar a resposta da bússola. Em seguida afasta o imã e fecha a chave S1 observando a resposta da agulha magnética da bússola.

Comentário: os alunos devem ser estimulados a pensar no que está acontecendo identificando o papel de cada elemento do circuito. Ao fechar a chave S1 permitimos a formação de uma corrente elétrica. Essa corrente percorre o circuito passando pelas espiras da bobina com núcleo de ferro (prego). A intensidade da corrente depende da fonte de tensão e da resistência elétrica do circuito que é neste caso dado pelos fios que formam a bobina. Como vimos esses três fatores se conjugam na lei de Ohm,

$$\mathbf{V = R.i}$$

Na bobina temos aproximadamente 20 m de fio de cobre recobertos com uma capa de verniz para isolamento. O cobre é um bom condutor de eletricidade. Usamos fios de cobre na instalação elétrica de nossas casas. Mesmo 20 m de fio de cobre com aproximadamente 0,5 mm de diâmetro tem uma resistência elétrica pequena. Na bobina da Figura 2.2 temos 4,0 Ω o que dá aproximadamente 0,2 Ω para cada metro de fio. Podemos calcular a corrente na bobina usando a lei de Ohm,

$$I = V/R = 3{,}0\ V / 4{,}0\ \Omega = 0{,}75\ A = 750\ mA$$

Mesmo para pilhas essa corrente é alta. Devemos ter cuidado e sempre acionar o eletroímã por curtos intervalos de tempo para evitar aquecimento e que se esgotem rapidamente. Nesse cálculo não estamos levando em conta a resistência interna das pilhas. Essas resistências, em aplicações usuais, podem ser consideradas pequenas, mas comparativamente no caso do eletroímã, não são tão pequenas. Um cálculo mais exato teria que levar em consideração a resistência interna das pilhas e resultaria numa corrente na bobina menor que os 0,75A. Como sabemos pela lei de Ampére, quanto menor a corrente na bobina menor será o efeito magnético (campo magnético). A pilha comum tamanho AA tem resistências internas maiores que as pilhas maiores, tamanho D. As pilhas alcalinas por sua vez têm resistências internas menores que as convencionais. Você pode usar pilhas grandes (tamanho D) para o acionamento do eletroímã ou, o que é mais recomendável, usar pilhas recarregáveis que possuem as menores resistências internas, gerando correntes máximas. Por outro lado, e não menos importante, são também mais ecológicas.

Vimos que a corrente ao passar pelo fio produz um campo magnético a sua volta. Se tomarmos um fio longo e o enrolamos em volta de um prego intensificamos o efeito magnético nas vizinhanças do prego. A compactação em espiras formando uma

bobina é uma geometria muito eficaz para isso. A presença do prego fabricado com um metal ferromagnético (ferro) atua intensificando o efeito magnético da bobina: é o efeito de magnetização. Como sabemos a magnetização do ferro induzida pelo efeito magnético da bobina não desaparece completamente quando sessamos a corrente na bobina. Esse fenômeno é conhecido como *histerese* e é esse fato que nos permite produzir imãs permanentes. No nosso caso, como as correntes usadas são muito pequenas o efeito da histerese é pequeno e o prego acaba por guardar pouca magnetização residual.

Evento POE – Como vimos, a pilha tem uma polaridade. Se trocarmos a polaridade no circuito do eletroímã o que acontece?

Nossa bobina se comporta como um pequeno imã. Quando ligada, ela possui um polo norte e um polo sul. Por isso, dizemos que nossa bobina é um eletroímã, ou seja, um "imã elétrico". Se invertermos o sentido de circulação da corrente invertemos os polos de nosso eletroímã.

2.1.2 - Segunda seção.

Nesta seção fazemos uma preparação para a próxima. Na seção seguinte realizamos o objetivo final da Atividade II.

Usando o eletroímã anterior acrescentamos contatos metálicos que servem para mostrar o princípio de funcionamento de um relé eletromecânico. Os relés comerciais são encapsulados e nem sempre é possível examinar o seu interior. Essa atividade foi concebida para facilitar a compreensão de como o relé funciona como uma chave abre-fecha comandada eletricamente. Nosso modelo é um relé artesanal, muito fácil de ser construído (ver Figura 2.4) e baseado no mesmo eletroímã usado na seção anterior. Na Figura 2.3 mostramos o circuito elétrico esquemático de acionamento do relé.

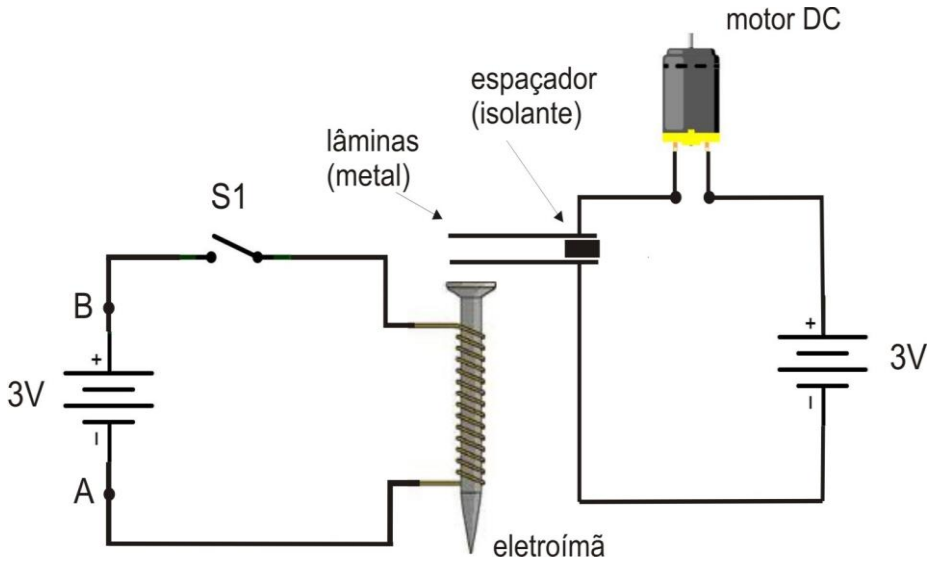


Figura 2.3- Circuito elétrico esquemático para o acionamento do relé.

O relé tem a função de chave num circuito auxiliar alimentado por um fonte independente de 3V e que aciona um pequeno motor DC. Nessa versão estamos usando um motor elétrico porque tem uma forte relação com o tema da atividade. O motor gira graças ao mesmo fenômeno que se manifesta no relé. Motores elétricos são muito importantes no mundo tecnológico que nos cerca e podemos encontrar aqui uma boa oportunidade de expandir a discussão sobre a lei de Ampère. Se o professor achar conveniente não iniciar essa discussão nesta fase ele pode facilmente substituir o motor por uma lâmpada incandescente tal como usamos na Atividade I, Seção I. O objetivo do circuito auxiliar é simplesmente mostrar o funcionamento da chave. É também possível utilizar no lugar do motor um multímetro na função de ohmímetro (medida de

resistência) para verificar continuidade. Neste caso, a vantagem do multímetro é a de que o professor estará praticando com a sua turma a utilização desse importante instrumento elétrico de larga aplicação nas mais diferentes atividades profissionais.

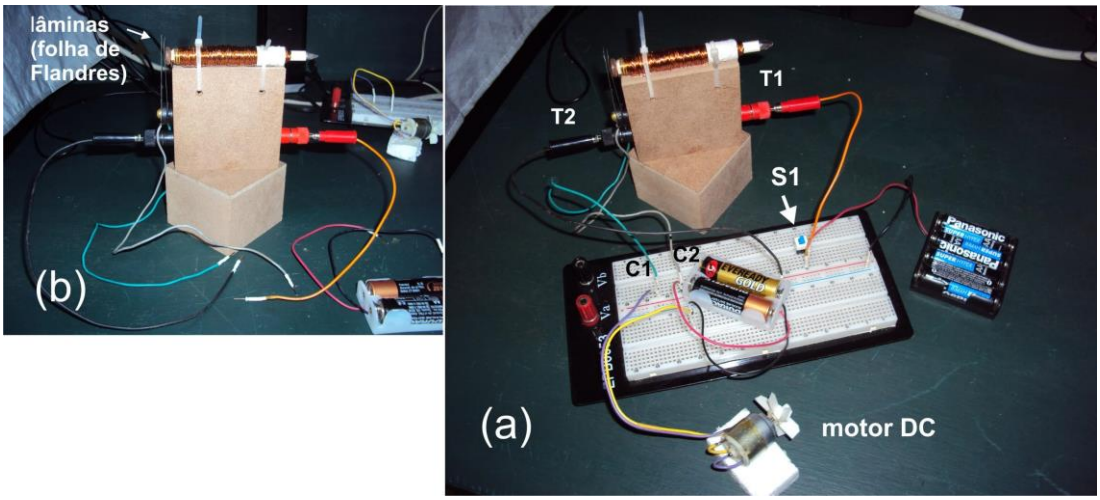


Figura 2.4 – (a) Circuito físico do esquema da Figura 2.3. T1 e T2 são os terminais de alimentação da bobina e C1 e C2 são os contatos de saída do relé. Na figura vemos o motor que será acionado através do relé. (b) Relé artesanal onde os contatos são lâminas recortadas de uma lata de leite em pó isoladas eletricamente por uma arruela de plástico (ver Anexo I).

Ação: O professor fecha a chave S1. As lâminas do relé incialmente separadas entram em contato pela ação magnética do eletroímã fechando o circuito auxiliar. O motor DC entra em funcionamento.

Evento POE – (1) Como vimos, a pilha tem uma polaridade. Se trocarmos a polaridade da fonte de alimentação (fonte de 3V) no circuito do eletroímã o relé vai funcionar?

(2) Se trocarmos a polaridade da fonte de tensão que alimenta o motor DC e acionarmos o relé o motor vai funcionar?

Comentário: os alunos devem ser estimulados a pensar no que está acontecendo identificando o papel de cada elemento do circuito. Ao fechar a chave S1 permitimos a formação de uma corrente elétrica no circuito do motor elétrico. O motor é DC e o sentido de rotação depende do sentido da corrente. Nesta atividade não discutimos ainda os motores elétricos, mas essa pode ser uma oportunidade para abrir o assunto. Existem várias experiências interessantes envolvendo motores elétricos e a placa Arduino. Neste ponto caberia a construção de um pequeno motor elétrico com rotor de espira única que poderia ser montado facilmente e ilustraria bastante bem o principio de funcionamento

de um motor de DC e que por sua vez segue sendo uma boa aplicação da lei de Ampére. Os endereços a seguir, são dois exemplos entre vários, que ensinam montagens muito simples e baratas de um motor elétrico,

- http://educador.brasilescola.com/estrategias-ensino/motor-eletrico.htm
- http://www.youtube.com/watch?v=GeJg8vL-Jqs

2.1.3 - Terceira seção.

Nesta seção vamos fazer uso de um relé comercial. Esse relé tem como característica o baixo consumo podendo ser acionado pela placa Arduino. Vamos usar mais especificamente o chamado **módulo relé** que consiste de uma pequena placa com terminais prontos para as conexões no *protoboard* e no circuito de potencia que vamos usar. Esse módulo relé é muito utilizado em aplicações com a placa Arduino. Na Figura 2.5 mostramos o relé e o módulo relé.

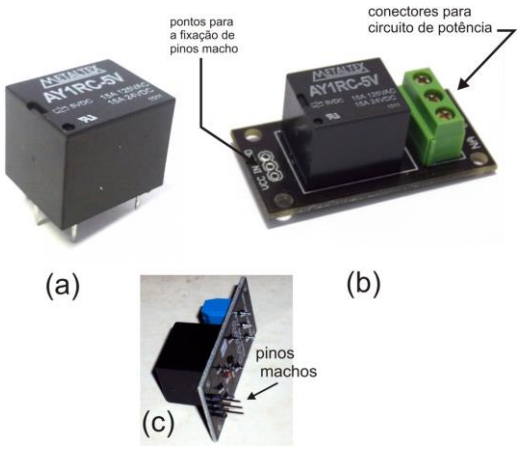


Figura 2.5 – (a) Relé de baixo consumo quando acionado e (b) módulo relé que incorpora numa pequena placa (*breakout board*) os conectores para adaptação ao *protoboard* e terminais para o circuito de potência. (c) Módulo relé com os três pinos machos já soldados.

É preciso chamar a atenção para o fato de que o módulo relé é fornecido com uma barra de três pinos (com o espaçamento padrão de 0,1’’), entretanto a barra não vem soldada no módulo. Com ferro de solda e um pouco de solda é muito fácil fixar os três pinos nos furos indicados na Figura 2.5(b). Depois de soldado o módulo fica como indicado na Figura 2.5(c). Com esses pinos podemos fixar facilmente o módulo relé no *protoboard*.

O circuito que vamos montar tem como finalidade fazer piscar uma lâmpada comum. No caso, selecionamos uma lâmpada vermelha de 15W /127 V. A Figura 2.6 mostra o esquema do circuito elétrico que vamos usar. Pela falta de símbolos convencionais para o módulo relé estamos usando uma representação informal bem simplificada.

Os três pinos do módulo relé são conectado na placa Arduino (via *protoboard* e cabos) da seguinte maneira:

- Pino VCC é conectado ao terminal 5V;
- Pino GND é conectado ao terminal GND;
- Pino IN é conectado a uma porta digital.

Quando a porta digital está no estado LOW (desligado) as conexões de saída C (comum) e NF (normalmente fechado) estão fechadas. Se conectarmos a lâmpada a esses dois terminais ele ficará acesa. Se colocarmos a porta digital no estado HIGH (ligado) as conexões C e NF se abrem cortando o fornecimento de energia para a lâmpada. O contrário irá ocorrer se ligarmos o circuito da lâmpada às conexões C e NA (normalmente aberto).

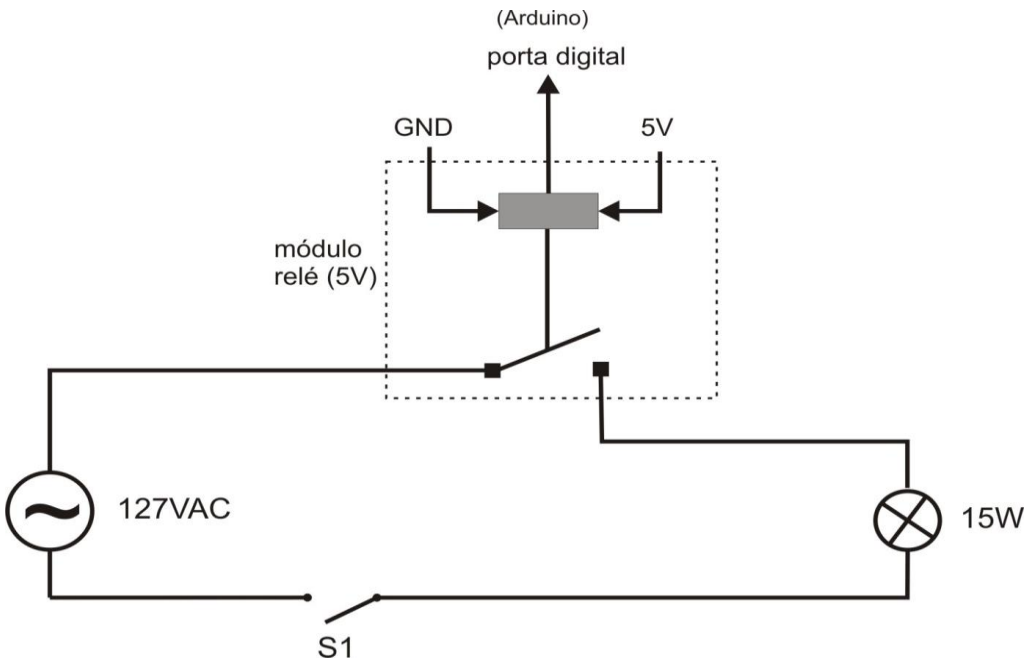


Figura 2.6 - Circuito elétrico esquemático com o módulo relé e uma lâmpada de 15 W. Quando este circuito é acionado a lâmpada pisca.

Na sequência os alunos são convidados a montar, junto com o professor, o circuito físico utilizando um *protoboard* e os componentes elétricos disponibilizados pelo professor. A Figura 2.7 é uma ilustração do circuito físico.

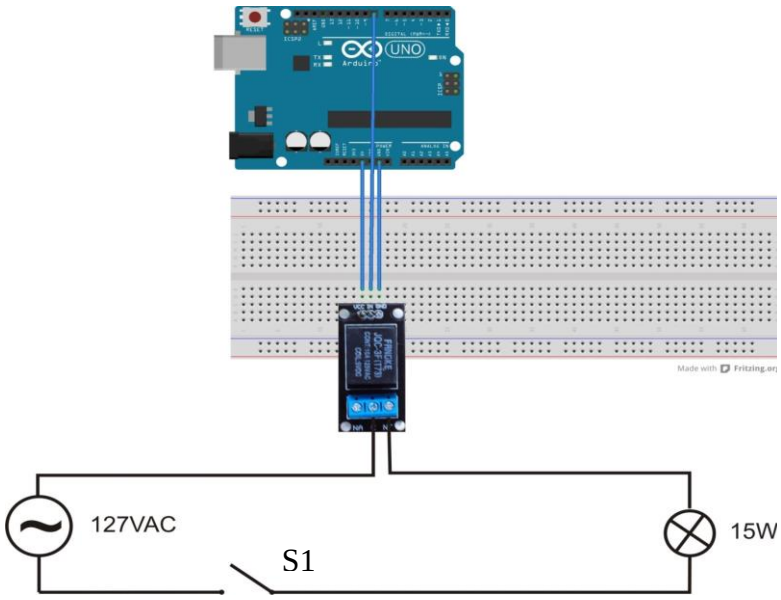


Figura 2.7 – Circuito físico do esquema da figura 2.6.

Por fim, temos que programar a placa Arduino para comandar o relé. Vamos estabelecer que a lâmpada deva piscar a cada 5 segundos e ficar acesa durante igual intervalo de tempo. O comando do relé é feito através de uma porta digital. O problema é similar ao problema de fazer o LED piscar que vimos na Atividade I.

```
//Atividade 2a - Lâmpada piscante

int relePin = 8;

void setup() {pinMode (relePin,
OUTPUT);}

void loop() {
    digitalWrite (relePin, HIGH);
    delay (5000);
     digitalWrite (relePin, LOW);
delay (5000);}
```

Ação: o professor escreve o esquete na IDE e clica no botão *VERIFY*.

Na sequência, salvamos o esquete num diretório apropriado e ligamos a Arduino a uma porta USB do PC através de um cabo próprio. Com a Arduino ligada ao PC vamos carregar (*upload*) o esquete: clicamos no botão *UPLOAD* (ver Atividade I). Antes porém, desligue o cabo de conexão ao pino digital 8. Como regra, faça o *upload* do esquete com os cabos de força desligados.

Ação 1: o professor clica no botão *UPLOAD*.

Ação 2: terminado o *upload*, o professor fecha a chave S1 e liga o cabo ao pino 8.

Vemos a lâmpada piscando com uma frequência de 0,1 Hz.

Evento POE – (1) Como vimos, a pilha tem uma polaridade, mas o mesmo podemos dizer da fonte de tensão que temos na tomada? Se trocarmos a orientação do pino na tomada a lâmpada vai piscar como antes?

(2) Se trocarmos as conexões da lâmpada dos terminais C e NF para C e NA, o que acontecerá?

Comentário: os alunos devem ser estimulados a pensar no que está acontecendo identificando o papel de cada elemento do circuito. O circuito da lâmpada é controlado por duas chaves: a chave S1, que é aqui usada mais como uma chave de segurança, e a chave relé. Quando fechamos a chave S1, a lâmpada passa a ser controlada apenas pelo relé. O circuito da lâmpada é alimentado pela tensão alternada de uso comum e da qual nossos alunos têm muita familiaridade. Nesse ponto da Atividade II ainda não discutimos a tensão alternada, mas isso não representa nenhuma limitação. Esse ponto pode ser abordado de forma puramente operacional, mas pode ser também um momento para começarmos a estabelecer algumas propriedades da tensão alternada em comparação com a fonte de tensão constante, já preparando a turma para um tratamento mais elaborado sobre o assunto. Nessa aplicação o esquete é similar ao esquete do LED piscante que vimos na Atividade I. Trocamos o LED pelo relé e isso por si só é um ponto importante de se chamar a atenção: com um mesmo esquete e um mesmo circuito básico podemos produzir diferentes resultados.

2.2 – Conclusão da Atividade II

Como conclusão da terceira seção, tal como fizemos na Atividade I, planejamos um conjunto de exercícios de fixação e problemas de aprofundamento. Novamente, a inserção dessas sugestões no plano de aula desta Atividade II deve ficar a cargo de cada professor.

Exercício II.1 (fixação) – (1) Como devemos modificar o esquete ***Atividade_2_a.pde*** para fazer com que o lâmpada pisque com uma frequência de 0,05 Hz. Queremos que a lâmpada fique três vezes mais tempo ligada do que desligada. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

Exercício II.2 (fixação) – É possível adicionar mais lâmpadas neste circuito? Colocando mais lâmpadas em paralelo, é possível apagar apenas uma ou teremos o mesmo efeito do pisca-pisca de natal?

Problema II.1 (aprofundamento) – Na atividade que realizamos, o acionamento da lâmpada é pré-programado, isto é, fixamos o intervalo de tempo em que a lâmpada acende e apaga. Podemos fazer diferente. Neste problema queremos programar a Arduino de forma a acender e apagar a lâmpada a partir do próprio computador. Usando o teclado do computador como podemos controlar a lâmpada?

Na Tabela 2.2 apresentamos as instruções usadas.

Tabela 2.2 – Comandos da linguagem de programação praticados na Atividade II

1	//	Esta tabela possui os comandos usados na atividade II, porém o que foi explicado nessa atividade é apenas parte do que cada comando pode fazer. Para aprofundar mais o seu conhecimento sobre a estrutura completa de cada comando e obter maiores informações a respeito, o professor pode acessar o site da página oficial da arduino em http://arduino.cc/en/Reference/HomePage
2	int	
3	setup()	
4	loop()	
5	void	
6	pinMode()	
7	digitalWrite()	
8	delay()	
9	for	
10	random()	

Problema extra, um desafio para ir mais longe: A atividade II teve como objetivo a construção de um circuito para fazer uma lâmpada piscar usando o relé e a placa Arduino. A conexão dessa placa com o computador é feita através da porta USB.

Existem muitos tipos de placas ou escudos (*shield*), que possuem diversas finalidades, que podem ser acopladas à placa Arduino deixando-a mais poderosa, com outras funções. O *Bluetooth Shield* é uma placa, ou escudo (*shield*), que possui um módulo *Bluetooth* integrado e pode ser facilmente usado com a Arduino para uma comunicação serial sem fio. Uma conexão do computador com a placa Arduino pode ser realizada sem a utilização de um cabo conectado a porta USB, utilizando esse *shield* (escudo) acoplado na Arduino.

Após *Bluetooth Shield* ser acoplado a Arduino, a conexão do Arduino com o computador será estabelecida através de sinal de rádio, assim como no sistema *wi-fi*, sem a necessidade de cabos ou da porta USB física do PC. Esse tipo de conexão é muito utilizada no nosso dia-a-dia. O sistema *Bluetooth* é muito usado para conectar computadores entre si, computadores e celulares, computadores e fones de ouvido, computadores e sistemas de alarme e etc, sempre envolvendo pequenas distâncias da ordem de 10 metros. O sistema *wi-fi* é parecido, mas é usado preferencialmente para conexões a internet uma vez que envolve maior potência de transmissão, podendo atingir distâncias de conexão maiores.

Através desse procedimento, utilizando o *Bluetooth shield*, construa um projeto para realizar a Atividade II, acionando a lâmpada de forma independente, sem utilizar cabos ou a porta USB, alimentando a placa Arduino através de uma bateria de 9 V.

2.3 – Solução dos Exercícios da Atividade II

Exercício II.1 (fixação) – (1) Como devemos modificar o esquete ***Atividade_2_a.pde*** para fazer com que o lâmpada pisque com uma frequência de 0,05 Hz. Queremos que a lâmpada fique três vezes mais tempo ligada do que desligada. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

Compare esse exercício com o exercício I.1. As perguntas são basicamente as mesmas e só modificamos a frequência de pulsação. Queremos fixar a ideia de que com um mesmo esquete podemos alcançar objetivos diferentes.

(1) Se o tempo desligado é t então o tempo ligado tem que ser 3t. O ciclo completo é pois t + 3t = 4t. Se a frequência (f) é 0,05 Hz o período é T = 1/f = 1/0,05 = 20 s. Assim 4t = 20 e portanto t = 5,0 s. A lâmpada deve ficar 5,0 s apagado e 15,0 s aceso. Para executar essa tarefa temos que alterar o esquete original ***Atividade_2_a.pde*** alterando o valor das duas instruções ***delay***, como se segue:

Esquete original		Esquete modificado (alterações em negrito)

//Atividade 1a: lâmpada piscante
 	
int relePin = 10;
 	
void setup() {pinMode(ledPin, OUTPUT);}
 	
void loop() {
digitalWrite (relePin, HIGH);
delay (1000);
digitalWrite (relePin, LOW);
delay (1000);}
		//Atividade **2b**: lâmpada piscante
		
		int relePin = 10;
		
		void setup() {pinMode(relePin, OUTPUT);}
		
		void loop() {
		digitalWrite (relePin, HIGH);
		delay (15000);
		digitalWrite (relePin, LOW);
		delay (5000);}

(2) Não é necessário modificar o circuito. A modificação é apenas quanto ao controle do acionamento do LED e, portanto um problema de programação, de software.

Exercício II.2 (fixação) – É possível adicionar mais lâmpadas neste circuito? Colocando mais lâmpadas em paralelo, é possível apagar apenas uma ou teremos o mesmo efeito do pisca-pisca de natal?

Sim, é perfeitamente possível adicionar mais lâmpadas a este circuito estando elas conectadas em paralelo. Porém não é possível apagar apenas uma lâmpada, pois esse sistema é controlado apenas por um relé, quando a chave magnética é desligada, todas as lâmpadas apagam. Para controlar cada lâmpada de uma forma independente, seria necessário o uso de vários relés, um para cada lâmpada e também disponibilizar um aporta digital para cada relé. Para esse circuito somente é possível aumentar o rabicho e colocar várias lâmpadas em paralelo com a primeira, quando ela acender todas acenderão junto e quando ela apagar, todas apagarão juntas. Desta forma, como ela é controlada através de um único relé, é impossível apagar apenas uma lâmpada e deixar outra acesa.

Problema II.1 (aprofundamento) – Na atividade que realizamos, o acionamento da lâmpada é pré-programado, isto é, fixamos o intervalo de tempo em que a lâmpada acende e apaga. Podemos fazer diferente. Neste problema queremos programar a Arduino de forma a acender e apagar a lâmpada a partir do próprio computador. Usando o teclado do computador como podemos controlar a lâmpada?

Nesta experiência não mudamos em nada o circuito elétrico da lâmpada piscante que aplicamos na Atividade II, só vamos alterar a programação da Arduino. O esquete que vamos propor é uma adaptação de um esquete proposto pelo fabricante do módulo relé que estamos usando, o Laboratório de Garagem[4].

[4] www.labdegaragem.org/ e http://labdegaragem.com/

```
//Atividade 2b - acionando o relé a partir do
//teclado do computador.
//Problema II.1

char leitura;      //define uma variável tipo char
                   // denominada leitura

#define rele 8     //define o nome rele como sendo 8

void setup() {

Serial.begin(9600); //Inicializa comunicação Serial
 pinMode(rele, OUTPUT); //Seta o pino indicado por rele
                        //como saída

 digitalWrite(rele,LOW); //Mantem rele desligado assim
                         // que iniciar o programa
 }

void loop() {

 while (Serial.available() > 0) {  //Verifica se há
                                   //conexão com a serial
  leitura = Serial.read();  //Lê o dado vindo da Serial
                          //e armazena na variável leitura

     if (leitura == 'd' || leitura =='D'){//Se a variável
                          //leitura for igual a 'd' ou 'D'
                    //ela desliga rele. As duas barras ||
                 //corresponde a operação booleana OU

     digitalWrite(rele,LOW);

     }

     /* Senão verifica se a variável leitura é
     igual a 'l' ou 'L'. Caso afirmativo, liga o rele */

       else if (leitura == 'l' || leitura =='L'){

       digitalWrite(rele,HIGH);
     }
         Serial.println(leitura);
   }
 }
```

O esquete é muito simples e vamos usar mais alguns recursos muito úteis da linguagem de programação da Arduino. Vamos ver cada item do esquete proposto no

quadro abaixo, mas antes, porém, vamos escrever o esquete no IDE e fazer o *upload* para a Arduino.

Ações:

- ligue a chave do circuito da lâmpada; a lâmpada permanece apagada.
- aperte a tecla ´M´; a lâmpada permanece apagada.
- aperte a tecla ´L´; a lâmpada acende.
- aperte a tecla ´P´; a lâmpada permanece acesa.
- aperte a tecla ´D´; a lâmpada apaga.

Tente agora com as letras maiúsculas. Você verá que o resultado é o mesmo. Nesta aplicação só as letra d (D) e l (L) podem acionar a lâmpada.

Vejamos as novas instruções usadas.

- char leitura; – definimos aqui uma variável denominada **leitura** e essa variável é do tipo **char**. A uma variável tipo **char** podemos atribuir um caractere literal. Podemos fazer **leitura = 'F'.** A variável leitura é igual ao caractere F. Entretanto a variável é armazenada como um número. A cada caractere que podemos acessar a partir do teclado do PC temos um número associado através de uma tabela. A tabela adotada e ASCII (http://arduino.cc/en/Reference/ASCIIchart). Nessa tabela o sinal de exclamação assume o valor 65. Você pode atribuir a variável leitura o sinal de exclamação, leitura = '!'. É necessário colocar aspas.

- #define rele 8 – a instrução #define atribui um nome a uma constante. Quando o esquete for processado, toda vez que aparecer o nome **rele** ele será substituído pelo número **8**. Essa forma não cria uma variável e com isso nenhuma memória é utilizada.

Na função **setup()** temos a instrução nova,

- Serial.begin(9600); – com essa instrução inicializamos a comunicação entre a Arduino e o PC através da porta serial (USB). Quando essa instrução é executada podemos enviar e receber dados para a Arduino a partir do PC. Essas informações podem ser enviadas/recebidas através da IDE usando a função ***Serial Monitor***. A Figura 2.8 mostra o acesso a essa função. O valor entre parêntesis é a taxa de transferência de dados. No caso estamos usando 9600 bits por segundo. O bit é a menor unidade de informação que podemos trocar entre dois sistemas. No momento oportuno será muito apropriado tratar da linguagem digital, isto é, de como a informação é trocada e processada nos computadores e por extensão nos microcontroladores. Nesse ponto em que estamos essa descriminação ainda não é essencial.

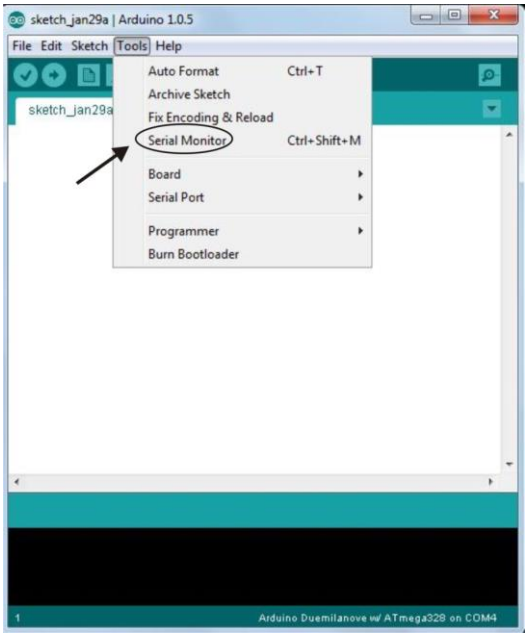


Figura 2.8 – A função Serial Monitor da IDE Arduino. Ao clicar em Serial Monitor uma janela de comunicação com a Arduino se abre.

As instruções seguintes já são conhecidas: ***pinMode(rele, OUTPUT)*** estabelece que o pino 8 (rele = 8) é um pino de saída de dados e ***digitalWrite(rele,LOW)*** estabelece que esse pino inicie no estado LOW (desligado).

Vejamos agora a função loop. A primeira instrução que encontramos é,

while (Serial.available() > 0) {...} – a instrução ***Serial.available()*** verifica a quantidade de dados que está chegando pela porta serial. Verifica se a quantidade de dados (bytes recebidos) é maior do que zero (> **0**). A instrução ***while***, por sua vez, é uma espécie de *loop* que executa o que está entre colchetes indefinidamente enquanto o teste colocado entre parêntesis for verdadeiro. Se nada for digitado pela porta serial nada acontece, isto é, ***Serial.available()*** retorna o valor zero e as instruções entre colchete não são executadas. Lembre-se que as instruções são enviadas a partir da IDE usando a função *Serial Monitor*. Assim que uma informação é enviada para a Arduino a partir do *Serial Monitor* as instruções entre colchetes passam a ser executadas. Por isso dissemos que a execução dessa primeira instrução da função loop verifica se há conexão com a porta serial.

A primeira instrução entre colchetes é,

leitura = Serial.read() – a instrução ***Seria.read()*** lê o que foi enviado pela porta serial e armazena essa informação na variável **leitura**.

Na sequência temos a instrução ***if(){...}.*** Essa instrução, como a instrução ***for***, é muito poderosa. Trata-se de uma instrução condicional: se o que está entre parêntesis for verdadeiro execute as instruções que estiverem entre colchetes. Se não for verdadeiro ignore tudo e siga em frente. O que fica entre parêntesis é um operador de comparação. Vejamos como esta escrita a instrução *if* no esquete:

if (leitura == 'd' || leitura =='D'){...} – a primeira parte do teste consiste em verificar se a variável ***leitura*** é igual a letra ***d*** minúscula; a segunda parte verifica se a variável ***leitura*** é igual a letra ***D*** maiúscula. Note que o termo de comparação é dado pelo símbolo **==, "***igual a"*. Já usamos outro termo de comparação acima quando usamos o símbolo > na instrução ***while***, com o significado de *"maior que"*. Outros símbolos de comparação podem ser usados. A lista completa é,

== (igual a)	**< (menor que)**	**<= (menor ou igual a)**
!= (não igual a)	**> (maior que)**	**>= (maior ou igual a)**

Entre os dois termos de comparação temos um outro operador ||. Esse é um operador lógico, ou booleano, e significa a conjunção **OU**. Assim, o teste é satisfeito se a variável leitura for igual à letra **d** minúscula **OU** igual à letra **D** maiúscula. Outros operadores lógicos podem ser usados,

&& - E
II - OU
! - NÃO

Agora, temos duas possibilidades: a condição é satisfeita ou não.

- ✓ Se o teste é aprovado, as instruções entre colchetes são executadas. No esquete a condição entre colchete é única,

digitalWrite(rele,LOW) – com a instrução ***digitalWrite()*** escrevemos na porta 8 (rele = 8) o estado LOW, desligado. Nessa condição o relé físico é desligado e consequentemente a lâmpada se apaga!

- ✓ Se o teste não é aprovado, isto é, a variável leitura não e igual a d ou D nada acontece e seguimos para a instrução seguinte onde executamos um teste complementar ao anterior,

else if (leitura == 'l' || leitura =='L'){...} – a instrução ***else if*** é similar a instrução ***if***. No caso, queremos fazer um teste em continuação ao teste anterior, isto é, queremos ver se a variável ***leitura*** é igual à letra ***l*** ou ***L***. Se for verdadeiro, executamos a instrução entre colchetes que no caso é,

digitalWrite(rele,HIGH) – neste caso escrevemos na porta 8 o estado HIGH, ligado, isto é, ligamos o relé e com isso ligamos a lâmpada.

Se o teste resultar falso, nada acontece e seguimos para a última instrução do esquete,

Serial.println(leitura) – a instrução ***serial.print()*** imprime na porta serial o que estiver indicado entre parêntesis, no caso em questão a variável ***leitura***. Com auxílio da função *Serial Monitor* da IDE vemos simultaneamente o que esta sendo digitado através do teclado. A instrução que estamos usando ***Serial.println()*** tem a terminação ***ln*** , que impõem após a impressão da variável a mudança automática de linha.

Atividade III – Controlando um Semáforo com uma Placa Arduino.

Metas da atividade: nesta atividade queremos mostrar alguns usos importantes da lei de Ohm aplicada a circuitos elétricos simples e mostrar a importância dos circuitos microcontrolados na realização de uma sequência de operações repetitivas que envolvem tomadas de decisão. Na Atividade I usamos a Arduino para acionar um LED e vimos como podemos estabelecer um controle automático sobre o estado ligado-desligado. Queremos agora mostrar como podemos usar esses recursos para o controle de operações importantes relacionadas ao dia a dia de uma cidade. Escolhemos para isso o controle de um semáforo de tráfego. Nesse exemplo queremos atingir um nível mais concreto da relação entre a ementa de Física que estamos seguindo e a área da tecnologia numa aplicação bem contextualizada. Nesta aplicação perseguimos mais intensamente o domínio do software. Em linhas gerais desenvolveremos competências e habilidades propostas no currículo mínimo como:

- Reconhecer, utilizar, interpretar e propor modelos explicativos para fenômenos naturais ou sistemas tecnológicos (Nova EJA e Ensino Médio Noturno).
- Dimensionar circuitos ou dispositivos elétricos de uso cotidiano (Ensino Médio Noturno).
- Compreender os conceitos de corrente, resistência e diferença de potencial elétrico (Ensino Médio Noturno).

Objetivos da atividade: construir o circuito elétrico físico do semáforo simples apenas para o controle dos automóveis; reconhecer os componentes elétricos comerciais (resistor, LED e fios) e suas propriedades elétricas; aplicar a lei de Ohm; introdução à placa Arduino.

Conceitos trabalhados: Diferença de potencial elétrico (DDP ou tensão), corrente e resistência elétrica (V, I e R); lei de Ohm.

Pré-requisito: Atividade I. Os alunos precisam ter participado da parte expositiva, onde os conceitos pertinentes a esta prática foram apresentados e discutidos.

Material utilizado: para a realização desta atividade, em uma seção única, serão necessários os itens relacionados na Tabela 3.1.

Tabela 3.1 – Relação de materiais para a Atividade III

	Item:	Quant.	Observação
1	*Protoboard*	01	
2	LED vermelho (5 mm)	02	
3	LED verde (5 mm)	02	
4	LED amarelo (5 mm)	01	
5	Resistor 150 Ω	03	
6	Arduino Uno (REV3)	01	
7	Cabos para conexão (Jumpers M/M)	QN	Usados na conexão do *protoboard* e portas da Arduino.

3.1 – Descrição.

3.1.1 - Seção Única.

Iniciamos apresentando o problema do controle de um semáforo. No caso particular de nossa escola, Colégio Estadual José Marti, temos um semáforo no acesso à escola pela Avenida Professor Plinio Bastos que é usada pelos nossos alunos diariamente. Pedimos aos nossos alunos que descrevam o seu funcionamento. Se necessário for podemos conduzir a turma até o semáforo para uma atenta observação de seu funcionamento. Neste semáforo temos o sinal de rua para os carros e o sinal de pedestre que é acionado por um botão fixo num poste (ver Figura 3.1).

(a) (b)

Figura 3.1 – (a) Foto da frente do Colégio que possui o sinal utilizado pelos alunos. Durante o dia a escola é usada pelo município e tem outro nome. (b) Foto destacando o botão de acionamento do sinal.

O sinal de rua funciona normalmente, com os intervalos padrão de alternância das cores. Quando o pedestre aciona o botão junto ao poste transfere o controle para o sinal do pedestre que fica sempre normalmente fechado, isto é, em vermelho. Com o acionamento do botão o sinal dos automóveis vai para o vermelho e o de pedestre muda para o verde liberando a passagem durante um intervalo de tempo dado. O sinal dos pedestres tem apenas as cores verde e vermelha. Após esse tempo o sinal do pedestre retorna para o vermelho e o sinal dos automóveis vai para o verde. Para evitar que o sinal de pedestre seja acionado indevidamente, um lapso de tempo mínimo deve transcorrer entre dois acionamentos sucessivos. Toda a questão deve ser discutida com turma para que se saiba claramente como devemos proceder na programação da placa Arduino que fará o controle do semáforo. Nosso semáforo será uma miniatura em que usaremos LED's coloridos em lugar de lâmpadas.

Na Figura 3.2 vemos o circuito físico do semáforo simples apenas para o controle dos automóveis. Neste caso, por simplicidade, não estamos apresentado o circuito esquemático do semáforo. Na sequência os alunos são convidados a montar, junto com o professor, o circuito físico utilizando um *protoboard* e os componentes elétricos disponibilizados pelo professor. Os três LED's da figura 3.2 correspondem ao sinal dos automóveis. Os LED's são conectados às portas digitais da Arduino. Para o sinal dos automóveis estamos usando as portas número 10 para o verde, 11 para o amarelo e 12 para o vermelho.

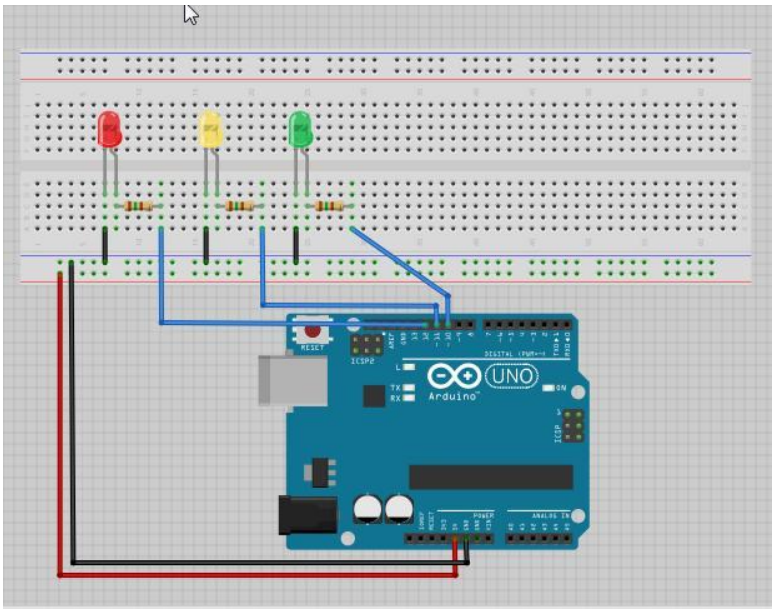


Figura 3.2 – Circuito físico do semáforo simples para automóveis

Temos agora que programar a Arduino para o controle do semáforo.

```
//Atividade 3a - semáforo simples para automóveis
int ledDelay = 10000;  //espera entre as alterações
int redPin = 12;
int yellowPin = 11;
int greenPin = 10;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop() {
  digitalWrite (redPin, HIGH);
  delay (ledDelay);

  digitalWrite (yellowPin, HIGH);
  delay(2000);

  digitalWrite (greenPin, HIGH);
  digitalWrite (redPin, LOW);
  digitalWrite (yellowPin, LOW);
  delay (ledDelay);

  digitalWrite (yellowPin, HIGH);
  digitalWrite (greenPin, LOW);
  delay (2000);

  digitalWrite(yellowPin, LOW);
}
```

Ação: o professor digita o esquete na IDE e clica no botão *VERIFY*, verificando assim a sua correção.

Na sequência, salvamos o esquete num diretório apropriado e ligamos a Arduino a uma porta USB do PC através de um cabo próprio. Com a Arduino ligada ao PC vamos carregar (*upload*) o esquete: clicamos no botão *UPLOAD* (ver Atividade I). Antes, porém, deligue o cabo de conexão do pino *ground*. Como regra, faça o *upload* do esquete com os cabos de força desligados.

Ação 1: o professor clica no botão *UPLOAD*.

Ação 2: terminado o *upload*, o professor conecta o cabo terra ao pino *ground* da Arduino fechando o circuito do semáforo.

O pequeno semáforo entra em operação. Inicialmente vemos a sequência em que são acesos os LED's vermelho, amarelo e verde do sinal para automóveis.

Evento POE – Como vimos, a pilha tem uma polaridade, mas o mesmo não podemos dizer da fonte de tensão que temos na tomada. Se trocarmos a orientação do pino na tomada, a lâmpada vai piscar como antes?

Comentário: os alunos devem ser estimulados a pensar no que está acontecendo identificando o papel de cada elemento. Toda a sequência de funcionamento do semáforo deve ser confrontada com o esquete utilizado. Ao fechar o circuito o sinal de veículos executa a sequência de estados mostradas na Figura 3.3. Esse padrão corresponde à convenção adotada no Reino Unido, pois estamos usando o esquete proposto por Michael McRoberts que adota essa convenção.

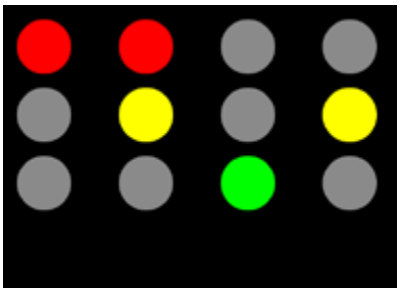


Figura 3.3 - Quatro estados do sistema de semáforos do Reino Unido (imagem do livro Arduino Básico de Michael McRoberts p.56)

É muito interessante discutir com os alunos as diferenças entre o padrão usado no Brasil e este, usado no Reino Unido. O tempo para a alternância do estado fechado para aberto e vice-versa e dado pela variável inteira ***ledDelay***. No exemplo que aqui adotamos tomamos esse tempo com sendo de 10 s (***int ledDelay = 10000;***). Esse valor é meramente ilustrativo e pode ser alterado segundo as conveniências de cada situação. A disposição dos LED's com resistores limitadores de corrente é idêntica à situação que estudamos na Atividade I. O anodo (+) do LED é ligado a um pino digital e o catodo (-) é ligado ao ***ground*** através de um resistor limitador de 150 Ω. Aqui temos um detalhe importante: estamos usando uma resistência um pouco maior do que os 100 Ω que usamos para acionar um único LED na Atividade I. O problema é que no padrão de acionamento do semáforo que estamos adotando dois LED's são acionados simultaneamente por certo tempo (2s). Podemos ver o que acontece através do esquema

mostrado na Figura 3.4. Quando os dois LED's são acionados simultaneamente a fonte interna alimenta as duas portas digitais, que no caso em questão são as de número 12 e 11. A fonte tem que suprir os dois ramos do circuito simultaneamente. Na Figura 3.4 indicamos essas correntes como i_1 e i_2. Após as ramificações as duas correntes se juntam e forma uma corrente única que é a soma das duas,

$$\mathbf{i_{total} = i_1 + i_2}$$

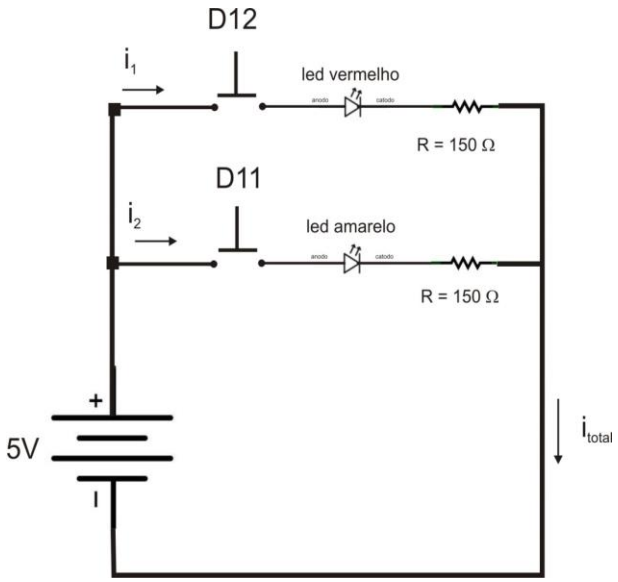


Figura 3.4 – Circuito elétrico esquemático para acionamento dos LED's vermelho e amarelo.

A corrente total não deve ser superior a 100 mA e a corrente em cada porta não deve ser superior a 40 mA. Qual a corrente em cada porta? Se aplicarmos a lei de Ohm em cada ramo temos que,

$$i_1 = i_2 = 5V / 150\Omega = 0{,}0333\ A = 33{,}3\ mA$$

o que dá para a corrente total,

$$i_{total} = i_1 + i_2 = 66{,}7\ mA$$

Esse valor está dentro da margem de tolerância. Se tivéssemos usado resistores de 100 Ω teríamos uma corrente total maior. É interessante observar que temos uma associação em paralelo de resistores e LED's. Com essa associação é como se tivéssemos uma resistência equivalente as duas, menor. Podemos refazer as contas com 100 Ω e vamos concluir que a i_{total} será de 100 mA o que já está no limite da capacidade da placa Arduino.

Vamos agora procurar entender as instruções que compõem o nosso esquete.

```
int ledDelay = 10000;
int redPin = 12;
int yellowPin = 11;
int greenPin = 10;
```

Definimos quatro variáveis no total. As três últimas variáveis definem os pinos onde são ligados os três LED's do semáforo. É muito importante dar nomes as variáveis que se associam a sua função. Isso ajuda muito na organização do esquete evitando erros de troca de variáveis no curso da edição de um esquete. Nesse caso, a variável y***ellowPin*** define o pino digital (11) em que será colocado o LED amarelo do sinal e a variável ***greenPin***, por exemplo, define o pino digital (10) em que será colocado o LED verde do sinal. As variáveis seguem uma terminologia em inglês (veja o exercício III.1).

Vejamos agora a estrutura da função ***setup ()***.

```
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin,
OUTPUT);
  pinMode(greenPin,
OUTPUT);
}
```

A instrução ***pinMode()*** é usada para definir os pinos digitais 10, 11 e 12 como pinos de saída (OUTPUT). O pino 2 é comandado pelo botão.

A função ***loop()*** é muito simples:

```
void loop() {
  digitalWrite(redPin, HIGH);
  delay(ledDelay);

  digitalWrite(yellowPin, HIGH);
  delay(2000);

  digitalWrite(greenPin, HIGH);
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  delay(ledDelay);

  digitalWrite(yellowPin, HIGH);
  digitalWrite(greenPin, LOW);
  delay(2000);

digitalWrite(yellowPin, LOW);
}
```

A primeira instrução ***digitalWrite(redPin, HIGH)*** coloca a porta digital 12 no estado ligado (HIGH), isto é, acende o LED vermelho. A instrução seguinte ***delay(ledDelay)*** estabelece que o LED deve ficar aceso por 10s, o valor adotado para a variável ***ledDelay***. Sem desligar o LED vermelho, a instrução seguinte manda ligar o LED amarelo através da instrução ***digitalWrite(yellowPin, HIGH)***. A partir desse instante os LED's vermelho e amarelo ficam acesos por um intervalo dado pela instrução seguinte ***delay(2000)***, isto é, 2s.

Na sequência, colocamos a porta digital 10 no estado alto através da instrução ***digitalWrite(greenPin, HIGH)***, ou seja, ligamos o LED verde. Em sequência imediata apagamos os LED's vermelho e amarelo através das duas instruções ***digitalWrite(redPin, LOW)*** e ***digitalWrite(yellowPin, LOW)***. A sequência em que mandamos os pinos 11 e 12 para o estado desligado (LOW) é muito rápida e não a conseguimos acompanhar visualmente. Para todos os propósitos o LED verde é aceso e os LED's vermelho e amarelo se apagam simultaneamente. A instrução seguinte ***delay(ledDelay)*** estabelece que o sinal permaneça verde por, também, 10s.

Por fim, após 10s, as instruções seguintes acende o LED amarelo e apaga o verde: ***digitalWrite(yellowPin, HIGH)*** e ***digitalWrite(greenPin, LOW)***.

O LED amarelo é mantido ligado por 2s através da instrução seguinte ***delay(2000)***. Terminado esse prazo, a instrução que segue apaga-o: ***digitalWrite(yellowPin, LOW)***.

Com essa última instrução chega-se ao fim do *loop* e a sequência de instruções é então integralmente repetida, indefinidamente. Para parar o semáforo será necessário desligar a Arduino.

3.2 – Conclusão da Atividade III

Como conclusão da seção, tal como fizemos na Atividade I, planejamos um conjunto de exercícios de fixação e problemas de aprofundamento. Novamente, a inserção dessas sugestões no plano de aula desta Atividade III deve ficar a cargo de cada professor.

Exercício III.1 (fixação) – Reescreva o esquete ***Atividade_3_a.pde*** de forma a dar novos nomes as variáveis utilizadas usando termos em português. Digite o novo esquete na IDE Arduino e use o botão *verify* para se certificar que não cometeu nenhum erro. Com o seu professor faça o *upload* do seu novo esquete e verifique o funcionamento.

Exercício III.2 (fixação) – (1) Como devemos modificar o esquete ***Atividade_3_a.pde*** para fazer com que o tempo de parada seja o dobro do tempo de movimento. Considere que os automóveis devem ficar retidos por 16 s. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

Problema III.1 (aprofundamento) – (1) Na atividade que realizamos o acionamento do semáforo obedece a convecção usada no Reino Unidos. Procure saber qual a convenção de acionamento de um semáforo em nosso país e em função disso reescreva o esquete para cumprir essa convenção. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

Na Tabela 3.2 apresentamos as instruções usadas.

Tabela 3.2 – Comandos da linguagem de programação praticados na Atividade III

1	//	Esta tabela possui os comandos usados na atividade III, porém o que foi explicado nessa atividade é apenas parte do que cada comando pode fazer. Para aprofundar mais o seu conhecimento sobre a estrutura completa de cada comando e obter maiores informações a respeito, o professor pode acessar o site da página oficial da arduino em http://arduino.cc/en/Reference/HomePage
2	int	
3	setup()	
4	loop()	
5	void	
6	pinMode()	
7	digitalWrite()	
8	delay()	
9	for	
10	random()	

Problema extra, um desafio para ir mais longe: Não chegamos a estudar a situação mais complicada na qual o semáforo é composto de dois sinais, um para automóveis e o outro para pedestre. Procure montar o projeto de um semáforo com os dois sinais para automóveis e pedestres como comentamos no início da atividade (SUGESTÃO: consulte o livro Arduino Básico de Michael McRoberts).

3.3 - Solução dos Exercícios da Atividade III

Exercício III.1 (fixação) – Reescreva o esquete ***Atividade_3_a.pde*** de forma a dar novos nomes as variáveis utilizadas usando termos em português. Digite o novo esquete na IDE Arduino e use o botão *verify* para se certificar que não cometeu nenhum erro. Com o seu professor faça o *upload* do seu novo esquete e verifique o funcionamento.

Neste exercício queremos que os alunos pratiquem a criação de variáveis. A referência importante é a de dar nomes às variáveis que correspondam a suas finalidades. No caso podemos apenas fazer uma adaptação dos termos em inglês. Damos aqui uma primeira sugestão e cada professor pode adaptar a solução do exercício segundo seus próprios critérios.

Esquete original		Esquete modificado (alterações em negrito)

//Atividade 3a - semáforo simples
. . .
int redPin = 12;
. . .
		//Atividade **3b** - semáforo simples
		...
		. . .
		int **vermPin** = 12;
		. . .

Exercício III.2 (fixação) – (1) Como devemos modificar o esquete ***Atividade_3_a.pde*** para fazer com que o tempo de parada seja o dobro do tempo de movimento. Considere que os automóveis devem ficar retidos por 16 s. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

(1) Neste caso temos que estabelecer dois tempos de espera. No esquete original adotamos um único tempo de espera de 10.000s que serve tanto para o tempo de parada quanto para o tempo de movimento. Assim sendo vamos precisar definir duas variáveis ao invés de apenas uma. Por simplicidade vamos chamá-las **ledDelay1** para o tempo em que os automóveis ficam parados e **ledDelay2** para o tempo de movimento. Segundo o enunciado do exercício devemos fazer **ledDelay1** = 16000 e **ledDelay2** = 32000.

Esquete original		Esquete modificado (alterações em negrito)

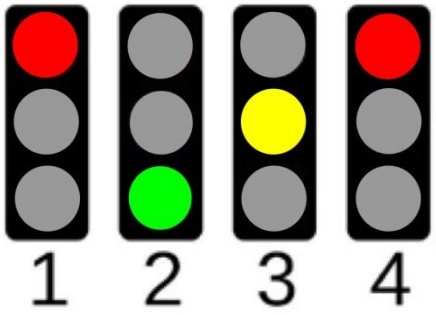
//Atividade 3a - semáforo simples
 	
int ledDelay = 10000;
int redPin = 12;
int yellowPin = 11;
int greenPin = 10;
 	
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}
 	
void loop() {
  digitalWrite (redPin, HIGH);
  delay (ledDelay);
 	
  digitalWrite (yellowPin, HIGH);
  delay(2000);
 	
  digitalWrite (greenPin, HIGH);
  digitalWrite (redPin, LOW);
  digitalWrite (yellowPin, LOW);
  delay (ledDelay);
 	
  digitalWrite (yellowPin, HIGH);
  digitalWrite (greenPin, LOW);
  delay (2000);
 	
  digitalWrite(yellowPin, LOW);
}
		//Atividade 3a - semáforo simples
		
		int **ledDelay1 = 16000**;
		int **ledDelay2 = 32000**;
		
		int redPin = 12;
		int yellowPin = 11;
		int greenPin = 10;
		
		void setup() {
		  pinMode(redPin, OUTPUT);
		  pinMode(yellowPin, OUTPUT);
		  pinMode(greenPin, OUTPUT);
		}
		
		void loop() {
		  digitalWrite (redPin, HIGH);
		  delay (**ledDelay1**);
		
		  digitalWrite (yellowPin, HIGH);
		  delay(2000);
		
		  digitalWrite (greenPin, HIGH);
		  digitalWrite (redPin, LOW);
		  digitalWrite (yellowPin, LOW);
		  delay (**ledDelay2**);
		
		  digitalWrite (yellowPin, HIGH);
		  digitalWrite (greenPin, LOW);
		  delay (2000);

		digitalWrite(yellowPin, LOW); }

(2) Não é necessário modificar o circuito. A modificação é apenas quanto ao controle do acionamento dos LEDs e, portanto um problema de programação, de software.

Problema III.1 (aprofundamento) – (1) Na atividade que realizamos o acionamento do semáforo obedece a convecção usada no Reino Unido. Procure saber qual a convenção de acionamento de um semáforo em nosso país e em função disso reescreva o esquete para cumprir essa convenção. (2) Para isso devemos modificar também o circuito elétrico que usamos na aula?

(1) O padrão de acionamento de um semáforo para automóveis no Brasil é ligeiramente diferente do padrão no Reino Unido. O esquema abaixo é uma representação do padrão brasileiro,



Compare com o esquema da Figura 3.3 da atividade III. No padrão brasileiro não temos o acionamento de duas cores simultaneamente.

```
//Atividade 3c - semáforo simples para automóveis com padrão brasileiro

int ledDelay = 10000;  //espera entre as alterações
int redPin = 12;
int yellowPin = 11;
int greenPin = 10;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop() {
  digitalWrite (redPin, HIGH);
  delay (ledDelay);

  digitalWrite (greenPin, HIGH);
  digitalWrite (redPin, LOW);
  delay (ledDelay);

  digitalWrite (yellowPin, HIGH);
  digitalWrite (greenPin, LOW);
  delay (2000);

  digitalWrite(yellowPin, LOW);
}
```

(2) Não é necessário modificar o circuito. A modificação é apenas quanto ao controle do acionamento do LED amarelo e, portanto um problema de programação, de software.

Anexo I - Relação de materiais

I.1 - Neste anexo apresentamos uma relação de materiais básicos para o desenvolvimento das atividades I, II e III.

Tabela I.1 – Relação de Materiais para as atividades I, II e III.

	Descrição:	Quantidade	Observação
1	*Protoboard*	01	20 x 30 pinos
2	Placa Arduino Uno (Rev 3)	01	
3	Fonte de Tensão (9V)	01	Fonte para Arduino
4	Ferro de solda (40 W)	01	
5	Estilete	01	
6	Tesoura	01	Tamanho médio
7	Alicate de corte	01	pequeno
8	Alicate de bico	01	pequeno
9	Chave de parafuso	01	pequeno
10	Caixa de ferramentas	01	Maleta do Professor
11	Multímetro de 3 1/2 dígitos	01	
	Componentes e materiais eletroeletrônicos		
1	Resistores, 100, 150, 500, 1.000, 10.000 Ω (1/8 W)	10 (cada)	
2	Led (vermelho, amarelo, verde)	10 (cada)	
3	Módulo Relé 5 VDC 10A	1	Fornecido pelo Laboratório de Garagem
4	Cabos para conexão (Jumpers M/M)	1	Pacote com 25 unidades

A primeira parte da tabela contém ferramentas que podem ser facilmente encontradas numa loja de materiais elétricos e de ferragens. Os componentes eletrônicos por sua vez são encontrados em lojas mais especializadas nem sempre acessíveis na própria localidade em que reside o professor. O suprimento de materiais eletrônicos através de lojas comerciais tem sofrido uma notável retração. Temos usado, em

substituição, as lojas virtuais que podem ser encontradas numa busca simples na internet. Os fornecedores de placas Arduino e acessórios que nos tem atendido com profissionalismo são,

- **Multilógica Shop** - http://multilogica-shop.com/
- **Laboratório de Garagem** - http://www.labdegaragem.org/loja/
- **FelipeFlop** - http://www.filipeflop.com/

Para o fornecimento de componentes eletrônicos de uso geral encontramos um bom sortimento na,

- **Farnell-Newark** - http://www.farnell.com.br/

I.2 - Na Atividade II fizemos uso de um pequeno motor elétrico DC. Nos fornecedores de material para a Arduino é possível conseguir motores DC, mas é possível conseguir também a partir de brinquedos quebrados. Consulte os seus alunos para encontrar eventuais doadores de peças e estimular a cultura da reciclagem.

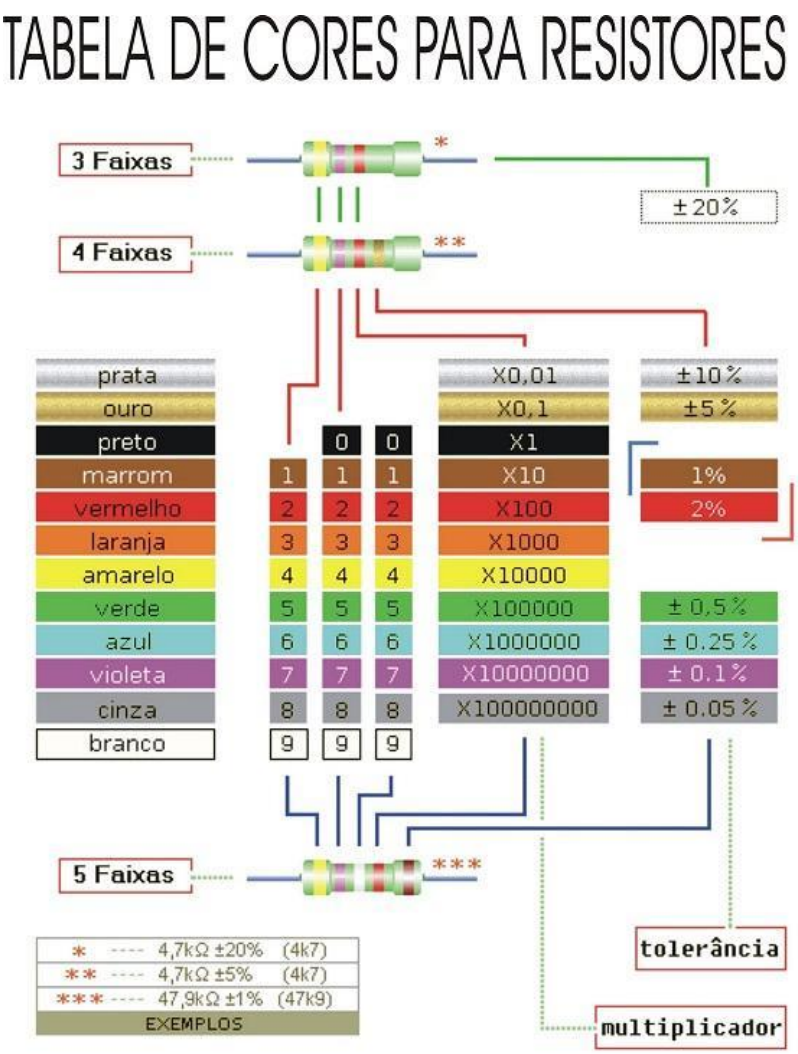
I.3 – Na Atividade II fazemos uso de um eletroímã. A construção do eletroímã é muito simples e várias indicações de como construir podem ser encontradas na internet, como por exemplo, http://www.manualdomundo.com.br/2012/06/como-fazer-um-eletroima-experiencia-de-fisica-eletromagnetismo/

O modelo que construímos faz uso de um prego (tamanho 19 x 36) e fio de cobre esmaltado. O problema maior é, em geral, conseguir o fio de cobre. Um modelo simples para ser acionado com pilhas de 1,5 V precisa de um bom suprimento de fio. No modelo que construímos, indicado na Figura 2.4 da atividade II, usamos aproximadamente 20 m de fio AWG 23 a 26, isto é, com aproximadamente 0,5 mm de diâmetro. Comprar fios de cobre esmaltados não é fácil e a melhor opção é reciclar. Podemos conseguir esses fios retirando de transformadores, indutores ou motores danificados. O relé que construímos para uso da Atividade II incorpora duas lâminas de metal separadas por um anel isolante, ao eletroímã. Essas duas lâminas são os terminais elétricos do relé. O dispositivo foi montado sobre uma estrutura feita a partir de uma pequena caixa de madeira MDF e sua tampa. Esse material é fácil de ser trabalhado e pode ser conseguido em pequenas quantidades em lojas de material para artesanato. Entretanto outros materiais mais simples podem ser usados como o isopor.

Anexo II – Tabela de Cores para Resistores

Fonte: http://www.digitei.com/tabela-de-cores-de-resistores/

Existem muitas tabelas de cores para resistores que podem ser obtidas diretamente na internet. Colocamos aqui a titulo de exemplo um modelo que consideramos simples e atualizado, que engloba a convenção para cinco faixas. Será muito útil se uma copia colorida puder ser fixada permanentemente em sala de aula e uma ou mais listas de exercícios puder ser aplicada na turma para a fixação do uso da convenção de cores. No CD que acompanha a dissertação colocamos um arquivo com imagem desta tabela.



REFERENCIAS BIBLIOGRÁFICAS

ALVES, R. M.; SILVA, A. L. C. da; PINTO, M. de C.; SAMPAIO F. F.; e ELIA M. da F. Uso do Hardware Livre Arduino em Ambientes de Ensino-aprendizagem, Jornada de Atualização em Informática na Educação, v.1, n.1, cap. 6, p. 162-187, 2012.

BRAGA, N. C.; Eletrônica Básica, 6ª ed, São Paulo Editora: Newton C. Braga. Maio de 2012.

BRAGA, N. C.; Curso Básico de Eletrônica, 4ª ed., São Paulo Editora Saber, 2001, p.55

CAVALCANTE, M. A.**;** Novas Tecnologias no Estudo de Ondas Sonoras**,** Cad. Bras. Ens. Fís., v. 30, n. 3, p. 579-613, dez. 2013.

CAVALCANTE, M. A.**;** RODRIGUES T.T.T.; BUENO D. A.; Controle Remoto: Principio de Funcionamento (parte 1 de 2), Cad. Bras. Ens. Fís., v. 30, n. 3, p. 554-565, dez. 2013.

CAVALCANTE M. A.; TAVOLARO C. R. C; MOLISANI E.; Física com Arduino para Iniciantes**,** Revista Brasileira de Ensino de Física, v. 33, n. 4, 4503, 2011.

GONÇALVES, L. R.; PASSOS, S. R. M. M. S. dos; PASSOS Á. M. dos. Novos rumos para o Ensino Médio Noturno – como e por que fazer?, Ensaio: aval. pol. públ. Educ., Rio de Janeiro, v.13, n.48, p. 345-360, 2005.

MENDELSON, P.; GREEN, T. R. G.; BRNA, P. (1990) Programming languages in education: the search for an easy start. In Hoc, J., Green, T., Gilmore, D. & Samway, R. (eds) Psychology of Programming, 175-200, London, Academic Press.

RICHTER, D.; BRAGA F. S.; FÜRKOTTER M; Informática no Processo Ensino-Aprendizagem: Contribuindo para uma Nova Escola, Revista Formação, v.2, n.13, p. 8-13, 2003

ROBERTS, M.; Arduino Básico, 1ª ed. São Paulo, Editora Novatec, 2011.

SCAICO P.D.; AZEVEDO S.; ALENCAR Y.; LIMA A. de A.; PAIVA L.F.; MENDES J.P.; SILVA J.B.B. da; RAPOSO E.H.; SCAICO A.; Ensino de Programação no Ensino Médio: Uma Abordagem Orientada ao Design com a linguagem Scratch. Revista Brasileira de Informática na Educação, v.21, n. 2, p. 93-103, 2013.

Secretaria Estadual de Educação do Estado do Rio de Janeiro, Currículo Mínimo de Física, http://www.conexaoprofessor.rj.gov.br/curriculo_aberto.asp Acesso em 03/09/2013.

Secretaria Estadual de Educação do Estado do Rio de Janeiro, Manual de Orientações do Nova EJA, http://projetoseeduc.cecierj.edu.br/principal/download/Manual_projeto_nova_eja_final.pdf Acesso em 04/10/2013.

SOUZA A.R. de; PAIXÃO A.C.; UZÊDA D.D.; DIAS M.A.; DUARTE S.; AMORIM H.S. de; A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC**,** Revista Brasileira de Ensino de Física, v. 33, n. 1, 1702. 2011.

TREAGUST D. F.; KEARNEY M.; YEO S. and ZADNIK M. G.; Student and Teacher Perceptions of the Use of Multimedia Supported Predict – 143 Observe – Explain Tasks to Probe Understanding, Research in Science Education, v. 31, n. 4, pp. 589 – 615, 2001.

VALENTE, J. A.; Mudanças na sociedade, mudanças na educação: o fazer e o compreender. In: VALENTE, J.A. (Org). O computador na sociedade do conhecimento. Campinas: Unicamp/NIED, 1999. p. 153. Apud RICHTER D.