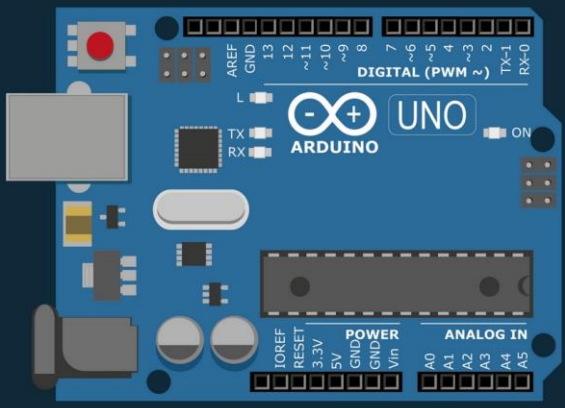
PRODUTO EDUCACIONAL

ARDUINO & ENSINO DE FÍSICA

Automação de práticas experimentais



OSÉIAS MOURÃO

Arduino & Ensino de Física

Automação de práticas experimentais

Oséias Mourão

1ª Edição

Capa
Design Canva

Dados internacionais de catalogação na publicação (CIP)

M929a Mourão, Oseias
Arduino & ensino de Física: automação de práticas experimentais / Oseias Mourão. – Tianguá: Clube dos Autores, 2018.
116 p.

1. Arduino. 2. Automação. 3. Ensino-Física. I. Mourão, Oseias. II Título.

CDD-530

www.fisicarduino.com

Dedico esta obra a minha linda esposa, Fátima, que foi muito compreensível enquanto me dedicava horas em experimentos e pesquisa. Porém, não poderia deixar de tributar a alguém muito especial, que surgiu na minha vida exatamente no período de criação deste Produto, minha filha Alice.

Sumário

Agradecimentos

Agradeço aos professores do Mestrado Nacional Profissional em Ensino de Física, integrantes do polo 56 UVA/IFCE, por terem contribuído no processo de formação profissional e orientação na elaboração deste Produto Educacional.

Aos meus alunos pelo apoio e contribuição na aplicação das atividades deste material educacional.

À Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e à SBF (Sociedade Brasileira de Física) pelo apoio financeiro e Iniciativa em promover o aprimoramento em um nível de pós-graduação *stricto sensu* a profissionais da educação básica.

À minha família, pelo apoio imprescindível.

Introdução

Este trabalho consiste em um produto educacional contendo propostas didáticas para o ensino de Física, sustentadas na automação de práticas experimentais, especialmente na aquisição de dados por meio de sensores, atuadores e da interface de prototipagem Arduino, a ferramenta mais bem-sucedidas para a educação STEAM[1] .

Este material sintetiza as principais funcionalidades do Arduino e disponibiliza 11 experimentos de Física com Arduino, abrangendo assuntos de Cinemática, Dinâmica, Termometria, Hidrostática e Eletricidade. Alguns experimentos inéditos e outros já conhecidos, porem reformulados, com a inserção de automatização na coleta de dados.

Cada capítulo contém um desses assuntos, e apresenta uma breve introdução, objetivos do experimento, fundamentação teórica, lista de materiais necessários para realização do experimento e sua automação, esquema de montagem, o código, instruções sobre a execução e coleta de dados; e finalmente, um tópico de questionamento e levantamento de hipóteses.

A estruturação da metodologia deste produto baseia-se em alguns pressupostos das teorias da aprendizagem, e principalmente nas ideologias de John Dewey, que pressupõem um ambiente educacional multidisciplinar na resolução de problemas legítimos.

[1] Recursos metodológicos citados em https://www.arduino.cc/en/Main/AboutUs

1. A plataforma Arduino

O Arduino pode ser considerado uma multiplataforma *open source*[2], pois constitui uma variedade de software, hardware e documentação livre, cujo modelo de desenvolvimento permite que *hobistas*, inventores, pesquisadores, professores e estudantes possam desenvolver e aplicar livremente suas ideias.

O projeto Arduino

A proposta original do projeto Arduino de Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis, iniciou em Ivrea, comuna italiana, no Interaction Design Institute, início dos anos 2000. Inspirados pelo projeto *Processing*, linguagem que ensina programação no cenário das artes visuais, e do projeto de Hernando Barragan relacionado a placas eletrônicas -*Wiring board*[3], conseguiram tornar mais acessível a prototipagem de projetos envolvendo eletrônica e programação. Assim, as primeiras placas começaram a ser usadas em 2005, a fim de auxiliar estudantes com pouco ou nenhum conhecimento em eletrônica no desenvolvimento de produtos tecnológicos.

A interface Arduino consiste em uma placa única com um microcontrolador ATMEL e estrutura integrada para entrada analógica e entrada/saída digital. Todos esses recursos de

[2] A definição do Open Source foi criada pela Open Source Iniciative (OSI) baseado no texto da *Debian Free Software Guidelines*, desinando um programa de código aberto e que este deve garantir: Distribuição livre, Código fonte, Trabalhos Derivados, entre outras garantias relacionadas à licenças.

[3] Termo em inglês que se refere a placa de circuito impresso. O modelo apresentado na tese de Hernando Barragan influenciou a implantação do projeto Arduino. A tese pode ser encontrada em http:// http://people.interactionivrea.org/h.barragan/thesis/thesis_low_res.pdf.

hardware podem ser controlados por meio de uma linguagem de programação fundamentada em C/C++.

Há uma variedade de produtos similares (**Figura 1**), mas os produtos Arduino oficiais são apresentados a seguir:

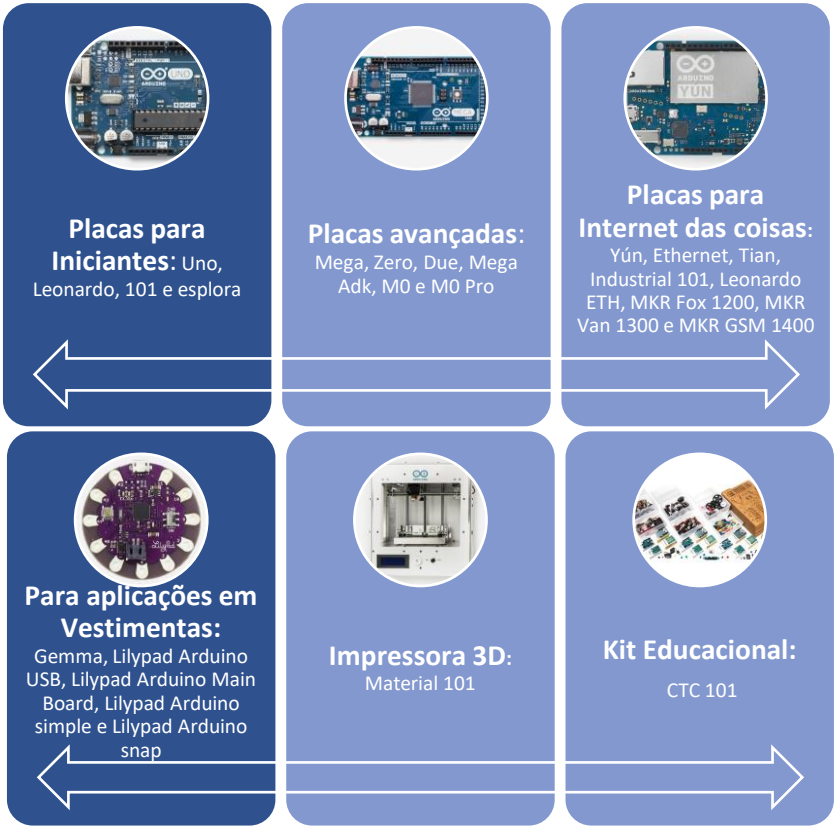


Figura 1 - Apresentação dos produtos oficiais Arduino, note a diversidade dos produtos, que variam desde placas simples para iniciantes à interfaces mais avançadas.

Fonte: arduino.cc/en/Main/Products.

Com o sucesso do Arduino, versões diferentes foram desenvolvidas com finalidades distintas. As placas podem ter

suas funcionalidades estendidas com a implementação de *shields*, recursos extras, como o *Ethernet shield V2* que possibilita a conexão do seu Arduino com a internet.

Principais componentes e acessórios

Para o desenvolvimento de experimentos alguns *Shields* podem ser montados em uma *protoboard* – veja a descrição sobre *protoboard* no **Quadro 1** – é o que potencializa a função destas montagens, os *shields* são os componentes e acessórios constituintes. Neste trabalho destacam-se os componentes apresentados no **Quadro 1**:

Quadro 1 - Há uma grande variedade de componentes eletrônicos no mercado, neste quadro é exibido apenas os componentes suas características e funções utilizados neste Produto Educacional.

Componente & acessórios	Características e funções
Buzzer	Dispositivo piezelétrico útil na geração de um sinal sonoro.
Fios *jumpers*	Fios condutores flexíveis com extremidades rígidas para conexão entre componentes, *protoboard* e pino do Arduino.
fotodiodo	Componente semicondutor conversor de luz em corrente elétrica. Os fotodiodos usados neste projeto são usados como sensores sensíveis a luz infravermelha.
Imã de neodímio	Compostos de neodímio, ferro e boro, tais ímãs possuem campo magnético bastante intenso, muito útil em experimentos de física para estudo do

	campo magnético e fenômenos de indução eletromagnética.
Laser	Dispositivo emissor com resistor integrado pronto para ser ligado no Arduino, na tensão de 5 volts. O tipo usado nos experimentos do Produto educacional possui potência de 5mW.
LDR	Resistor dependente de luz. Muito usado como sensor de luminosidade, a que é sensível à luz visível. Esse componente tem sua resistividade elétrica alterada em função da intensidade luminosa, portanto pode ser usado como experimento de física, testando a 2ª lei de Ohm.
LED	Diodo emissor de Luz. Esse componente tem uso recorrente no Produto educacional como indicador de algum estado ou como fonte luminosa para algum sensor. Para este último caso optou-se, convenientemente, usar o LED Infravermelho. Notar que o LED possui polaridade definida, deixando a corrente fluir apenas em um único sentido; logo, para determinar a polaridade basta notar a diferença de tamanho nos terminais do LED, o maior possui polaridade positiva e o menor negativa.
Potenciômetro	Consiste em um dispositivo com resistência elétrica variável. Muito útil para o controle de da tensão elétrica em certos trechos de um circuito.

Protoboard	É uma placa de ensaio pronta para realizar conexões entre diversos componentes eletrônicos, os furos em uma mesma coluna na *protoboard* estão conectados por um condutor. Nas extremidades da placa existem linhas que estão conectadas, ideal para extensão da fonte de energia.
Push-Button	Chave tátil para alterar estado lógico de um pino digital.
Resistor	Componente com resistência elétrica específica. Nos experimentos apresentados no produto educacional os resistores a estão especificados quanto a resistência, conforme o código de cores.
Sensor de temperatura LM35	Este sensor apresenta uma boa precisão para medida de temperatura. Um dos terminais mantem tensão elétrica linear proporcional à temperatura, especificamente, 10mV para cada grau Celsius.
Termistor	Consiste em um resistor cuja resistência elétrica é determinada pela temperatura onde se encontra.
Transistor de efeito hall	Componente transdutor capaz de detectar a polaridade magnética, podendo reverter o estado lógico de um pino de digital no Arduino em função do polo magnético detectado. No PE foi utilizado um transistor de efeito Hall denominado S41.

Fonte: do autor.

Esses componentes contribuem significativamente para a captação de dados, facilitando a análise e comprovação de algumas teorias física.

Funcionalidades e configurações

A utilização de um Arduino envolve basicamente controle dos pinos disponíveis na placa, cuja quantidade depende da versão em uso. No Arduino UNO (**Figura 2**), o mais popular, há 14 pinos digitais (0 -13) e 6 entradas analógicas (A0 -A5).

Figura 2- Imagem de um Arduino UNO. Na imagem, os pinos digitais estão localizados na região superior, os pinos analógicos na região inferior. Na região inferior esquerda, o bloco de pinos contém um pino de 5 volts e outro de 3,5 volts, além dos pinos destacados pela faixa branca, onde contem 2 pinos GND, correspondentes ao *ground*, ou seja, um aterramento.

Fonte: arduino.cc.

Os pinos digitais podem ser configurados como entrada ou saída digital, operando em um nível lógico alto ou baixo, que na prática correspondem a 5 e 0 volts, respectivamente. A configuração de um pino digital como entrada é útil quando deseja-se registrar alguma informação externa, portanto, para medidas sem ruído, o pino deve ser montado com um resistor de *pull-down*, que define o pino como *baixo*; ou com um resistor de *pull-up*, que define o pino como *alto*. Como exemplo para o

primeiro caso (*pull-down*), se uma chave tátil conectada for pressionada o estado lógico do pino muda para alto; no caso *pull-up* ao pressionar a chave o pino mudará para *baixo*. Vejamos (**Figura 3**):

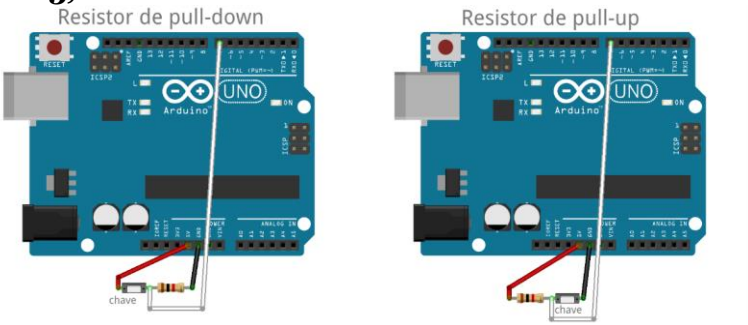


Figura 3- Imagem de um Arduino UNO em uma montagem de resistor de *pull-down* e resistor de *pull-up*. Ambos com chave tátil.

Fonte: Imagem do autor extraída e editada no Fritzing.[4]

As configurações dos pinos digitais são feitas na programação e normalmente são usadas as expressões HIGH e LOW para indicar os estados lógicos respectivos a 0 e 5 volts. Se um determinado pino está configurado como saída é possível acender e apagar um LED, por exemplo.

Os pinos correspondentes as entradas analógicas operam em uma faixa de valores de 0 a 1023, em razão da resolução do conversor analógico-digital do Arduino UNO que é de 10 bits, correspondente a faixa de tensão de 0 a 5 volts, previamente definido no Arduino. Ou seja, a faixa de 0 a 5V será representada por 1024 valores discretos.

Normalmente os sensores são ligados aos pinos analógicos. Tais sensores, de acordo com suas características, fornecem valores analógicos de 0 a 5 volts que é convertido pelo

[4] O *Fritzing* é um sistema de *software open-source* ideal para o design de hardware eletrônico, programação e com o banco de imagens predefinidas ou editáveis. Sistema muito útil para simular montagem de protótipos envolvendo Arduino e seus componentes, em função da quantidade de imagens relacionadas. Mais informação sobre o software e download em: *fritizing.org*.

Arduino para valores discretos até 1023; isso pode ser muito útil na automação.

Alguns pinos digitais podem ser usados ainda como saídas analógicas, por meio da geração de uma onda quadrada - PWM. Neste caso o resultado analógico é simulado pelo controle da largura da onda, que significa controlar o percentual de tempo que a onda permanecerá em um estado lógico alto, podendo gerar valores de 0 a 255; em termos da tensão elétrica, esses valores no pino podem ser de 0 a 5 volts. Logo, o PWM pode ser usado para controlar a velocidade de um motor, a posição de um servo ou mesmo o brilho de um LED. Para descobrir qual pino pode ser usado como PWM basta identificar o pino que contém o caractere ~.

- **O uso do Arduino IDE**

Um dos *softwares* que permite a programação é o Arduino IDE, que facilita a criação e compilação de códigos, normalmente denominado de *sketches*, e carregamento na memória flash do Arduino. Para o desenvolvimento do produto educacional foi utilizado o Arduino 1.8.4. É possível instalar e executar este software nos sistemas operacionais Windows, Mac e Linux. O download do programa pode ser feito no site https://www.arduino.cc/en/Main/Software. Além dessa opção há a opção de trabalhar online com o Arduino Web Editor, que pode ser acessado no mesmo site.

Após instalação do software, sua utilização estará quase pronta, necessitando de uma breve configuração, que consiste na especificação do tipo de Arduino utilizado e a porta COM. Assim, sabendo do tipo de Arduino, basta acessar a opção *ferramentas* na barra de menu e selecionar a placa correta, veja o procedimento na **Figura 4**:

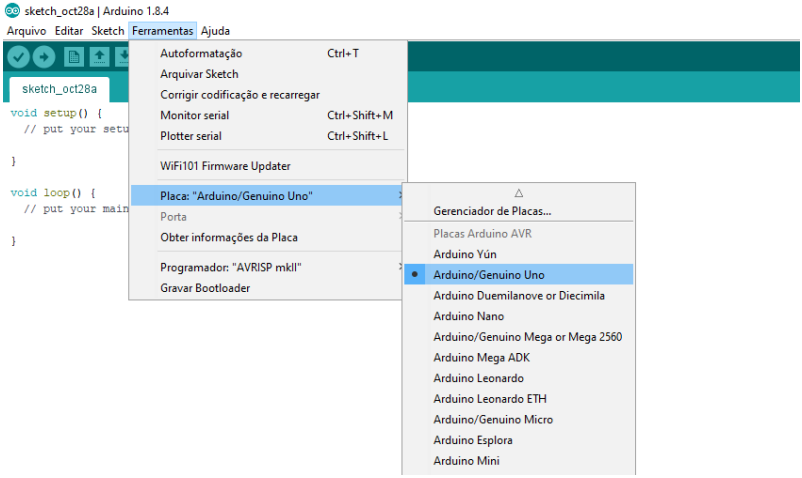


Figura 4 - Software IDE Arduino 1.8.4. Após a instalação do software é necessário especificar a versão da placa que será utilizada, opção disponível em ferramentas, no menu do programa.

Fonte: do autor

Após conectar o Arduino no computador será possível selecionar a porta COM apropriada, normalmente a porta COM3. Essa alteração pode ser feita também na opção *ferramentas*.

Os sketches do Arduino são divididos em três partes básicas, estrutura, variáveis e funções.

A estrutura apresentada (ver **Quadro 2**) ao criar um novo sketch é a seguinte:

Quadro 2 - Estrutura de um novo sketch novo.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fonte: Arduino.cc

As estruturas `setup()` e `loop()` são imprescindíveis na programação. Quando o Arduino for iniciado a função `setup()` será executada uma única vez, iniciando variáveis, definindo as configurações dos pinos digitais ou incluindo bibliotecas. Logo em seguida a função `loop()` inicia e se repete executando seu conteúdo. Há algumas estruturas de controle que, inclusive serão recorrentes neste trabalho, como o: `if` e `if...else`, por exemplo. Quanto à sintaxe, é exigido no final de uma linha o "ponto e vírgula" e para inserção de comentários usa-se `//` para uma linha ou `/* */` para comentários multilinhas. Pode se usar operadores aritméticos, booleanos, comparativos ou de atribuição, conforme necessidade.

As variáveis são necessárias na programação para guardar valores, que devem ser declaradas no programa com a indicação do tipo de variável. Por exemplo, os valores lidos pela entrada analógica `A0`, devido a um sensor conectado neste pino, podem ser armazenados em uma variável, cuja denominação é atribuída pelo programador, por exemplo:

```
int valorSensor=analogRead(A0);
```

Neste caso, sugestivamente, os valores do sensor ficam armazenados na variável denominada `valorSensor`. O tipo de variável é especificado pela expressão int. Essa variável é sempre usada para armazenar valores inteiros desde `-32.768` a `32.767`. para armazenar valores maiores é preciso usar outros tipos de variáveis, como a variável `long`. No site oficial [5]do Arduino há uma lista com os tipos de dados e exemplos.

As funções também são usadas constantemente e são conjunto de instruções já definidas na arquitetura do software, ou que podem ser criadas pelo programador como um bloco contendo códigos específicos. No exemplo anterior aparece a função `analogRead()` que já vem previamente definida no

[5] http://playground.arduino.cc/Portugues/Referencia

software; essa função faz a leitura da entrada analógica especificada nos parênteses.

Após criar um *sketch* ou colar um já pronto pode ser feito um teste que verifica se há algum erro de edição, clicando no "v" na barra de menu, conforme **Figura 5**.

Figura 5 - Software IDE Arduino 1.8.4. Ao editar algum sketch verifique se não há algum erro, clicando no botão conforme indicação da imagem, ou use as teclas *crtl+R*. A verificação consiste também na compilação do *sketch*, para a placa Arduino.

Fonte: do autor.

Carregar o *sketch* – o código – é bem simples, o software do Arduino faz quase todo o serviço, necessitando apenas que o programador faça o carregamento à placa, o termo mais comum para esse procedimento é o *upload*. Consiste em clicar no botão contendo a seta orientada para a direita, na barra de menu, ou simplesmente pressionando as teclas *crtl+U* no teclado do computador.

Após essa verificação, caso não haja erro, basta carregar o código na placa Arduino devidamente conectada no computador. A conexão entre o Arduino e o computador é feita por meio de um cabo USB A-B, o mesmo tipo de cabo usado para conexão entre computador e impressora – (Ver **Figura 6**).

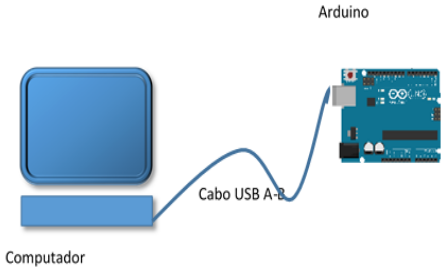


Figura 6 - Esquema da conexão entre computador e Arduino. Usar cabo USB do tipo A-B, mesmo tipo usado em conexões entre computador e impressora.

Fonte: do autor

Evidentemente, mesmo para quem não tem tanta familiaridade com programação e conhecimento sobre eletrônica pode montar e executar algum projeto com sucesso.

Monitor serial

A monitoração de um experimento e coleta de dados, como é caso dos experimentos de Física propostos neste produto educacional são facilitadas por um ferramenta disponível no Arduino IDE, o monitor serial (**Figura 7**), que pode ser iniciado quando a placa está conectada no computador.

Figura 7 - Aparência do monitor serial. Essa interface exibirá os dados experimentais e onde ocorrerão algumas interações. Todos os experimentos deste produto necessitam do monitor.

Fonte: do autor.

O Monitor serial é a interface de exibição de dados e de interação. Todos os experimentos apresentados aqui usam o monitor serial, onde grandezas como velocidade, aceleração, tempo, força, temperatura e resistência elétrica são imprimidas. Alguns experimentos aceitam a inserção de valores de entrada no monitor serial, alterando convenientemente alguns parâmetros da programação. Para utilização do monitor é necessário iniciar e especificar a taxa de transmissão, isso é feito pela função `Serial.begin(9600)` dentro do `void setup()`. Para abrir o monitor deve-se acessar no menu a opção ferramentas, ou usar o atalho *crtl+shift+m*, no teclado do computador.

As versões mais recentes do Arduino IDE, como a 1.8.4, plotam um gráfico em tempo real dos valores das variáveis informadas no comando `Serial.println()`, mesmo comando para exibição de valores no Monitor serial. Para iniciar a plotagem das medidas o usuário deve localizar e abrir a opção *Plotter serial* em ferramentas.

2. Movimento Uniforme

Observar como um corpo muda sua posição no espaço e no tempo é fundamental para o entendimento do significado do movimento. A compreensão da cinemática, que estuda os movimentos, permite, a princípio, esse entendimento, implicando no estabelecimento de modelos físicos, capazes de preverem trajetórias para um corpo. Em busca desses modelos, o experimento deste capítulo aprecia um movimento ideal, livre da atuação de forças, o Movimento Uniforme.

Fundamento Teórico

No movimento uniforme um corpo percorre distancias iguais em intervalos de tempos iguais. É importante ressaltar que movimentos assim são na realidade difíceis de serem reproduzidos, uma vez que os corpos podem estar sujeitos à ação das mais diversas forças, o que altera quaisquer uniformidades no movimento, porem há casos reais que podem ser representados por estimativas com uma margem de erro admissível.

Objetivos
- Analisar o movimento de um objeto com velocidade constante.
- Comparar intervalos de tempo para deslocamentos idênticos.

A taxa com que o corpo muda de posição para cada unidade de tempo é definida como velocidade (V).

$$V = \frac{S - S_0}{t - t_0} \qquad (2.1)$$

A variação da posição é da pela diferença entre as posições final (S) e inicial (S_0). Já o intervalo de tempo pode ser

determinado pela diferença entre o instante de tempo final (t) e inicial (t_o). Portanto, para determinar se um corpo desenvolve um movimento uniforme é suficiente calcular a velocidade do mesmo para alguns trechos.

Montagem

Disponha cada componente na *protoboard* e conectando-os com os fios jumpers à placa. Observe a polaridade correta dos LEDs, dos receptores infravermelhos e que os fios brancos deverão estar ligados aos pinos digitais 2, 3 e 4 da placa Arduino, eles que informarão o estado lógico de cada sensor.

Materiais e componentes:

- 5 resistores 330Ω
- 3 resistores 10kΩ
- 1 LED verde
- 1 LED vermelho
- 03 LEDs infravermelhos
- 03 Receptores infravermelhos
- Arduino e *protoboard*
- Fios *jumpers*
- Trilho para Movimento retilíneo

Cada par LED infravermelho/Receptor infravermelho constitui um sensor, note que haverá três pares, ou seja, três sensores.

Ver **Figura 8** para montagem

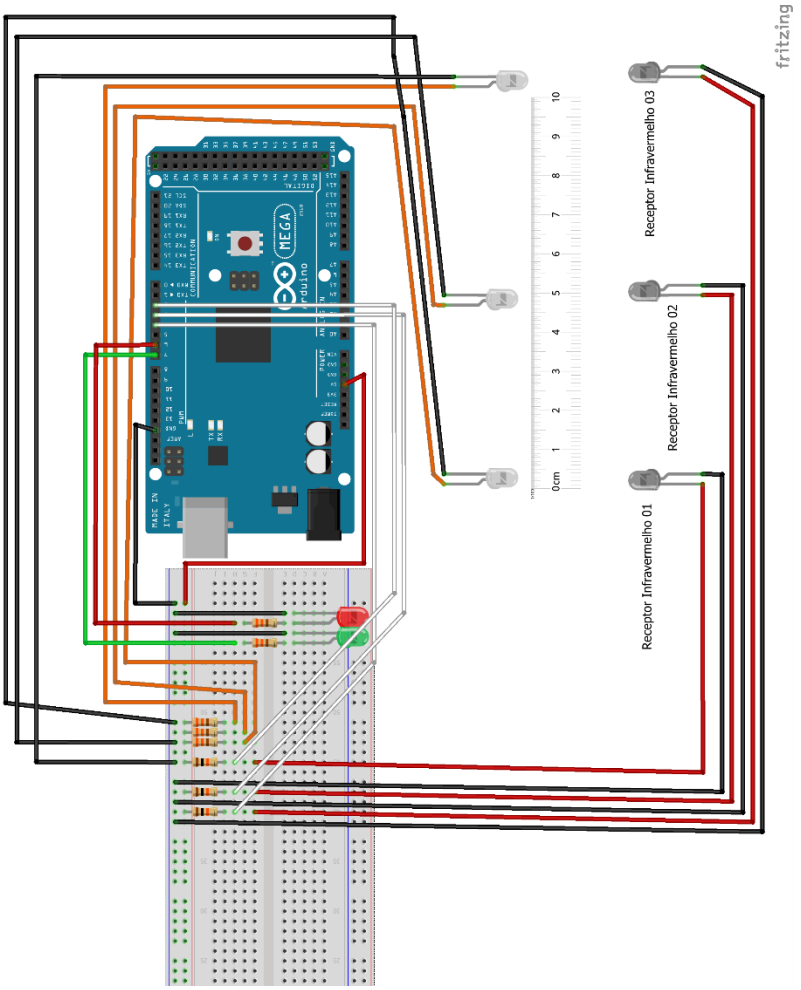


Figura 8 - Esquema de montagem de sensores para estudo de cinemática. Fixe os Leds emissores e os receptores infravermelhos no trilho que será utilizado, posicione o receptor infravermelho 02 com seu respectivo led emissor a meia distância dos demais.

Fonte: Imagem do autor extraída e editada no Fritzing.

O código

Após a montagem abra o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo de Cinemática.
 * O programa calcula o intervalo de tempo entre três sensores
infravermelho,
 * e determina se o movimento é uniforme ou uniformemente variado
 *Este programa poderá ser usado para se determinar a aceleração
gravitacional
 */

#define ledVerde 7                    // O LED verde deverá ser ligado no
pino digital 7
#define ledVermelho 6                 // O LED vermelho deverá ser ligado
no pino digital 6
#define sensorUM 2                    // Configura o pino digital 2 para
o PRIMEIRO SENSOR
#define sensorDOIS 3                  // Configura o pino digital 3 para
o SEGUNDO SENSOR
#define sensorTRES 4                  // Configura o pino digital 4 para
o TERCEIRO SENSOR

int estadoUM;                         // variável que guarda estado do
primeiro sensor
int estadoDOIS;                       // variável que guarda estado do
segundo sensor
int estadoTRES;                       // variável que guarda estado do
terceiro sensor
int UltEstSenUM;                      // variável a guardar o último
estado do primeiro sensor
int UltEstSenDOIS;                    // variável a guardar o último
estado do segundo sensor
int UltEstSenTRES;                    // variável a guardar o último
estado do terceiro sensor
long elapsedTimeOne ;                 // tempo decorrido no primeiro
intervalo
long elapsedTimeTwo ;                 // tempo decorrido no segundo
intervalo
long instanteUM = 0;                  // tempo total decorrido desde que
o programa foi executado
long instanteDOIS = 0;                // tempo total decorrido desde que
o programa foi executado
long instanteTRES= 0;                 // tempo total decorrido desde que
o programa foi executado
long tempoTotal;
```

```
void setup()
{
    Serial.begin(9600);

Serial.println("..................................................");
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................");
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................");
Serial.println("CINEMATICA");
Serial.println("..................................................");
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................");

Serial.println("Analise do tipo de movimento e Calculo da Velocidade");
Serial.println("..................................................");
delay(1000); //Pausa de 2 segundos
Serial.println("::INICIE O EXPERIMENTO::");
Serial.println("..................................................");
delay(1000); //Pausa de 1 segundos

    pinMode(ledVermelho, OUTPUT);       // Configura o pino 6 como saída
    pinMode(ledVerde, OUTPUT);          // Configura o pino 7 como saída
    pinMode(sensorUM, INPUT);           // Configura o pino 2 como
entrada
    pinMode(sensorDOIS, INPUT);         // Configura o pino 3 como
entrada
    pinMode(sensorTRES, INPUT);         // Configura o pino 4 como
entrada
    digitalWrite(ledVermelho, LOW);     // desliga LED vermelho
    digitalWrite(ledVerde, LOW);        // desliga LED verde

}

void loop()
{
estadoUM = digitalRead(sensorUM);         // Ler o sensor UM e armazena
em estadoUM
 if(estadoUM == HIGH){
  digitalWrite(ledVerde, HIGH);           // Liga o LED VERDE
  digitalWrite(ledVermelho, LOW);         // desliga o LED VERMELHO
  instanteUM = millis();                  // armazena o tempo Total
decorrido para Sensor UM

}

else
{
digitalWrite(ledVerde, LOW);             // desliga os lEDs VERDE e
VERMELHO
digitalWrite(ledVermelho, LOW);
}
```

```
estadoDOIS = digitalRead(sensorDOIS);   // Ler o sensor DOIS e armazena
em estadoDOIS
if(estadoDOIS == HIGH) {

digitalWrite(ledVerde, LOW);            // desliga LED verde
digitalWrite(ledVermelho, HIGH);        // liga LED vermelho
instanteDOIS = millis();                // armazena o tempo Total
decorrido para Sensor DOIS

elapsedTimeOne = (instanteDOIS - instanteUM);  // cálculo do primeiro
intervalo de tempo

}
else
{
digitalWrite(ledVerde, LOW);
digitalWrite(ledVermelho, LOW);                //desliga os LED verde e
Vermelho

}
estadoTRES = digitalRead(sensorTRES);         // Ler o sensor TRÊS e
armazena em estadoTRES
if(estadoTRES == HIGH) {

digitalWrite(ledVerde, HIGH);                 // liga LED Verde
digitalWrite(ledVermelho, HIGH);              // liga LED Vermelho
instanteTRES = millis();                      // armazena o tempo Total
decorrido para Sensor TRÊS

elapsedTimeTwo = (instanteTRES - instanteDOIS);
      tempoTotal = elapsedTimeTwo;
Serial.println("..................................................
.........");
Serial.print(" 1o intervalo (ms) = ");
Serial.println(elapsedTimeOne);       // imprime no monitor serial o
Primeiro Intervalo de Tempo
Serial.println("..................................................
.........");
Serial.println("..................................................
.........");
Serial.print(" 2o intervalo (ms) = ");
Serial.println(elapsedTimeTwo);       // imprime no monitor serial o
Primeiro Intervalo de Tempo
Serial.println("..................................................
.........");
Serial.println("..................................................");
Serial.println("Anote a distancia entre os Sensores.");
Serial.println("..................................................");
Serial.println("..................................................");
Serial.println("calcule as velocidades para o primeiro intervalo e para
o segundo.");
Serial.println("..................................................");
Serial.println("..................................................");
Serial.println("Qual o tipo de movimento descrito pelo objeto?");
Serial.println("..................................................");
```

```
delay(tempoTotal); //tempo de espera para efetuar nova leitura
}
else
{
digitalWrite(ledVerde, LOW);
digitalWrite(ledVermelho, LOW);   // desliga LED verde e Vermelho
}
}
```

Execução e coleta de dados

A montagem descrita na **Figura 8** pode ser usada em qualquer trilho horizontal, comum em laboratórios didáticos. Use como móvel uma esfera com dimensões compatíveis ao trilho sem colchão de ar. Se o trilho for do tipo colchão de ar, use o móvel adequado do experimento. Porém, note que o móvel deve deslocar-se sem interrupções até o fim do trilho, enquanto isso, ser detectado pelos sensores. Caso não tenha acesso a algum desses trilhos é possível construir um que se adeque as suas necessidades, os autores Alcides Goya e Samir El Halabi (2011) dão dicas de confecção e experimentação com um "Trilho Multifuncional para Ensino de Mecânica"[6].

Para que os alunos envolvidos na experiência possam observar o movimento uniforme é necessário que o professor desnivele o trilho sem colchão de ar, deixando a extremidade da origem do movimento ligeiramente mais elevado que a extremidade oposta, a razão disso é para que o a esfera não desacelere, devido a ação do atrito e da resistência do ar, verifique o esquema na **Figura 9**. É importante também não elevar demais a extremidade da origem para que a esfera não acelere; a sugestão é que o professor teste a altura certa, observando no Monitor serial do Arduino se o 1º intervalo e o 2º intervalo de tempo são idênticos.

A fim de verificar o funcionamento dos sensores passe algo em frente dos mesmos, o LED verde acenderá enquanto algum objeto for posicionado em frente ao sensor 1, para o sensor

[6] Disponívelem:http://www.uel.br/ccb/biologiageral/eventos/erebio/painel/T170.pdf

2 se acenderá o LED vermelho; para o sensor 3, os LEDs verde e vermelho ao mesmo tempo. Retirando-se os objetos todos LEDs deverão apagar.

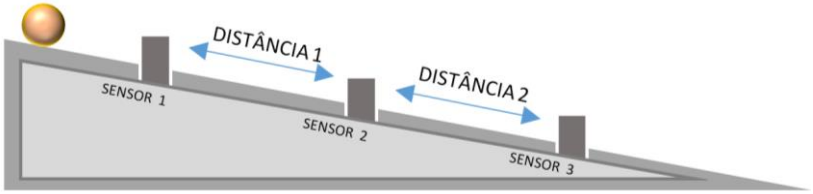


Figura 9 - Esquema para disposição dos sensores no trilho sem colchão de ar. Fixe os sensores ao trilho distanciando-os a uma mesma distância. Quanto maior for a precisão nas distancias dos sensores mais confiável será o resultado.

Fonte: do autor.

Após ter feito a montagem do circuito elétrico na *protoboard* e Arduino, posicionado os sensores e ajustado a inclinação do trilho abra o monitor serial e inicie a experiência: impulsione o móvel antes do primeiro sensor. Ao findar o percurso, observe os dados obtidos no monitor serial e registre-os.

A unidade de medida de tempo exibidas no monitor serial serão em milésimos de segundo (ms), converta para segundos ao preencher a **Tabela 1** a seguir:

Tabela 1 - Modelo de tabela para registro dos dados.

Tipos de dados	**Valores dos dados**
1º intervalo (em segundos)	
2º intervalo (em segundo)	
Qual a distância, em metros, entre o primeiro e o segundo Sensor?	
Qual a distância, em metros, entre o segundo e o terceiro Sensor?	

Fonte: do autor

Enquanto o Arduino estiver conectado ao computador é possível repetir o experimento por várias vezes. Ou seja, logo após o móvel passar pelo terceiro sensor o programa reiniciará, aguardando novos registros, isso ocorre porque a parte do programa que coleta os dados dos sensores encontra-se dentro da função *void loop*, que executa o código enquanto o Arduino estiver ligado.

- **Questionamentos e levantamento de hipóteses**

1. *Calcule as velocidades para o primeiro e para o segundo intervalo de tempo.*
2. *O movimento descrito pelo objeto utilizado aproxima-se de um movimento uniforme? Explique.*

3. Movimento Uniformemente Variado

O movimento uniformemente variado – MUV – envolve uma variação constante, mas uniforme na velocidade de um móvel. No dia-a-dia ocorre em casos específicos, como na análise do tempo e espaço percorrido por um veículo ao ser acelerado ou ter os freios acionados, compreensão útil para uma direção defensiva e desenvolvimento de dispositivos de segurança, ou mesmo no dimensionamento de pistas de pouso em aeroportos.

Fundamentos Teóricos

Quando a velocidade de um móvel varia ($\Delta V = V - V_0$) em uma taxa constante tem-se uma aceleração constante, isso define um movimento como uniformemente variado. A aceleração é uma grandeza importante neste estudo, e nos casos em que a velocidade é conhecida em qualquer instante podemos calcular a aceleração média (a_m):

Objetivos

- Compreender a relação entre velocidade e posição de um móvel.
- Aplicar a equação de

$$a_m = \frac{\Delta V}{\Delta t} \quad (3.1)$$

Em um MUV a posição varia ($\Delta S = S - S_0$) cada vez mais rápido e pela função horária da posição podemos determiná-la a qualquer instante:

$$S = S_0 + V_0 t + \frac{1}{2} a t^2 \quad (3.2)$$

É possível, ainda, relacionar variação na velocidade e na posição usando a equação de Torricelli:

$$V^2 = V_0^2 + 2a\Delta S \quad (3.3)$$

Montagem

Utilizaremos aqui a mesma montagem descrita no capítulo anterior, para o Movimento Uniforme, enfatizando que a montagem servirá para os estudos de cinemática, seja para Movimento uniforme ou Uniformemente variado, uma vez que a automatização apresentada nesta unidade consiste em aferir os intervalos de tempo entre os sensores utilizados.

O código

Apesar de a montagem ser idêntica à do capítulo anterior o programa que utilizaremos contém algumas alterações em afinidade com o objetivo do capítulo, como a inserção das variáveis **velocidadeUM** e **velocidadeDOIS** que armazenam o cálculo das velocidades V_o e V. Também foi definido a distância entre os sensores 1 e 2 e entre 2 e 3, como **deslocamentoUM** e **deslocamentoDOIS**, respectivamente. Essas duas distâncias na linha 28 e 29 do código, são iguais a 500 milímetros, mas se na prática não for possível manter essas distâncias é possível alterá-las convenientemente, enfim, a distância medida entre os sensores devem coincidir com os valores informados na programação. Então, abra o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo de Cinemática.
 * O programa calcula o intervalo de tempo entre três sensores
infravermelho,
 * e determina se o movimento é uniforme ou uniformemente variado
```

```
 *Este programa poderá ser usado para se determinar a aceleração
gravitacional
 */

#define ledVerde 7                    // O LED verde deverá ser ligado
no pino digital 7
#define ledVermelho 6                 // O LED vermelho deverá ser
ligado no pino digital 6
#define sensorUM 2                    // Configura o pino digital 2 para
o PRIMEIRO SENSOR
#define sensorDOIS 3                  // Configura o pino digital 3 para
o SEGUNDO SENSOR
#define sensorTRES 4                  // Configura o pino digital 4 para
o TERCEIRO SENSOR
int estadoUM;                         // variável que guarda estado do
primeiro sensor
int estadoDOIS;                       // variável que guarda estado do
segundo sensor
int estadoTRES;                       // variável que guarda estado do
terceiro sensor
long interTempoUM ;                  // tempo decorrido no primeiro
intervalo
long interTempoDOIS ;                 // tempo decorrido no segundo
intervalo
long instanteUM = 0;                  // tempo total decorrido desde que
o programa foi executado
long instanteDOIS = 0;                // tempo total decorrido desde que
o programa foi executado
long instanteTRES= 0;                 // tempo total decorrido desde que
o programa foi executado
long velocidadeUM;
long velocidadeDOIS;
long deslocamentoUM = 500;           // distancia sugerida em milimetro
entre sensor 1 e 2. Mude a medida se for conveniente
long deslocamentoDOIS = 500;         // distancia sugerida em milimetro
entre sensor 2 e 3. Mude a medida se for conveniente
long tempoTotal;

void setup()
{
    Serial.begin(9600);
Serial.println("...................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("...................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("...................................................")
;
Serial.println("CINEMATICA");
Serial.println("...................................................")
;
delay(2000); //Pausa de 2 segundos
```

```
Serial.println("................................................")
;
Serial.println("Analise do tipo de movimento e Calculo da Aceleração");
Serial.println("................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("::INICIE O EXPERIMENTO::");
Serial.println("................................................")
;
delay(1000); //Pausa de 1 segundos
    pinMode(ledVermelho, OUTPUT);       // Configura o pino 6 como
saída
    pinMode(ledVerde, OUTPUT);          // Configura o pino 7 como
saída
    pinMode(sensorUM, INPUT);           // Configura o pino 2 como
entrada
    pinMode(sensorDOIS, INPUT);         // Configura o pino 3 como
entrada
    pinMode(sensorTRES, INPUT);         // Configura o pino 4 como
entrada
    digitalWrite(ledVermelho, LOW);     // desliga LED vermelho
    digitalWrite(ledVerde, LOW);        // desliga LED verde
}
void loop()
{
estadoUM = digitalRead(sensorUM);         // Ler o sensor UM e armazena
em estadoUM
 if(estadoUM == HIGH){
  digitalWrite(ledVerde, HIGH);           // Liga o LED VERDE
  digitalWrite(ledVermelho, LOW);         // desliga o LED VERMELHO
  instanteUM = millis();                  // armazena o tempo Total
decorrido para Sensor UM
}
else
{
digitalWrite(ledVerde, LOW);             // desliga o lED VERDE e
VERMELHO
digitalWrite(ledVermelho, LOW);
}
      estadoDOIS = digitalRead(sensorDOIS);   // Ler o sensor DOIS e
armazena em estadoDOIS
if(estadoDOIS == HIGH) {
digitalWrite(ledVerde, LOW);            // desliga LED verde
digitalWrite(ledVermelho, HIGH);        // liga LED vermelho
instanteDOIS = millis();                // armazena o tempo Total
decorrido para Sensor DOIS
  interTempoUM = (instanteDOIS - instanteUM);  // cálculo do primeiro
intervalo de tempo
velocidadeUM = deslocamentoUM/interTempoUM; // calculo da velocidade
média entre 1 e 2 sensor
}
else
{
digitalWrite(ledVerde, LOW);
digitalWrite(ledVermelho, LOW);               //desliga os LED verde e
Vermelho
```

```
}
estadoTRES = digitalRead(sensorTRES);         // Ler o sensor TRÊS e
armazena em estadoTRES
if(estadoTRES == HIGH) {

digitalWrite(ledVerde, HIGH);                 // liga LED Verde
digitalWrite(ledVermelho, HIGH);              // liga LED Vermelho
instanteTRES = millis();                      // armazena o tempo Total
decorrido para Sensor TRÊS

interTempoDOIS = (instanteTRES - instanteDOIS);
velocidadeDOIS = deslocamentoDOIS/interTempoDOIS; // calculo da
velocidade média entre 1 e 2 sensor
      tempoTotal = interTempoDOIS;
Serial.println("..................................................
.........");
Serial.print(" |||| Velocidade Aproximada (em m/s) no sensor 2 ||||
V0= ");
Serial.println(velocidadeUM);       // imprime no monitor serial o a
velocidade entre sensor 1 e 2
Serial.println("..................................................
.........");
Serial.println("..................................................
.........");
Serial.print(" |||| Velocidade Aproximada (em m/s) no sensor 3 ||||
V= ");
Serial.println(velocidadeDOIS);       // imprime no monitor serial o a
velocidade entre sensor 2 e 3
Serial.println("..................................................
.........");
Serial.println("................................................")
;
Serial.println("Anote as velocidades V0 e V aqui exibidas.");
Serial.println("................................................")
;
Serial.println("................................................")
;
Serial.println("Realize a medida e anote a distancia entre os Sensores
2 e 3.");
Serial.println("................................................")
;
Serial.println("................................................")
;
Serial.println("Admitindo Uniformidade na variação da Velocidade,
calcule a aceleração do móvel, usando a Equação de Torricelli.");
Serial.println("................................................")
;

delay(tempoTotal); //tempo de espera para efetuar nova leitura
}
else
{
digitalWrite(ledVerde, LOW);
digitalWrite(ledVermelho, LOW);   // desliga LED verde e Vermelho
}
}
```

Execução e coleta de dados

Para este tipo de experimento será necessário posicionar o trilho a um ângulo em que seja perceptível o MUV (sugestão: 45°). Após posicionar o trilho e os sensores a 500mm entre si, e ter feito o upload do código, inicie o monitor serial (atalho: *crtl+ shift+m*). Abandone o móvel do ponto mais alto do trilho. Como a aceleração gravitacional é constante o móvel descreverá um MUV. No monitor serial será exibido a velocidade (V_0) do móvel no 2º sensor e a velocidade (V) no 3º sensor. Registre os valores na **Tabela 2**.

Tabela 2 - Modelo de tabela para registro dos dados.

Tipos de dados	Valores dos dados
Velocidade Aproximada (em **m/s**) no sensor 2. V_0 =	
Velocidade Aproximada (em **m/s**) no sensor 3. V =	
Distância (em **m**) entre os sensores 1 e 3. ΔS =	

Fonte: do autor.

Questionamentos e levantamento de hipóteses

1. *Admitindo Uniformidade na variação da Velocidade, calcule a aceleração do móvel, usando a Equação de Torricelli.*
2. *O que é a aceleração? Use o valor encontrado neste experimento para exemplificar.*
3. *Estime a velocidade do móvel ao passar, hipoteticamente, por um 4º sensor posicionado a uma distância do 3º idêntica a distâncias entre os demais sensores.*

4. Aceleração da Gravidade

A alteração da velocidade de um objeto pode ser aferida e a medida desta variação em função do tempo é definida como aceleração, grandeza importante para o estudo do movimento e suas causas. Este capítulo apresenta uma programação útil no estudo do movimento de objetos sujeitos a aceleração gravitacional. A montagem do Arduino, idêntica a dos capítulos 2 e 3 facilita a determinação da aceleração gravitacional local. O programa apresentará, quando executado, a aceleração escalar média do objeto no trecho com sensores.

Fundamentos Teóricos

É possível fazer uma análise simples do movimento de um objeto desconsiderando algumas características vetoriais. A determinação da aceleração escalar média permite o estudo de um movimento uniformemente variado, prevendo como o valor da velocidade evolui. Observando a medida da velocidade inicial e final de um móvel, para um instante de tempo inicial e final respectivamente, calcula-se a razão entre a variação da velocidade e do instante de tempo como a aceleração escalar - ver Eq. 3.1.

Objetivos
- Determinação do módulo da aceleração gravitacional.

O valor da aceleração escalar carrega, *a priori*, dois tipos de informações: um se refere a intensidade da variação da velocidade; o outro, indica se o móvel está acelerado ou desacelerado, basta verificar se o valor da aceleração é positivo (acelerado) ou negativo (desacelerado).

Montagem

Mais uma vez utilizaremos aqui a mesma montagem descrita no capítulo anterior, para o Movimento Uniforme. Fixe os sensores em um trilho vertical, conforme **Figura 10**.

Materiais e componentes:

- 5 resistores 330Ω
- 3 resistores 10kΩ
- 1 LED verde
- 1 LED vermelho
- 03 LEDs infravermelhos
- 03 Receptores infravermelhos
- Arduino e protoboad
- Fios *jumpers*
- Trilho vertical para queda livre

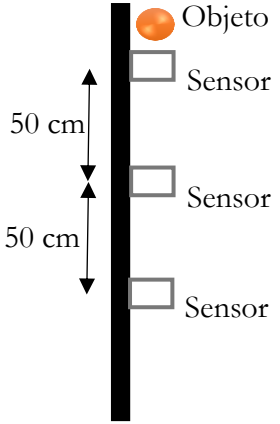


Figura 10 - Esquema para disposição dos sensores no trilho vertical. Posicione os três sensores a 50 cm um do outro.

Fonte: do autor.

O código

Abra o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo de Cinemática.
 * O programa calcula o intervalo de tempo entre três sensores
infravermelho,
 * e determina se o movimento é uniforme ou uniformemente variado
 *Este programa poderá ser usado para se determinar a aceleração
gravitacional
 */

#define ledVerde 7                    // O LED verde deverá ser ligado
no pino digital 7
#define ledVermelho 6                 // O LED vermelho deverá ser
ligado no pino digital 6
#define sensorUM 2                    // Configura o pino digital 2 para
o PRIMEIRO SENSOR
#define sensorDOIS 3                  // Configura o pino digital 3 para
o SEGUNDO SENSOR
#define sensorTRES 4                  // Configura o pino digital 4 para
o TERCEIRO SENSOR

int estadoUM;                         // variável que guarda estado do
primeiro sensor
int estadoDOIS;                       // variável que guarda estado do
segundo sensor
int estadoTRES;                       // variável que guarda estado do
terceiro sensor
long interTempoUM ;                 // tempo decorrido no primeiro
intervalo
long interTempoDOIS ;                 // tempo decorrido no segundo
intervalo
long instanteUM = 0;                  // tempo total decorrido desde que
o programa foi executado
long instanteDOIS = 0;                // tempo total decorrido desde que
o programa foi executado
long instanteTRES= 0;                 // tempo total decorrido desde que
o programa foi executado
long velocidadeUM;
long velocidadeDOIS;
long deslocamentoUM = 500;          // distancia sugerida em milimetro
entre sensor 1 e 2. Mude a medida se for conveniente
long deslocamentoDOIS = 500;        // distancia sugerida em milimetro
entre sensor 2 e 3. Mude a medida se for conveniente
long tempoTotal;
long g;                               // aceleração gravitacional
```

```
void setup()
{
    Serial.begin(9600);

Serial.println("..................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("CINEMATICA");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................")
;

Serial.println("Afericao da Aceleracao Gravitacional (g)");
Serial.println("..................................................")
;
delay(1000); //Pausa de 2 segundos
Serial.println("::INICIE O EXPERIMENTO::");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos

    pinMode(ledVermelho, OUTPUT);       // Configura o pino 6 como
saída
    pinMode(ledVerde, OUTPUT);          // Configura o pino 7 como
saída
    pinMode(sensorUM, INPUT);           // Configura o pino 2 como
entrada
    pinMode(sensorDOIS, INPUT);         // Configura o pino 3 como
entrada
    pinMode(sensorTRES, INPUT);         // Configura o pino 4 como
entrada
    digitalWrite(ledVermelho, LOW);     // desliga LED vermelho
    digitalWrite(ledVerde, LOW);        // desliga LED verde

}

void loop()
{
estadoUM = digitalRead(sensorUM);         // Ler o sensor UM e armazena
em estadoUM
 if(estadoUM == HIGH){
  digitalWrite(ledVerde, HIGH);           // Liga o LED VERDE
  digitalWrite(ledVermelho, LOW);         // desliga o LED VERMELHO
  instanteUM = millis();                  // armazena o tempo Total
decorrido para Sensor UM

}
```

```
else
{
digitalWrite(ledVerde, LOW);             // desliga os lEDs VERDE e
VERMELHO
digitalWrite(ledVermelho, LOW);
}

estadoDOIS = digitalRead(sensorDOIS);   // Ler o sensor DOIS e armazena
em estadoDOIS
if(estadoDOIS == HIGH) {

digitalWrite(ledVerde, LOW);            // desliga LED verde
digitalWrite(ledVermelho, HIGH);        // liga LED vermelho
instanteDOIS = millis();                // armazena o tempo Total
decorrido para Sensor DOIS

interTempoUM = (instanteDOIS - instanteUM);  // cálculo do primeiro
intervalo de tempo
velocidadeUM = deslocamentoUM/interTempoUM; // calculo da velocidade
média entre 1 e 2 sensor
}
else
{
digitalWrite(ledVerde, LOW);
digitalWrite(ledVermelho, LOW);                //desliga os LED verde e
Vermelho

}
estadoTRES = digitalRead(sensorTRES);         // Ler o sensor TRÊS e
armazena em estadoTRES
if(estadoTRES == HIGH) {

digitalWrite(ledVerde, HIGH);                 // liga LED Verde
digitalWrite(ledVermelho, HIGH);              // liga LED Vermelho
instanteTRES = millis();                      // armazena o tempo Total
decorrido para Sensor TRÊS

interTempoDOIS = (instanteTRES - instanteDOIS);
velocidadeDOIS = deslocamentoDOIS/interTempoDOIS; // calculo da
velocidade média entre 1 e 2 sensor
g=((velocidadeDOIS*velocidadeDOIS)-
(velocidadeUM*velocidadeUM))/(2*deslocamentoDOIS);    // eq. De
Torricelli
      tempoTotal = interTempoDOIS;
Serial.println("..................................................
.........");
Serial.print(" |||| Aceleracao Gravitacional (em m/s^2) ||||   g= ");
Serial.println(g);       // imprime no monitor serial o valor
aproximado da aceleração gravitacional
Serial.println("..................................................
.........");
Serial.println("..................................................
.........");
delay(tempoTotal); //tempo de espera para efetuar nova leitura
}
```

```
else
{
digitalWrite(ledVerde, LOW);
digitalWrite(ledVermelho, LOW);   // desliga LED verde e Vermelho
}
}
```

Importante mencionar as dificuldades técnicas no cálculo do tempo de queda de um objeto usando um cronômetro, uma vez que o tempo de reação para acionamento do cronometro deve ser muito menor que o tempo de queda. A rapidez na atuação dos sensores conectados ao Arduino permite determinar intervalos tempo muito pequenos na ordem de milésimos de segundo.

- **Execução e coleta de dados**

Como o objetivo deste experimento consiste na determinação da aceleração gravitacional, segure um objeto logo acima do sensor 1 (**Figura 10**), certificando-se que as montagens estão corretas, abandone o objeto. Como a montagem é a mesma do capitulo 2 e 3, apenas com alguns ajustes na programação, o LED verde piscará quando o objeto passar pelo primeiro sensor, depois o LED vermelho e finalmente no terceiro sensor os dois LEDs piscarão simultaneamente. Feito isso, será exibido no monitor serial o valor aproximado de g em m/s^2, anote.

- **Questionamentos e levantamento de hipóteses**

1. *Compare o valor da aceleração gravitacional encontrado pelo dispositivo com o valor normalmente apresentado nos livros de Física. Levante hipóteses e pesquise a respeito de possíveis divergências entre esses valores.*

5. Movimento Circular Uniforme

Implementações tecnológicas referente ao domínio do Movimento Circular Uniforme – MCU – abrangem desde a determinação do raio de curvatura em via de transporte rápido, a fim de minimizar o desconforto da alteração na orientação do vetor velocidade, até analisar e prever o movimento periódico da orbita de um satélite. Veremos, portanto, quais os principais fundamentos do MCU no âmbito da Cinemática, porém esse estudo já aponta algumas causas para o entendimento da natureza do MCU, com a apreciação do Princípio Fundamental da Dinâmica. Neste capitulo será apresentada uma automação que permite medir a velocidade angular de certos objetos que desenvolva um movimento circular. Conforme descrito na montagem utilizaremos um transistor de efeito Hall como sensor.

Objetivos

- Compreender principais características do movimento circular uniforme.
- Identificar e aplicar fundamentos do MCU no dia-a-dia.

Fundamentos Teóricos

O movimento circular uniforme consiste em um movimento cuja trajetória não é retilínea conotando a existência de uma aceleração orientada sempre para o centro de curvatura, implicando na alteração da orientação da velocidade; contudo sem alterar seu módulo. Tal aceleração (a) é inversamente proporcional ao raio de curvatura (R):

$$a = \frac{v^2}{R} \quad (5.1)$$

A velocidade em questão tem módulo constante, o que caracteriza a uniformidade do movimento, além disso em qualquer ponto da trajetória circular a velocidade é tangente.

$$\omega = \frac{2\pi}{T} \quad (5.2)$$

A velocidade angular tem unidade de medida definida como rad/s. Em termos da velocidade angular a velocidade escalar, e dada por:

$$\omega = \frac{v}{R} \quad (5.3)$$

Montagem

A montagem apresentada (**Figura 11**)consiste em uma automação para o estudo do movimento circular, para isso, será necessária a utilização de algo que descreva um movimento circular, como sugestão pode-se utilizar uma parafusadeira.

Materiais e componentes:

- Transistor de efeito hall - S41.
- Arduino.
- Resistor 10kΩ.
- Jumper.
- Ímã neodímio.
- Motor ou parafusadeira.

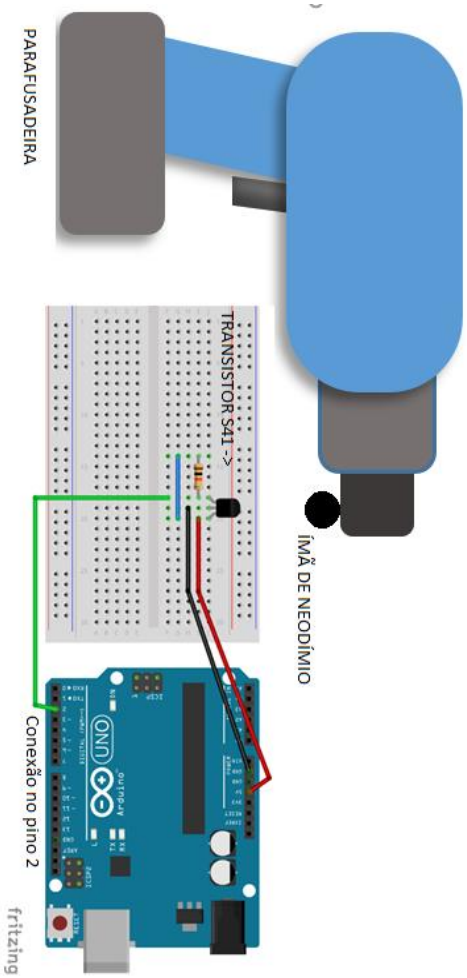


Figura 11 - Obedeça a montagem conforme a figura, usando uma parafusadeira, ou outro objeto que apresente movimento circular, mas observe que é usado um ímã que alternará o estado lógico do pino digital 2 do Arduino em cada rotação, com o código carregado na placa o Arduino exibirá a quantidade de rotação por minuto. Atenção: fixe firmemente o ímã na região de rotação!

Fonte: Imagem do autor extraída e editada no Fritzing.

Para esta montagem e programação é possível medir a frequência angular de uma parafusadeira com boa precisão. Na *protoboard* conecte cada componente, ligue os *jumpers* ao Arduino, atente-se para a conexão do pino digital 2 com o terminal de saída o transistor de efeito hall – s41. Na ponta extrema do eixo de rotação da parafusadeira fixe o ímã, importante que seja um imã cilíndrico de neodímio, cuja intensidade da força magnética mantenha-o preso durante a rotação.

O transistor especificado nesta automação é do tipo liga/desliga, ou seja, apresenta um nível lógico alto (HIGH) ou baixo (LOW), determinado pela polaridade norte ou sul do ímã que se aproxima. Ao fixar o ímã em algum rotor e aproximá-lo do transistor, este alternará sua saída digital entre HIGH e LOW à medida que o ímã gira.

A programação a seguir permitirá a contagem de apenas um dos estados lógicos para cada segundo, o que será equivalente a frequência angular do rotor. Esse tipo de automação pode ser facilmente ajustado como tacômetro, dispositivo que permite a medição do número de *rotações por minuto* -RPM- de um motor. Por fim, verifique cuidadosamente a montagem, faça upload do código usando o software e siga as instruções do tópico execução e coleta de dados.

O código

Abra o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo do Movimento Circular Iniforme.
 * O programa calcula o valor aproximado da frenquencia angular.
 * utiliza-se no Arduino um transistor de efeito hall, conveniente
 * para registrar uma das polaridades de um ímã de neodímio fixado
 * no eixo de rotação de um motor (parafusadeira), contando o
 * número de voltas por segundo.
 */
```

```
long numVoltas=0; //Cria uma variável para contagem do numero de voltas

void setup()
{
Serial.begin(9600);
Serial.println("..................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("MOVIMENTO CIRCULAR UNIFORME");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("Medindo a Velocidade Angular");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("::INICIE O EXPERIMENTO::");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos

}

void loop()
{
numVoltas = 0;
attachInterrupt(0, armVoltas, FALLING); //interrompe o loop para
incrementar o numero de voltas(ativando a função "armaVoltas"), a
partir do transistor s41 no pino 2
delay(1000); //Aguarda 1 segundo
detachInterrupt(0); //Desabilita a interrupção
numVoltas = numVoltas;
Serial.println("velocidade angular");
Serial.print(numVoltas); //Imprime o numero de voltas por segundo na
serial, a velocidade angular
Serial.println(" rad/s"); //Imprime a unidade de medida da velocidade
angular
}

void armVoltas() // função para armazenar o numero de voltas
{
numVoltas++; //soma o numero de rotações quando a variação na
polaridade do ímã
}
```

Execução e coleta de dados

Com este experimento fica fácil a análise do movimento circular uniforme, permitindo a compreensão de algumas grandezas, próprias do MCU, como a frequência angular, a velocidade tangencial em um ponto a uma distância R do centro de rotação e principalmente a compreensão da aceleração centrípeta.

Portanto, utilizando uma régua, meça o raio (R) do ímã e registre a medida do raio. Converta a unidade de medida para metros. Tendo iniciado o monitor serial aproxime a parafusadeira, em operação, do transistor de efeito Hall, perceba no monitor que a frequência angular será registrada, mantenha a mesma velocidade de rotação na parafusadeira, pode se aplicar a potência máxima, onde será mostrada frequência angular máxima da parafusadeira. Registre também a frequência angular (ω).

Questionamentos e levantamento de hipóteses

1. *Utilizando a frequência angular registrada, calcule a velocidade tangencial da extremidade do imã.*
2. *Calcule o módulo da aceleração centrípeta atuante nas extremidades do ímã.*
3. *Explique qual o papel da aceleração centrípeta no MCU.*

6. Força Peso

O peso de um corpo está relacionado a aceleração gravitacional em que o corpo está sujeito e sua massa. Na Terra, um corpo qualquer é seis vezes mais pesado do que se estivesse na Lua. Neste experimento mediremos o peso de algumas massas aferidas usando um dinamômetro simples acoplado a um módulo laser e LDR, que conectados ao Arduino mostram a força peso em Newtons, na superfície da Terra e em alguns corpos celestes do sistema solar.

Objetivos
- Aplicar 2ª Lei de Newton.
- Determinação do peso de massas aferidas.

Fundamentos Teóricos

O peso de um corpo é determinado pelo campo gravitacional onde o corpo se encontra, como o peso é um tipo de força, sua determinação tem origem na aplicação da 2ª Lei de Newton. A aplicação de uma força sobre um corpo de massa m, produz uma aceleração. A relação matemática é dada por,

$$F = m \cdot a \quad (6.1)$$

Considerando que um objeto esteja sujeito exclusivamente a ação de uma força gravitacional, a aceleração produzida é definida como a aceleração gravitacional (g). Nesses termos, a força determinada é chamada de peso (p):

$$p = m \cdot g \quad (6.2)$$

No dia-a-dia o termo “peso” é usado indevidamente referindo-se a “massa”. O instrumento que mede realmente o peso de um objeto é o dinamômetro, enquanto que as balanças medem a massa, obviamente o peso pode ser calculado a partir

da massa aferida em uma balança. O cálculo seria simplesmente o produto da massa e da aceleração gravitacional.

Com um dinamômetro é possível aferir o peso de um corpo. Quando uma massa suspensa à mola de um dinamômetro está em equilíbrio, as forças atuantes são o peso e a força elástica. Nesta situação, as forças têm módulos iguais, logo o peso do objeto será proporcional a deformação na mola.

Montagem

A montagem apresentada neste capítulo consiste na automação de um dinamômetro simples, a intenção da automação é permitir a aferição da força peso, em newtons, de objetos. Com o Arduino será possível análise do peso, seja na superfície da Terra ou em outros corpos celestes.

Materiais e componentes:

- Dinamômetro
- Massas aferidas
- Arduino
- Resistor 10 kΩ
- Protoboard
- Jumpers
- Laser

No monitor serial será possível verificar o peso na Terra, na Lua, no Sol e nos demais planetas do sistema solar, incluindo Plutão.

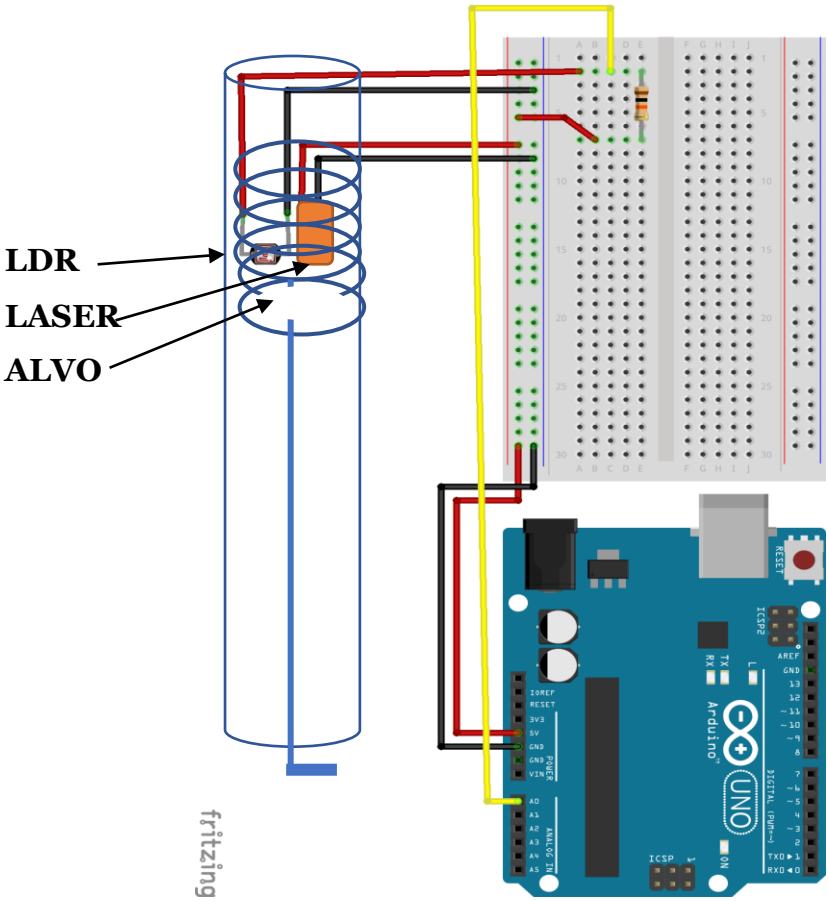


Figura 12 - Para a execução deste experimento é necessário desmontar a parte superior do dinamômetro, se isso não for possível será necessário serrá-lo totalmente na superior, isto para inserir o Laser e o LDR. Além disso, um alvo branco deve ser fixado na espira inferior (a última) a fim de refletir a luz do laser na direção do LDR. Sugestão: caso não caiba o Laser e o LDR no interior do dinamômetro você pode fixar o LDR na parte externa do dinamômetro, paralelo ao Laser.

Fonte: Imagem do autor extraída e editada no Fritzing.

Para executar este experimento é importante desmontar o dinamômetro para fixar o módulo laser/LDR em uma posição a, aproximadamente, 1cm acima da extremidade móvel inferior da mola do dinamômetro. Essa mesma extremidade deverá estar pintada com cor clara, o que permite a reflexão do laser e devida sensibilização do LDR. Veja a **Figura 12**.

Para iniciar o experimento fixe verticalmente, em um suporte, o dinamômetro e prepare as massas aferidas que você utilizará.

O código

Abra o software IDE do Arduino e digite/cole o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo da força
 * peso de um objeto,
 */
char leitura;
int valorLDR;
int forcIN;
int forcFIM;
float valFOR;
int Fdim = 2; // ajuste aqui a medida(em N) máxima da força que pode
ser registrada no seu dinamômetro
float FdimP; // variável armazena múltiplo de Fdim, para controlar
precisão

void setup() {
  Serial.begin(9600);


Serial.println("................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("................................................")
;
Serial.println("FORCA PESO");
```

```
Serial.println("................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("................................................")
;
Serial.println("MEDINDO O PESO DE MASSAS AFERIDAS");
Serial.println("................................................")
;
delay(1000); //Pausa de 2 segundos
  Serial.println("Calibrando o dinamometro");
  delay(2000);
Serial.println("posicione o dinamometro na posicao inicial");
delay(5000);
 forcIN = analogRead(A0);
 delay(1000);
Serial.println("...");
Serial.println("Agora posicione o dinamometro na posicao final");
delay(3000);
forcFIM = analogRead(A0);
delay(1000);
Serial.println("calibrado!");
delay(1000);
Serial.println("para exibir o peso do objeto pressione a letra inicial
do planeta e aperte Enter");
Serial.println("Peso na Terra (T ou t)");
Serial.println("Peso na Lua (L ou l)");
Serial.println("Peso no Sol (S)");
Serial.println("Peso em Venus (V ou v)");
Serial.println("Peso em Mercurio (m)");
Serial.println("Peso em Marte (M)");
Serial.println("Peso em Jupiter (J ou j)");
Serial.println("Peso em Saturno (s)");
Serial.println("Peso em Netuno (N ou n)");
Serial.println("Peso em Plutao (P ou p)");
Serial.println("Peso em Urano (U ou u)");
}

void loop() {
FdimP = Fdim*100.00;

valorLDR = analogRead(A0);

 valFOR = map(valorLDR, forcFIM, forcIN, FdimP, 0.00);
delay(500);
while (Serial.available() > 0) {
//Lê o dado vindo da Serial e armazena na variável leitura
leitura = Serial.read();
if (leitura == 'T' || leitura =='t'){// As duas || é a operação
booleana OU

Serial.println("Peso na Terra:");
Serial.print("F= ");
Serial.print(valFOR/100);
Serial.println(" N");
if ((valFOR/100)< -0.02) {
  Serial.println("::::sensor descalibrado, calibre-o novamente::::");
```

```
  }
}
else if (leitura == 'L' || leitura =='l'){
Serial.println("Peso na Lua:");
Serial.print("F= ");
Serial.print(valFOR/600);
Serial.println(" N");
}

else if (leitura == 'S'){
Serial.println("Peso no Sol:");
Serial.print("F= ");
Serial.print(valFOR*28.02/100);
Serial.println(" N");
}

else if (leitura == 'm'){
Serial.println("Peso em Mercurio:");
Serial.print("F= ");
Serial.print(valFOR*0.38/100);
Serial.println(" N");
}
else if (leitura == 'V' || leitura =='v'){
Serial.println("Peso no Venus:");
Serial.print("F= ");
Serial.print(valFOR*0.90/100);
Serial.println(" N");
}
else if (leitura == 'M'){
Serial.println("Peso em Marte:");
Serial.print("F= ");
Serial.print(valFOR*0.38/100);
Serial.println(" N");
}
else if (leitura == 'J' || leitura =='j'){
Serial.println("Peso em Jupiter:");
Serial.print("F= ");
Serial.print(valFOR*2.53/100);
Serial.println(" N");
}
else if (leitura == 's'){
Serial.println("Peso em Saturno:");
Serial.print("F= ");
Serial.print(valFOR*1.07/100);
Serial.println(" N");
}
else if (leitura == 'U' || leitura =='u'){
Serial.println("Peso em Urano:");
Serial.print("F= ");
Serial.print(valFOR*0.89/100);
Serial.println(" N");
}
else if (leitura == 'N' || leitura =='n'){
Serial.println("Peso Netuno:");
Serial.print("F= ");
Serial.print(valFOR*1.14/100);
```

```
Serial.println(" N");
}
else if (leitura == 'P' || leitura =='p'){
Serial.println("Peso em Plutao:");
Serial.print("F= ");
Serial.print(valFOR*0.07/100);
Serial.println(" N");
}

                                        }
}
```

Para que seu dispositivo funcione corretamente, o programa exigirá calibração do dinamômetro antes de iniciar a realização das medidas. Isso em razão de possíveis variações na posição do LDR e das flutuações de luminosidade no LDR. Além disso é importante que você observe a medida máxima do seu dinamômetro e digite essa medida na linha 14 substituindo o valor que lá se encontra. Veja que nesta linha aparece a variável *Fdim = 2*, por padrão este programa mede adequadamente força de até 2N. Portanto, verifique até quanto varia seu dinamômetro e substitua o valor da variável *Fdim*.

- **Execução e coleta de dados**

Depois de conferir toda a montagem e a inserção do código no software do Arduino IDE, faça *upload* para o seu Arduino. Inicie o Monitor serial, note que após a introdução do experimento que é exibida no Monitor serial, surgirá a mensagem:

```
"Calibrando o dinamometro"
```

Será o momento de calibrar o dinamômetro para só então dar inicio as medidas. A mensagem seguinte será:

```
"posicione o dinamometro na posicao
          inicial"
```

A partir da exibição da mensagem você terá apenas 5 segundos para deixar o dinamômetro em sua posição inicial, ou seja, sem nenhuma massa suspensa em sua extremidade inferior

aguarde até que surja a mensagem para o próximo procedimento. Caso não consiga efetuar este procedimento dentro dos 5 segundo reinicie o monitor Serial, ou o Arduino, e repita. Ao findar os 5 segundos, a nova mensagem será exibida:

```
"Agora posicione o dinamometro na
      posicao final"
```

Este é o último procedimento e consiste em apenar estender a mola do Arduino até sua marca máxima, correspondente ao valor máximo da força que o mesmo pode registrar. Também deverá ser mantida essa posição durante os 5 após a exibição da mensagem.

Tudo pronto, você já pode medir o peso da massa que você escolheu! Caso o dinamômetro esteja mal calibrado, surgirá a seguinte mensagem quando estiver sem a massa:

```
"::::sensor descalibrado, calibre-o
          novamente::::"
```

Para resolver isso, inicie novamente o monitor serial ou o Arduino e repita os dois procedimentos.

Note que quando você suspender alguma massa no dinamômetro nenhuma medida aparecerá no Monitor até que você pressione a letra inicial correspondente ao corpo celeste em que supostamente a massa aferida estivesse. Ou seja, suspenda a massa no dinamômetro digite *T* ou *t* no monitor, pressione ENTER e pronto, será exibido o peso da massa aferida na Terra. Para saber a massa em qualquer outro corpo celeste verifique a relação a seguir: Peso na Terra (T ou t); Peso na Lua (L ou l); Peso no Sol (S); Peso em Vênus (V ou v); Peso em Mercúrio (m); Peso em Marte (M); Peso em Júpiter (J ou j); Peso em Saturno (s); Peso em Netuno (N ou n); Peso em Plutão (P ou p); Peso em Urano (U ou u).

Tabela 3 - Modelo de tabela para registro dos dados

Tipo de dado	**Valor do dado**
Massa aferida (kg):	
Corpo celeste	**Peso da massa aferida**
Júpiter	
Lua	
Marte	
Mercúrio	
Netuno	
Plutão	
Saturno	
Sol	
Terra	
Urano	
Vênus	

Fonte: do autor.

Registre a medida da massa que você escolheu em Kg na **Tabela 3** e as medidas do peso desta massa apresentado em cada corpo celeste.

Questionamentos e levantamento de hipóteses

1. *Porque a mesma massa apresenta pesos diferentes em cada corpo celeste?*
2. *Calcule a aceleração gravitacional em cada corpo celeste usando a massa que você usou e os respectivos pesos apresentados.*
3. *Pense e comente sobre como a gravidade influencia na órbita de um corpo celeste.*

7. Lei de Hooke

Muitos instrumentos de medição utilizam molas como parte do sistema mecânico de medição, como é o caso de algumas balanças, dinamômetros e aparelhos de medidas de grandezas elétricas. Essas aplicações são possíveis em virtude da deformação da mola manter uma relação de proporcionalidade entre a força aplicada e deslocamento.

Objetivos

❖ Compreender a relação de proporcionalidade na deformação de uma mola.

Fundamentos Teóricos

Quando se aplica uma força sobre uma mola surge uma deformação na mola, ou seja, a mola pode esticar ou ser comprimida dependendo do sentido da força aplicada. A relação de proporcionalidade entre a força e a deformação elástica foi apresentada ainda no século XVII pelo físico inglês Robert Hooke, a partir do comportamento de uma mola, obviamente, esta relação aplica-se a quaisquer materiais que apresentem deformação elástica, ou seja que retornem as suas características originais quando a força é retirada. Eis a relação:

$$F = k \cdot x \quad (7.1)$$

F representa a força em Newtons (N), x a deformação em metros (m) e k uma constante de proporcionalidade denominada constante elástica da mola, cuja unidade é N/m.

Montagem

Usar a mesma montagem apresentada no capítulo 6.

O código

Abra o o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo da Lei de Hooke
 */
char leitura;
int valorLDR;
int forcIN;
int forcFIM;
float valFOR;
int Fdim = 2; // ajuste aqui a medida(em N) maxima da força no seu
dinammetro
float FdimP; // variável armazena multiplo de Fdim, para controle
precisão

void setup() {
  Serial.begin(9600);


Serial.println("..................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("FORCA PESO");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("MEDINDO O PESO DE MASSAS AFERIDAS");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos

  Serial.println("Calibrando o dinamometro");
  delay(2000);
Serial.println("posicione o dinamometro na posicao inicial");
delay(5000);
 forcIN = analogRead(A0);
 delay(1000);
Serial.println("...");
Serial.println("Agora posicione o dinamometro na posicao final");
```

```
delay(3000);
forcFIM = analogRead(A0);
delay(1000);
Serial.println("calibrado!");
delay(1000);
}
void loop(){
FdimP = Fdim*100.00;

valorLDR = analogRead(A0);

 valFOR = map(valorLDR, forcFIM, forcIN, FdimP, 0.00);
delay(500);
Serial.print("F= ");
Serial.print(valFOR/100);
Serial.println(" N");
if ((valFOR/100)< -0.02) {
  Serial.println("::::sensor descalibrado, calibre-o novamente::::");
  }
}
```

Instruções sobre o código e possíveis ajustes consulte o capítulo 6.

- **Execução e coleta de dados**

Depois de enviar o código ao Arduino, realize a calibragem como descrita no capítulo 6. Use diversas massas aferidas e pendure-as no dinamômetro, registrando a força apesentada no monitor serial. Use a **Tabela 4**.

Tabela 4 - Modelo de tabela para registro dos dados.

Massa aferida (g)	Força peso (N)	Deformação na mola (x)

Fonte: do autor.

- **Questionamentos e levantamento de hipóteses**

1. *O dinamômetro que você utilizou obedeceu a Lei de Hooke? Sugestão: calcule a constante elástica para mais de um par de medida da força e da deformação.*
2. *Qual a constante elástica da mola do dinamômetro que você utilizou?*

8. Hidrostática: Empuxo

A hidrostática é parte da mecânica dos fluidos, área da física que estuda os efeitos das forças sobre os fluidos, ou seja, gases e líquidos. Os princípios abordados na hidrostática são preponderantes para se compreender o comportamento dos fluidos em situação de repouso, bem como a atuação de certas forças e a distribuição da pressão. Conceitos fundamentais que ajudam a compreender e replicar conhecimentos, como, por que navios pesadíssimos não afundam? Por que nos sentimos mais leves ao entrarmos em piscina? São questões curiosas, mas que podem ser facilmente compreendidas tomando os princípios hidrostáticos. Para essas questões aplica-se suficientemente o princípio de Arquimedes, portanto este capítulo será dedicado ao estudo da força de empuxo, com a construção de um simples dispositivo feito de garrafa PET e um Arduino Mega (pode ser feito com UNO, mas com limitações de entradas digitais), capaz de medir o empuxo sobre algum corpo.

Objetivos

- Compreender o conceito de empuxo.
- Relacionar densidade, aceleração gravitacional e volume deslocado no princípio de Arquimedes.

Fundamentos Teóricos

Segundo o princípio de Arquimedes:

Um corpo completamente (ou parcialmente) imerso em um fluido receberá a ação de uma força (empuxo) para cima igual ao peso do fluido deslocado.

As forças atuantes em um corpo imerso em um fluido são o peso do corpo e o empuxo (E). Como o peso do fluido deslocado (p_{desl}) determina a intensidade do empuxo, temos:

$$E = p_{desl} = m_{desl} \cdot g \tag{8.1}$$

A massa deslocada tem volume dado por:

$$m_{desl} = v_{desl} \cdot d \tag{8.2}$$

Logo o empuxo (E) em termos do volume deslocado (v_{desl}) é

$$E = v_{desl} \cdot d \cdot g \tag{8.3}$$

Na prática com este princípio é possível determinar a densidade de líquidos ou objetos. É possível determinar, inclusive, o peso (p) do objeto, quando o mesmo se encontra em equilíbrio, onde a força resultante é nula, ou seja o empuxo e o peso do objeto tem mesmo módulo,

$$p = E \tag{8.4}$$

Montagem

Utilize uma garrafa PET como recipiente. É importante que a garrafa seja cilindrica na região que será analisado o volume deslocado, normalmente alguns modelos tem a região central cilindrica.

Ver **figura 13** para montagem

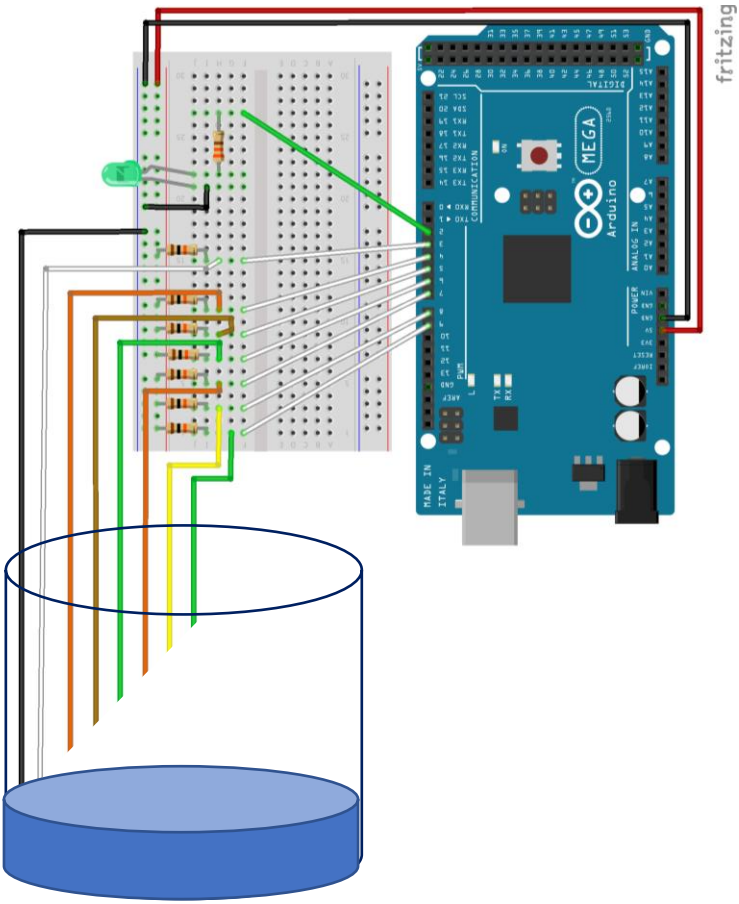


Figura 1 – Montagem para o Capítulo 8. Observe, conforme a imagem, a distância vertical entre os fios, fixa-los bem para evitar que mudem de posição durante manuseio.

Fonte: Imagem do autor extraída e editada no *Fritzing*.

Corte a garrafa na extremidade superior no ponto onde a mesma se afunila. Fixe os oito fios na garrafa, note que o fio preto e branco estão no mesmo nível, equivalente a posição zero na referência da régua. Os demais fios devem ter suas extremidades distantes de 0,5cm, na vertical. Para que os fio não mudem de posição use cola instântea em todo a extensão dos fios em contato com a garrafa.

Uma variação de meio centímetro no nível do líquido equivale a uma variação no volume de aproximademente 33 ml. Para garantir esses parâmetros utilize apenas garrafas com 9,5 cm de diametros. A montagem apresentada pode medir uma variação volumétrica de 0 a 200ml.

Materiais e componentes:

- Fios condutores compridos, cerca 40 cm.
- 7 resistores 10kΩ.
- 1 resistor 330Ω.
- 1 LED.
- Arduino Uno (ou Mega para usar mais entradas digitais).
- Protoboard.
- Garrafa PET.
- Régua.

Após a montagem dispeje água no recipiente até o nível inicial (0 cm). Assim, o experimento consiste em monitorar o volume de líquido deslocado quando um objeto for abandonado sobre o líquido. No monitor serial será apresentado o último volume deslocado e o empuxo sobre o objeto. Abandone sobre o líquido apenas objetos pequenos com volume máximo de 200ml, sem que enconste na garrafa. Para que a análise não apresente erros os objetos podem ser bexigas apenas com aguá (sem ar) e com volume de 33ml, 66ml, 100ml, 133ml, 166ml e 200ml.

A detecção do nível ocorre quando o líquido deslocado atinge o fio do respectivo nível, fechando o circuito, logo é necessário que o líquido conduza eletricidade. Para haver condução elétrica ponha um pouco de sal de cozinha na água.

O código

Abra o o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo do Principio de Arquimedes, ou seja,
 * aferição do empuxo sobre objetos que ocupam até 200ml de volume
total.
 */

int ledPin = 2;
int inPinUM = 3;          //nivel UM inicial 0cm  | volume deslocado 0
ml
int inPinDOIS = 4;         //nivel dois    0,5cm  | volume deslocado
~33 ml
int inPinTRES = 5;         //nivel tres    1,0cm  | volume deslocado
~66 ml
int inPinQUATRO = 6;      //nivel quatro   1,5cm  | volume deslocado
~100 ml
int inPinCINCO = 7;        //nivel cinco   2,0cm  | volume deslocado
~133 ml
int inPinSEIS = 8;         //nivel seis    2,5cm  | volume deslocado
~167 ml
int inPinSETE = 9;         //nivel sete    3,0cm  | volume deslocado
~200 ml
int valUM = 0;
int valDOIS = 0;
int valTRES = 0;
int valQUATRO = 0;
int valCINCO = 0;
int valSEIS = 0;
int valSETE = 0;
float empuxo;
void setup() {
  Serial.begin(9600);


Serial.println("................................................")
;
```

```
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("HIDROSTATICA: EMPUXO");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("Aferição do volume deslocado e o EMPUXO");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos
Serial.println("::INICIE O EXPERIMENTO::");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos

    pinMode (ledPin, OUTPUT);
pinMode (inPinUM, INPUT);
pinMode (inPinDOIS, INPUT);
pinMode (inPinTRES, INPUT);
pinMode (inPinQUATRO, INPUT);
pinMode (inPinCINCO, INPUT);
pinMode (inPinSEIS, INPUT);
pinMode (inPinSETE, INPUT);
}

void loop() {

// NIVEL  UM 0 ml
valUM = digitalRead(inPinUM);

if (valUM == HIGH){
  digitalWrite(ledPin, LOW);
  Serial.println("Atencao liquido Abaixo do Nivel complete o
Recipiente!");

  } else {
   digitalWrite(ledPin, HIGH);

Serial.println("..................................................")
;
    Serial.println("Tudo Pronto! Abandone Um objeto na superficie do
Liquido.");

Serial.println("..................................................")
;
    }

// NIVEL DOIS 33,3 ml
```

```
valDOIS = digitalRead(inPinDOIS);

if (valDOIS == HIGH){
    Serial.println(".");
  } else {

Serial.println("................................................")
;
    Serial.print("ULTIMO VOLUME DETECTADO =");
    Serial.println(" 33 mililitros");
    Serial.print("EMPUXO =");
    Serial.println(" 0,33 newtons");

Serial.println("................................................")
;
      }

    // NIVEL  TRES 66,7 ml
valTRES = digitalRead(inPinTRES);

if (valTRES == HIGH){
    Serial.println(".");
  } else {
Serial.println("................................................")
;
    Serial.print("ULTIMO VOLUME DETECTADO =");
    Serial.println(" 66,7 mililitros");
    Serial.print("EMPUXO =");
    Serial.println(" 0,65 newtons");

Serial.println("................................................")
;
      }

// NIVEL QUATRO 100 ml
valQUATRO = digitalRead(inPinQUATRO);

if (valQUATRO == HIGH){
    Serial.println(".");
  } else {

Serial.println("................................................")
;
    Serial.print("ULTIMO VOLUME DETECTADO =");
    Serial.println(" 100 mililitros");
    Serial.print("EMPUXO =");
    Serial.println(" 0,98 newtons");

Serial.println("................................................")
;
    }

    // NIVEL CINCO 133 ml
valCINCO = digitalRead(inPinCINCO);

if (valCINCO == HIGH){
```

```
    Serial.println(".");
  } else {

Serial.println("................................................")
;
    Serial.print("ULTIMO VOLUME DETECTADO =");
    Serial.println(" 133,3 mililitros");
    Serial.print("EMPUXO =");
    Serial.println(" 1,3 newtons");

Serial.println("................................................")
;
    }

    // NIVEL SEIS ~167 ml
valSEIS = digitalRead(inPinSEIS);

if (valSEIS == HIGH){
    Serial.println(".");
  } else {

Serial.println("................................................")
;
    Serial.print("ULTIMO VOLUME DETECTADO =");
    Serial.println(" 167");
    Serial.print("EMPUXO =");
    Serial.println(" 1,63 newtons");

Serial.println("................................................")
;
    }

    // NIVEL  SETE ~200 ml
valSETE = digitalRead(inPinSETE);

if (valSETE == HIGH){
    Serial.println(".");
  } else {

Serial.println("................................................")
;
    Serial.print("ULTIMO VOLUME DETECTADO =");
    Serial.println(" 200 mililitros");
    Serial.print("EMPUXO =");
    Serial.println(" 1,9 newtons");

Serial.println("................................................")
;
    }


delay(2000);
}
```

- **Execução e coleta de dados**

Após a montagem e upload do código inicie o monitor serial. Caso o nível inicial do líquido não esteja correto a seguinte mensagem será exibida:

Atenção líquido Abaixo do Nível complete o Recipiente!

Se o nível estiver correto, a mensagem a ser exibida será:

Tudo Pronto! Abandone Um objeto na superfície do Líquido.

Separe os objetos e abandone-os individualmente no recipiente. Observe, no monitor serial, que serão exibidos o último volume deslocado e o empuxo sobre o objeto. Registrar na **Tabela 5**. Não revele nem o volume nem a massa dos objetos aos seus alunos, deixe que eles determinem estes valores nos questionamentos a partir das medidas apresentadas no monitor serial.

Tabela 5 - Modelo de tabela para registro dos dados.

Objeto 1	
Volume deslocado:_____	Empuxo:_____
Objeto 2	
Volume deslocado:_____	Empuxo:_____
Objeto 3	
Volume deslocado:_____	Empuxo:_____
Objeto 4	
Volume deslocado:_____	Empuxo:_____
Objeto 5	
Volume deslocado:_____	Empuxo:_____
Objeto 6	
Volume deslocado:_____	Empuxo:_____

Fonte: do autor.

Questionamentos e levantamento de hipóteses

1. *Com base nos dados coletados determine a densidade do líquido.*
2. *É possível calcular a massa dos objetos a partir dos dados apresentados? Em que circunstância? Se possível calcule cada massa.*
3. *Reúna-se com seus colegas e pensem em possíveis aplicações que utilizem a montagem apresentada, registre-as.*

9. Escalas Termométrica

A aferição da temperatura de um corpo ou mesmo do ambiente requer a utilização de algum instrumento que apresente variação em alguma propriedade física quando sua temperatura também varia. Em virtude da natureza das propriedades físicas, há vários tipos de termômetros, que diferem quanto ao tipo de grandeza termométrica, por exemplo, termômetros de mercúrio, cuja grandeza termométrica corresponde a dilatação ou contração térmica; termômetros digitais, que possuem uma resistência elétrica que varia em função da temperatura; há ainda termômetros Ópticos que medem a radiação térmica da matéria. Neste capítulo iremos construir um termômetro que fornece a temperatura nas escalas Celsius, Fahrenheit e Kelvin.

Objetivos

- Analisar o funcionamento de termômetros baseados em resistência variável
- Relacionar as principais escalas termométrica

Fundamentos Teóricos

Uma das escalas de temperatura mais utilizadas é a escala Celsius, baseada na temperatura do ponto de fusão do gelo (0°) e de ebulição da água (100°). Há ainda a escala Fahrenheit e Kelvin, também muito conhecidas, a última mais no domínio da comunidade científica.

Vejamos algumas relações matemáticas entre as escalas citadas.

Celsius x Fahrenheit:

$$\frac{T_C}{5} = \frac{T_F - 32}{9} \qquad (9.1)$$

Celsius x Kelvin:

$$T_K = T_C + 273 \qquad (9.2)$$

Montagem

Neste experimento usa-se como sensor de temperatura um resistor cuja resistividade é sensível à temperatura, esse componente eletrônico é conhecido como termistor, que normalmente pode medir temperatura na faixa de -55°C a 125°C.

Materiais e componentes:

- Arduino
- 4 Resistores 10kΩ
- 3 Chave Táctil Push-Button
- Termistor
- Fios jumpers

Siga o esquema apresentado na montagem (**Figura 14**). Para facilitar a realização de medidas é importante que os dois fios condutores ligados ao termistor sejam longos o suficiente.

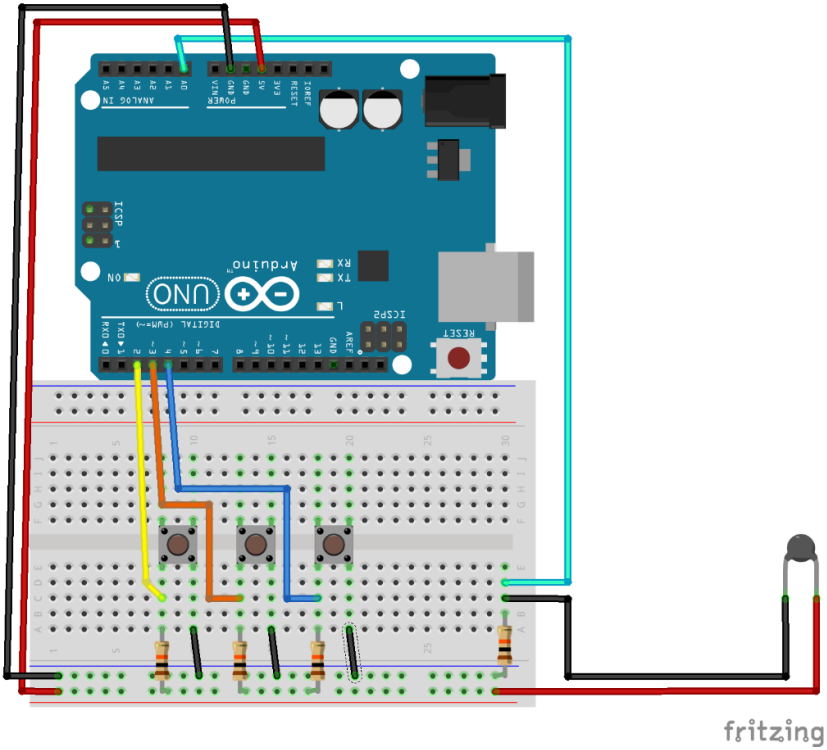


Figura 14 - Conforme a montagem e as instruções no *sketch* ao pressionar os botões (chave tátil) da esquerda para a direita, se obtém a temperatura na escala Celsius, Fahrenheit e Kelvin.

Fonte: Imagem do autor extraída e editada no Fritzing.

O código

Abra o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado ao estudo das escalas termométrica.
```

```
 * Usando um termistor de 10k para aferição da temperatura
 * em Celsius, Fahrenheit ou Kelvin.
 */
#include <Thermistor.h> // caso não tenha adicionado esta biblioteca,
adicione em Sketch-> incluir biblioteca-> adicionar biblioteca .ZIP
#define inPinUM 2
#define inPinDOIS 3
#define inPinTRES 4
Thermistor temp(0);
int tempF;
int tempK;
int valUM;
int valDOIS;
int valTRES;

void setup() {
Serial.begin(9600);
Serial.println("..................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("Escalas Termometricas");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("Medindo temperatura");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos
Serial.println("::::::::::::Primeiro Botao-> Escala Celsius::::::::::
");
Serial.println("::::::::::::Segundo Botao-> Escala Fahrenheit::::::::
");
Serial.println("::::::::::::Terceiro Botao-> Escala Kelvin::::::::::::
");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos

pinMode(inPinUM, INPUT);
pinMode(inPinDOIS, INPUT);
pinMode(inPinTRES, INPUT);
}

void loop()
{

int tempC = temp.getTemp(); //pega a temperatura na escala Celsius
```

```
tempF= 32 + (tempC*9)/5;        //temperatura em Fahrenheit
tempK= tempC+273;            // temperatura em Kelvin

valUM = digitalRead(inPinUM);
valDOIS = digitalRead(inPinDOIS);
valTRES = digitalRead(inPinTRES);
if( valUM == LOW){

Serial.print("Temperatura em graus Celsius = ");
Serial.println(tempC);

  } else{
    Serial.println("...");
    }

if( valDOIS == LOW){

Serial.print("Temperatura em graus Fahrenheit = ");
Serial.println(tempF);

  } else{
    Serial.println("...");
    }

if( valTRES == LOW){
Serial.print("Temperatura em Kelvin = ");
Serial.println(tempK);

  } else{
    Serial.println("...");
    }

delay(1000);

}
```

Para que este programa funcione é necessário manter a biblioteca "Thermistor.h" que já está incluída no código, porem a mesma deve estar devidamente instalada no software IDE, para fazer isso baixe a biblioteca em www.fisicarduino.com.

Feito download instale a biblioteca em:

Sketch > Inclui Biblioteca > Adicionar biblioteca .ZIP.

Observe na **figura 15**.

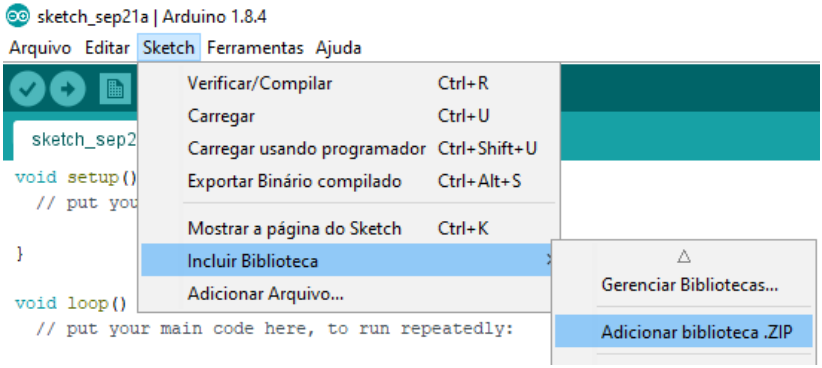


Figura 15 - Procedimento para inclusão da biblioteca Termistor.h. Porem antes deste procedimento baixe a biblioteca disponível em www.fisicarduino.com.

Fonte: do autor.

Execução e coleta de dados

Após fazer o upload do código será possível medir a temperatura de objetos e do ambiente. Abra o monitor serial para acompanhar a medidas de temperatura. O dispositivo mostrará a temperatura em uma das três escalas termométricas aqui apresentadas, Celsius, Fahrenheit ou Kelvin, para isso basta pressionar a chave tátil correspondente. A primeira chave exibirá a temperatura em graus Celsius, a segunda em fahrenheit e a terceira em kelvin. Para entender as relações entre as escalas faça o registro (**usar Tabela 6**) da temperatura ambiente nas três escalas:

Tabela 6 - Modelo de tabela para registro dos dados do Capítulo 9.

Escala termométrica	Registro de temperatura
Celsius	
Fahrenheit	
Escala Kelvin	

Fonte: do autor.

A exibição da temperatura ocorrerá no monitor serial, mas caso você queira deixar o dispositivo mais portátil, é possível usar uma bateria como fonte de alimentação, além de um display de LCD. Para uso do LCD é necessário também inclusão da biblioteca `"LiquidCrystal.h"`, além das configuração necessárias para que os valores sejam imprimidos no LCD.

Questionamentos e levantamento de hipóteses

Verifique se as temperaturas registradas pelo dispositivo obedecem às relações matemáticas apresentadas nos fundamentos teóricos deste capítulo.

10. Lei Zero da Termodinâmica

A ideia para a criação de uma escala termométrica apoia-se em pressupostos da Lei Zero da termodinâmica, no cerne desta lei encontra-se o significado de equilíbrio térmico, útil na determinação de parâmetros termométricos. A automação no experimento deste capítulo permitirá a verificação da Lei zero.

Objetivos
- Comparar temperaturas entre corpos.
- Verificar a Lei zero da Termodinâmica.

Fundamentos Teóricos

"Se dois corpos A e B estão em equilíbrio térmico com um terceiro corpo T, então, estão em equilíbrio térmico um com o outro" (Halliday, Resnick, & Walker, 1996).

Por ser um conceito fundamental, e naturalmente compreendido pelos cientistas, já com a vigência da Primeira e Segunda Lei da termodinâmica, o princípio foi definido como Lei zero da Termodinâmica.

O equilíbrio térmico entre dois corpos ocorre quando não há mais troca de calor entre eles, o que coincide quando suas temperaturas são equivalentes. Assim um termômetro, quando utilizado para medir a temperatura de uma pessoa, deve ser posto em contato com a pele durante um certo intervalo de tempo até que ocorra equilíbrio térmico entre o termômetro e pele da pessoa, o que será registrado no termômetro será uma temperatura igual à da pessoa.

Montagem

Observe a simplicidade da montagem para realizar as medidas de comparação de temperatura, fixe os dois termômetros (sensores LM35) em uma superfície, local onde os

corpos usados para comparação de suas temperaturas serão colocados. Veja esquema na **Figura 16**.

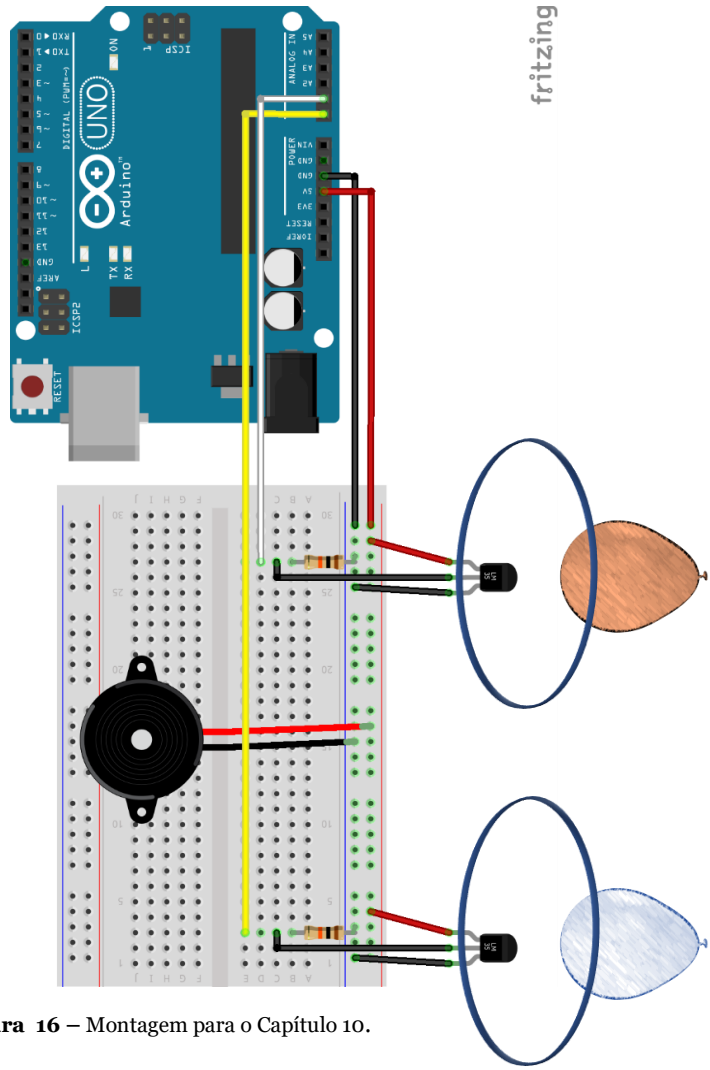


Figura 16 – Montagem para o Capítulo 10.

Fonte: Imagem do autor extraída e editada no Fritzing.

Materiais e componentes:

- 01 Arduino
- 02 LM35
- 02 Resistores 1kΩ
- Buzzer
- Protoboard
- Fios jumpers
- Balões com água

Os dois corpos, A (balão azul) e B (balão laranja) por exemplo, devem estar a princípio com temperaturas diferentes, para o estudo do equilíbrio térmico. Use *Jumpers* longos entre os sensores e a *protoboard*, o que facilitará a manipulação e posicionamento.

O código

Abra o o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo do equlíbrio térmico entre corpos
 * ou seja a Lei zero da Termodinamica
 */

const int LM35A = A0; // Define o pino que lera a saída do LM35 A
const int LM35B = A1; // Define o pino que lera a saída do LM35 B
float temperaturaA; // Variável que armazenará a temperatura A medida
float temperaturaB; // Variável que armazenará a temperatura B medida
float seno;
const int buzzer = 9;
int frequencia;

void setup() {
Serial.begin(9600); // inicializa a comunicação serial
pinMode(9,OUTPUT);
Serial.println(".................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
```

```
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("LEI ZERO DA TERMODINAMICA");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("Determinacao do equilibrio termico entre dois corpos A
e B");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos
Serial.println("::POSICIONE OS DOIS CORPOS SOBRE OS TERMOMETROS::");
Serial.println("..................................................")
;
Serial.println("::AGUARDE A CALIBRAGEM DOS TERMOMETROS::");
delay(10000); //espera para divergencia inicial da temperatura entre
termometros
Serial.println("::TUDO PRONTO! QUANDO OCORRER O EQUILIBRIO TERMICO UM
ALARME SOARA::");
delay(60000);
}


void loop() {

temperaturaA = (float(analogRead(LM35A))*5/(1023))/0.01;
temperaturaB = (float(analogRead(LM35B))*5/(1023))/0.01;
if(abs(temperaturaA - temperaturaB)<= 2 ){
Serial.println("Corpo A em equilíbrio térmico com o corpo B");
for(int x=0;x<180;x++){
seno=(sin(x*3.1416/180));
  //gera uma frequência a partir do seno
frequencia = 2000+(int(seno*1000));
 tone(buzzer,frequencia);
 delay(2);
 }
 }
 else{
   noTone(buzzer);
    }
}
```

Execução e coleta de dados

Separe dois objetos com temperaturas diferente, pode ser dois balões com o mesmo volume de água. Deixe um dos balões por alguns minutos em um refrigerador, o outro à

temperatura ambiente. Quando o código for descarregado no Arduino e o monitor serial for iniciado, o programa solicita que dois corpos A e B sejam posicionados próximos aos termômetros. É mportante, agora, colocar os dois balões em um local isolado termicamente do meio externo, um caixa de isopor por exemplo. O objetivo é mostrar quando ocorre o equilíbrio térmico entre os dois corpos. Neste experimento os dois corpos trocarão calor entre si e com a caixa, quando as temperaturas dos corpos forem iguais conclui-se que estarão em equilíbrio térmico, neste momento o *buzzer* emitirá um som de sirene.

Questionamentos e levantamento de hipóteses

1. *Em caso da utilização de volumes menores, o tempo para que ocorra o equilíbrio térmico ocorre também é menor?*
2. *Como os dois sensores podem medir temperaturas de forma independente, reflita sobre possíveis aplicações que solucionaria problemas do cotidiano. Apresente sua reflexão ao professor e lembre-se, seu desenvolvimento pode torna-se em um excelente projeto para feiras de ciências.*

11. Primeira e Segunda Lei de Ohm

Compreender a natureza da resistência elétrica em um material é crucial para o desenvolvimento de novos componentes eletrônicos ou mesmo torna-los mais eficientes. Algumas pesquisas sobre esse tema se voltam para a precisão de componentes eletrônicos e sensores. Já outras investigações buscam o desenvolvimento de materiais cerâmicos que não apresentam resistividade elétrica, propriedade que define os supercondutores, importantes para a criação de circuitos que não perdem energia por *efeito Joule*. Neste capítulo será possível compreender um pouco mais sobre a resistência elétrica. Os fundamentos teóricos a serem discutidos referem-se aos esforços do físico alemão George Simon Ohm sobre a condução elétrica.

Objetivos

- Analisar comportamento dos resistores ôhmicos.
- Compreender as relações de proporcionalidades entre correntes, tensão e resistência elétrica.

Fundamentos Teóricos

Os estudos de Georg Simon Ohm foram importantes para a compreensão da resistência elétrica nos materiais, que *a priori* tratam-se das dificuldades de movimento, enfrentada pelos elétrons nos condutores, quando submetido a uma diferença de potencial. Cada material apresenta uma resistência distinta que é calculada pela razão entre a tensão (U) e a corrente elétrica(i) no material:

$$R = \frac{U}{i} \qquad (11.1)$$

As unidades de medidas de cada grandeza para tensão, corrente e resistência elétrica, corresponde ao volt(V), ampère(A) e ohm(Ω), respectivamente.

Para alguns materiais, a resistência revela-se constante ao variar até certo limite a tensão elétrica do mesmo. Esse fenômeno é conhecido como a Primeira lei de Ohm, ou seja, quando dizemos que um condutor obedece a Primeira lei de Ohm, entendemos que a resistência do mesmo é constante, consequentemente um gráfico da *Corrente* x *tensão* para esse resistor apresentam uma certa linearidade.

A Segunda lei de Ohm relaciona as características físicas que determinam a resistência de um condutor:

$$R = \rho \frac{L}{A} \qquad (11.2)$$

Portanto, a resistividade elétrica (ρ) do tipo de material, a área da seção transversal (A) do condutor e seu comprimento(L) determinam sua resistência elétrica. O conhecimento desta relação é extremamente útil quando for conveniente alterar a resistência elétrica em uma parte de um circuito elétrico. No experimento a seguir usaremos um potenciômetro que ajudará comprovar a segunda lei e um resistor para a primeira.

Montagem

Observe a disposição de cada componente na **Figura 17**. Os elementos que serão analisados serão o resistor, que está associado em paralelo com o LED, e o potenciômetro. A análise do resistor consistirá no registro das tensões e da corrente para averiguação da Primeira lei de Ohm. O potenciômetro, tem sua resistência modificada quando alterada a posição angular do seu eixo de rotação. Essa modificação ocorre devido a variação no comprimento do condutor. Nesse caso, o potenciômetro demostrará aplicabilidade da Segunda lei de Ohm.

Materiais e componentes:

- Arduino
- Fios jumpers
- 2 resistores de 330Ω (laranja-laranja-marrom)
- 01 Potenciômetros 10kΩ
- 01 LED
- Multímetro (opcional)

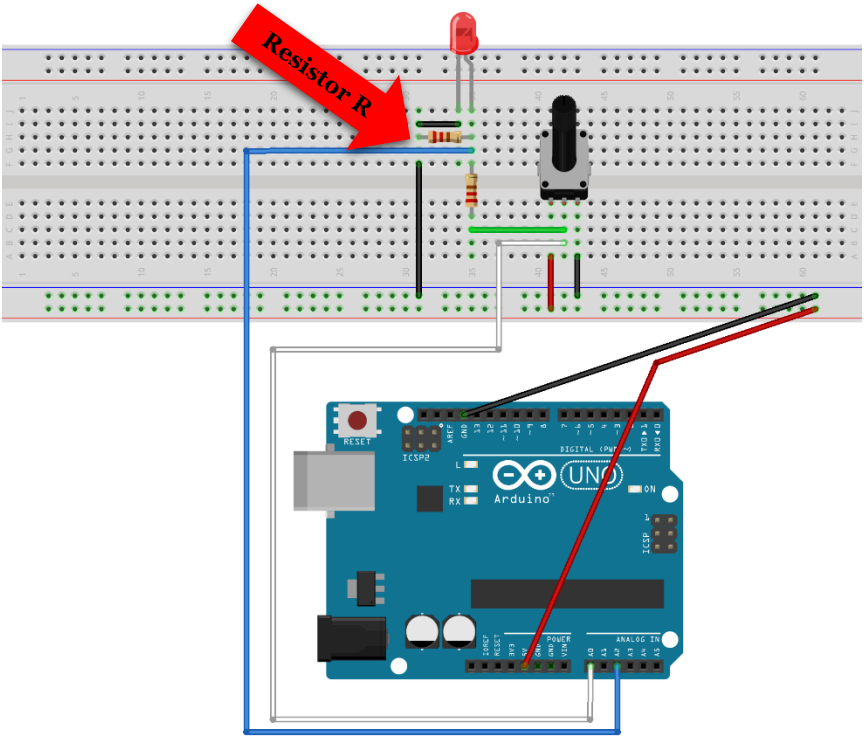


Figura 17 - Esquema de montagem para demonstração da 1º e 2ª Lei de Ohm.

Fonte: Imagem do autor extraída e editada no Fritzing.

O código

Feita a montagem abra o software IDE do Arduino e copie e cole o código a seguir no Software. Uma outra opção é baixar a *sketch* em *www.fisicarduino.com* e executá-lo em seu computador. Observe os comentários no programa (texto precedido de //), pois contêm informações sobre as funções utilizadas e sugestão de edição no programa para algum ajuste necessário. Por exemplo, a o valor da resistência R na linha 29 deve ser igual a resistência do resistor utilizado, o sugerido é o resistor de 330 Ohm, mas caso seja utilizado um resistor com outro valor, faça a alteração. Para uma maior precisão determine a resistência do resistor R usando o multímetro, e altere o valor no programa (linha 29).

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo das Leis de Ohm
 */

#define potenPin 0    // define a entrada analógica para o
Potenciômetro
#define voltPin 2     // define a entrada analógica para o resistor
float Rx;             // variável que armazena valor da resistência em
OHM do potenciômetro 0 a 10k
float R0;             // variável que armazena valor da resistência
float corrente;       // variável para armazenar valor da corrente
float voltvalor;      // cria variável para armazenar o valor da tensão
do RESISTOR observado
float potenvalor;     // cria variável para armazenar valor da
resistência 0 a 1023.
float U;
float Ux;

void setup()
{

Serial.begin(9600); //inicial a serial
Serial.println("..................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
```

```
delay(2000);//Pausa de 2 segundos
Serial.println(".................................................")
;
Serial.println("PRIMEIRA E SEGUNDA LEI DE OHM");
Serial.println(".................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println(".................................................")
;
R0 = 330; // Você pode editar o valor de R. resistencia em Ohms para
saber o valor exato use um ohmimetro, ou veja o código de cores para
valor aproximado
Serial.println("Medidas para um Resistor de 330");
Serial.println(".................................................")
;
delay(2000); //Pausa de 2 segundos
}


void loop()
{

voltvalor = analogRead(voltPin);// Lê o valor registrado no resistor e
armazena na variável "voltvalor"
potenvalor = analogRead(potenPin);// Lê o valor registrado no
potenciometro e armazena na variável "potenvalor"
  U = voltvalor/204.6;
  corrente = U/R0;//relação matemática para 1ª lei de Ohm.
  Rx = -((potenvalor/102.3)-10);
Serial.println(".................................................");
Serial.print("Resistencia do potenciometro em quiloohm Rx="); //Imprime
na serial O TEXTO ENTRE ASPAS.
Serial.println(Rx);  //Imprime na serial o valor da resistencia do
potenciometro

Serial.print("Tensao (em Volts) no Resistor R U= "); //Imprime na
serial "tensão(em volts) = "
Serial.println(U);  //Imprime na serial o valor lido
Serial.print("Corrente (em Ampere) no Resistor R  i= ");

Serial.println(corrente, 6); //Imprime na serial o valor lido com 6
casas decimais
Serial.println(".................................................")
;

delay(2000); //Aguarda 2 segundos



}
```

Execução e coleta de dados

Feito todos os procedimentos conecte seu Arduino ao computador e faça o *upload* do código. Abra o Monitor Serial (crtl+shift+m) e verifique se os valores são mostrados. Se tudo estiver correto ao findar o upload e iniciar o monitor serial será exibido informações semelhantes a estas:

Quadro 3 - Na interface do monitor serial será exibido os valores da resistência variável do potenciômetro, da tensão e da corrente elétrica no resistor analisado. DICA: pressione as teclas crtl+shift+m do seu computador para exibir a interface.

```
.....................................................................
MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA - MNPEF
PRODUTO DESENVOLVIDO POR: OSEIAS MOURÃO
.....................................................................
.....................................................................
PRIMEIRA E SEGUNDA LEI DE OHM
.....................................................................
.....................................................................
Medidas para um resistor de 330 ohm
.....................................................................
.....................................................................
Resistencia do potenciometro em quiloohms Rx=
Tensão (em volts) no resistor R.   U=
Corrente (em ampère) no resistor R. i=
.....................................................................
```

fonte: do autor

Os valores que serão exibidos a cada 2 segundos serão: a resistência Rx do potenciômetro em quiloohms(kΩ); a tensão U (em volts) no resistor R; e a corrente i (em ampère) no resistor R, conforme **Quadro 3**

Enquanto o Monitor exibe os valores, gire cuidadosamente o potenciômetro variando sua resistência, partindo do seu mínimo até o máximo. Registre na **tabela 7** a seguir os valores exibidos no monitor. Depois de fixada a posição do potenciômetro, caso as medidas oscilem, anote os valores que mais se repetem, ou para maior precisão, calcule a média aritmética dos valores.

Tabela 7 - Use esta tabela para anotar os dados que serão exibidos no monitor serial.

Ajuste o potenciômetro para estas resistências	1	2	3	4	5	6	7	8	9
Medidas da tensão (u) para o resistor **R**									
Medidas da corrente (i) para o resistor **R**									

Fonte: do autor.

Questionamentos e levantamento de hipóteses

1. *Elabore um gráfico U x i para o Resistor R e verifique se a 1ª lei de Ohm é satisfeita. Explique.*
2. *Observe que a rotação do eixo do potenciômetro implica na variação da tensão e da corrente do resistor R. As verificações dos valores dessas grandezas podem ser autenticadas com o uso de um multímetro, ou simplesmente percebendo a variação da intensidade luminosa do LED. Portanto, porque a variação da posição angular do eixo de rotação do potenciômetro causa alterações na tensão do resistor R?*
3. *Apresente uma proposta para um projeto de ciência em que se possa aplicar conhecimentos sobre resistência elétrica.*

12. Fotorresistividade de um semicondutor

Como vimos no capítulo anterior, cada tipo de material pode oferecer certa dificuldade a passagem da corrente elétrica, isso caracteriza a grandeza Física denominada de resistência elétrica. Sabendo que as dimensões e resistividade de um material condutor são preponderantes no valor da resistência, alguns componentes foram desenhados permitindo variações em características específicas, como nas dimensões, no caso de um potenciômetro, ou na resistividade como ocorre nos termistores, varistores e fotorresistores. Normalmente a aplicação destes componentes está relacionada a algum tipo de sensoriamento. Será descrito a seguir o funcionamento de um tipo de fotorresistor e como a intensidade de luz poderá afetá-lo.

Objetivos

- Verificar propriedades fotorresistivas de um semicondutor
- Fortalecer o conhecimento sobre resistência elétrica;
- Analisar relações de proporcionalidade entre medidas.

Fundamentos Teóricos

Um resistor dependente de luz (do inglês LDR) é constituído basicamente de um material semicondutor cuja resistência é determinada pela intensidade da luz incidente. O LDR apresenta uma resistência elétrica mínima quando exposto a uma alta intensidade luminosa e uma resistência elétrica máxima na ausência de luz. A resistência de um LDR é alterada em função da variação da resistividade do material semicondutor que o constitui, a explicação fundamental para essa alteração é o efeito fotoelétrico que consiste na emissão

de elétrons em um metal, quando exposto a certas frequências de luz. No LDR os elétrons não chegam a ser emitidos do material, mas a energia dos fótons de luz é suficiente para que os elétrons saiam da banda de valência para a banda de condução, o que implica na diminuição da resistência elétrica do material; ou seja, quanto maior a intensidade de luz menor será a resistência do material.

Montagem

Faça as conexões dos componentes conforme são descritas na montagem. As Conexões entre as duas *protoboards* devem ser feitas com fio de aproximadamente 20 cm, para que seja possível a variação da distância entre o LED e o LDR.

Observe que o LDR e o resistor estão em série e na divisão de tensão relacionada a um dos terminais do LDR é inserido um condutor conectado a porta analógica zero (`A0`) do Arduino (**Figura 18**). Essa porta receberá os valores referentes a tensão elétrica entre os terminais do LDR. Os valores indicam a medida da intensidade luminosa incidente sobre o LDR. Ao preparar a montagem é importante que o dispositivo fique em um local escuro, onde a luz mais intensa seja a do LED.

Materiais e componentes:

- LDR
- LED
- Resistor 10kΩ
- Resistor 330Ω
- Arduino
- 02 Protoboard
- Jumpers
- Régua ou trena

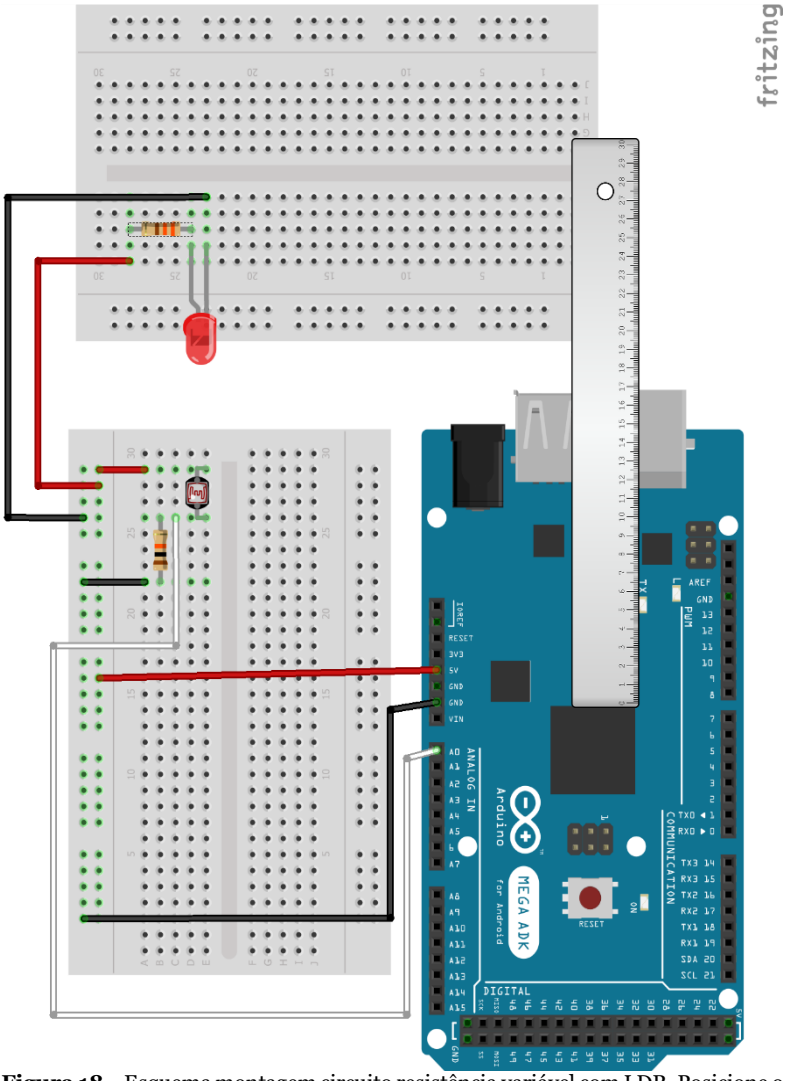


Figura 18 - Esquema montagem circuito resistência variável com LDR. Posicione o LDR e o LED frente à frente e varie a distância entre ambos conforme indicação da tabela 8.

Fonte: Imagem do autor extraída e editada no Fritzing.

O código

Abra o software IDE do Arduino e digite o código a seguir:

```
/*MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA-SBF
 * INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DO CEARÁ-IFCE
 * UNIVERSIDADE ESTADUAL VALE DO ACARAÚ-UVA
 * PRODUTO DESENVOLVIDO POR: OSÉIAS DE SOUSA MOURÃO
 *
 * Programa destinado para estudo da fotorresistiidade de um
semicondutor.
 * O programa permite perceber a variação da resitência elétrica em
materiais
 * semicondutores quando variada a intensidade luminosa.
 * utiliza-se no Arduino um LDR e LED para verificar essa variação.
 */

int ldrPin = 0; // indica que o LDR deve esta no pino analógico 0
int ldrValor = 0; // armazena medidas do LDR

 void setup()
{
Serial.begin(9600);

digitalWrite(13,LOW);
Serial.println("..................................................")
;
Serial.println("MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FISICA -
MNPEF");
Serial.println("PRODUTO DESENVOLVIDO POR: OSEIAS MOURAO");
Serial.println("..................................................")
;
delay(2000);//Pausa de 2 segundos
Serial.println("..................................................")
;
Serial.println("FOTORRESISTIVIDADE DE UM SEMICONDUTOR");
Serial.println("..................................................")
;
delay(2000); //Pausa de 2 segundos
Serial.println("::INICIE O EXPERIMENTO::");
Serial.println("..................................................")
;
delay(1000); //Pausa de 1 segundos
digitalWrite(13,HIGH);
}

void loop() {
ldrValor = analogRead(ldrPin); // lê o valor do LDR
delay(1000); // tempo de espera de 1 segundo
Serial.println(ldrValor);

}
```

De acordo com a programação definimos a variável `ldrValor` para armazenar a medida vinculada à resistência, vinculada porque não é literalmente a medida da resistência. O valor lido na porta analógica expressa a tensão elétrica entre os terminais do LDR explicitado de 0 a 1023 (0 a 5 volts). Para entender melhor essa relação veja o capítulo 1 na seção sobre a resolução do conversor analógico-digital do Arduino. O comando `Serial.println` imprimirá o valor do LDR no Monitor Serial, cuja medida será proporcional à resistência elétrica do mesmo.

Execução e coleta de dados

Feito todos os procedimentos conecte seu Arduino ao computador e faça o *upload* do código. Abra o Monitor Serial (crtl+shift+m) e verifique se os valores são mostrados. Se tudo estiver correto a cada 1 segundo o valor do LDR será atualizado em uma nova linha.

Posicione as *protoboards* de tal forma que o LDR e o LED fiquem sempre alinhados e anote os valores do LDR registrados no Monitor Serial para cada distância que você escolheu, conforme **Tabela 8**:

Tabela 8 - Modelo de tabela para registro dos dados do Capítulo 12

REGISTROS	**01**	**02**	**03**	**04**	**05**	**05**	**07**	**08**	**09**	**10**
DISTÂNCIAS ENTRE LDR E LED										
VALORES DO LDR (Ver monitor serial)										
MÉDIA										

Nota. As distâncias devem ser determinadas pelo professor. Para facilitar a análise é importante adotar distâncias de até 2 metros.
Fonte: do autor.

Questionamentos e levantamento de hipóteses

1. *Elabore um gráfico com os dados coletados relacionando os valores do LDR com a distância do LED.*
2. *Você deve ter verificado que a resistência do LDR (valores do LDR) mudou com a variação da intensidade da luz ao mover o LED. Qual a relação de proporcionalidade entre a intensidade da luz e a resistência do LDR?*
3. *Reflita sobre possíveis aplicações no dia-a-dia usando esse recurso da variação da resistência elétrica em um circuito baseado na variação da intensidade luminosa. Levante novas hipótese. Comente a respeito e, se possível, crie com a ajuda do seu professor algum projeto de pesquisa; podendo inclusive expor suas descobertas ou propostas de intervenção em eventos públicos, como em feiras de ciências por exemplo.*

Orientações Didáticas

A partir de experiências pedagógicas obtidas em aulas de Robótica Educacional ministradas aos alunos da escola de Ensino Médio Liceu de Tianguá José Ni Moreira[7], notou-se a conveniência da motivação no processo de ensino. Como um elemento propulsor constante, a motivação sempre exige uso de ferramentas que suportem esse elemento. Notou-se, a princípio, que o uso constante de interfaces eletrônicas na montagem de projetos despertava a curiosidade, impelindo os alunos a concretizarem suas tarefas. O caráter livre da condução das atividades foi um fator preponderante, uma vez que o manuseio dos equipamentos eletrônicos, os testes com sensores e atuadores exigiam certa liberdade.

As experiências foram inspiradoras na formatação deste Produto Educacional, produto este, que usa os mesmos elementos das aulas de robótica da referida escola, no entanto, adaptados à construção e reconfiguração de experimentos de Física. Será apreciado neste recurso a implementação em sala de aula dos experimentos automatizados, os aspectos estruturais do produto e orientações pertinentes à aplicação.

Antes de prosseguirmos é importante advertir sobre a natureza desta seção, que consiste essencialmente em um texto descritivo sobre condições, orientações e aplicabilidade dos experimentos. A natureza dos experimentos deste produto harmoniza-se muito bem com concepções pedagógicas pragmáticas. Não ocorrendo, portanto, inovação teórica no campo da educação, a não ser na automatização de práticas de conteúdos de Física.

Cada capítulo do Produto Educacional começa com uma introdução contextualizando o tema de Física com situações do

[7] Escola pública da rede estadual de Educação do Estado do Ceará, localizada na cidade de Tianguá, na região norte do Estado.

cotidiano ou enfatizando o tema que será abordado. Após a introdução, os fundamentos teóricos de Física, relacionados à temática do capítulo, são apresentados. Essas partes dos Capítulos servem de subsídio à aula do professor. É importante que o professor reforce tais fundamentos, antes da execução da atividade.

Um quadro, posterior à introdução do capítulo, contém os objetivos que poderão ser atingidos no desenvolve das atividades. O propósito é promover orientação ao plano de aula do professor, ou simplesmente permitir que o professor verifique se há consonância com um plano de ensino já estabelecido.

Na fundamentação é considerado apenas os conceitos e princípios mais relevantes, essenciais à compreensão do tema. Ainda sobre os aspectos da teoria, as fórmulas exibidas neste ponto são apresentadas com o objetivo de permitir que o aluno desenvolva a capacidade de relacionar as grandezas físicas de forma conveniente. Esta etapa, proporciona ao aluno o domínio expressivo do conteúdo; para o professor, proporciona um plano imediato, complementar a sua prática. As expressões matemáticas contidas nesta etapa também são relevantes, ao professor ou aluno que queira apropriar-se dos conteúdos dos *sketches*, a fim de editá-los, no ensejo de alterar algum parâmetro experimental. Logo, compreender os códigos contidos nos *sketches* para possíveis alterações posteriores da atividade exige entendimento matemático das fórmulas apresentadas.

Antes das orientações sobre a montagem, é listado os materiais e componentes necessários. A placa Arduino UNO, *protoboard,* e *jumpers* para a conexão entre os pinos são comuns a todos os experimentos. Durante a montagem é importante recorrer a essa lista para verificar as características dos componentes, como exemplo, o valor da resistência de um resistor ou qual tipo de transistor.

A etapa de montagem do experimento exige bastante atenção do professor ou do aluno. Por isso a seção referente à montagem contém instruções essenciais, além da figura desenvolvida apropriadamente ao experimento, através do

software *Fritzing*. A figura representa uma montagem experimental atestada durante o desenvolvimento do produto. Obviamente, a montagem dever ser executada fielmente para sucesso do experimento, entretanto, com prática e conhecimento das funcionalidades da *protoboard* é possível realizar montagens diferentes para o mesmo experimento, no caso de melhorias ou para obtenção de novos dados experimentais da atividade.

O código é exibido no Produto Educacional, permitindo que o professor faça uso livre do mesmo. Quando toda a montagem for feita resta enviar o programa para o Arduino, cujo código pode ser copiado para um novo *sketch*. Para facilitar esse processo o código é apresentado como o conteúdo de uma tabela. E como mais uma forma de acesso, minimizando possíveis problemas de formatação do texto do código no ato de copiar e colar, o *sketch* referente aos experimentos poderá ser baixado em *fisicarduino.com*, site de divulgação do Produto Educacional.

A etapa de Execução e Coleta de Dados, consiste em instruir quanto à interação do usuário (professor/alunos) com o programa e dispositivos associados ao Arduino. Há, por exemplo, experimentos que necessitam de calibração prévia, logo essa etapa cuida em fornecer informações necessárias para a calibração. Nessa mesma etapa do Produto há uma seção para coleta de dados, que especifica como e o que deve ser coletado no experimento. Normalmente os sensores usados gerarão dados que serão exibidos no monitor serial do Software do Arduino. Esses dados deverão ser anotados conforme orientação dessa seção.

Em um primeiro momento os experimentos afiguram-se inflexíveis, porém, é importante perceber que as automatizações presentes em todos os experimentos, o código disponível, os dados iniciais obtidos, além dos questionamentos apresentados no fim de cada capítulo possibilitam novas abordagens mediadas pelo professor. Na etapa final do capítulo há questionamentos específicos que ajudam a fixar o conhecimento sobre os fundamentos teóricos, e questionamentos divergentes, que induzem o estudante a um pensamento livre e intervencionista.

Certamente, fazendo uso das diversas ferramentas que compreendem as atividades, o aluno pode desenvolver novas investigações.

Recomenda-se que durante a aplicação, o professor siga a sequência estabelecida: fundamentação teórica e Objetivos, lista de materiais e componentes, montagem do experimento, o código, execução e coletas de dados, e questionamentos e levantamento de hipóteses. O tempo necessário para as atividades é de aproximadamente 90 minutos para o capítulo 1, momento em que há instrução sobre as funcionalidades e aplicações do Arduino. Posteriormente, aplica-se a atividade mais conveniente para o professor e alunos. Com o material em disponibilidade, e um número reduzido de alunos, até 5 por atividade, o tempo de execução será também de 90 minutos, que corresponde aproximadamente à carga horária semanal mínima da disciplina de Física. Para a solução dos Questionamentos & levantamento de hipóteses o professor deverá estipular um prazo maior, marcando um terceiro encontro, ocasião em será apresentado os resultados, dúvidas e propostas de ações intervencionistas.

É importante que o professor sempre estimule seus alunos a proporem alguma ação intervencionista baseada no que compreenderam.

Principais Resultados para os Experimentos

A automação do experimento do Capítulo 2, que trata sobre o **Movimento Uniforme** exibirá dois intervalos de tempo, cujos valores devem ser quase idênticos, indicando um movimento uniforme, todavia, os dois valores podem divergir com outros valores do mesmo experimento em condições diferentes, ou seja, a resistência ao movimento é difícil de controlar, seja pelas características do móvel ou do trilho utilizado. Entretanto, se os procedimentos de montagem e

programação foram obedecidos, as velocidades, que devem ser calculadas na questão 1 do experimento, devem ser aproximadamente iguais. Se isso ocorrer, conclui-se que a montagem está correta e o professor com seus alunos terão um dispositivo para análise de Movimento Uniforme.

Com a mesma montagem do Capítulo 2, o Capítulo 3, que trata do Movimento Uniformemente variado, deverá informar, com a automação, duas velocidades diferentes, o que permitirá ao aluno, através da equação de Torricelli, calcular a aceleração do móvel.

O Capítulo 4, ainda com a mesma montagem (Arduino e seus componentes) dos capítulos anteriores, tem seus sensores dispostos verticalmente para medida da aceleração da gravidade. Neste experimento espera-se que o professor e seus alunos encontrem medidas próximas de $10m/s^2$. Entretanto, em laboratório, os testes revelaram valores médios de $13\ m/s^2$. Erro que, hipoteticamente, pode ser suprimido na reformulação da atividade, aumentado as distâncias entre os sensores, implicando em maior precisão no experimento.

Os questionamentos no Capítulo 5 – Movimento Circular Uniforme – suscitam respostas calculadas a partir do número de voltas realizadas por segundo, pelo objeto em rotação. Logo, para cada implementação desse capítulo haverá uma medida, que dependerá das características do objeto utilizado; no mesmo capítulo sugere-se o uso de uma parafusadeira ou furadeira. Em testes realizados em laboratórios com os alunos, foi usado uma furadeira da marca *Mondial,* modelo *Power tool*, cuja rotação máxima por minuto é de 2800 RPM (Rotações Por Minuto). Foram obtidos, no experimento, resultados plausíveis. A automação com Arduino contou aproximadamente 39 voltas por segundo, ou seja, 39 rad/s ou 2340 RPM.

A automação do Capítulo 6 exibirá o peso de objeto de até 200g na superfície de alguns corpos celestes do sistema solar. Os valores exibidos são aproximados tomando o peso na terra como referência, além de admitir o valor $g=10m/s^2$ para a aceleração da gravidade na superfície terrestre. Logo poderá haver diferença

nos cálculos da gravidade em comparação com os valores já conhecidos. Reiterando, os valores exibidos no monitor serial do Arduino dependerão do valor da massa escolhida pelo professor ou aluno.

No Capítulo 7, onde é estudado a Lei de Hooke, os valores exibidos para a força deverão ser característicos de cada experimento. A deformação na mola do dinamômetro não é fixa para todos os dinamômetros, portanto a constante elástica da mola é única para cada tipo de dinamômetro. Porém, se espera que ao verificar a lei de Hooke, a mola apresente a mesma constante para mais de um par, força *versus* deformação.

Os valores imprimidos no monitor Serial no Capítulo 8, que aborda o Empuxo, são prefixados, ou seja, será exibido apenas os valores determinados na **Tabela 9**. Logo, os valores são aproximações, pois o dispositivo determina o empuxo sob o objeto com uma resolução de 0,33 newtons, dentro do intervalo de 0 a 2 newtons.

Tabela 9 - Valores fixados para exibição no monitor serial (Capítulo 8).

Volume deslocado (ml)	Empuxo (N)
33	0,33
66	0,65
100	0,98
133	1,30
166	1,63
200	1,98

Fonte: do autor

Com estas informações prefixadas a resposta do questionamento 1, desse Capítulo, dever ser aproximadamente 1,0 kg/litro.

Os Capítulos 9 e 10 apresentam um dispositivo de aferição de temperatura, portanto, será útil em diversas atividades que necessitem de monitoramento de temperatura, nas principais escalas, dentro dos limites de temperatura

estabelecidos pelos sensores, normalmente de -55 a 125°C. As repostas aos questionamentos desses capítulos são bem subjetivas e pessoais, conforme orientações do plano de aula do professor, que decidirá com seus alunos o que medirão.

No Capítulo 11 é abordado a primeira e a segunda Lei de Ohm. Os questionamentos no final do capítulo iniciam com a sugestão para elaboração de um gráfico da Tensão *versus* a corrente elétrica para o resistor R, em análise.

Na atividade há a sugestão de ajustar a tensão elétrica sobre o resistor R, com o potenciômetro, variando sua resistência de 1 a 9 kΩ. Seguindo essa sugestão é possível encontrar valores próximos ao exibido na **Tabela 10**, cujas medidas são reais, coletadas por uma dupla de alunos envolvidos na atividade.

Entretanto para verificação da Primeira Lei de Ohm é possível ajustar a tensão para qualquer valor, até o limite do dispositivo, que nos teste revelou-se ser de 2 volts.

Tabela 10 - Medidas obtidas por um aluno (Capítulo 11)

Tensão elétrica (V)	Corrente elétrica (A)
0,24	0,000914
0,49	0,001884
0,75	0,002854
1,01	0,003843
1,25	0,004757
1,49	0,005690
1,76	0,006716

Fonte: do autor

A plotagem dos dados referente a **Tabela 10** apresenta a linearidade esperada para um resistor ôhmico, conforme roga a primeira Lei de Ohm. Vejamos no **Gráfico 1**.

A análise da Segunda Lei de Ohm é mais subjetiva, nessa atividade, onde o aluno deve expor sua compreensão sobre a relação da mudança de tensão provocada pela variação da posição angular do eixo de rotação do potenciômetro.

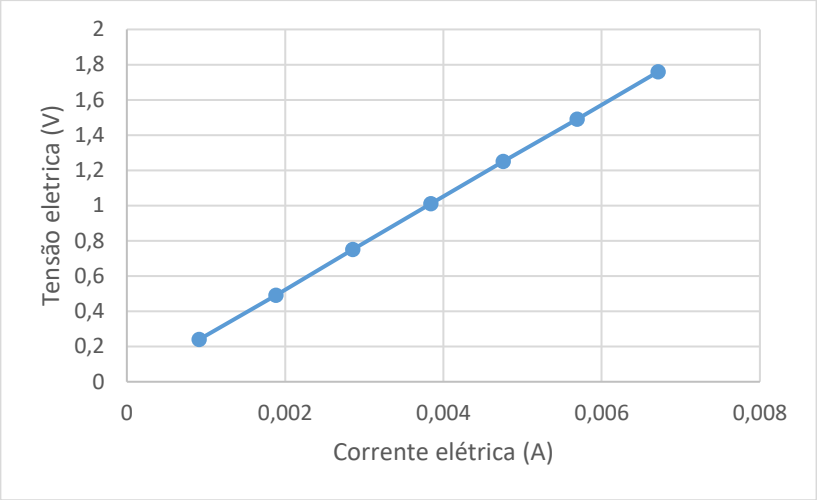


Gráfico 1 - Relação entre tensão e corrente elétrica para o resistor R da atividade do Capítulo 11 do Produto Educacional. A partir dos dados coletados de uma dupla participante dos testes do produto.

Fonte: do autor.

As atividades do Produto Educacional encerram-se no Capítulo 12, com um experimento de fotorresistividade, aproveitando a compreensão obtida no Capítulo 11 sobre resistividade. Os valores exibidos não são prefixados, e dependerão das condições de montagem, luminosidade do ambiente e potência do LED.

Testes feitos para cinco distâncias arbitrárias entre a fonte de luz e o LDR forneceram a relação gráfica para os níveis de luminosidade e distancia da fonte, vejamos no **Gráfico 2**.

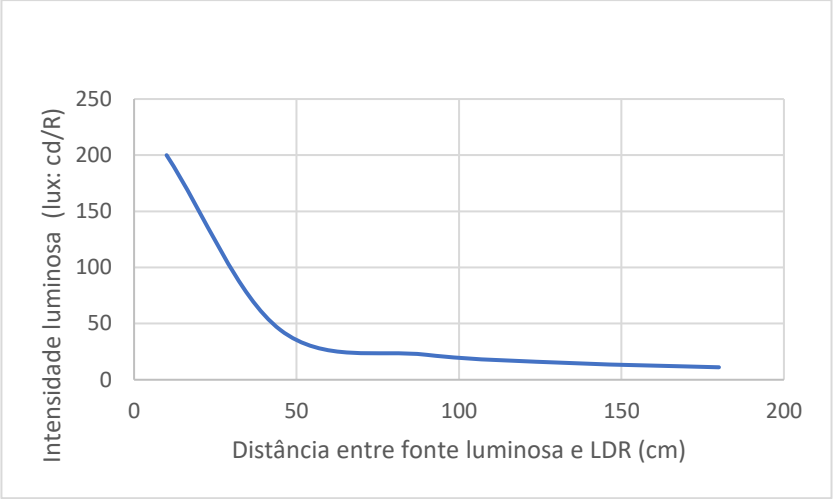


Gráfico 2 - Relação entre o nível de luminosidade e a distância entre a fonte luminosa e o LDR. As distâncias consideradas foram, 10, 45, 90, 135 e 180 cm. A partir de dados coletados por equipes de alunos do 3° ano do Ensino Médio. O parâmetro para a intensidade luminosa foi o especificado pelo fabricante do LED com medida de 20 cd.

Fonte: do autor.

Códigos de erros recorrentes

Os códigos contidos neste produto educacional foram todos testados, logo os problemas que podem ocorrer são erros durante a cópia, caso se esqueça de selecionar parte do código, um dos mais recorrente são estes:

```
stray '\240' in program
```

Ocorre em algumas versões do software ao copiar e colar um *sketch*. A correção exige a identificação da linha com erro e digitação da linha, a fim de suprimir possível incompatibilidade de formatação do texto.

```
expected ';' before '}' token:
```

Erro devido à falta de ponto e vírgula, para correção apenas digite ";".

```
was not declared in this scope
```

Ocorre quando alguma variável não for declarada. Veja que o erro é informado no momento da verificação do sketch (**Figura 19**).
Para corrigir esse tipo de erro, simplesmente localize a variável e a declare especificando o tipo.

Eis a correção do problema apresentado na figura 19:

```
int valor;
```

A correção consistiu em inserir o tipo de variável e o caractere ";".

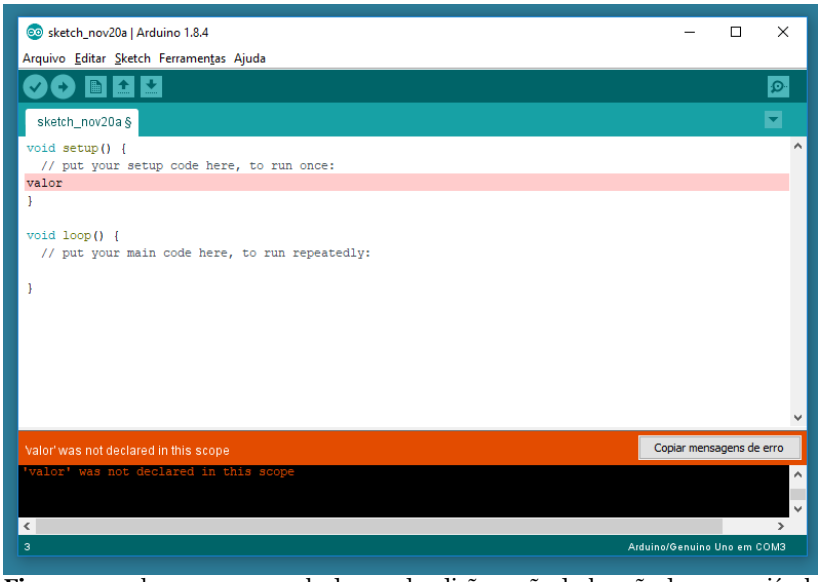


Figura 19 - observe um exemplo de erro de edição, a não declaração de uma variável.

Fonte: do autor

Para outros erros, consulte a documentação no site oficial Arduino

- Resolução de problemas: https://www.arduino.cc/en/Guide/Troubleshooting
- Perguntas frequentes: https://www.arduino.cc/en/Main/FAQ

Bibliografia

BALDO, D. A., et al. Aparato educacional para estudo da queda livre com análise do movimento. **Caderno Brasileiro de Ensino de Física**, *33*, pp. 1064-1078, 2016.

CAVALCANTE, M. A., TAVOLARO, C. R., & MOLISANI, E.. Física com Arduino para iniciantes. **Revista Brasileira de Ensino de Física**, *33*, p. 9pp, 2011.

GOYA, A.; HALABI, S. E. **Trilho Multifuncional para Ensino de Mecânica**. Disponível em: <http://www.uel.br/ccb/biologiageral/eventos/erebio/painel/T170.pdf>. Acesso em: 29 de Jan. 2018.

HALLIDAY, D., RESNICK, R., & WALKER, J. (1996). **Fundamentos de Física** (4ª ed., Vol. 2). Rio de Janeiro, RJ: LTC, 1996.

McROBERTS, M. **Arduino Básico**. São Paulo: Novatec, 2011.

MONK, S. **30 Projetos com Arduino** (2 ed.). Porto Alegre: Bookman, 2014.