UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

DEPARTAMENTO DE FÍSICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENSINO DE FÍSICA

MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA

PRODUTO EDUCACIONAL

AQUISIÇÃO DE DADOS COM ARDUINO E SMARTFONE: EXPERIMENTO DO PÊNDULO SIMPLES.

Autor: Prof. João Bosco Araújo Fernandes

Orientador: Prof. Dr. Jairo Ricardo Rocha de Oliveira

Recife

2020

Sumário

1. Apresentação

Esse Produto Educacional foi elaborado para o Mestrado Nacional Profissional em Ensino de Física, polo UFRPE no ano de 2020. Seu objetivo principal é permitir aos professores de ensino médio, e também a qualquer interessado, uma iniciação no desenvolvimento de projetos usando Arduino e smartphone Android.

Como esse objetivo não é simples de ser atingido, foram tomados alguns cuidados para diminuir a complexidade envolvida:

a) Uso de uma sequência didática com curva de aprendizado suave para estudo do Arduino e de alguns dispositivos eletrônicos. Ela foi desenvolvida ao longo de vários anos para originalmente fazer a iniciação de educandos e outros interessados em projetos com Arduino e foi atualizada para atender as especificidades dos professores.

b) Para evitar a tarefa nada simples de desenvolver um app de comunicação bluetooth entre o smartphone e o Arduino foi usado o app gratuito Serial Bluetooth Terminal. Esse app Android utiliza uma interface muito simples e amigável para enviar e receber dados por essa conexão.

c) O experimento para determinar a expressão do período do pêndulo simples foi selecionado devido à simplicidade de sua construção e estudo experimental. É importante observar que pequenas modificações apenas no software e na disposição dos componentes desse experimento, permitem fazer uma série de outros experimentos para medição de tempo. A inclusão de uns poucos novos dispositivos simples de usar permite elevar em muito essa quantidade de experimentos.

Assim, no capítulo II é apresentado o Arduino e visto em detalhes como usá-lo para acionar dispositivos elétricos tais como LED's e reles. A seguir é analisado como esse microcontrolador manipula sensores, em particular, o sensor de luz. E no final do capítulo é descrito como é feita a comunicação via Bluetooth entre o Arduino e um smartphone Android.

No capítulo III é apresentado o Serial Bluetooth Terminal, um programa que instalado em um smartphone Android permite de forma muito simples enviar e receber dados para qualquer dispositivo que use uma conexão bluetootch.

Em seguida, no capítulo IV é discutido em detalhes um dispositivo de medição do período do pêndulo simples por aquisição de dados com Arduino e smartphone.

Por último, no capítulo V é apresentada uma sugestão de experimento para obtenção da expressão período do pêndulo simples.

2. A plataforma Arduino

2.1. O micro-controlador Arduino

O Arduino foi desenvolvido em 2005 no Interaction Design Institute, na cidade italiana de Ivrea e o nome Arduino foi dado à nova placa em referência a um bar frequentado por professores e estudantes desse instituto. Foi idealizado por Massimo Banzi, um professor preocupado em encontrar um meio barato e fácil de seus estudantes de arte e design trabalharem com tecnologia, em contrapartida aos produtos caros e relativamente difíceis de usar existentes até então. As duas exigências básicas eram, primeiro que fosse barato e com preço não podendo ser mais caro que o custo da pizza que um estudante fosse comprar e em segundo lugar que fosse um sistema que qualquer um pudesse utilizar. Assim Banzi solicitou a David Cuartielles, pesquisador visitante da Universidade sueca de Malmö, desenhar os circuitos da placa, ao mesmo tempo em que David Mellis, aluno de Banzi, desenvolvia o software de controle da placa. Em seguida, um engenheiro local chamado Gianluca Martino foi contratado para orientar os alunos do Design Institute no desenvolvimento dos projetos (Arduino.cc-Origem 2019).

Como era trabalhoso para os estudantes encontrar todos os componentes e dispositivos necessários, foram produzidos kits contendo as placas e os itens eletrônicos mais comuns dos seus projetos. Foi um grande sucesso e o estoque inicial de duzentos kits foi rapidamente vendido, sendo necessária a fabricação de mais unidades para suprir a demanda. Designes, artistas e profissionais de outras áreas se interessaram pela novidade e passaram a usá-lo em seus projetos e a sua popularidade cresceu ainda mais (Arduino-Origem 2019). O sucesso foi confirmado com a obtenção de uma menção honrosa na categoria Comunidades Digitais em 2006, pela Prix Ars Electronica, além da marca de mais de 50.000 placas vendidas até outubro de 2008 (Build It-Arduino 2019). O grande público percebeu que além de sua relativa facilidade de uso e custo baixo ele era uma excelente introdução para

programação de microcontroladores e dessa forma poderia ser usado por qualquer pessoa interessada (Arduino-Origem 2019).

O Arduino é uma plataforma eletrônica baseada em hardware livre e software de código aberto. Assim, pode-se comprar o Arduino pré-pronto ou adquirir seus componentes separadamente e conectá-los de acordo com as necessidades do projeto a ser desenvolvido. É possível encontrar, principalmente em sites da internet, todas as rotinas necessárias ao projeto já prontas para uso ou somente parte delas e desenvolver só o restante que falta ou ainda desenvolver totalmente os algoritmos necessários ao projeto. Tudo dependendo da necessidade, tempo e interesse do usuário (Arduino.cc-Intro 2019).

Ele pode ser usado para desenvolver objetos interativos independentes ou ser conectado a um computador, a uma rede ou até mesmo à internet para recuperar e enviar dados do Arduino e trabalhar com eles. Assim, pode ser usado em soluções IoT, wearable, impressão 3D e ambientes incorporados. Como exemplo simples, pode enviar um conjunto de dados recebidos de sensores para um site e exibi-los na forma de tabela ou de um gráfico (Arduino-Intro 2019).

Com o passar dos anos, o Arduino tem permitido o desenvolvimento de milhares de projetos, desde objetos do cotidiano até instrumentos científicos complexos. Uma crescente comunidade mundial de entusiastas (estudantes, amadores, artistas, programadores e profissionais de diversas áreas) utilizando de forma intensa essa plataforma contribuiu para o desenvolvimento de uma incrível quantidade de literatura livre, de fácil acesso e de grande ajuda para iniciantes e especialistas (Arduino.cc-Intro 2019).

Uma pesquisa na Internet por “Arduino” mostrará um vasto número de sites dedicados ao Arduino e que apresentam projetos interessantes em que ele foi utilizado para ler dados e controlar uma enorme quantidade de dispositivos. Por isso ele é um dispositivo incrível e com um pouco de vontade

para aprender como usá-lo será possível a criação de quase tudo, desde obras de arte interativas até robôs (Mcroberts 2011).

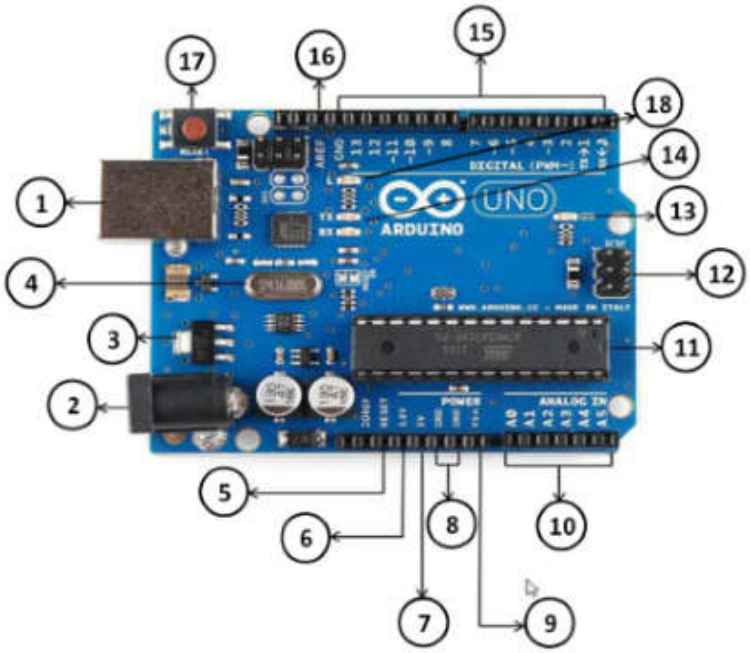
O Arduino original tem sua programação feita diretamente pelo IDE oficial do Arduino, o layout da placa é padronizada, está devidamente documentado no site oficial (www.arduino.cc), está devidamente licenciado para exibir o nome e o logotipo do Arduino e é produzido por fabricantes autorizados e oficiais que atualmente são SmartProjects na Itália, Sparkfun nos EUA e DogHunter em Taiwan / China (Arduino.cc-clone 2019).

Como ele tem hardware aberto, há muitas não oficiais disponíveis para compra com base no Arduino original, ou que podem ser criadas a partir de um diagrama de fácil compreensão. Assim é possível comprar os componentes apropriados e montar uma placa Arduino completamente funcional em uma matriz de pontos ou em uma PCB (Printed Circuit Board, placa de circuito impresso) feita em casa. A equipe do Arduino impõe apenas que não seja utilizado o nome “Arduino” nem o seu logotipo, pois eles são de uso exclusivo da placa oficial (Mcroberts 2011).

O Arduino original ou oficial usa um microcontrolador da família ATmega (Oliveira 2018). Apenas no site do Arduino são disponíveis para compra dezenas de diferentes tipos de placas, cada uma com uma finalidade específica, além de módulos para expansão (shelds) e kits que estão agrupados em Entry Level (Nível de Entrada), Enhanced Features (Recursos Aprimorados), Internet of Things (Internet das Coisas), Education (Educação), Wearable (Vestível) além do grupo Retired (Aposentado) que mostra a história de diversos modelos de Arduino e placas auxiliares que foram descontinuados por ficarem obsoletos (Arduino.cc-Produ 2019).

Atualmente a placa Uno, pertencente ao grupo Entry Level e mostrada na Figura 1, é o modelo de referência para a plataforma Arduino, sendo a placa mais utilizada nos projetos em geral e a mais fácil de fazer a iniciação em

Arduino. Ela é a primeira de uma série de placas USB Arduino e com micro-controlador baseado no ATmega328P (Arduino-Desc 2019).



Componente	Descrição
1	Conector USB – Permite comunicação com um PC via cabo USB
2	Conector de alimentação – Entrada de tensão (6 V a 20 V DC).
3	Regulador de voltagem – Estabiliza as tensões DC em toda a placa.
4	Oscilador de cristal de 16 MHz – Temporizador do sistema.
5	Pino de reset da placa – Reinicializa a placa com botão externo.
6	Pino de 3,3 V – Fornece 3,3 V de saída.
7	Pino de 5 V – Fornece 5 V de saída.
8	Pino GND (Ground) – Pino terra e de retorno de corrente.
9	Pino Vin – Entrada de tensão por fonte de energia externa AC.
10	Pinos analógicos de A0 a A5.
11	Microcontrolador da placa – Atualmente no Uno é o ATmega328P.
12	Pino ICSP – Usado para transferir programas/firmwares e também para executar tarefas administrativas.
13	LED indicador – Indica que a placa está ligada corretamente.
14	LEDs TX e RX – O led TX pisca enquanto envia os dados seriais. A O led RX pisca durante o processo de recebimento.
15	Pinos digitais – A placa Arduino UNO possui 14 pinos de E/S digitais
16	Pino AREF – Define uma tensão de referência externa (entre 0 e 5 Volts) como o limite superior para os pinos de entrada analógica.
17	Botão de reset da placa – Reinicializa a placa.
18	LED built-in – conectado geralmente ao pino digital 13.

Figura 1 – Arduino Uno.
Adaptado de:
https://www.tutorialspoint.com/arduino/arduino_board_description.htm.
Acesso em 10-01-2019.

Através do conector USB (componente 1 da Figura 1) o Arduino pode ser alimentado e controlado diretamente pelo computador. Para tornar-se autônomo, um programa deve ser desenvolvido no computador e transferido para o Arduino também através da conexão USB. Depois de desconectado, é necessário que através do conector de alimentação (componente 2 da Figura 1) a placa receba energia de alguma fonte com tensão contínua entre 6 V e 20 V. (Monk, 2014, p. 1).

É importante observar que os pinos digitais de 0 a 13 podem ser configurados como pinos de entrada para ler valores lógicos (0 ou 1) ou como pinos de saída para acionar diferentes dispositivos como LEDs, relés, etc. Os pinos 3, 5, 6, 9, 10 e 11 são rotulados com “~” pois podem ser usados também para fornecer saída do tipo PWM (Pulse Width Modulation) ou Modulação de Largura de Pulso (Arduino-Desc 2019).

2.2. A IDE Arduino

Para controlar a placa do Arduino é necessário desenvolver um sketch (esboço) que é um algoritmo de controle e fazer sua gravação na placa através de uma conexão USB. Esse desenvolvimento é feito no Arduino Software (IDE) e que atualmente disponível em duas opções (Arduino.cc-Start 2019):

a) IDE on-line (Arduino Web Editor) – Deve-se usar essa opção somente se disponível uma conexão de Internet confiável. As placas originais e algumas placas clones licenciadas funcionam prontamente sem necessidade de nenhuma instalação adicional. Essa interface permite que os sketchs criados sejam salvos na internet, disponibilizando-os a partir de qualquer dispositivo online e fazendo backup automático. Sempre será executada a versão mais atualizada do IDE sem a necessidade de instalar atualizações ou bibliotecas geradas pela comunidade.

b) IDE off-line – Quando não se dispuser de uma conexão confiável com a internet ou utilizar placas clones não compatíveis com a opção online a solução é baixar a versão mais recente do IDE e fazer a sua instalação no PC. Após a instalação aparece o ícone da IDE do Arduino indicado pela Figura 2 (Arduino.cc-Start 2019).

Figura 2 - Ícone da IDE Arduino
Fonte: IDE do Arduino.

2.2.1. Configurando a IDE off-line do Arduino.

Assim que a IDE for aberta aparece algo similar a Figura 3

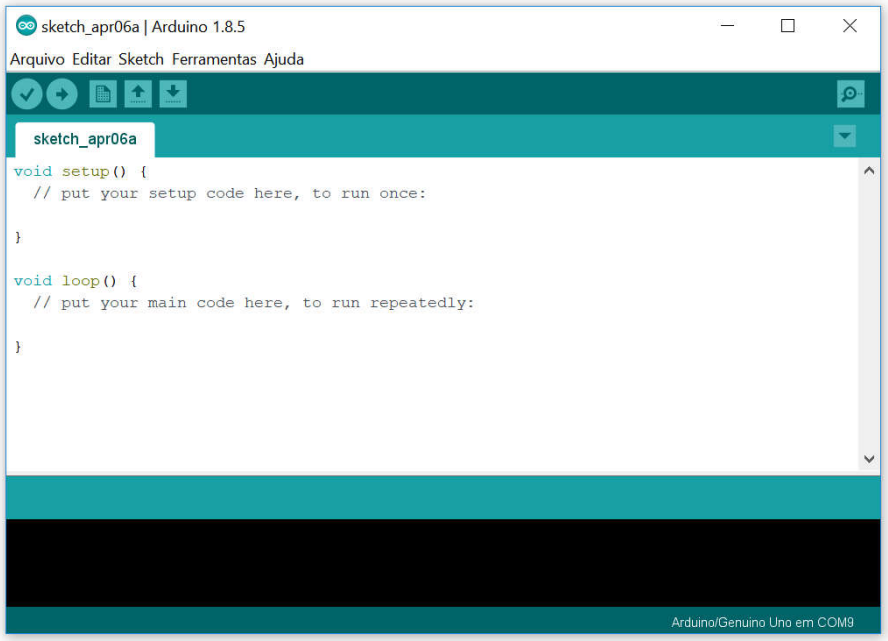


Figura 3 – IDE do Arduino quando aberto pela primeira vez.
Fonte: IDE do Arduino.

Acionando o item de menu **Ferramentas** clicar em **Placa** (Figura 4).

Figura 4 – Menu Ferramentas do Arduino.
Fonte: IDE do Arduino.

A seguir será apresentada a uma lista de placas (Figura 5). Deve-se selecionar a opção apropriada a partir dessa lista.

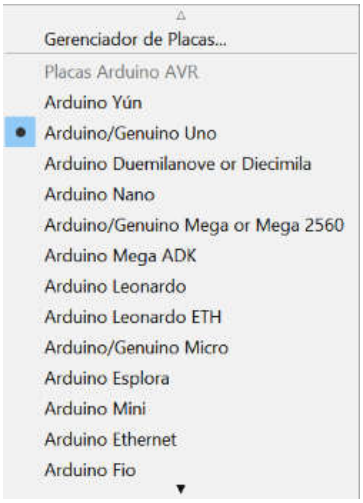


Figura 5 – Item de menu Placas do Arduino.
Fonte: IDE do Arduino.

Como último passo da configuração, deve-se acionar novamente o menu **Ferramentas**, clicar em **Porta** e escolher na lista a porta apropriada para a placa do Arduino (Figura 6).

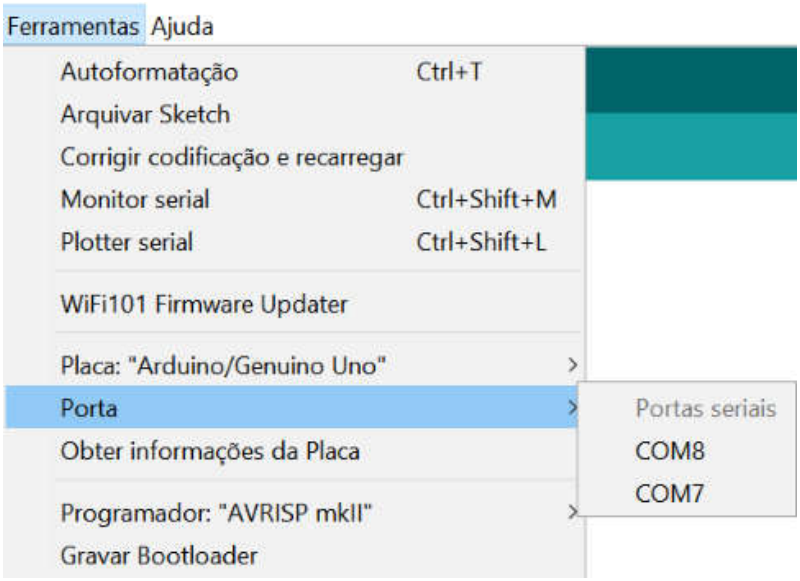


Figura 6 – Item de menu Porta do Arduino.
Fonte: IDE do Arduino.

O IDE do Arduino é bem simples, e permite rápido aprendizado à medida que é utilizado. Conforme se torna mais hábil no uso da IDE padrão, pode ser considerado experimentar alguma IDE profissional, como Eclipse, ArduIDE, GNU/Emacs, AVR-GCC, AVR Studio e até mesmo o XCode da Apple (Mcroberts 2011).

2.3. A linguagem de programação do Arduino

A linguagem de programação do Arduino é baseada em Wiring (Arduino.cc-Intro 2019). Wiring é uma linguagem de programação de código aberto para microcontroladores e baseada em C/C++ (Wiring 2019). Um programa em Arduino é chamado sketch (esboço em inglês) e é composto por uma sequência de instruções que só podem ser escritas com palavras-chave e que seguem um conjunto bem definido de regras da linguagem (Mcroberts 2011).

O termo função na programação do Arduino é extremamente importante e tem vários significados. Primeiro, ele pode ser usado com o mesmo sentido de uma função matemática, pois recebe parâmetros como entrada (o domínio

da função) e retorna um valor de saída (a imagem da função). Como exemplo simples, a execução da instrução com sintaxe ***sqrt(25)*** calcula a raiz quadrada de 25. Ela tem como parâmetro de entrada o valor 25 e como parâmetro de retorno ou de saída o valor 5. Aqui o termo função está associado a apenas uma única instrução pertencente ao grupo de palavras-chave do Arduino. Podem existir funções sem parâmetros de entrada e nesse caso não há nada entre seus parênteses e podem existir funções sem parâmetros de saída em que é obrigatório colocar a palavra-chave ***void*** antes de serem declaradas. Assim, a instrução ***void setup()*** indica que setup é uma função sem parâmetros de entrada e também sem parâmetros de saída (Mcroberts 2011).

Em um sentido diferente, uma função pode ser também um conjunto de comandos agrupados em um bloco de código que se inicia com o símbolo { e termina com o símbolo }. Tudo entre esses dois símbolos fará parte da função. Nesse sentido, uma função pode ter muitas linhas de código, cada linha contendo uma ou várias instruções e pode-se executar esse código inúmeras vezes simplesmente chamando o nome da função, em vez de ter de reescrever o código cada vez que for usá-lo. É obrigatório para um sketch Arduino ter uma função ***setup()*** e uma função ***loop()***, do contrário, ele não funcionará. Por não terem nenhum parâmetro de entrada não há nada entre seus parênteses. Ambas devem ser precedidas da palavra-chave void para indicar que são funções que não retornam nenhum parâmetro. A figura 3 mostra uma possível imagem da IDE logo após sua abertura e nela as duas podem ser vistas (Mcroberts 2011).

A função ***setup()*** é executada uma única vez no início da execução do programa e antes que o loop principal ser executado. Ela contem instruções de inicialização do programa, como a definição dos modos dos pinos, das taxas de transmissão serial etc, Ela inicia com void setup() para informar ao compilador que a função é chamada setup, que ela não retorna nenhum dado (void) e que não recebe nenhum parâmetro (parênteses vazios) (Mcroberts 2011).

A função ***loop()*** é a principal função do programa e será executa continuamente enquanto o Arduino estiver ligado. As declarações dentro da função são executadas uma por vez, até a sua última linha quando é reiniciando o loop num processo repetitivo até que o Arduino seja desligado ou o botão Reset pressionado. De modo similar ao setup, ela inicia com void loop() para informar ao compilador que não retorna nenhum dado e que não recebe nenhum parâmetro (Mcroberts 2011).

Por exemplo, se num programa for desenvolvida uma função (ou um bloco de código) de nome myFunc, em que a sintaxe da escrita seja algo como:

```
int myFunc (int x, int y) {
  // instruções da função
}
```

Então a função ***myFunc*** recebe como entrada dois números inteiros, x e y. E quando seu processamento for concluído, será retornado um valor inteiro no ponto de execução imediatamente após onde ela tiver sido chamada no programa. Para indicar que ela retorna um valor inteiro é obrigatório preceder seu nome com a palavra-chave int. Pelo mesmo motivo deve-se fazer isso com cada parâmetro de entrada inteiro (Mcroberts 2011). É sempre necessário preceder o nome da função e cada um de seus parâmetros de entrada com a correspondente palavra-chave indicativa tipo de dado (Arduino.cc-LanRef 2019).

À medida que um sketch se torna mais complexo, e passar a ter dezenas, centenas ou talvez milhares de linhas, comentários serão vitais para facilitar a compreensão de como cada seção funciona. Pode-se desenvolver um trecho incrível de código, mas provavelmente não será possível lembrar-se de como ele funciona quando for tentar entende-lo dias, semanas, meses ou até anos após a sua criação. Por isso, inserir comentários é essencial para a compreensão do código. Além disso, faz parte do espírito do Arduino, e de toda a comunidade de fonte aberta, o compartilhamento de código e projetos. E o

uso dos comentários pode ajudar muito alguém interessado na compreensão de um determinado algoritmo (Mcroberts 2011).

Um comentário que ocupa toda uma linha ou parte dela deve iniciar com *//*, e qualquer texto a direita será desconsiderado no momento da execução dessa linha do programa. Caso o comentário possua varias linhas é melhor usar outro formato: um bloco linhas de texto entre os sinais */** e ***/***. No anexo A as duas formas são usadas para fazer comentários no sketch Blink.

As instruções da linguagem de programação do Arduino estão agrupadas em três partes principais: **functions** (funções), **values** (valores variáveis e constantes) e **structure** (estrutura). Na IDE do Arduino as instruções do grupo functions são escritas em vermelho, as do grupo values então em azul e aquelas do grupo structure em marrom. Já os comentários ficam em cinza e os demais termos numéricos e alfanuméricos não pertencentes ao grupo de palavras-reservadas ficam na cor preta (Arduino.cc-LanRef 2019).

O grupo functions reúne as instruções que servem para controlar a placa do Arduino e realizar cálculos matemáticos. Como exemplo, no trecho de código

```
void setup() {
  pinMode(13, OUTPUT);    // seta o pino digital 13 como output
}
```

A função setup é usada para configurar o pino digital 13 para o modo de entrada de dados através do uso da função ***pinMode*** que pertence ao grupo functions da linguagem do Arduino. É importante observar que por exigência da linguagem, é obrigatório finalizar cada instrução com um ponto e vírgula. Se desejado, após uma instrução pode-se colocar outra instrução ou um comentário (Arduino.cc-LanRef 2019).

O grupo ***values*** contém as constantes e variáveis que serão usadas ao longo do programa. Constantes são expressões predefinidas na linguagem Arduino. Elas são usadas para tornar os programas mais fáceis de ler. É importante observar que a linguagem diferencia caracteres maiúsculos e minúsculos. Como no exemplo, na instrução pimMode é necessário especificar como primeiro parâmetro o número do pino e como segundo parâmetro o comportamento elétrico do pino, ou seja, se é um pino de entrada (INPUT) ou de saída (OUTPUT). Pinos configurados como INPUT adquirem um estado de alta impedância e consumem correntes extremamente pequenas no circuito que eles estão conectados, equivalente a um resistor em série de 100 MΩ conectado do pino. Isso os torna úteis para ler um sensor. Pinos configurados como OUTPUT passam a ter um estado de baixa impedância, podendo fornecer até 40 mA de corrente para outros circuitos. Isso os torna úteis para alimentar LEDs e dispositivos que usam menos de 40 mA de intensidade de corrente. Cargas maiores que 40 mA, como motores, exigirão um transistor ou outro circuito de interface (Arduino.cc-LanRef 2019).

No grupo ***values***, além das constantes, pode-se trabalhar com variáveis. Uma variável é um local na memória do computador em que é possível armazenar dados de diversos tipos, tais como texto (string), inteiro (int), decimal (float) etc. Assim, uma variável de tipo int, ou inteiro é um número entre −32.768 e 32.767 e a instrução

```
int ledPin = 10;
```

Atribui à variável de tipo inteiro o nome ledPin e dá a ela um valor de 10 (Mcroberts 2011).

Pelas regras da linguagem Arduino, todas as variáveis devem iniciar com uma letra maiúscula ou minúscula e o restante do nome pode ser formado por letras, números e underscores. É proibido utilizar palavras-chave da linguagem como nomes de variáveis. Para ajudar a evitar isso, todas as

palavras-chave no sketch serão mostradas em vermelho, azul ou verde (Mcroberts 2011).

Toda variável pode ter um escopo local ou global em relação ao programa. Se ela for global deverá ser declarada no início do algoritmo e antes da função setup e existirá enquanto o Arduino estiver em funcionamento, podendo ser acessada e alterada por qualquer função do algoritmo. Já uma variável local deverá ser declarada dentro do bloco de instruções de uma função. Poderá ser vista e modificada apenas pelo código dentro do seu próprio bloco e será destruída ao final da execução dessa função. Assim, no trecho de código

```
int ledPin = 10;

void setup() {
  int x = 0;
  // instruções da função
}
```

São criadas a variável global inteira ***ledPin*** e a variável local inteira ***x***. A variável x é uma variável local e só poderá ser usada dentro da função setup. Já a variável ledPin é uma variável global e poderá ser utilizada em qualquer outra função do algoritmo (Mcroberts 2011).

Structure é o terceiro grupo da linguagem Arduino e contém os elementos do squeth (setup e loop), de controle de estrutura (if, for, while, etc), dos operadores (aritméticos, de comparação, booleanos, etc) e elementos de sintaxe adicional (#define, #include, etc) (Arduino.cc-LanRef 2019).

2.3.1. Controle de acionamento do led built-in

Um LED é um Diodo Emissor de Luz (Light Emitting Diode em inglês). Um diodo é um dispositivo que permite a corrente elétrica passar em um único sentido por ele. Caso a corrente tente passar no sentido oposto, o diodo impede que ela o faça. Diodos podem ser úteis para prevenir que se faça uma conexão errada que reverta o sentido da corrente e danifique algum

componente. Os LEDs impedem essa reversão e também emitem luz. Eles são fabricados com diferentes tamanhos, níveis de luminosidade e cores, incluindo o espectro ultravioleta e infravermelho como os LEDs dos controles remotos do televisor, condicionador de ar e outros aparelhos eletrônicos (Mcroberts 2011).

Na placa Arduino existem quatro LEDs. O primeiro é o Power LED, na cor verde, que permanece aceso enquanto o Arduino estiver em funcionamento. Os dois LEDs de comunicação serial, na cor laranja, só ficam acesos quando a placa envia ou recebe dados do computador através do cabo USB. O LED TX informa quando o Arduino envia dados para o PC e o LED RX informa quando o PC recebe dados do PC. Finalmente, o Status LED ou LED built-in ou ainda LED on-board, também de cor laranja, está conectado eletricamente a um dos pinos digitais (LED-Onboard 2019).

Para referir-se ao LED built-in em um comando é usada a constante predefinida LED_BUILTIN. Ela tem valor inteiro e igual ao do pino digital a que o LED está conectado e na grande maioria das placas é o pino 13. Se houver certeza do número do pino, seu valor pode ser usado no lugar de LED_BUILTIN (Arduino.cc-Const 2019).

A IDE Arduino apresenta, no menu Arquivo, item Exemplo, uma grande quantidade de exemplos de sketchs que podem ser abertos, analisados estudados e modificados. Um deles, o Blink (anexo A) é o exemplo ideal para início de estudo do Arduino. Sua codificação precisa de poucos comandos para fazer o LED built-in de forma repetitiva acender por um segundo e em seguida apagar por um segundo. Sem as linhas de comentário, seu algoritmo é

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

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

```
}
```

A primeira e única linha do setup simplesmente informa através do comando pinMode que o pino LED_BUILTIN será de saída e por isso recebe o parâmetro OUTPUT. Essa indicação deverá ser sempre usada quando se desejar acionar um dispositivo conectado a qualquer porta digital. Caso o pino fosse usado para fazer leitura de um sensor deveria ser usado o parâmetro INPUT (Arduino.cc-Const 2019).

Dentro da função loop, digitalWrite é a primeira instrução e permite definir um valor HIGH ou LOW para o pino. Quando se define um pino como HIGH, ele passa a receber 5 volts do Arduino. Quando se define como LOW, o pino recebe 0 volt, tornando-se um terra. Assim, a primeira instrução envia 5 V para o pino LED_BUILTIN e acende o LED built-in (Arduino.cc-Const 2019).

Quando um pino é configurado com saída, ele é posto em estado de baixa impedância e pode fornecer (drenar) até 40 mA de corrente para um circuito externo. Essa configuração é perigosa para a placa Arduino. Por isso, quando um pino for usado como saída de dados, deve sempre haver a preocupação em não ultrapassar o limite de corrente e danificar o pino ou mesmo queimar o microcontrolador (Arduino.cc-DigitalPins 2019).

É muito importante também observar que pelo datasheet do microcontrolador do Arduino Uno, o ATMega 3208, a soma das correntes drenadas por todos os pinos não pode ultrapassar 200 mA com temperatura de 25 ºC e apenas 100 mA quando a 125 ºC. (ATMega-DataSheet 2019, p. 15)

A segunda instrução do loop usa o comando delay e simplesmente informa ao Arduino para esperar 1.000 milissegundos (um segundo) antes de executar a próxima instrução. A terceira instrução do loop anula a tensão que vai para o pino LED_BUILTIN apagando o LED. Por fim a quarta instrução, um delay, manda que se espere por mais 1.000 milissegundos para ir para o final do loop e reiniciar o algoritmo num processo infinito (Mcroberts 2011).

Uma possível melhoria desse código é:

```
int pinoLED = LED_BUILTIN;

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

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

Nele é utilizada uma variável global inteira, através da instrução int, para representar o número do pino digital, o que tornaria o programa em

A grande vantagem dessa modificação é que se for desejado utilizar qualquer pino digital para fazer outra atividade é só alterar uma única linha, a primeira, colocando o valor adequado do pino.

2.3.2. Controle de acionamento de um LED externo.

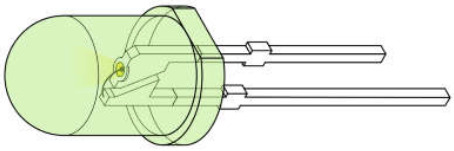


Figura 7– Detalhe de um LED verde com 5mm de tamanho.
Fonte: https://commons.wikimedia.org/wiki/File:LED,_5mm,_green_(en).svg#/media/File:LED,_5mm,_green_(unlabelled,_full).svg. Acesso em 05-04-2019.

Em um examine cuidadoso de um LED percebem-se dois detalhes: os terminais têm comprimentos diferentes, e um lado do LED é chanfrado, em vez de cilíndrico (Figura 7). Esses detalhes servem para indicar qual terminal é o ânodo (positivo) e qual é o cátodo (negativo): o terminal mais comprido (ânodo) é conectado à tensão positiva e o terminal mais curto e com o lado chanfrado (cátodo) vai para o terra. Se o LED for conectado ao contrário, isso não o danificará se ele não tiver que conduzir correntes muito elevadas.

De acordo com o datasheet da Atmega, um pino digital emite 5 V e corrente máxima de 40 mA (Mcroberts 2011). Já um LED de alto brilho branco de 5 mm suporta, de acordo com seu datasheet (disponível emhttp://pdf1.alldatasheet.com/datasheet-pdf/view/303904/OPTOSUPPLY/OSM5DA5111A.html), uma tensão direta da ordem de 3 V e uma corrente de 20 mA. Assim, ligar diretamente um LED a uma porta digital irá sobrecarregá-lo, sendo por isso necessário utilizar alguma estratégia para diminuir a tensão que chega ao LED. A mais simples é a utilização de um resistor em série a ele.

Um resistor é um dispositivo projetado para provocar resistência a uma corrente elétrica, causando uma queda na voltagem em seus terminais e sendo por isso utilizado para diminuir a voltagem e a corrente para os dispositivos conectados a ele. Assim, o terminal curto do LED (cátodo) deve ser conectado ao terra (GND) e o terminal longo (ânodo) deve ser ligado em série com um resistor e conectado a um pino digital para reduzir os 5 V do pino para os 3 V requeridos pelo LED e evitar a sua queima. Para determinar o valor R da resistência adequada é necessário usar a relação (Young 2009, pág 142, equação 25.9)

$$R = \frac{V}{I}$$

Em que $\boldsymbol{V}$ é a tensão nos terminais do resistor e $\boldsymbol{I}$ a corrente que o percorre. Mas a tensão no resistor deve ser a diferença entre a tensão fornecida pelo pino digital ($\boldsymbol{V_S}$) e a tensão requerida pelo LED ($\boldsymbol{V_L}$), então

$$R = \frac{V_S - V_L}{I}$$

Como $\boldsymbol{V_S}$ = 5 V, $\boldsymbol{V_L}$ = 3 V e $\boldsymbol{I}$ = 20 mA, o valor necessário da resistência seria de $\boldsymbol{R}$ = (5 – 3) / 0.02, ou seja, um valor de 200 Ω. Resistores são fabricados com valores-padrão e o valor disponível mais próximo seria de 220Ω. É prudente sempre escolher um resistor com valor mais próximo e maior

ao valor necessário, caso contrário uma corrente muito intensa pode atravessar o resistor e danificar o LED (Mcroberts 2011).

Em geral os resistores são muito pequenos para conter informações de fácil leitura, por isso, é utilizado um código de cores. Ao redor do resistor encontram tipicamente quatro anéis de cores e é possível descobrir o valor de sua resistência utilizando o código de cores contido na Tabela 1.

Cor	1ª Faixa	2ª Faixa	3ª Faixa	4ª Faixa
Preto	0	0	x 10^0	
Marrom	1	1	x 10^1	± 1%
Vermelho	2	2	x 10^2	± 2%
Laranja	3	3	x 10^3	
Amarelo	4	4	x 10^4	
Verde	5	5	x 10^5	± 0,5%
Azul	6	6	x 10^6	± 0,25%
Violeta	7	7	x 10^7	± 0,1%
Cinza	8	8	x 10^8	± 0,05%
Branco	9	9	x 10^9	
Dourado			x 10^{-1}	± 5%
Prata			x 10^{-2}	± 10%
Nenhuma				± 20%

Tabela 1 – Código de cores dos resistores
Fonte: Mcroberts 2011, pág. 48.

Assim, para um resistor de 220 Ω é necessário um 2 na primeira faixa, que é vermelho, seguido por um 2 na faixa seguinte, também vermelho. A terceira faixa se refere ao expoente da potência de 10 que deve ser multiplica ao valor já computado nas duas primeiras faixas e que no caso é 10^1. Ou seja, o expoente é 1, resultando em marrom na terceira faixa. A faixa final indica a tolerância do resistor. Caso o resistor tenha uma faixa dourada, ele tem uma tolerância de ±5% e isso significa que o valor, de fato, do resistor varia entre 209 Ω e 231 Ω. Dessa forma, um LED que requer 3 V de tensão e 20 mA de

corrente, necessitará de um resistor com uma combinação de faixas vermelho, vermelho e marrom (Mcroberts 2011).

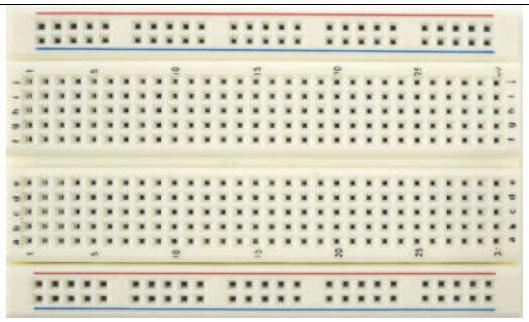


Figura 8 – Protoboard.
Fonte: https://upload.wikimedia.org/wikipedia/commons/b/be/Protoboard_Unitec.jpg. Acesso em 20/04/2019.

Para conectar um LED a um dos pinos digitais, será necessária também uma protoboard (Figura 8). Ela é uma placa muito usada para montagem de circuitos experimentais, pois é reutilizável e não necessita de realização de solda elétrica. O tipo mais comum e usado na maioria dos projetos é a que mede 16,5 cm por 5,5 cm e apresenta 840 furos (ou pontos) na placa. Os furos são conectados eletricamente por tiras de metal condutor alojados no interior da placa. Os furos das duas linhas de tiras paralelas ao topo e a base estão conectados eletricamente somente aos furos da mesma linha, e são projetados para carregar o barramento de alimentação e o barramento do terra. As tiras no centro correm a 90 graus dos barramentos de alimentação e do terra, e há um espaço vazio no meio para que se possa colocar circuitos integrados, de modo que cada pino do chip vá para um conjunto diferente de furos e, portanto, para um barramento diferente (Mcroberts 2011).

As conexões elétricas entre os dispositivos em geral são feitas com fios jumper. Eles são encontrados comercialmente com pontas moldadas para facilitar sua inserção na protoboard. Porém, é possível usar também pedaços de fios rígidos de núcleo único em que se retira cerca de 6 mm do material isolante de cada ponta (Mcroberts 2011).

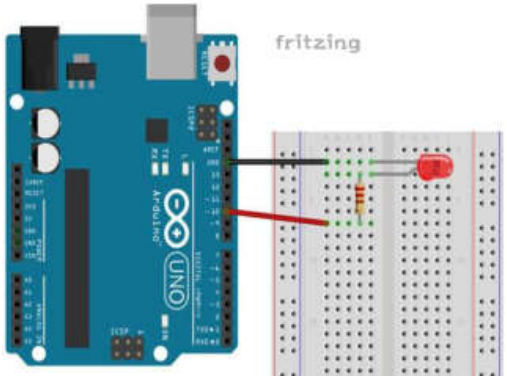


Figura 9 – Esquema de ligação de um LED externo ao Arduino.
Adaptado de: http://www.comofazerascoisas.com.br/projeto-arduino-pisca-led.html.
Acesso: 21/04/2019.

A Figura 9 mostra o diagrama elétrico para acionar o LED. O jumper preto conecta o cátodo do LED ao pino GND. O anodo do LED é conectado com o resistor de 220 Ω e esse é conectado ao pino 10 do Arduino pelo jumper vermelho.

Com as conexões elétricas concluídas, é necessário desenvolver o algoritmo no Arduino. Como exemplo, um código para fazer o LED conectado ao pino digital 10 repetidamente ficar aceso por um segundo e em seguida ficar apagado por um segundo é apresentado abaixo:

```
int pinoLED 10

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

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

Esse código tem uma única diferença em relação ao código para controle do LED built-in: a numeração da porta. E poderia até ser igual se o

circuito fosse conectado ao pino do LED built-in. A reutilização do código ou de parte dele é uma das coisas mais úteis na programação do Arduino.

2.3.3. Eletroímã.

Em abril de 1820, o cientista dinamarquês Hans Christian Orsted descobriu que se um fio condutor percorrido por uma corrente elétrica fosse colocado próximo a uma bússola, a sua agulha era defletida da posição de equilíbrio. Verificou assim que uma corrente elétrica ao atravessar um fio condutor cria ao seu redor um campo magnético (UNESP-Eletroímã-2019).

Enrolando-se um fio condutor de tal modo que forme uma sequência de voltas (espiras) muito próximas em forma de um tubo retilíneo produz-se uma bobina de fios chamado solenóide. Se por ele passar uma corrente elétrica, é gerado um campo magnético no sentido perpendicular à seção reta do solenóide. O resultado final é que o solenóide possui pólos norte e sul, tal como um ímã natural (UNESP-Eletroímã 2019).

O ferro doce (ferro puro) tem a propriedade de só se imantar enquanto estiver próximo de um campo magnético. Assim, usando um núcleo de ferro doce em um solenóide é criado um imã temporário relativamente forte chamado eletroímã. Os eletroímãs apresentam inúmeras aplicações práticas tais como telégrafos, fones, alto falantes, telefones, campainhas e relés entre outras (USP-eletroímã 2019).

Um relé do tipo eletromecânico (EMR – Electromechanical Relay) é simplesmente um interruptor eletromecânico. Como mostra a Figura 9, ele é composto por um eletroímã, uma armadura (haste de material ferromagnético) e pelos contatos: comum (C), normalmente aberto (NA) e normalmente fechado (NF). Quando a bobina está desligada, C e NF estão conectados eletricamente. Quando uma tensão é aplicada na bobina, ela se magnetiza e atrai a armadura, levando C a se desconectar de NF e a se conectar com NA. Quando a bobina é

desligada, ocorre o contrário, C desconecta-se de NA e volta a ter contato com NF (Embarcados–Arduino 2019).

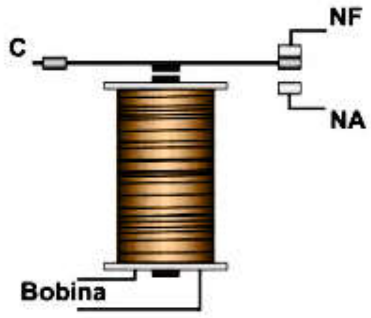
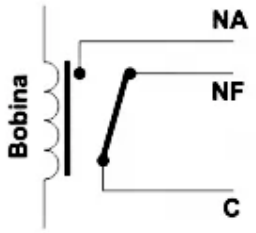
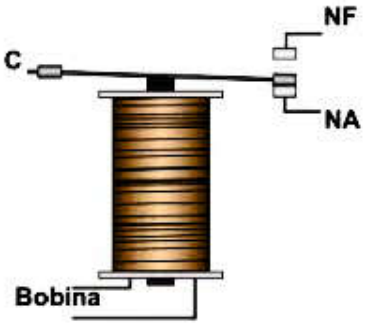
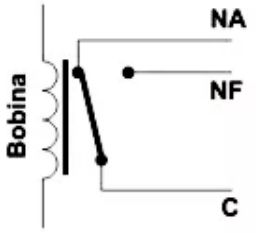
Situação do relé	Ilustração	Esquema elétrico
Desligado		
Acionado		

Figura 11 – Esquema de um relé EMR simples.
Fonte: http://autocorerobotica.blog.br/controlando-lampadas-por-controle-remoto-com-arduino/. Acesso em maio de 2019.

2.3.4. Controle de acionamento de dispositivos elétricos com um módulo relé.

Quando for desejado acionar um dispositivo que utilize um valor de tensão superior aos 5 V ou correntes maiores que 50 mA fornecidos pela placa do Arduino Uno, é necessário utilizar alguma interface que suporte tal carga. Um relé seria uma opção, mas quando se liga e desliga uma bobina, ela induz correntes de valores bem superiores aos limites suportados pelo Arduino e para evitar isso é necessário o uso de circuito de proteção. Esse circuito, não é trivial de ser construído, mas pode ser encontrado pronto num módulo relé, que é um relé montado em uma placa junto com o circuito auxiliar para seu

acionamento, e por isso é frequentemente usado com o Arduino (Embarcados–Arduino 2019).

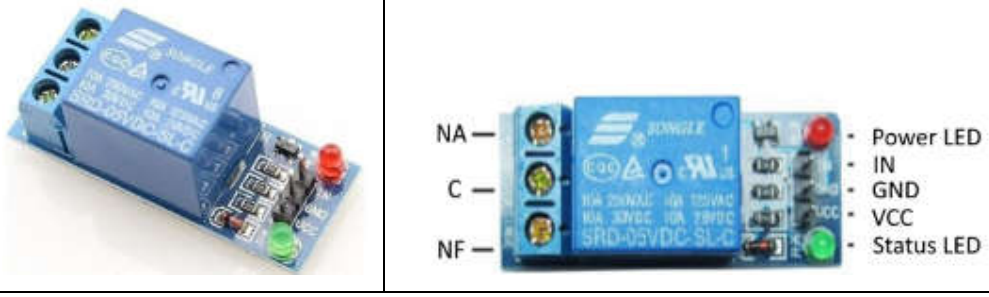


Figura 11 – Módulo relé de 1 canal.
Disponível em http://mekhos.com.br/mekhos/produto/modulo-relé-de-1-canal/. Acesso em 10 de maio de 2019.

Num módulo relé, além dos pinos C, NA e NF, existem os pinos VCC, GND e IN. Os pinos VCC e GND devem ser conectados aos pinos +5 V e GND do Arduino, respectivamente. Já o pino IN deve ser conectado à porta digital do Arduino responsável pelo controle de acionamento (VidaSil-Relé 2019).

Atualmente, são fabricados vários modelos de módulos relés com diversas quantidades de relés, cada um deles podendo controlar um circuito diferente dos outros. Cada circuito controlado por um relé é chamado canal e comercialmente, encontram-se placas com 1 (Figura 11), 2, 4, 8 e até 16 canais (VidaSil-Relé 2019).

Como exemplo, a Figura 12 mostra um esquema elétrico em que o pino 10 do Arduino é conectado ao pino IN do módulo relé para controlá-lo. A lâmpada tem um de seus terminais conectado ao contato NA do relé e o outro ao neutro da rede. Finalmente, o contato C do relé é conectado ao fase da rede. Desta forma, se for usado o algoritmo desenvolvido para o LED externo piscar, a lâmpada ficará acendendo e apagando com intervalo de um segundo.

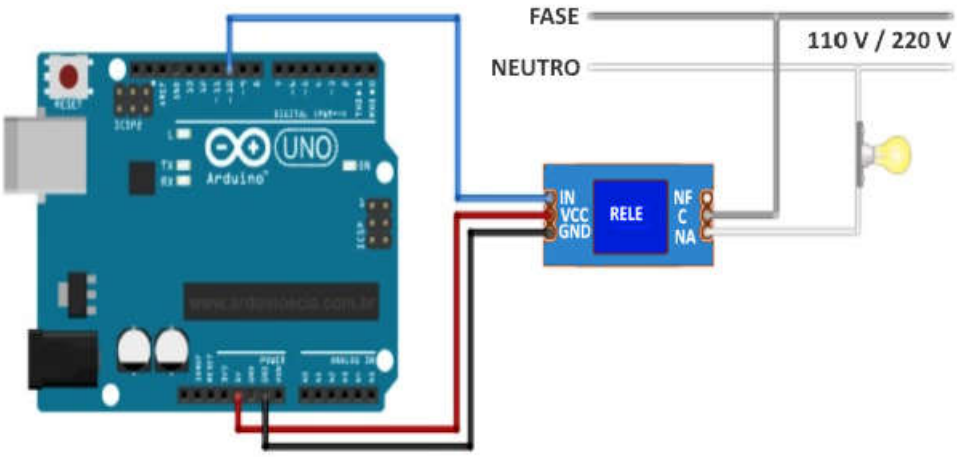


Figura 12 – Esquema de acionamento de uma lâmpada por um módulo relé.
Adaptado de: https://www.arduinoecia.com.br/ligando-uma-lampada-com-modulo-rele-arduino/. Acesso em 20 de maio de 2019.

2.3.5. Controle da recepção de sinal de um sensor de toque.

Existem aplicações em que é necessário converter energia elétrica em outras formas de energia. De forma contrária, em certas aplicações é necessário converter alguma forma de energia em energia elétrica. Esses dispositivos que convertem uma forma de energia em outra são denominados “transdutores”. (Braga 2005, p. 99).

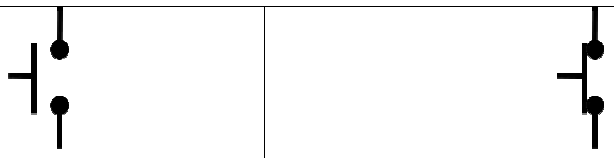
Segundo Banzi (2015, p. 39), um dispositivo interativo eletrônico é capaz de detectar sinais vindos do ambiente usando transdutores eletrônicos chamados sensores, que convertem alguma grandeza física medida no mundo real em sinais elétricos. O microcontrolador do dispositivo processa as informações detectadas seguindo um comportamento previamente determinado por um software e decide como interagir com o mundo usando atuadores, ou seja, transdutores eletrônicos que podem converter um sinal elétrico em uma ação física de reação. Um LED, um relé ou um motor são exemplos de atuadores.

Roggia (2016, p. 23) define um sensor como um dispositivo sensível a uma forma de energia do ambiente (energia luminosa, cinética, sonora,

térmica, entre outras), relacionando informações sobre uma grandeza física que precisa ser medida como luminosidade, temperatura, pressão, vazão, posição e corrente. E de acordo com a natureza do sinal que geram, os sensores podem ser classificados em sensores digitais (discretos) e sensores analógicos (contínuos).

Ainda segundo Roggia (2016, p. 23), os sensores digitais são utilizados para monitorar a ocorrência ou não de um determinado evento. Apresentam como sinal apenas dois estados: O primeiro, quando a corrente pode fluir por ele, é chamado de fechado ou ligado (ON) ou alto (HIGH) ou nível 1; O segundo, quando não há corrente passando por ele, é chamado de aberto ou desligado (OFF) ou nível baixo (LOW) ou nível 0.

Os sensores digitais são de ativação/desativação (ON/OFF) e por isso são chamados comutadores, pois comutam entre os estados aberto (desligado) e fechado (ligado), como mostrado na Figura 13. O comutador alternado é o tipo mais usado, mudando de um estado para outro apenas quando acionado. O Interruptor elétrico e a chave liga/desliga são seus exemplos mais comuns. Já os comutadores do tipo momentâneo, quando pressionados passam do estado aberto para o fechado e retornam ao estado anterior quando a pressão termina. Seus exemplos mais encontrados são o botão da campainha elétrica e o botão de pressão ou push button (Banzi, p. 64, 2015).

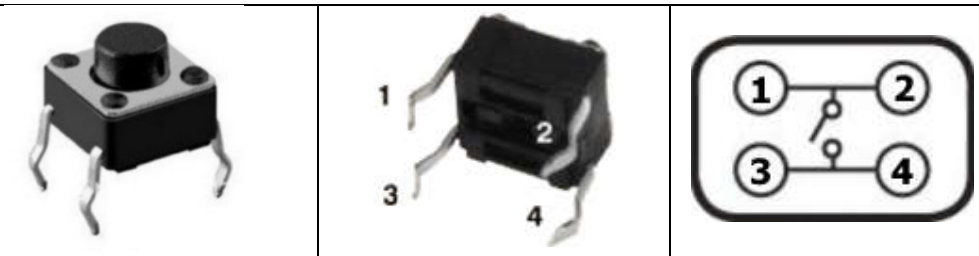


Aberto	**Fechado**

Figura 13 – Simbologia elétrica dos estados de um comutador elétrico.
Adaptado de: http://eletronicaparaartistas.com.br/arduino-5-resistores-de-pull-up/. Acesso em 11-07-2019.

O botão de pressão também é conhecido como botão momentâneo, botão táctil ou botão de apertar (push button). O Tipo mais comum tem quatro pinos e seu aspecto é mostrado nas Figuras 14 (a) e 14 (b) e seu símbolo

elétrico é visto na figura 14 (c). Com o botão não pressionado, só os pinos 1 e 2 estão em contato elétrico entre si, assim como os pinos 3 e 4. Os quatro pinos ficarão conectados eletricamente enquanto o botão for pressionado, (Kamei 2019, p 8).



(a) (b) (c)

Figura 14 – Apresentação e símbolo do botão de pressão de quatro pinos.
Adaptado de: Gueddes 2017, p 44 e Kamei 2019, p 7.

Um sensor digital é comumente conectado ao Arduino em um pino digital configurado como de entrada de dados. Nesse modo o pino é posto em um estado de alta impedância, equivalente a um resistor de 100 MΩ em série com o circuito a ser monitorado e consumindo corrente extremamente pequena. Isso significa que é necessária uma corrente muito baixa para mudar seu estado lógico de um valor para outro. E significa também que um pino configurado como entrada sem nada conectado a ele, ou com fios conectados a ele que não estão conectados a outros circuitos, estará em um estado lógico indefinido e seu nível de tensão apresentará alterações aparentemente aleatórias, captando ruído elétrico do ambiente ou acoplamento capacitivo do estado de um pino próximo (Arduino.cc-DigitalPins 2019).

Uma tentativa não eficiente de evitar que o estado do pino seja indefinido é mostrada nas Figuras 15 (a) e 15 (b), em que o botão não está pressionado e o pino encontra-se com estado indefinido. Com o botão pressionado, o estado do pino passa a ser LOW como é visto nas figuras 15 (c) e HIGH em 15 (d).

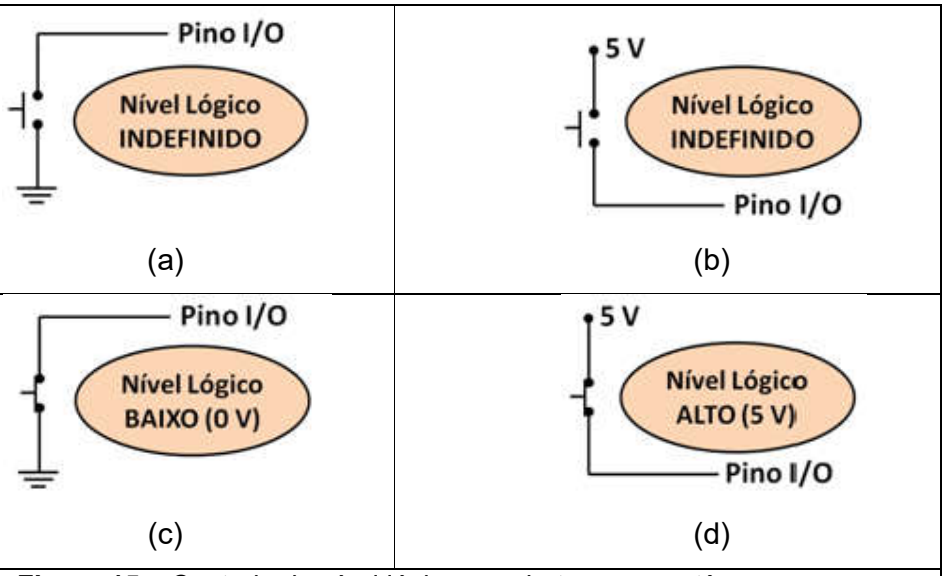


Figura 15 – Controle do nível lógico com botão momentâneo.
Adaptado de: http://eletronicaparaartistas.com.br/arduino-5-resistores-de-pull-up/. Acesso em 11-07-2019.

As conexões com resistores pull up e pull down mostradas na Figura 16 são as soluções mais simples para que o pino nunca esteja em um estado indefinido. Em ambas é usado um resistor geralmente de 10K conectado a um botão de pressão. Na ligação do tipo pulll up, enquanto o botão está livre o estado lógico do pino é HIGH e enquanto o botão está pressionado o pino está no estado LOW. Já na conexão pull down ocorre o contrário, com o pino digital no estado LOW com o botão livre e HIGH com o botão pressionado (Arduino-Guia 2019).

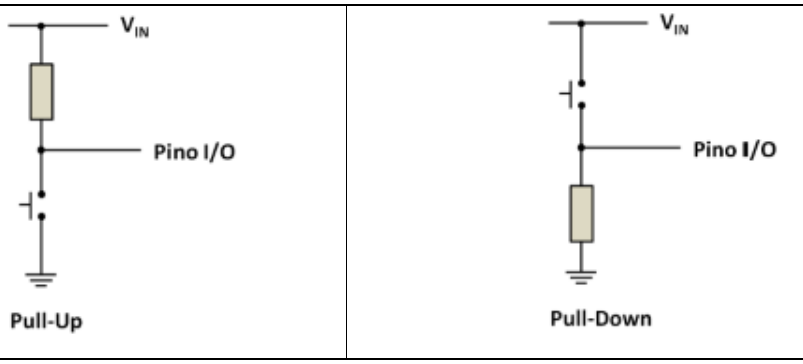


Figura 16 – Resistor de pull up e resistor de pull down.
Adaptado de: https://www.embarcados.com.br/pinos-digitais-do-arduino/. Acesso em 15/07/2019.

Em ARDCC-BUTTON (2019) é apresentado um projeto que usa uma ligação pull-down conectada ao pino digital 7 para controlar o LED builtin e a Figura 19 mostra o esquema elétrico do circuito.

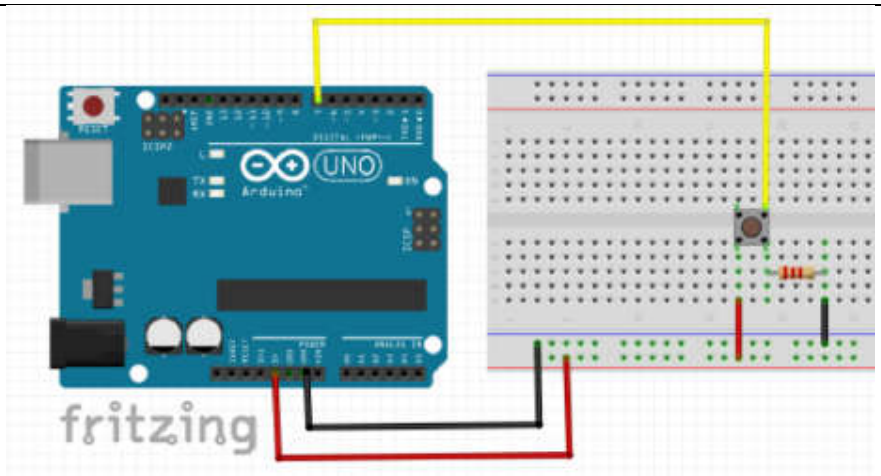


Figura 17 – Botão pull-down para controlar acionamento do LED builtin.
Adaptado de: https://www.arduino.cc/en/tutorial/button.
Acesso em 20-07-2019.

Num sketch do Arduino, se for desejado que o pino digital de número **val_pino** seja usado para fazer a entrada de dados, ele deve ser previamente configurado no setup com a instrução **pinMode**(val_pino, INPUT), e para ler seu valor lógico deve ser usada a instrução **digitalRead**(val_pino). O retorno dessa instrução será **HIGH** ou **LOW**. Em Arduino.cc-DigitalRead (2019) é encontrado um exemplo de código para controle do circuito da figura 17:

```
int ledPin = 13; // Atribui a variável ledPin o valor do pino do LED builtin.
int inPin = 7;     // Atribui a variável inteira inPin o valor do pino digital 7.
int val = 0;       // Atribui a variável inteira val o valor 0.

void setup() {
  pinMode(ledPin, OUTPUT); // Configura o pino 13 como saída de dados
  pinMode(inPin, INPUT);     // Configura o pino 7 como entrada de dados
}

void loop() {
  val = digitalRead(inPin);    // Lê o pino de entrada
  digitalWrite(ledPin, val);    // Aciona o LED com o valor lido do botão
}
```

Esse programa é bem simples: inicialmente são criadas as variáveis inteiras **ledPin** com valor **13**, i**nPin** com valor **7** e **val** com valor **0**. Dentro do setup do sketch **ledPin** é configurada para **saída** de dados e **inPin** para **entrada** de dados. Finalmente, na primeira instrução do loop, **val** recebe constantemente o valor do estado de **inPin** e na segunda instrução esse valor é colocado em led Pin.

Pode-se escrever uma versão simplificada para esse algoritmo, sem uso de variáveis. Nessa versão, o valor do pino 7 é colocado diretamente como valor do pino 13.

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(7, INPUT);
}

void loop() {
  digitalWrite(13, digitalRead(7));
}
```

É importante observar que em um sketch mais extenso, em que varias portas são usadas em muitos comandos, é muito comum a necessidade de troca de uma porta ou de varias delas e o uso de variáveis torna muito mais simples a realização dessas mudanças.

Em ARDCC-Button (2019) e ARDCC-ButtonP (2019) encontram-se outras versões para esse algoritmo que fazem uso de um teste lógico para saber se o botão está pressionado e acender o LED caso o resultado seja verdadeiro ou apagar o LED em caso contrário. A instrução ***if*** serve para verificar se uma condição é verdadeira ou falsa e, segundo Arduino.cc-IF (2019), sua sintaxe é

```
if (condição) [bloco de instruções];
```

A condição deve sempre estar entre parêntesis. Ela é um teste lógico que só possui dois resultados possíveis: ***true*** (verdadeiro) ou ***false*** (falso). O

bloco de instruções dentro dos colchetes somente será executado se o resultado da condição for ***true***.

Caso seja necessário executar alguma instrução somente se o resultado do teste for falso, deve-se, segundo Arduino.cc-Else (2019), usar

```
if (condição) [bloco de instruções]
else [bloco de instruções];
```

O uso de ***if..else*** traz maior controle sobre o fluxo do código quando comparado com a instrução ***if*** sozinha. Permite também que vários testes sejam agrupados. O ***else*** pode realizar outro teste ***if***, criando a possibilidade de executar simultaneamente vários testes mutuamente exclusivos.

.

Assim, uma versão reduzida do sketch com uso de teste lógico, encontrado em ARDCC-ButtonP (2019), pode ser

```
void setup() {
  pinMode(13, OUTPUT;
  pinMode(7, INPUT);
}

void loop() {
  if (digitalRead(7) == HIGH) [digitalWrite(13, HIGH)]
  else [digitalWrite(13, LOW)];
}
```

Em um teste lógico de igualdade é necessário usar dois sinais seguidos de igual, pois na linguagem do Arduino um único sinal de igual é usado para realizar atribuição de variáveis (Arduino.cc-IF 2019).

.

É facil perceber que no projeto que usa uma ligação pull-down para controlar o LED builtin o teste condicional além de ser opcional, traz uma complexidade desnecessária. Porém, existem incontáveis situações em que seu uso é indispensável.

2.3.6. Controle da recepção de sinal de um sensor de luz.

De acordo com Roggia (2016, p. 23), além do sensor digital, existe o sensor analógico que servem para medir a grandeza analógica, ou seja, a grandeza física com valor compreendido entre um mínimo e um máximo de uma faixa contínua de valores. O sensor digital apresenta em sua saída um sinal de tensão, corrente ou resistência proporcional ao valor da grandeza física monitorada.

Existem diversos tipos de sensores analógicos e entre eles, Braga (2005, p. 106) define sensores resistivos como transdutores que apresentam a característica de mudar sua resistência elétrica na presença de um sinal de energia qualquer, normalmente luz, temperatura ou pressão e convertendo-o em um sinal elétrico analógico.

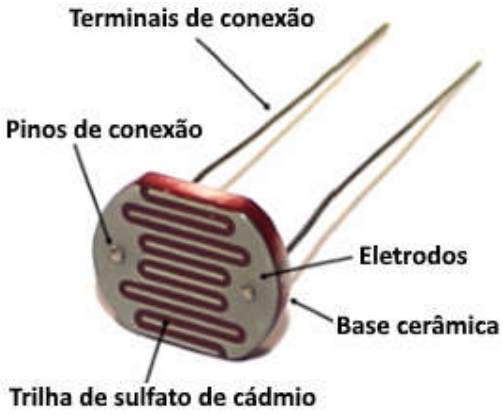
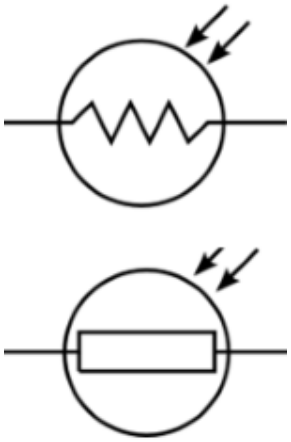
Aspecto e elementos	Símbolos
	

Figura 18 – Símbolos e aspecto de um LDR.
Fonte: http://tot.eng.br/sensor-luz-ldr-arduino/.
Acesso em 21/07/2019.

Segundo Braga (2005, p.107), o tipo mais comum de sensor resistivo de luz é o LDR (Ligh Dependent Resistor), também conhecido como foto-resistor ou ainda célula de sulfeto de cádmio (CdS). Os LDRs têm diferentes formas, tamanhos e valores de resistência, com aspecto mais comum e símbolos mostrados na Figura 18. Um LDR é constituído por uma base de cerâmica

isolante sobre a superfície da qual existem duas placas condutoras que constituem dois eletrodos separados por uma fina trilha ondulada de sulfeto de cádmio. Os dois eletrodos são conectados cada um com os terminais de conexão do LDR.

O sulfeto de cádmio apresenta resistência de milhões de ohms na escuridão total, mas como é sensível a todo espectro de luz visível, o aumento da luminosidade incidente sobre ele, produz a liberação cada vez maior de grande quantidade de portadores de cargas elétricas, diminuindo drasticamente sua resistência, até atingir algumas dezenas de ohms. Assim, ele permite detectar se o ambiente está claro, escuro ou com qualquer valor de luminosidade. É importante observar que o LDR, ao contrário do LED, não tem polaridade Braga (2005, p.107).

Como a resistência do LDR é uma grandeza analógica, para medi-la com o Arduino é necessário usar uma porta analógica. Diferentemente de um pino digital, não é necessário definir um pino analógico como de entrada ou saída e para ler o seu valor é utilizado o comando analogRead(num_pino) em que num_pino é um dos seis pinos analógicos, ou seja, A0, A1, A2, A3, A4 ou A5. Cada pino analógico tem um conversor analógico-digital de 10 bits, o que corresponde a 2^{10} valores inteiros. Assim, um pino analógico ao receber voltagens entre 0 e 5 volts, converte-as em valores inteiros entre 0 (0 V) e 1023 (5 V) (Mcroberts 2011, p. 76).

Embora a principal função dos pinos analógicos nas aplicações em geral seja a leitura de sensores analógicos, os pinos analógicos também possuem toda a funcionalidade de pinos de entrada/saída de uso geral (GPIO), ou dos pinos digitais de 0 a 13. Assim, caso seja preciso de mais pinos digitais para uso geral, e nenhum pino analógico estiver em uso, os pinos analógicos poderão ser usados como se fossem GPIO (Arduino.cc-AnalogInputPins 2019).

Praticamente todos os circuitos sensores usam de algum modo, um divisor de tensão ou divisor de potencial. Um divisor de tensão padrão utiliza dois resistores ligados em série, como mostra a Figura 19 (Netto 2019, p. 52).

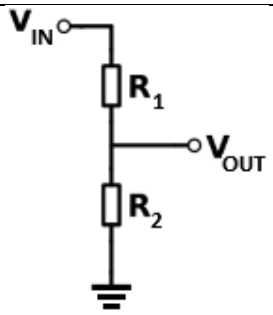


Figura 19 – Circuito Resistivo de um divisor de tensão.
Adaptado de: http://nerdeletrico.blogspot.com/2011/04/circuito-divisor-de-tensao.html. Acesso em 24/07/2019.

Sendo V_1 e V_2 as tensões em R_1 e R_2, a voltagem de entrada V_{IN} será reduzida (dividida) para uma tensão de saída V_{OUT}, assim

$$V_{IN} = V_1 + V_2$$

Se i_1 e i_2 são as correntes em R_1 e R_2, então, $V_1 = i_1 \cdot R_1$ e $V_{OUT} = V_2 = i_2 \cdot R_2$ e admitindo que $i_{IN} = i_1 \approx i_2$, ou seja, que pouca corrente é drenada entre R_1 e R_2, ou que $i_{OUT} \ll i_{IN}$, logo

$$V_{IN} = (R_1 + R_2) \cdot i_{IN}$$

Mas como

$$i_{IN} \approx i_2 = \frac{V_2}{R_2} = \frac{V_{OUT}}{R_2}$$

Logo

$$V_{IN} = (R_1 + R_2) \cdot \frac{V_{OUT}}{R_2}$$

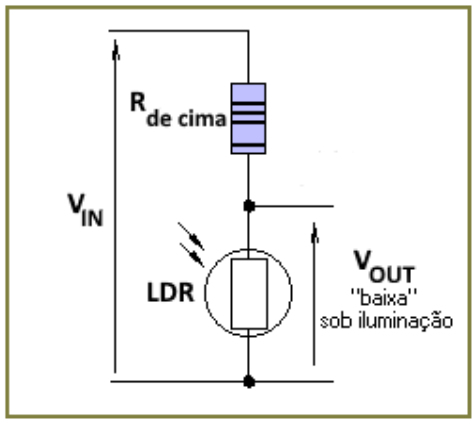
Assim

$$V_{OUT} = \frac{R_2}{R_1 + R_2} V_{IN}$$

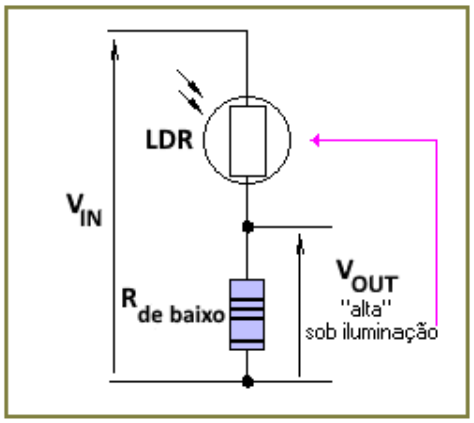
Ou

$$V_{OUT} = \frac{V_{IN}}{1 + R_1/R_2}$$

Mostrando que é a razão entre os valores dos resistores e não os seus valores em si que importam para determinar V_{OUT} (Mcroberts 2011, p. 118).



(a) Saída de tensão BAIXA



(b) Saída de tensão ALTA

Figura 20 – Divisor de tensão com LDR.
Adaptado de: Netto 2019, p. 53-54. Acesso em 04/10/2019.

Substituindo R_1 ou R_2 por um LDR no divisor de tensão, pode-se obter duas situações, como mostra a Figura 20. Na situação (a), obtém-se na saída uma tensão BAIXA quando o LDR está intensamente iluminado e uma tensão ALTA quando o LDR é posto no escuro. Já na situação (b), obtém-se o contrário, ou seja, uma tensão ALTA quando o LDR está intensamente iluminado e uma tensão baixa quando o LDR é posto no escuro. A escolha de uma delas só depende da leitura desejada no sensor (Netto 2019, p. 53-54).

R_1	R_2 (LDR)	V_{OUT}	Claridade
10 kΩ	100 kΩ	4,54 V	Mais escuro
10 kΩ	73 kΩ	4,39 V	25%
10 kΩ	45 kΩ	4,09 V	50%
10 kΩ	28 kΩ	3,68 V	75%
10 kΩ	10 kΩ	2,5 V	Mais claro

Tabela 2 – Valores de V_{OUT} (voltagem de saída) para LDR com 5 V como V_{in} (voltagem de entrada).
Fonte: Mcroberts 2011, p. 119.

Como exemplo, a Tabela 2 mostra as voltagens obtidas conforme a luminosidade sobre o LDR se altera no modo de tensão de saída baixa, usando V_{IN} = 5 V e um LDR com resistência de 100 kΩ no escuro e 10 kΩ quando bem iluminado.

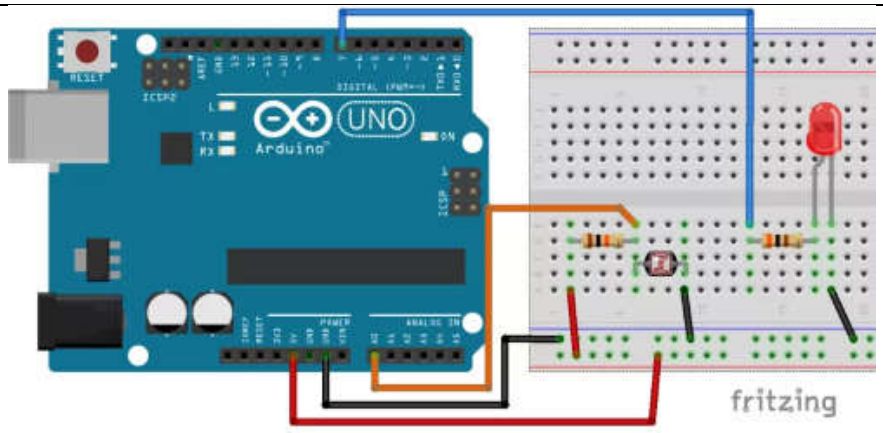


Figura 21 – Controle do acionamento de um LED com um LDR.
Adaptado de: https://portal.vidadesilicio.com.br/sensor-de-luz-com-ldr/. Acesso em 25-07-2019.

.

Em VidaSil-LDR (2019) é apresentado um projeto simples capaz de detectar a luminosidade e acender o LED em um ambiente escuro, e apagar em um ambiente claro. Como mostra a Figura 21, o LED está conectado com a porta digital 7 e em série com o um resistor de 300 Ω. O divisor está conectado com a porta analógica A0, tem saída de tensão baixa sob iluminação e possui além do LDR, um resistor de 10 kΩ. Também foi definido que, uma leitura da porta analógica entre 0 e 500, indica pouca luminosidade e o LED deve ser aceso. Já uma leitura entre 501 e 1023, significa muita luminosidade e o LED deve ser apagado. O sketch do projeto é

```
int ldrValor = 0; //Valor lido do LDR

void setup() {
  pinMode(7,OUTPUT); //define a porta 7 como saída
}

void loop() {
  ldrValor = analogRead(A5); //Lê valor do LDR – valor estará entre 0 e 1023
  if (ldrValor > 500) {
     digitalWrite(ledPin,HIGH); //Liga LED se o valor > 500
```

```
  }
  else {
    digitalWrite(ledPin,LOW); // senão, apaga o led
  }
  delay(100);
}
```

Nesse algoritmo, primeiro é criada a variável inteira ***ldrVal*** para armazenar o valor da leitura do LDR e com valor inicial nulo. A seguir, dentro do setup, a porta digital 7 é configurada para saída de dados. Depois, dentro do loop, a primeira instrução faz a leitura da porta analógica e armazena seu valor em ***ldrVal***. A instrução seguinte realiza o teste lógico: Se o valor de ***ldrVal*** for maior que 500 o LED será aceso, caso contrário será apagado. Por último, ocorre uma pequena espera de 100 ms antes do reinício do loop.

2.3.7. Transmissão via porta serial.

Toda placa Arduino possui pelo menos uma porta serial, também chamada UART ou USART. Essa porta serve para fazer a comunicação entre o Arduino e um computador através da conexão USB. Além disso, como está conectada aos pinos RX (de recepção) e TX (de transmissão), usa níveis lógicos TTL para fazer comunicação com outros dispositivos. Na placa Uno, o RX corresponde ao pino 0 e o TX ao pino 1. Como o upload do computador para o Arduino também é realizado pela conexão USB, durante sua realização nenhum outro dispositivo deve estar conectado ao RX ou ao TX, pois pode ocorrer interferência nessa comunicação, causando falhas no upload (ARDCC-SERIAL 2019).

A plataforma Arduino possui diversas funções para manipulação da comunicação serial. Segundo EMBA-SERIAL (2019) as principais são: Serial.begin(taxa), usada para iniciar a porta serial e definir a taxa de velocidade para transmissão de dados seriais em bits por segundo; Serial.available(), usada para obter o número de bytes (caracteres) disponíveis para leitura na porta serial; Serial.read(), que permite ler o byte mais recente apontado no buffer de entrada da serial; Serial.print(texto), que permite escrever um texto na serial; e Serial.println(texto), que além de escrever um

texto na serial, acrescenta ao seu final o caractere de fim de linha e o caractere de nova linha.

Ainda, segundo EMBA-SERIAL (2019), o monitor serial Integrado a IDE do Arduino é uma das interfaces mais simples para fazer comunicação serial. Seu uso exige que o Arduino esteja devidamente conectado ao computador e para iniciá-lo, basta clicar no botão com ícone de lupa localizado na parte superior da janela e à direita da barra de ferramentas da IDE (Figura 22). Após a abertura do monitor, deve-se usar o botão de ajuste da velocidade de troca de dados, localizado na parte inferior direita da interface, para configurar a velocidade com o mesmo valor definido no sketch do Arduino. Para enviar dados é preciso fazer sua inserção na caixa de envio, localizada na parte superior da janela do monitor e acionar o botão enviar ou a tecla enter. Os dados de retorno da placa para o computador, caso existam, serão exibidos na caixa localizada no centro da janela do monitor.

Figura 22 – Monitor serial integrado a IDE do Arduino.
Fonte: IDE do Arduino.

Em ARDCC-SERIALDR (2019) é encontrado um exemplo de aplicação para monitorar o estado de um botão momentâneo usando a comunicação serial. O circuito é praticamente igual ao mostrado na Figura 17, tendo como única diferença a substituição do pino 7 pelo pino 2.

O sketch sem os comentários para realizar a comunicação serial é

```
int pushButton = 2;

void setup() {
  Serial.begin(9600);
  pinMode(pushButton, INPUT);
}

void loop() {
  int buttonState = digitalRead(pushButton);
  Serial.println(buttonState);
  delay(1);
}
```

Inicialmente é criada a variável inteira ***pushButton*** com valor 2, o do pino de leitura do estado do botão. No setup, a velocidade de comunicação serial entre a placa e o computador é configurada para 9600 bits de dados por segundo. Depois o pino digital com valor definido na variável ***pushButton*** é configurado como uma entrada de dados. No loop, o estado do pino é lido e seu valor colocado na variável inteira ***buttonState***. Em seguida, usando o comando ***Serial.println***, esse valor é enviado para a porta serial. Assim será escrito na caixa central do monitor serial uma sequência de "0"s, um abaixo do outro, se o botão estiver aberto ou de "1"s em caso contrário. Por último, A instrução delay(1) é usada para criar um atraso entre as leituras e gerar estabilidade no programa.

Em outro exemplo, encontrado em ARDCC-LDR&LED (2019), similar ao encontrado em VidaSil-LDR (2019) e cujo circuito elétrico é apresentado na Figura 23, um LDR controla o acionamento de um LED, acendendo-o quando a leitura na porta A5 é maior que 800 e apagando-o em caso contrário. Além disso, escreve na serial o valor encontrado na porta analógica.

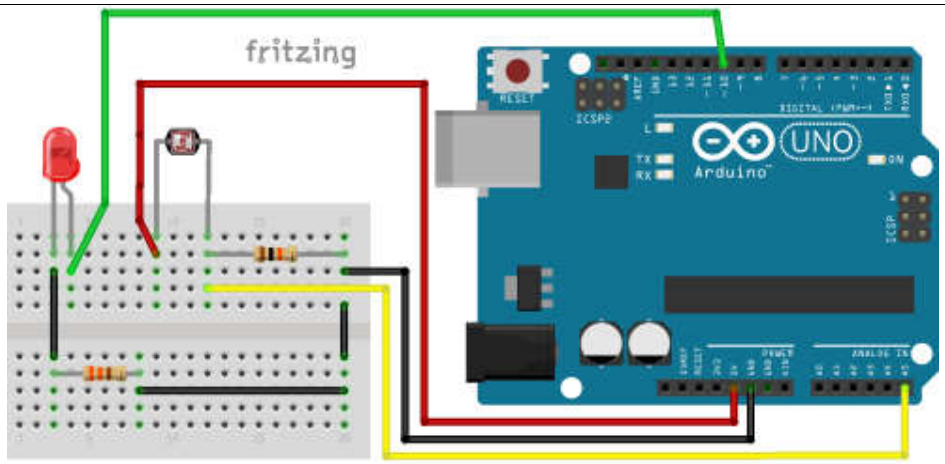


Figura 23 – Circuito para monitoramento do estado de um botão momentâneo.
Adaptado de ARDCC-LDR&LED (2019).
Acesso em 25-07-2019.

O seu sketch sem os comentários para controle do projeto é

```
int LedPin = 10;
int sensorPin = A5;
int sensorValor = 0;

void setup() {
  Serial.begin(9600);
  pinMode(LedPin, OUTPUT);
}

void loop() {
  int sensorValor = analogRead(sensorPin);
  Serial.println(sensorValor);
  if (sensorValor > 800) {
    digitalWrite(LedPin, HIGH);
  }
  else {
    digitalWrite(LedPin, LOW);
  }
}
```

No algoritmo, primeiro é criada a variável inteira ***LedPin,*** com valor 10, o do pino de leitura do estado do botão. Depois é criada a variável inteira **sensorPin**, com valor A5, o pino analógico que será usado. Em seguida é criada a variável inteira ***sensorValor*** para armazenar o estado do sensor e inicialmente com valor nulo. No setup, a velocidade de comunicação serial

entre a placa e o computador é configurada para 9600 bits de dados por segundo. Depois o pino digital com valor definido na variável ***sensorValor*** é configurado como uma entrada de dados. No loop, o estado do pino é lido e seu valor colocado na variável ***sensorValor***. Em seguida, usando o comando ***Serial.println***, esse valor é enviado para a porta serial. Por fim um teste é realizado e o led é aceso se o valor da leitura do sensor for maior que 800 e apagado em caso contrário.

2.3.8. Transmissão via bluetooth.

Bluetooth é uma tecnologia desenvolvida pela Ericsson em 1994. Usa ondas de rádio com frequências entre 2,4 GHz e 2,483 GHz. O objetivo inicial era fazer conexão sem fio (wireless) em substituição aos cabos que conectam a curta distância entre computadores e seus periféricos. Posteriormente, ganhou o suporte de empresas tais como Intel, IBM, Toshiba, Nokia, Lucent, Motorola entre outras e juntas criaram o Bluetooth Special Interest Group (SIG), para ser responsável por escrever as especificações do protocolo e lançar versões atualizadas e novas funcionalidades. Atualmente, suas aplicações incluem o controle de mídia (áudio, vídeo, envio e imagens) e de sistemas de comunicações com diversos dispositivos tais como fones de ouvido, equipamentos de som, TVs, home theathers, computadores, telefones celulares, câmeras digitais, dispositivos de interface humana (mouses, joysticks, teclados), entre outras (TELE-Blue 2019).

Os principais requisitos que norteiam o desenvolvimento do Bluetooth são o baixo consumo de energia, o baixo custo, a capacidade de transmissão de voz, dados e sinalização e por último, uma área de cobertura pequena, entre um e cem metros, sendo por isso classificada como rede sem fio PAN (Personal Area Network), que não precisa de autorização governamental para ser utilizada (TELE-Blue 2019).

Ainda, segundo TELE-Blue (2019), uma rede Blootooth é chamada piconet. Ela é composta por um dispositivo mestre e até sete outros

dispositivos, chamados escravos. O mestre pode solicitar e aceitar conexão (pareamento), enquanto o escravo só pode aceitar. Assim, toda comunicação ocorre somente entre o mestre e os escravos, não existindo comunicação direta entre dispositivos escravos.

Devido à facilidade de uso e preço relativamente baixo, o módulo Bluetooth HC-06 é um dos mais usados com Arduino. O HC-06 apresenta transmissão full duplex, ou seja, pode transmitir e receber ao mesmo tempo. Trabalha apenas em modo escravo com versão 2.0+EDR e o perfil SPP (Serial Port Profile). Tem potência de 2,5 mW e alcance aproximado de 10 metros (ELETRO-Blue 2019).

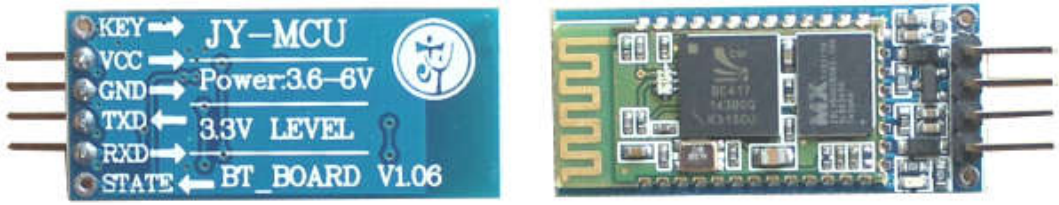


Figura 24 – Módulo HC-06.
Adaptado de https://www.fecegypt.com/uploads/dataSheet/1522750308_bluetooth%20module%20hc06.pdf, p. 1. Acesso em 28/07/2019.

Como mostra a Figura 24, ele apresenta quatro pinos:

- VCC que deve receber tensão de alimentação entre 3,6 V e 6,0 V;
- GND ou pino terra;
- TXD que envia sinais de transmissão com nível de tensão 3,3 V;
- RXD que recebe sinais de transmissão com nível de tensão de 3,3 V.

O Arduino recebe sem problemas um sinal de 3,3 V vindo do pino TXD como nível lógico alto. Já no sentido de comunicação oposto, é necessário converter os 5 V saídos do pino TX do Arduino nos 3,3 V suportados pelo pino RXD. Para fazer essa redução a maneira mais simples e barata é usar um divisor de tensão em que uma das resistências seja o dobro da outra, pois essa é a razão aproximada entre a tensão de 3,3 V no pino RXD e a queda de tensão entre os pinos do Arduino e do HC-06:

$$\frac{3{,}3\ \text{V}}{5\ \text{V} - 3{,}3\ \text{V}} = \frac{3{,}3}{1{,}7} \approx \frac{2}{1}$$

Por isso são usados resistores de 1 kΩ e 2 kΩ em ELETRO-Blue (2019), ou de 7,5 kΩ e 15 kΩ em WUSTL-HC-06 (2019), ou ainda de 10 kΩ e 20 kΩ em ENGIN-HC-06 (2019).

A Figura 25 mostra um circuito que usa o HC-06 como interface de conexão bluetootch para enviar comandos de um smartphone para o Arduino ativar e desativar um led conectado em sua porta digital 13.

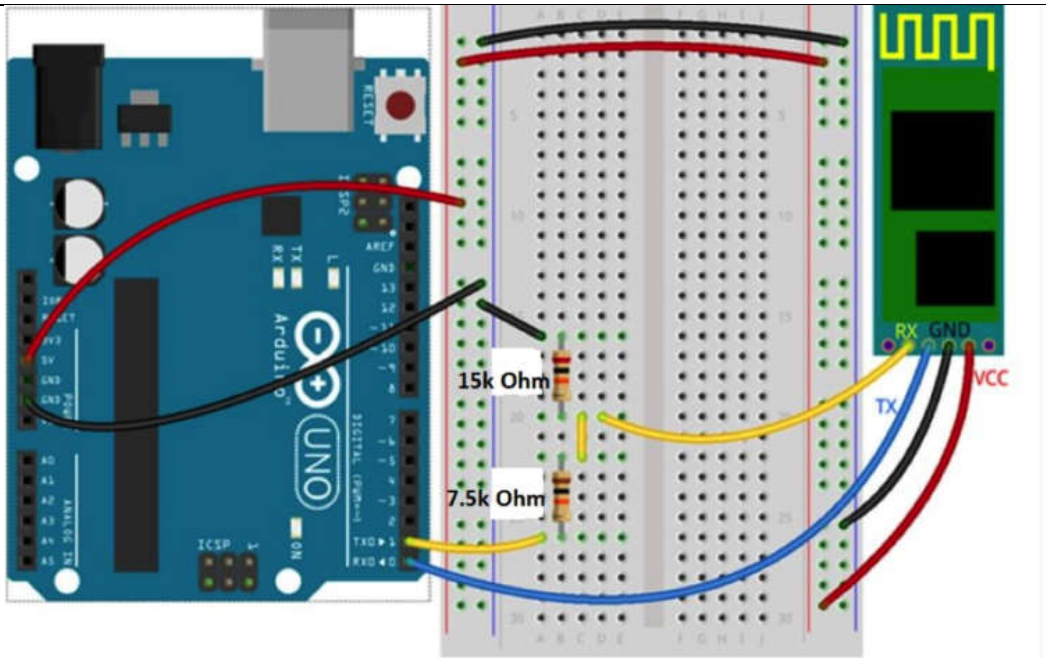


Figura 25 – Circuito para acionamento da placa bluetootch HC-06.
Fonte: https://classes.engineering.wustl.edu/ese205/core/index.php?title=Bluetooth_Module_(HC-06)_%2B_Arduino. Acesso em 30/07/2019.

Em FILIP-HC-05 (2019) encontra-se o sketch do projeto:

```
char buf;

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  while(Serial.available() > 0) {
    buf = Serial.read();
    if (buf == 'L') {
      digitalWrite(13, HIGH);
    }
    if (buf == 'D') {
      digitalWrite(13, LOW);
    }
  }
}
```

A primeira linha do algoritmo cria a variável ***buf*** que irá armazenar os valores lidos na porta serial. No setup, o pino 13 é configurado para saída e a taxa de comunicação da serial estipulada em 9600. No loop, a primitiva ***while*** verifica se existe algum valor a ser lido na porta serial e se existir coloca esse valor na variável ***buf***. Em seguida, no primeiro ***if***, verifica se ela é igual a ‘L’, colocando o pino 13 em HIGH e acendendo o LED onboard. Já o segundo if verifica se buf tem valor ‘D’, colocando o pino 13 em LOW desligando o LED em caso verdadeiro.

Como no Arduino Uno as comunicações com o computador e com o módulo bluetooth utilizam os mesmos pinos, somente após o final do upload do sketch é que devem ser conectados os pinos TX e RX do Arduino aos pinos RX e TX do HC-06. Para evitar isso, pode-se incluir no sketch uma chamada a biblioteca externa SoftwareSerial, nativa da IDE Arduino, para simular por software uma porta serial em dois outros pinos do Arduino e manter livre sua serial em hardware para ser utilizada no debug e gravação do código a ser executado (VIDASIL-HC06 2020).

Quando o HC-06 está energizado, seu LED pisca repetidamente, indicando que o mesmo está aguardando conexão com um dispositivo mestre. Nesse momento, utilizando algum app para gerenciamento de conexão bluetooth instalado num celular, ao acessar a lista de varredura de dispositivos bluetooth no app, deve-se encontrar o nome HC-06. Ao selecioná-lo, caso seja a primeira tentativa de conexão, será solicitada uma senha (que é 1234 por padrão). Com o pareamento estabelecido, o LED para de piscar indicando que a placa está apta a transmitir e receber dados (VIDASIL-HC06 2020).

3. Usando o app Serial Bluetooth Terminal

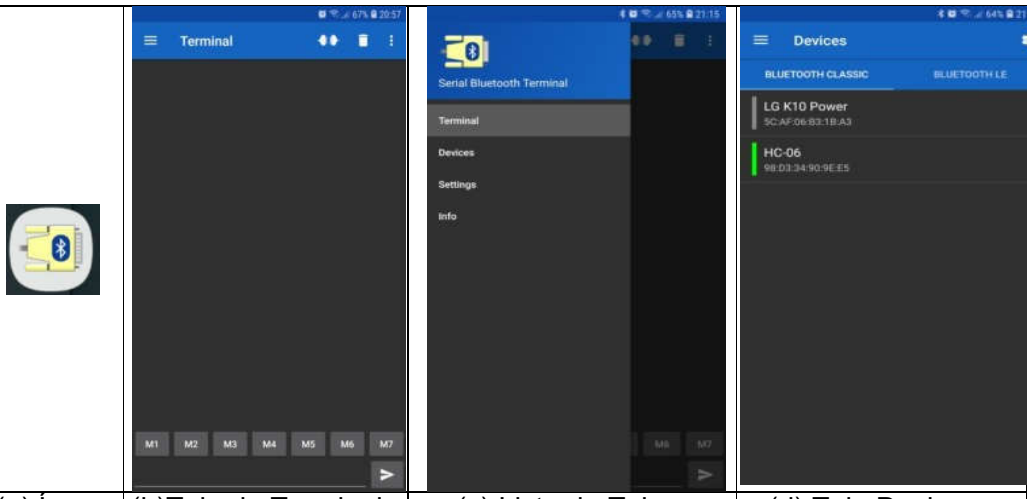
O 'Serial Bluetooth Terminal' é um aplicativo (app) para Android completamente gratuito. Foi desenvolvido por Kai Morich e seu download pode ser feito pelo Google Play em

https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

Utiliza uma interface bem simples e intuitiva tipo terminal/console orientado a linha para comunicação de dispositivo Bluetooth pareado a um smartfone Android.

Ele suporta comunicação com diferentes dispositivos bluetooth e utiliza diferentes versões do bluetooth, mas no presente trabalho foi apenas usada a comunicação com a placa HC-06 com a versão do Bluetooth clássico.

Após sua instalação no smartfone o ícone na Figura 26(a) aparece.



(a) Ícone	(b)Tela do Terminal	(c) Lista de Telas	(d) Tela Devices

Figura 26 – Ícone e principais telas do Serial Bluetooth Terminal
Fonte: app Serial Bluetooth Terminal

Ao abrir o programa entra-se inicialmente na tela do **Terminal**, como mostra a Figura 26(b).

	Mudança de telas
	Conectar/desconectar dispositivo bluetooth
	Limpar lista de mensagens
	Configurações

Tabela 3 – Botões de comando do Serial Bluetooth Terminal
Fonte: app Serial Bluetooth Terminal

A parte superior da tela contém quatro botões de comandos cuja função encontra-se na Tabela 3. Na região central da tela encontra-se uma lista com horário de mensagens, comandos e dados enviados/recebidos e que inicialmente está vazia. Finalmente na parte inferior encontra-se a linha de comandos a serem enviados e acima dela um conjunto de sete botões de macros.

A primeira tarefa a ser feita é parear o smartfone com a placa HC-06. Para isso deve-se acionar o botão de mudança de telas. Após isso, aparece uma lista das quatro telas disponíveis, sendo uma delas **Devices**, como indica a Figura 26(c). Ao acionar essa opção é mostrada uma tela com duas abas, uma com a relação de dispositivos disponíveis com versão bluetooth clássica e a outra com a versão bluetootch LE, como mostra a Figura 26(d).

Se for a primeira vez de uso do dispositivo selecionado será solicitada uma senha. Para o caso da HC-06, a senha padrão é 1234. Na região de Mensagens, será apresentado um aviso indicando o sucesso no pareamento ou que um erro ocorreu em caso contrário.

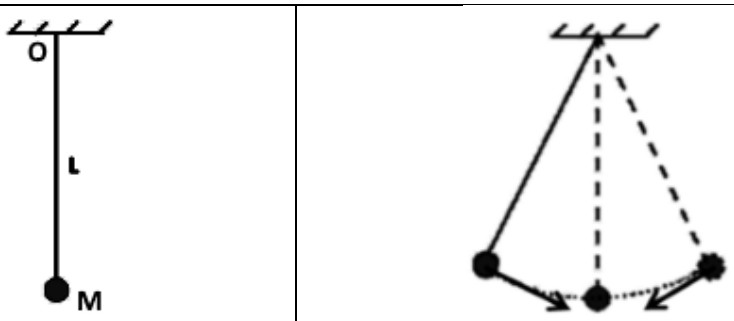
Após realizar o pareamento é necessário tornar a conexão ativa com o botão conectar e só depois é possível transmitir e receber dados. Nova mensagem informa o sucesso da conexão ou uma falha na sua tentativa.

Com a conexão ativa é possível enviar diversos comandos e receber inúmeras informações. Após algum tempo a tela enche-se de dados que não são mais úteis e assim pode-se usar o botão de limpeza de mensagens.

Pode-se automatizar o envio de comandos e dados através de macros configurando um ou mais dos sete botões de macros existentes. Para isto, basta pressionar um deles por dois ou três segundos até abrir uma tela de edição de macro. Insira o nome da macro, ou se preferir deixe o nome padrão, insira o valor desejado e selecione o modo de edição (Texto, Hexadecimal ou Texto de múltiplas linhas) e a ação (Enviar ou inserir) e confirme as alterações no botão superior direito.

4. Construção do pêndulo simples

Um pêndulo simples foi constituído por um objeto de massa M concentrada e suspenso em um ponto fixo O por um fio inextensível e de comprimento L. Quando afastado de sua posição de equilíbrio e abandonado, o corpo oscila em torno desta posição com um período de tempo T.



(a) Pêndulo em repouso | (b) Pêndulo em movimento oscilatório em torno da posição de equilíbrio

Figura 27 – Pêndulo simples
Fonte: do próprio autor

Componentes do pêndulo.

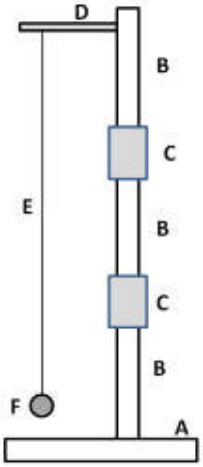


Figura 28 – Componentes do pêndulo simples
Fonte: do próprio autor

A figura 28 apresenta os componentes do pendulo usado no projeto e é composto pó:

A. Uma base de madeira com dimensões aproximadas de 20 cm x 30 cm x 2 cm e com um furo circular de raio 10 cm.
B. Três hastes de madeira com 40 cm de comprimento e raio de 10 mm.
C. Dois pedaços de cano PVC de 20 mm e 15 cm de comprimento.
D. Uma haste metálica com 15 cm de comprimento com fixador de 20 mm.
E. 1,5 m de fio de naylon fino.
F. Uma esfera de aço com aproximadamente 1 cm de diâmetro.

Preparação do conjunto:

1. Para facilitar o transporte, a haste de madeira do pêndulo é constituída por três pedaços que são facilmente encaixados um ao outro usando os tubos de pvc. Deve-se primeiro fixar um deles na base de madeira e só depois fazer o encaixe dos outros dois.
2. Amarrar num extremo da linha de nylon a esfera de aço. Uma forma simples de amarração é envolver a esfera em um pedaço de pano ou plástico e amarrá-lo com o nylon.
3. Amarrar o outro extremo da linha de nylon na haste metálica.
4. Encaixar a haste metálica na haste de madeira a uma altura de aproximadamente 50 cm.
5. Colocar o aparato de medição sobre a base de madeira e diretamente abaixo da esfera metálica.

5. Dispositivo de medição do período do pêndulo simples

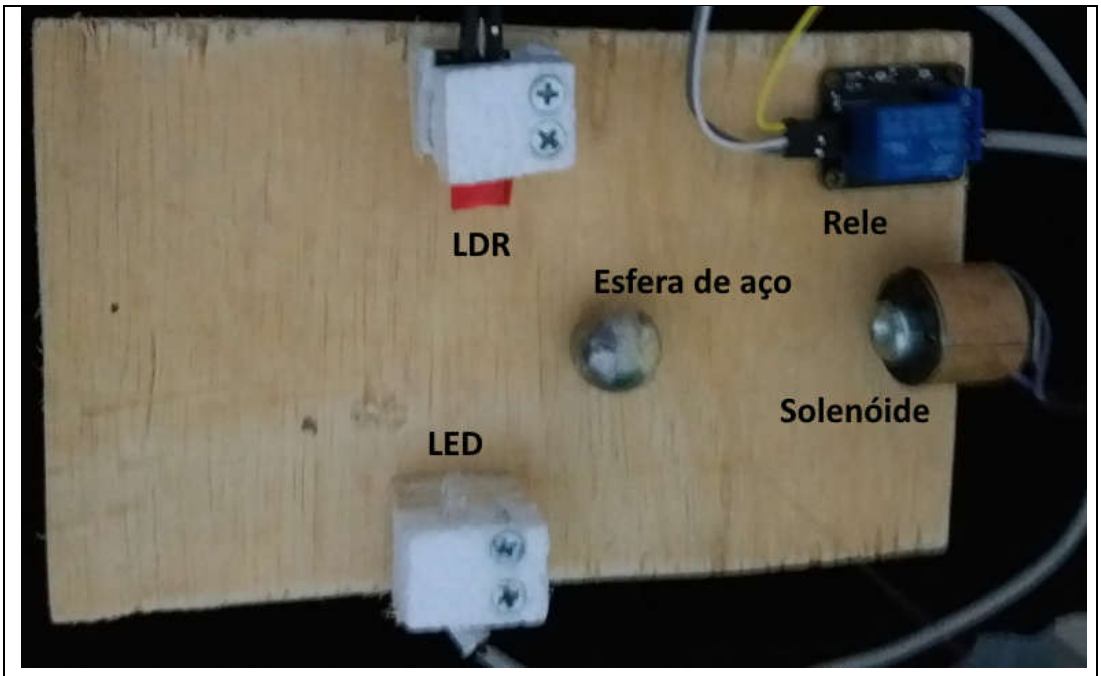


Figura 29 – Vista superior do dispositivo de medição de tempo.
Fonte: do próprio autor

Para medir o período do pêndulo, como mostra a figura 29, o LED aponta seu feixe luminoso para o LDR enquanto a esfera do pêndulo oscila em um plano vertical perpendicular ao feixe de luz. Quando ela estiver entre eles, a luminosidade sobre o LDR diminui e gera uma variação de sinal que é enviado ao Arduino. Assim o intervalo de tempo entre duas variações do sinal corresponde a metade do período do pêndulo.

O solenóide é usado para liberar a esfera de aço e iniciar a oscilação, o que permite evitar à soltura dela com os dedos. As vantagens do emprego do solenóide são que a esfera é liberada sempre da mesma posição (sempre com mesmo ângulo inicial), com velocidade praticamente nula e com a corda de nylon do pêndulo esticada o que produz medidas mais precisas

A lista de material necessário para construir o dispositivo de medição é:

a) Uma placa Arduino Uno.
b) Um computador com a IDE do Arduino instalada.
c) Um cabo USB para fazer a conexão Computador Arduino
d) Uma protoboard
e) Um LED de alto brilho
f) Um Resistor de 470 Ω para ligação com LED
g) Um LDR
h) Um Resistor de 10 kΩ para ligação com LDR
i) Um módulo Bluetooth HC-06
j) Um Resistor de 470 Ω para o divisor de tensão do HC-06
k) Um resistor de 560 Ω para o divisor de tensão do HC-06
l) Um resistor de 220 Ω para o divisor de tensão do HC-06
m) Um solenóide 12 V para carga de 0,5 kg
n) Um módulo rele para acionamento do solenóide
o) Fonte de alimentação 12 V para o solenóide
p) Fonte de alimentação 9 V para o Arduino
q) Jumps de conexão
r) 1 metro de cabo com 1 mm de diâmetro
s) Base de madeira com aproximadamente 20 cm x 12 cm
t) Pedaços de isopor para fixar o LED e o LDR

E apresenta do ponto de vista eletrônico quatro circuitos:

a) O circuito de acionamento do LED:

É similar ao circuito da Figura 21. Porém, com o pino digital 12 do Arduino sendo usado para controle do LED e com um resistor de 470 Ω no lugar do resistor de 220 Ω.

b) O circuito de leitura do LDR:

É similar ao circuito da Figura 21.

c) O circuito de acionamento do solenóide:

É similar ao circuito da Figura 12, com a substituição da lâmpada pelo solenóide energizado por 12 V e controlado pelo pino digital 11 do Arduino.

Foram usadas fontes de alimentação diferentes para o Arduino e para o solenóide com objetivo de evitar interferência entre eles.

d) O circuito de conexão com o módulo Bluetooth HC-06:

É similar ao circuito da Figura 25, com a colocação de dois resistores de 560 Ω e 220 Ω ligados em série no lugar do resistor de 15 kΩ e da substituição do resistor de 7,5 kΩ por outro com 470 Ω.

6. Aplicação da proposta didática

A aplicação da proposta didática ocorreu na Escola Técnica Estadual Cícero Dias, com uma turma da 2ª série do ensino médio, no transcorrer da 2ª unidade do ano de 2019 e em um total de 4 aulas de 50 minutos.

Adaptando a proposta desenvolvida em ROSA (2012), o experimento foi dividido em quatro etapas:

a) 1ª Aula de Matemática: O professor de matemática fez uma atividade de revisão de logaritmos em uma de suas aulas.

b) 1ª Aula de Física: Etapa Pré-Experimental.

   - Uma pequena introdução teórica foi apresentada com base do livro texto.

- A Etapa 1 da Ficha de experimentos foi apresentada aos educandos, momento em que eles foram divididos em 11 grupos com uma média de quatro educandos por grupo.

- O dispositivo de medição do período foi apresentado aos educandos.

- O app Serial Bluetooth Terminal foi apresentado aos educandos

c) 2ª Aula de Física: Etapa Experimental.

Para tornar a medição mais ágil, um grupo de educandos ficou responsável por coordenar e orientar os demais grupos.

Como havia muitos grupos e apenas um dispositivo de medida, foi estabelecido que cada grupo determinasse o período para um único comprimento do pêndulo e o resultado compartilhado com os demais grupos para preenchimento da Tabela 1.

d) 3ª Aula de Física: Etapa Pós-Experimental.

A Etapa 3 de análise dos dados foi executada e apenas 8 dos 10 grupos chegaram a um resultado próximo do teórico. Dois grupos erraram em alguma passagem matemática e só conseguiram finalizar a atividade quando receberam orientação externa.

Ao final da atividade foi questionada qual a maior dificuldade na realização da atividade e a resposta unânime foi trabalhar com logarítmo.

7. Considerações finais.

A realização as aulas teóricas, as aulas de experimentos exige do professor um preparo adicional em relação às aulas teóricas, bem como um gasto de tempo maior com para a preparação e aplicação.

Mas é impressionante a capacidade de um experimento tem de atrair a atenção do educando e induzi-lo ao pensamento crítico e a uma aprendizagem mais intensa.

Na realização dessa atividade experimental com mais de quarenta educandos, foi marcante o alto grau de interesse e aprendizagem dos educandos participantes.

Apêndice A – Ficha de experimento do pêndulo simples

Escola Técnica Estadual Cícero Dias Disciplina: Física Turma: ____ Data: ___/___/_____	Componentes: _____ _____ _____ _____

Ficha de experimento

Atividade:

A expressão de dependência do período com o comprimento de um pêndulo simples.

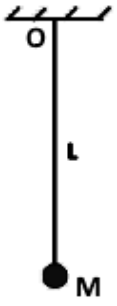
1. ETAPA PRÉ-EXPERIMENTAL

Organização

1.1. Objetivo

Determinar e expressão que relaciona o período com o comprimento de um pêndulo simples a partir da medição experimental dos períodos de pêndulos simples com diversos comprimentos.

1.2. Introdução

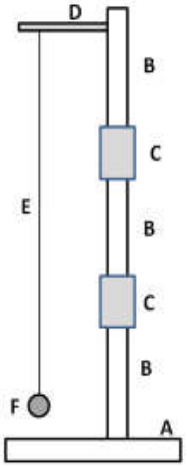


Pêndulo em repouso

Pêndulo em movimento oscilatório em torno da posição de equilíbrio

Um pêndulo simples é constituído por um objeto de massa **M** concentrada e suspenso em um ponto fixo **O** por um fio inextensível e de comprimento **L**. Quando afastado de sua posição de equilíbrio e abandonado, o corpo oscila em torno desta posição com um período de tempo **T**.

1.3. Material utilizado no dispositivo do pêndulo



G. Uma base de madeira com dimensões aproximadas de 20 cm x 30 cm x 2 cm e com um furo circular de raio 10 cm.
H. Três hastes de madeira com 40 cm de comprimento e raio de 10 mm.
I. Dois pedaços de cano PVC de 20 mm e 15 cm de comprimento.
J. Uma haste metálica com 15 cm de comprimento com fixador de 20 mm.
K. 1,5 m de fio de naylon fino.
L. Uma esfera de aço com aproximadamente 1 cm de diâmetro.

1.4. Outros materiais

A. Aparato de medição de tempo composto por placa arduíno, sensor de luz, solenóide, LED e placa Bluetooth.
B. Smartfone (ou tablet, ou computador) com apps de comunicação serial por Bluetooth e planilha eletrônica.
C. Fita métrica com 1,5 m.
D. Lápis.
E. Borracha.
F. Régua.

1.4. Montagem

Preparação da parte mecânica:

A. Para facilitar o transporte, a haste de madeira do pêndulo é constituída por três pedaços que são facilmente encaixados um ao outro usando os tubos de pvc. Deve-se primeiro fixar um deles na base de madeira e só depois fazer o encaixe dos outros dois.

B. Amarrar num extremo da linha de nylon a esfera de aço. Uma forma simples de amarração é envolver a esfera em um pedaço de pano ou plástico e amarrá-lo com o nylon.
C. Amarrar o outro extremo da linha de nylon na haste metálica.
D. Encaixar a haste metálica na haste de madeira a uma altura de aproximadamente 50 cm.
E. Colocar o aparato de medição sobre a base de madeira e diretamente abaixo da esfera metálica.

1.5. Hipótese

O período do pêndulo é proporcional a uma potência de seu comprimento:

$$\mathbf{T = c \,.\, L^{b}}$$

Então

$$\log T = \log\left(c \,.\, L^{b}\right)$$

$$\log T = \log c + \log L^{b}$$

$$\log T = \log c + b.\log L$$

Fazendo $y = \log T$
$a = \log c$
$x = \log L$

Encontramos $\mathbf{y = a + b.x}$

Ou seja, o logaritmo do período é uma função do primeiro grau do logaritmo do comprimento do pêndulo.

2. ETAPA EXPERIMENTAL

Coleta de dados

2.1. Procedimento de coleta de dados

A. Ajuste a distância L entre a haste metálica e o centro da esfera para 50 cm.
B. Afastar a esfera da vertical aproximadamente 15 cm, tendo o cuidado de deixar a linha de nylon esticada;
C. Acionar o smartfone para liberar a esfera e medir o valor do período T_I da oscilação;
D. Repetir outras três vezes para medida de T_{II}, T_{III} e T_{IV}, registrando na tabela 2;
E. Repetir a medida do período para todos tamanhos de L na tabela 2.
F. Calcular o período médio da oscilação para cada comprimento usando

2.2. Tabela de registro de dados

$$T = \frac{T_I + T_{II} + T_{III} + T_{IV}}{4}$$

Tabela 1					
L (m)	T_I (s)	T_{II}(s)	T_{III} (s)	T_{IV} (s)	T (s)
0,50					
0,60					
0,70					
0,80					
0,90					
1,00					

OBS: Trabalhar com apenas duas casas decimais

3. ETAPA PÓS-EXPERIMENTAL

Análise de dados

3.1. Conversão das medidas para seus logaritmos

Como visto em 1.5:

Hipótese: $\mathbf{T = c \,.\, L^b}$

Então
$$\log T = \log\left(c \,.\, L^b\right)$$
$$\log T = \log c + \log L^b$$
$$\log T = \log c + b.\log L$$

Fazendo
$$y = \log T$$
$$a = \log c$$
$$x = \log L$$

Encontramos $\mathbf{y = a + b.x}$

Para achar a e b, preencha a tabela 2 seguindo as orientações a seguir:

A. Transcreva para a coluna 3 todos os períodos para cada comprimento do pêndulo da tabela 1;
B. Calcule $x_i = \log L_i$ para cada comprimento e registre o resultado na coluna 4;
C. Calcule $y_i = \log T_i$ para cada período e registre o resultado na coluna 5;

Tabela 2

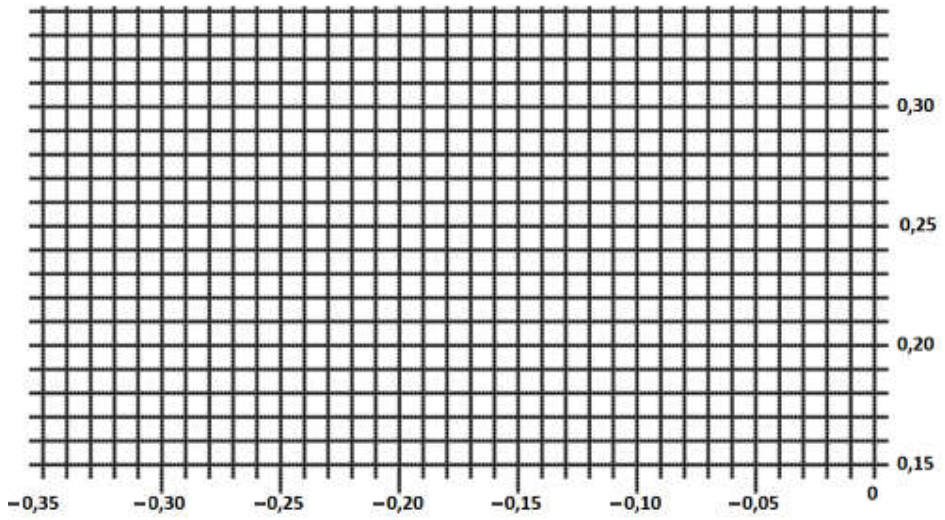
i	L_i (m)	T_i(s)	x_i	y_i
1	0,50			
2	0,60			
3	0,70			
4	0,80			
5	0,90			
6	1,00			

3.2. Fazendo o gráfico x por y

A. Transcreva os valores de x e y para o Gráfico 1;

B. Trace a melhor reta para os pontos no Gráfico 1;

Gráfico 1



3.3. Obtendo a expressão de dependência a partir do gráfico

A. Como $\mathbf{y} = \mathbf{a} + \mathbf{b}.\mathbf{x}$, o valor de $\mathbf{a}$ é igual ao valor de $\mathbf{y}$ para $\mathbf{x}$ = 0, assim

$$a = \_\_\_\_\_$$

B. Selecione dois pontos $(x_I,\ y_I)$ e $(x_{II},\ y_{II})$ sobre a reta e calcule o valor de a

$$b = \frac{y_{II} - y_I}{x_{II} - x_I} = \_\_\_\_\_ = \_\_\_\_\_$$

C. Calcule o valor de $\boldsymbol{c}$ usando

$$a = \log_{10} c \quad \rightarrow \quad c = 10^a = \_\_\_\_\_$$

D. Finalmente escreva a relação

$$T = c\,.\,L^b \quad \rightarrow \quad \mathbf{T} = \_\_\_\_\_.\mathbf{L}$$

3.4. Comparação Teoria x Prática

A expressão teórica é:

$$T = 2\pi \sqrt{\frac{L}{g}}$$

$$T = \frac{2\pi}{\sqrt{g}}\sqrt{L}$$

$$T \approx \frac{2 \cdot 3{,}14}{\sqrt{9{,}8}}\sqrt{L}$$

$$\mathbf{T \approx 2{,}007 . L^{0{,}5}}$$

A expressão experimental é:

$$\mathbf{T = \_\_\_\_\_ . L}$$

Apêndice B – Registros fotográficos do projeto

Figura: Dispositivo de medição do período do pêndulo

Fonte: do próprio autor.

Figura: Educandos durante medição do período do pêndulo

Fonte: do próprio autor.

Figura: Educandas durante medição do período do pêndulo

Fonte: do próprio autor.

Referências Bibliográficas

ARDCC-BOTAO. Pushbutton. Disponível em https://www.arduino.cc/en/Tutorial/Pushbutton/. Acesso em 20-09-2019.

ARDCC-BUTTON. ARDUINO TUTORIAL. Button. 2015. Disponível em: <httpshttps://www.arduino.cc/en/tutorial/button>. Acesso em: 20 jul. 2019.

ARDCC-LDR&LED. LDR & LED Light. Disponível em https://create.arduino.cc/projecthub/Kenpoca_Dias/ldr-led-light-1147c3. Acesso em 20 jul. 2019.

ARDCC-SERIAL. Serial. Disponível em https://www.arduino.cc/reference/en/language/functions/ communication/serial/. Acesso em 20-09-2019.

ARDCC-SERIALDR. Digital Read Serial. Disponível em: https://www.arduino.cc/en/Tutorial/ DigitalReadSerial. Acesso em 25-09-2019.

ARDUINO.CC-ANALOGINPUTPINS. Analog Input Pins. Disponível em https://www.arduino.cc/en/Tutorial/ AnalogInputPins. Acesso em 09-09-2019.

ARDUINO.CC-BLINKINGLED. Arduino Blinking LED. Disponível em https://www.arduino.cc/en/Tutorial-0007/BlinkingLED. Acesso em 01/04/2019.

ARDUINO.CC-CLONE. Send in the clones. Disponível em https://blog.arduino.cc/2013/07/10/send-in-the-clones/. Acesso em 05/03/2019.

ARDUINO.CC-CONST. Arduino Constantes. Disponível em https://www.arduino.cc/reference/en/language/variables/constants/constants/. Acesso em 06/04/2019.

ARDUINO.CC-DIGITALPINS. Digital Pins. Disponível em https://www.arduino.cc/en/Tutorial/DigitalPins. Acesso em 10/10/2019.

ARDUINO.CC-DIGITALREAD. digitalRead(). Disponível em https://www.arduino.cc/reference/en/language/ functions/digital-io/digitalread/. Acesso em 10/10/2019.

ARDUINO.CC-ELSE. Else. Disponível em https://www.arduino.cc/reference/en/language/structure/ control-structure/else/. Acesso em 16-10-2019.

ARDUINO.CC-HOME. ARDUINO Home. Disponível em: https://www.arduino.cc. Acesso em: 03/02/2019.

ARDUINO.CC-IF. IF. Disponível em https://www.arduino.cc/reference/en/language/structure/ control-structure/if/. Acesso em 16-10-2019.

ARDUINO.CC-INTRO. Arduino Introduction. Disponível em: https://www.arduino.cc/en/Guide/ Introduction. Acesso em 10/02/2019.

ARDUINO.CC-LANREF. Arduino Language Reference. Disponível em https://www.arduino.cc/ reference/en/. Acesso em 06/04/2019.

ARDUINO.CC-PRODU. Arduino Products. Disponível em: https://www.arduino.cc/en/Main/Products. Acesso em 04/04/2019.

ARDUINO.CC-START. Getting Started with Arduino and Genuino products. Disponivel em: https://www.arduino.cc/en/Guide/HomePage. Acesso em 04/04/2019.

ARDUINO.CC-UNO. Disponível em: https://store.arduino.cc/usa/arduino-uno-rev3. Acesso em 04/04/2019.

ARDUINO-DESC. Arduino – Descrição da placa. Disponível em 70https://www.tutorialspoint.com/arduino/arduino_board_description.htm. Acesso em 04/04/2019.

ARDUINO-GUIA. O que é Arduino – O guia definitivo. Disponível em https://blog.silvatronics.com.
br/arduino-o-guia-definitivo/. Acesso em 11-10-2019.

ARDUINO-ORIGEM. Arduino - A Origem. Disponível em http://www.natalmakers.com/
arduino-a-origem/. Acesso em 15-3-2019.

ARGENTO, Heloisa. TEORIA CONSTRUTIVISTA. Disponível em http://penta3.ufrgs.br/midiasedu/modulo11/etapa2/construtivismo.pdf. Acesso em 25/02/2020.

ATMEGA-DATASHEET.
ATmega808/1608/3208/4808 – 32-Pin. Disponível em http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega808-1608-3208-4808-32-Pin-40002017C.pdf. Acesso em 13-10-2019.

BANZI, Massimo. Getting Started with Arduino. First Edition. U.S.A: O'Reilly Media, 2008.

BANZI, Massimo; Michael Shiloh. Primeiros Passos com Arduino. 2ª edição. São Paulo: Novatec Editora Ltda, 2015.

BRAGA, C Newton. Eletrônica básica para mecatrônica. São Paulo: Saber, 2005.

BUILD IT-ARDUINO. Build It. Share It. Profit. Can Open Source Hardware Work? . Disponível em
https://www.wired.com/2008/10/ff-openmanufacturing/
Acesso em 09/02/2019.

BUTTON. Disponível em https://www.arduino.cc/en/tutorial/button. Acesso em 20-09-2019.

CARVALHAES, Claudio G. I; Suppes, Patrick I. O cálculo de alta precisão do período do pêndulo simples. Rev. Bras. Ensino Física. vol. 31 nº. 2. São Paulo: Abril/Junho 2009.

EG PROJETS. Stepper Motor Speed and Direction Control Using Arduino and Bluetooth HC-06 Module through an Android App. 2019. Disponível em: <https://www.engineersgarage.com/arduino/stepper-motor-controlled-with-hc06-bluetooth-module/>. Acesso em: 10 fev. 2020.

ELETRO-BLUE. Módulos Bluetooth HC05 e HC06 para comunicação com dispositivos móveis com Arduino. Disponível em https://blog.eletrogate.com/modulos-bluetooth-hc05-e-hc06-para-comunicacao-com-dispositivos-moveis-com-arduino/. Acesso em 17-09-2019

EMBA-SERIAL. Arduino - Comunicação Serial. Disponível em: https://www.embarcados.com.br/
arduino-comunicacao-serial/. Acesso em 10-09-2019.

EMBARCADOS–ARDUINO. Acionamento de uma lâmpada com Arduino. Disponível em https://www.embarcados.com.br/acionamento-de-uma-lampada-com-arduino/. Acesso em 15/05/2019.
Gueddes, Mark. Manual de projetos Arduino. São Paulo: Novatec, 2017.

.

FILIP-HC-05. Como usar o Arduino Bluetooth HC-05 em modo mestre. 2019. Disponível em: <https://www.filipeflop.com/blog/tutorial-arduino-bluetooth-hc-05-mestre/>. Acesso em: 10 fev. 2020.

HC-06. Arduino and HC-06 (ZS-040). Disponível em http://www.martyncurrey.com/ arduino-and-hc-06-zs-040/. Acesso em 17-09-2019.

KAMEI, Camila A. N.; Barbosa, Lucas F. L. Introdução a Programação Programando com Intel Galileo. Experimento 1. Recife: UFPE - Centro de Informática - Grupo de Engenharia da Computação. Disponível em https://www.cin.ufpe.br/~lflb/Tutorial%201%20-20conceitos_basicos_galileo.pdf Acesso em 14-10-2019.

LED-ONBOARD. Arduino Onboard embedded LED's Their Function and Color. Disponível em http://electricarena.blogspot.com/2015/01/arduino-onboard-embedded-leds-function.html. Acesso em 08-05-2019.

MARQUES, Gil da Costa. Mecânica Clássica para Professores. São Paulo: Editora Universidade de São Paulo, 2014.

MARTINS, A. F. P. (2005). Ensino de ciências: desafios à formação de professores. Revista Educação Em Questão, 23(9), 53-65. Recuperado de https://periodicos.ufrn.br/educacaoemquestao/article/view/8342.
Acesso em 23-02-2020.

MCROBERTS, Michael. Arduino Básico. São Paulo: Novatec, 2011.

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

MOREIRA, Marco Antonio. Teorias da aprendizagem. São Paulo: EPU, 1999

NETTO, Luiz Ferraz. Circuitos Eletrônicos. Disponível em http://netsaber.com.br/apostilas/apostilas/38.doc. Acesso em 14-10-2019.

OLIVEIRA, Cláudio Luís Vieira. Aprenda Arduino - Uma abordagem prática. – Duque de Caixas: Katzen Editora, 2018.

PALMA, Daniel A.; Oliveira, Maria Stella N.; Jesus, Vitor L. B. de. As representações integrais e o ensino de matemática nas licenciaturas em física: Um exemplo simples baseado em observações experimentais. Revista Perspectivas da Ciência e Tecnologia vol. 1, nº. 1. Rio de Janeiro: jan-jun 2009.

PCN+. Orientações Educacionais Complementares aos Parâmetros Curriculares Nacionais. Ciências da Natureza, Matemática e suas Tecnologias. Disponível em http://portal.mec.gov.br/seb/arquivos/pdf/CienciasNatureza.pdf. Acesso em 23/02/2020.

RELÉ. Como funciona um relé de estado sólido? Disponível em https://blog.rhmateriaiseletricos.com.br/como-funciona-um-relé-de-estado-solido/. Acesso em 15/05/2019.

ROGGIA, Leandro. Automação industrial / Leandro Roggia, Rodrigo Cardozo Fuentes. – Santa Maria: Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria, Rede e-Tec Brasil, 2016.

ROSA, Cleci T. Werner da; ROSA, Álvaro Becker da. Aulas experimentais na perspectiva construtivista. Física na Escola, v. 13, n. 1, 2012. Disponível em http://www1.fisica.org.br/fne/phocadownload/Vol13-Num1/a021.pdf. Acesso em 22/02/2020.

SERWAY, Raymond A.; Jewett Jr, John W. Física para cientistas e engenheiros, volume 2: oscilações, ondas e termodinâmica. – São Paulo: Cengage Learning, 2011.

TELE-BLUE. Bluetooth: O que é? Disponível em https://www.teleco.com.br/tutoriais/ tutorialblue/pagina_1.asp. Acesso em 17-09-2019.

UNESP-ELETROÍMÃ. Eletroímã. Disponível em http://www2.fc.unesp.br/experimentosdefisica/ele11. htm. Acesso em 14/09/2019.

USP-ELETROÍMÃ.
Aplicações do 1º fenômeno eletromagnético. Disponível em http://efisica.if.usp.br/eletricidade/basico/campo_corrente/aplic_prim_fenom_eletromag/. Acesso em 14/09/2019.

VIDASIL-LDR. Sensor de Luz – Aprendendo a usar o LDR com Arduino.
Disponível em https://portal.vidadesilicio.com.br/sensor-de-luz-com-ldr/.
Acesso em 15-10-2019.

VIDASIL-HC06. MÓDULO BLUETOOTH HC-06 E HC-05 – ARDUINO. Disponível em: <https://portal.vidadesilicio.com.br/modulo-bluetooth-hc-05-e-hc-06/>. Acesso em: 27 fev. 2020.

VIDASIL-RELÉ. Módulo relé – Acionando cargas com Arduino. Disponível em https://portal.vidadesilicio.com.br/modulo-rele-com-arduino/.
Acesso em 15-05-2019.

WIRING. Disponível em http://wiring.org.co/. Acesso em 06-09-2019.

WUSTL-HC-06. Bluetooth Module (HC-06) + Arduino.. Disponível em: <https://classes.engineering.wustl.edu/ese205/core/index.php?title=Bluetooth_Module_(HC-06)_%2B_Arduino&oldid=12261>. Acesso em: 10 fev. 2019.

YOUNG, Hugh D. Física III: eletromagnetismo – São Paulo: Addison Wesley, 2009.