UNIVERSIDADE FEDERAL DE UBERLÂNDIA
INSTITUTO DE FÍSICA – CAMPUS SANTA MÔNICA

HERMES GUSTAVO FERNANDES NERI

UTILIZAÇÃO DA PLATAFORMA ARDUINO PARA CONTROLE DE EXPERIMENTOS REMOTOS DE FÍSICA

UBERLÂNDIA
2014

HERMES GUSTAVO FERNANDES NERI

UTILIZAÇÃO DA PLATAFORMA ARDUINO PARA CONTROLE DE EXPERIMENTOS REMOTOS DE FÍSICA

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Uberlândia (UFU), como parte das exigências do Programa do curso de Licenciatura em Física, para obtenção do Título de Graduado.

Orientador: Prof. Dr. Eduardo Kojy Takahashi

UBERLÂNDIA
2014

HERMES GUSTAVO FERNANDES NERI

UTILIZAÇÃO DA PLATAFORMA ARDUINO PARA CONTROLE DE EXPERIMENTOS REMOTOS DE FÍSICA

COMISSÃO EXAMINADORA

_____

Prof. Dr. Eduardo Kojy Takahashi - Orientador
Universidade Federal de Uberlândia

_____

Prof. Dr. Sílvia Martins dos Santos
Universidade Federal de Uberlândia

_____

Prof. Dr. Ricardo Kagimura
Universidade Federal de Uberlândia

Uberlândia, 01 de Setembro de 2014

RESUMO

O objetivo deste trabalho consistiu em elaborar um projeto que possibilitasse a disponibilização de experimentos reais para acesso remoto, por meio da internet, dimensionando os componentes eletromecânicos e os equipamentos necessários para tal empreendimento. O experimento piloto disponibilizado foi o experimento da determinação da razão carga/massa do elétron, conforme realizado por Joseph John Thomson em 1897. Para o processo de controle desse experimento foi escolhida a placa de prototipagem Arduino que, intermediada por um circuito de potência, atua nos motores de passo conectados aos potenciômetros das fontes de tensão, permitindo ao usuário a regulagem do potencial acelerador do feixe de elétron, do potencial colimador desse feixe e da corrente elétrica geradora do campo magnético nas bobinas de Helmholtz do aparato experimental. A utilização didática do Arduino na construção de dispositivos simples pode contribuir para o desenvolvimento do raciocínio lógico, viabilizado pela programação computacional, além de estimular a aprendizagem significativa de conceitos da eletrônica e da mecânica.

Palavras-chave: Experimentação remota; Arduino; experimento de Thomson.

SUMÁRIO

1. INTRODUÇÃO

A atual situação estrutural da educação brasileira inviabiliza a construção de laboratórios de Física e o acesso de professores e estudantes a essas atividades experimentais (CARDOSO; TAKAHASHI, 2011).

Com o objetivo de disponibilizar um experimento real de Física a alunos e professores do ensino médio, foi projetado e desenvolvido um sistema de experimentação remota através da internet, no qual os usuários podem manipular e coletar os dados necessários para a compreensão do experimento de forma eficiente e em tempo real. Para esse sistema, foi escolhido o experimento de Thomson, que possibilitou a descoberta do elétron (SILVA, SANTOS e DIAS, 2011), sendo esse, um marco histórico e de grande relevância para a Ciência e Tecnologia contemporâneas.

Para realizar o controle, o sistema deveria regular os potenciômetros das fontes de tensão conectadas ao experimento. Foram feitas diversas pesquisas sobre as melhores opções de hardware para a montagem desse sistema de controle. Dentre as opções disponíveis, a plataforma Arduino foi a que se mostrou mais eficiente devido a diversos motivos, dentre eles:

- A plataforma possui hardware e software livres;
- Boa relação custo-benefício;
- Facilidade de programação, gravação do software na placa;
- Comunicação via USB embutida na placa, o que facilita a conexão com o servidor;
- A linguagem utilizada para sua programação é de fácil compreensão.

Para realizar o controle das variáveis do experimento, utilizamos motores de passo com precisão de 7,5° por passo, acoplados por eixos fixos de alumínio e tecnil aos potenciômetros das fontes de tensão e corrente. Utilizando a placa Arduino como módulo de controle dos motores de passo, é possível regular cada um desses motores individualmente através da internet. A placa Arduino recebe os comandos da internet através de um servidor local e atua nos motores conectados aos potenciômetros da fonte.

Este é um modelo de controle eficaz devido a sua precisão, velocidade de resposta e custo de montagem. A partir desse modelo é possível projetar e controlar outros experimentos, inclusive de outras áreas da Física, através de pequenas adaptações no circuito final de controle.

2. Experimentação Remota

Para a Física, a experimentação exerce um papel de suma importância para o processo de ensino/aprendizagem dessa ciência, uma vez que facilita a compreensão do conteúdo por parte dos estudantes e favorece a construção de uma aprendizagem mais significativa. Entretanto, existem algumas restrições relacionadas ao uso de laboratórios presenciais, tais como, a limitação do tempo de uso, a duplicação de equipamentos experimentais, os custos com materiais e manutenção, dentre outros (CARDOSO; TAKAHASHI, 2011).

Além disso, as práticas experimentais ficam restritas ao cumprimento de procedimentos pré-determinados de tomadas de medidas e análises dessas medidas, não permitindo a exploração de aspectos formativos relacionados à compreensão dos conhecimentos envolvidos na construção e funcionamento do aparato experimental, assim como, dos aspectos históricos, científicos e tecnológicos que viabilizaram a concepção e o uso dos equipamentos experimentais.

Para possibilitar uma utilização mais efetiva da prática experimental na construção do conhecimento em Física, pode-se conceber um laboratório remoto constituído de experimentos reais. Um experimento remoto consiste em um equipamento que pode ser manipulado a distância e disponibilizado para acesso através da internet em um ambiente virtual de aprendizagem (AVA). Esse tipo de equipamento oferece soluções para alguns dos problemas encontrados na utilização de experimentos tradicionais, além de possuir diversas outras vantagens, tais como:

- Diminuição de custos com equipamentos e materiais;
- Combate a falta de suporte técnico e a escassez de equipamentos experimentais;
- Não possuir limite de tempo e espaço para sua utilização;

- Ser um ambiente seguro, disponibilizado pela internet;
- Possibilidade de ser disponibilizado para cursos à distância.

3. A Placa Arduino

Arduino é uma plataforma de hardware *Open Source* de fácil utilização e ideal para a criação de dispositivos que permitam interagir com o ambiente através de sensores de temperatura, sensores de som, *leds*, motores, *displays*, alto falantes, entre outros (BANZI, 2011). Segundo CAVALCANTE, M.A (2011) "Resumidamente, a plataforma consiste em uma placa de circuitos com entradas e saídas para um microcontrolador (...), um ambiente de desenvolvimento e o bootloader que já vem gravado no microcontrolador."

Criada em 2005, na Itália, o objetivo inicial do projeto era desenvolver uma placa de prototipagem com um preço acessível para os estudantes de engenharia. Atualmente, ela é utilizada para aplicação e desenvolvimento de projetos em diversas áreas do conhecimento. Além disso, ela foi adotada como uma opção para artistas, hobbistas e pessoas interessadas em ensinar e aprender sobre robótica e eletrônica, por exemplo. Suas funcionalidades e aplicações são inúmeras, assim como, a quantidade de sites, fóruns e materiais (livros, e-books, apostilas, vídeos, tutoriais) disponíveis na internet e que em sua maioria são gratuitos.

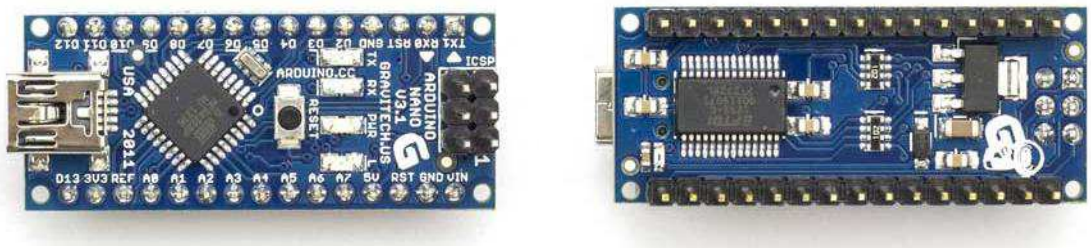
3.1. Alguns modelos de placas

A placa Arduino é constituída basicamente por pinos de entrada e saída (I/O), que podem ser digitais ou analógicos e um microcontrolador, que é responsável pelo gerenciamento dos processos de comunicação e controle dos pinos (I/O). O Arduino se baseia na ideia do hardware livre, assim, além dos modelos fabricados pela empresa, existem também os modelos produzidos e modificados por usuários da placa. Dentre os modelos produzidos pela empresa, temos:

3.1.1. Arduino Nano

A Figura 1 mostra as duas faces de uma placa Arduino modelo Nano.

Figura 1 – Imagem das vistas frontal e traseira da placa Arduino Nano.



Fonte: Disponível em: http://arduino.cc/en/Main/ArduinoBoardNano.

Na Tabela 1 encontram-se as especificações técnicas da referida placa.

Tabela 1 – Especificações técnicas da placa Arduino Nano.

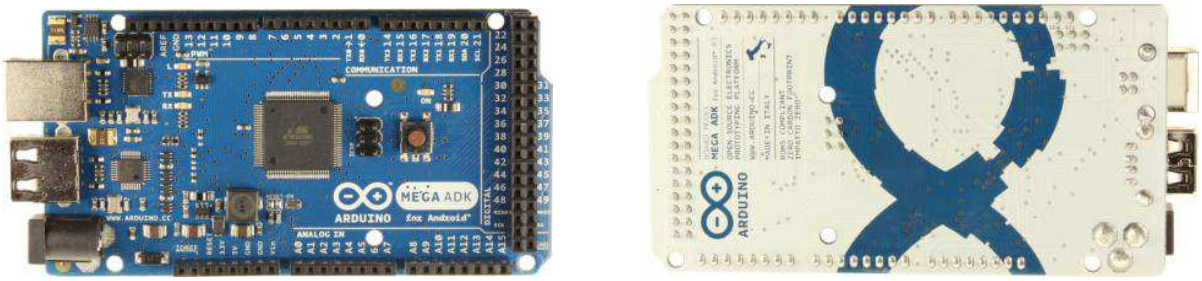
ESPECIFICAÇÕES	
Dimensões (cm x cm)	1,85 x 4,32
Microcontrolador	Atmel ATmega168 ou ATmega328
Velocidade de Clock	16 MHz
Corrente DC por I/O	40 mA
Pinos analógicos de Entrada	8
Pinos de I/O Digital	14 (dos quais 6 fornecem saída PWM)
Tensão de alimentação (recomendado)	7 a 12 Volts
Tensão de alimentação (limites)	6 a 20 Volts
Tensão de operação (nível lógico)	5 Volts

Fonte: Disponível em: http://arduino.cc/en/Main/ArduinoBoardNano.

3.1.2. Arduino Mega ADK

A Figura 2 mostra as duas faces de uma placa Arduino Mega ADX, enquanto que a Tabela 2 apresenta suas especificações técnicas.

Figura 2 – Imagem das vistas frontal e traseira da placa Arduino Mega ADK.



Fonte: Disponível em: http://arduino.cc/en/Main/ArduinoBoardMegaADK.

Tabela 2 – Especificações técnicas da placa Arduino Mega ADK.

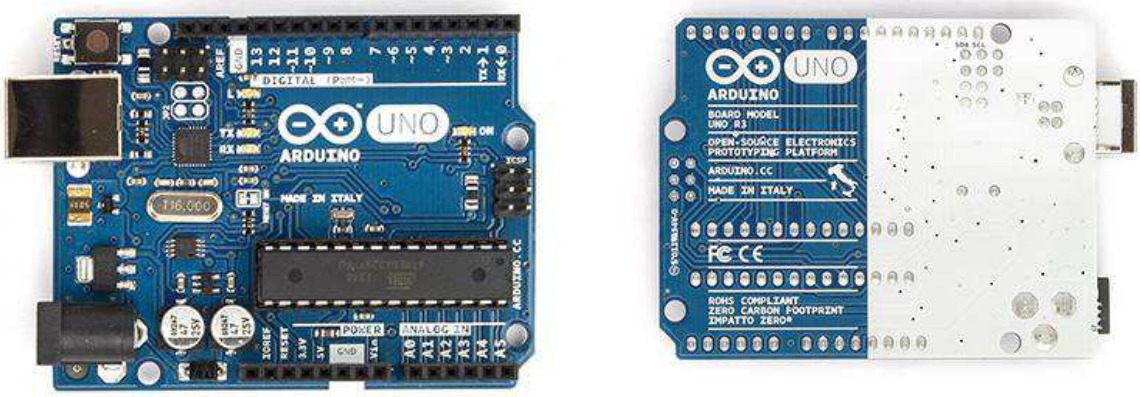
ESPECIFICAÇÕES	
Microcontrolador	ATmega2560
Velocidade de Clock	16MHz
Corrente DC por I/O	40mA
Corrente DC por pino 3,3V	50mA
Pinos analógicos de Entrada	16
Pinos de I/O Digital	54 (dos quais 15 fornecem saída PWM)
Tensão de alimentação (recomendado)	7 a 12 Volts
Tensão de alimentação (limites)	6 a 20 Volts
Tensão de operação (nível lógico)	5 Volts

Fonte: Disponível em: http://arduino.cc/en/Main/ArduinoBoardMegaADK.

3.1.3. Arduino UNO

A Figura 3 mostra as duas faces de uma placa Arduino Uno, enquanto que a Tabela 3 apresenta suas especificações técnicas.

Figura 3 – Imagem das vistas frontal e traseira da placa Arduino Uno.



Fonte: Disponível em: http://arduino.cc/en/Main/ArduinoBoardUno.

Tabela 3 – Especificações técnicas da placa Arduino Uno.

ESPECIFICAÇÕES	
Microcontrolador	ATmega328
Velocidade de Clock	16MHz
Corrente DC por I/O	40mA
Corrente DC por pino 3,3V	50mA
Pinos analógicos de Entrada	6
Pinos de I/O Digital	14 (dos quais 6 fornecem saída PWM)
Tensão de alimentação (recomendado)	7 a 12 Volts
Tensão de alimentação (limites)	6 a 20 Volts
Tensão de operação (nível lógico)	5 Volts

Fonte: Disponível em: http://arduino.cc/en/Main/ArduinoBoardUno.

Como utilizamos, em nosso trabalho, a placa Arduino Uno, iremos descrever melhor suas características na próxima sessão.

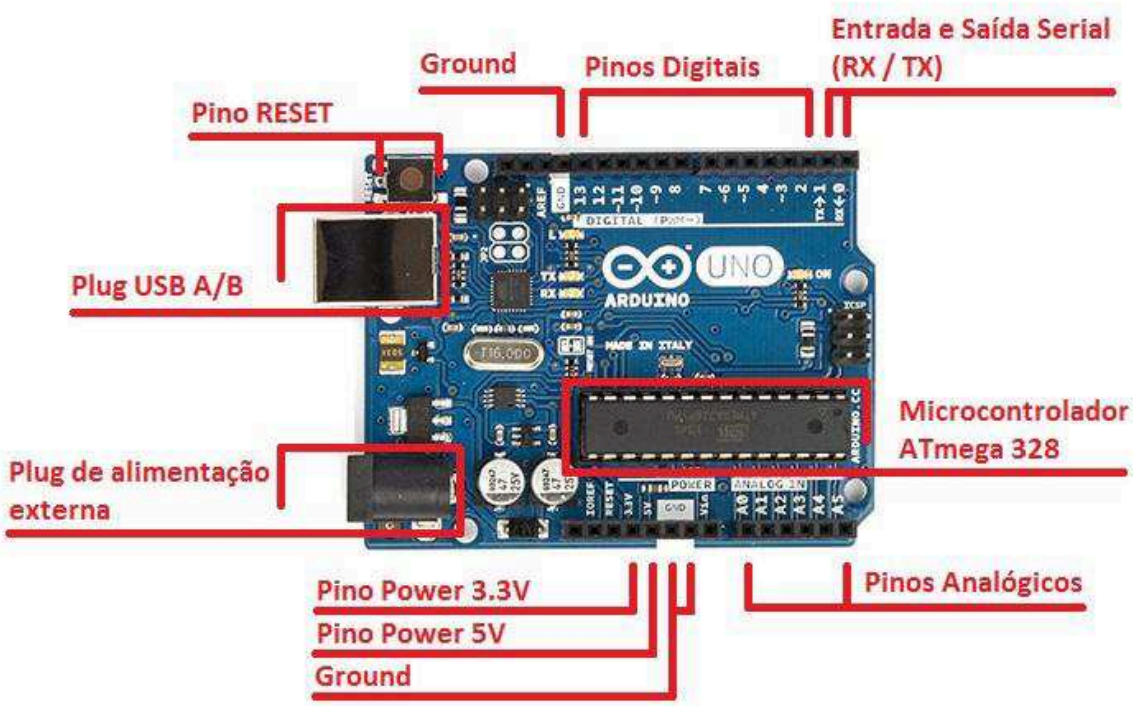
4. Conhecendo a placa Arduino Uno

Além das características de funcionamento e desempenho das placas, o custo desses equipamentos também foi levado em consideração. Segundo dados da loja virtual Robocore, o preço da placa Mega ADK varia entre 200,00 e 280,00 reais, enquanto o modelo Nano pode ser encontrado por aproximadamente 135,00 reais. Para este trabalho, foi escolhida a placa Arduino modelo Uno, que além de possuir o menor valor dentre os modelos analisados (em média, 80,00 reais) possui outros fatores relevantes para o projeto, como por exemplo:

- Quantidade de portas digitais e analógicas;
- Relação custo e benefício;
- Facilidade para realização de sua compra em mercados nacionais;
- Estrutura para conexão de cabos e plugs externos.

Assim, apresentamos na Figura 4 as especificações e os detalhamentos dos principais componentes do hardware utilizado.

Figura 4 – Disposição dos principais componentes da placa.



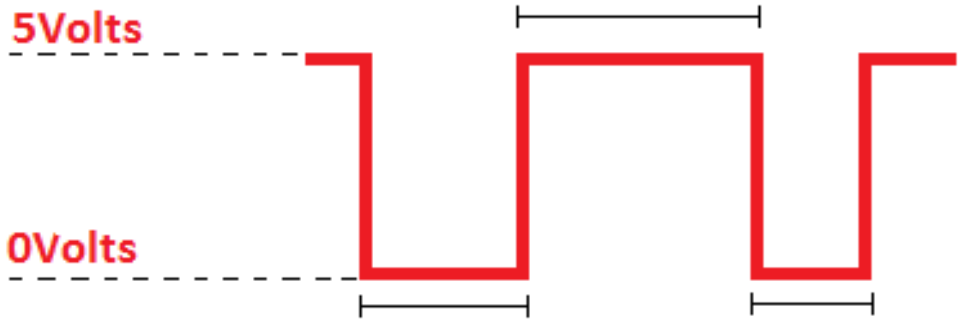
Fonte: http://arduino.cc/en/Main/ArduinoBoardUno (Adaptado).

A Placa Arduino Uno possui um microcontrolador ATmega 328. Este é o componente responsável pelo controle das operações realizadas pela placa, tais como, controle de sinais digitais e analógicos nos pinos e a comunicação serial com outros equipamentos e dispositivos, através da porta USB.

A placa pode ser alimentada através do cabo USB A/B, conectado a um computador pessoal, notebook, tablet, smartphone ou a qualquer outro dispositivo que possua uma porta de comunicação USB. Além disso, uma outra opção consiste em alimentá-la através de uma fonte externa que pode ser conectada ao seu *plug*. Para isso, o fabricante recomenda a utilização de fontes que forneçam um valor de tensão entre 7V à 12V.

Dentre os 14 pinos digitais existentes na placa Arduino Uno, 6 deles possuem função PWM (*pulse-width modulation*), que é a mudança da duração do pulso, ou seja, o tempo que o sinal digital permanece em nível lógico 1 ou em nível lógico 0, conforme Figura 5.

Figura 5 – Demonstração do pulso PWM.

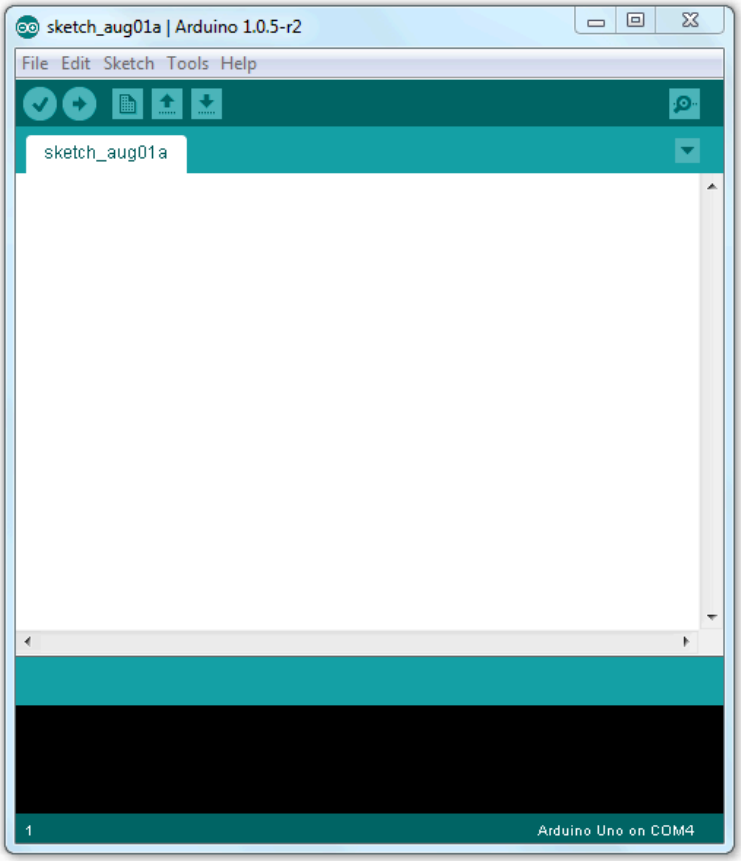


O nível lógico 1 é indicado por uma tensão de 5Volts, enquanto o nível lógico 0 é indicado por uma tensão de 0Volts. O tempo em que o sinal permanece em cada nível é o que determina diretamente a largura do pulso, ou seja, quanto maior o tempo de permanência, maior a largura. O PWM é um recurso que pode ser utilizado para controle de motores de passo, servomotores, intensidade luminosa de leds, entre outras aplicações.

5. Especificações de Software

Segundo o site oficial, o ambiente de desenvolvimento Arduino (Figura 6) utilizado para programar e gravar o código na placa é escrito em Java e baseado nas linguagens Processing, avr-gcc e em outros softwares Open Source. Ele funciona em sistemas operacionais Windows, Mac OS X e Linux. Para baixar gratuitamente o ambiente, basta entrar na página oficial da placa e clicar na aba *Download*, ou acessar o seguinte link: http://arduino.cc/en/Main/Software.

Figura 6 – Ambiente de desenvolvimento.



Os principais recursos do ambiente de programação estão apresentados na Tabela 4.

Tabela 4 – Principais recursos do ambiente de programação.

Principais recursos	
	**Verify:** verifica se há algum erro na programação do código
	Upload: transfere o código para a placa Arduino
	New: Cria uma nova janela em branco
	Open: abre um arquivo já existente
	Save: salva as alterações realizadas no código

5.1. Estrutura de programação

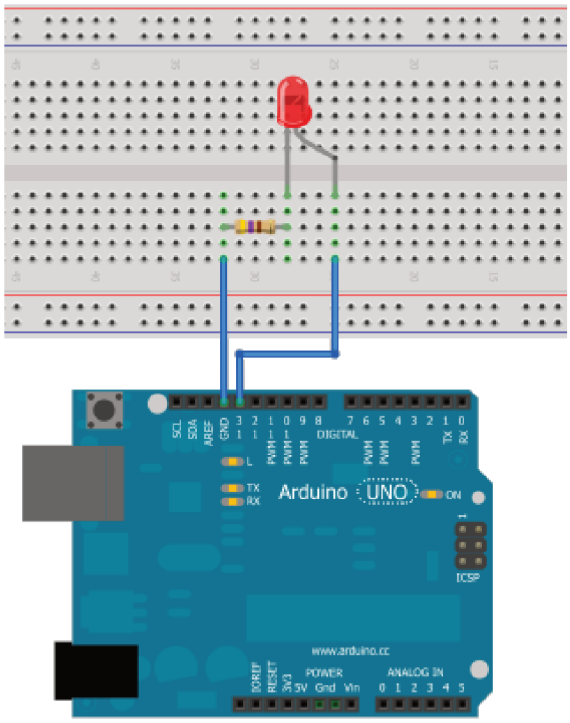
No ambiente de programação disponibilizado pelo fabricante existem exemplos de códigos para algumas funções básicas. A linguagem de programação é baseada na linguagem Wiring, que é específica para microcontroladores.

Para demonstrar a estrutura de programação utilizaremos o código *Blink* (Figura 7), que é um exemplo já existente no ambiente de programação e cuja função é piscar o led conectado ao pino 13 da placa, a cada 1 segundo. Para tanto, a conexão elétrica do led na placa Arduino Uno deve ser conforme a Figura 8. A descrição de cada comando apresentado na Figura 7 encontra-se na Tabela 5.

Figura 7 – Exemplo código *Blink.*

Na figura 8, temos a representação do circuito eletrônico montado na protoboard para utilização do código Blink.

Figura 8 – Conexão do led na placa Arduino para utilização do código *Blink*.



Na tabela 5 temos a descrição completa de cada estrutura do código Blink.

Tabela 5 – Descrição do código *Blink*.

COMANDOS	DESCRIÇÃO
int led = 13;	Nomeia a porta 13 como “led”.
void setup() { }	O código escrito dentro desta função será executado somente uma vez, ou quando a placa for reiniciada através do botão RESET.
pinMode(led, OUTPUT);	Configura a porta como uma saída.
void loop() { }	O código escrito dentro desta função é executado continuamente até que a placa seja desligada.
digitalWrite(led, HIGH);	Envia o comando HIGH (sinal em nível lógico 1) para a porta “led”, ou seja, liga o led conectado à porta 13.
delay(1000);	Faz com que o código pare de ser executado por 1 segundo (1000ms).
digitalWrite(led, LOW);	Envia o comando LOW (sinal em nível lógico 0) para a porta “led”, ou seja, desliga o led conectado à porta 13.

6. O experimento de Thomson

O experimento de Thomson realizado pelo físico inglês Joseph John Thomson no final do século XIX, em Cambridge, na Inglaterra, possibilitou a confirmação de que existia uma partícula elementar carregada negativamente, o elétron (Takahashi, 2013). Seu controle consiste basicamente na manipulação do nível de tensão (0 à 300V) que é utilizado para aquecer o colimador e na manipulação da corrente (0 à 5A), que por sua vez, gera o campo magnético ao ser aplicado nas bobinas de Hellmoltz, localizadas na proximidade do bulbo (Figura 9). Abaixo, temos uma fotografia do experimento utilizado.

Figura 9 - Experimento de Thomson.

Fonte: Disponível em: nutec.ufu.br/experimentos/thomson/exp/expaluno/experimento.html

Se um elétron de massa m e carga ***e*** é acelerado por uma diferença de potencial U, este adquire energia cinética. Pelo Teorema da Energia Cinética sabemos que o Trabalho da força resultante é igual à variação da energia cinética.

O elétron é acelerado devido a uma diferença de potencial e o trabalho realizado sobre o elétron pode ser calculado pelo produto entre a carga e a diferença de potencial. Igualando o produto da carga elétrica e da diferença de potencial com a variação da energia cinética temos:

$$e \cdot U = \frac{m_0 \, v^2}{2} \qquad (1)$$

onde v é a velocidade do elétron. Em um campo magnético de intensidade B, a força de Lorentz em um elétron com velocidade v é:

$$F = e.V \times B \qquad (2)$$

Se o campo magnético é uniforme, como é o caso no interior das Bobinas de Helmholtz, o elétron descreve uma trajetória em espiral às linhas de força magnética, que se torna um círculo de raio r se a velocidade v for ortogonal a B. Como a força centrípeta assim gerada é igual à força de Lorentz, obtemos:

$$v = \frac{eBr}{m_0} \qquad (3)$$

onde B é o valor absoluto do vetor campo B.

Substituindo a equação (3) na equação (1) obtemos a expressão para a razão carga/massa do elétron:

$$\frac{e}{m_0} = \frac{2U}{B\ r^2} \qquad (4)$$

Para o arranjo de Helmholtz de duas bobinas de raio R, a distância entre o plano das duas bobinas com número de espiras *n* também é R. O Campo B no centro entre as bobinas é dado por:

$$B = \left(\frac{4}{5}\right)^{3/2} u_0 n \frac{I}{R} \qquad (5)$$

Assim, o procedimento experimental consiste em fixar um valor para o raio da trajetória do feixe *r,* pelo ajuste de pares de valores do potencial acelerador *U* e do campo magnético *B* (por meio da variação da corrente elétrica *I* na equação 5). Construindo-se, então, um gráfico de *U* em função de *B*, deve-se obter uma reta, cujo coeficiente angular é $er^2/(2m_0)$. Calculando-se esse coeficiente angular a partir do gráfico, obtém-se o valor da razão e/m_o. Esse procedimento pode ser repetido para outros valores de r, resultando em diversos valores para e/m_o. A média dos valores encontrados fornece o resultado experimental da razão carga/massa do elétron. O valor teórico da razão carga-massa (e/m) é de aproximadamente 1,759 x 10^{11} C/kg. Os valores experimentais, obtidos com o equipamento que dispomos, podem ter incertezas da ordem de apenas 2% nessa razão.

Para manipular o potencial acelerador do feixe de elétron, o potencial colimador desse feixe e a corrente elétrica geradora do campo magnético nas bobinas de Helmholtz são necessários controlar os potenciômetros das fontes de tensão conectadas ao experimento. Assim, projetamos três eixos, cada um com um corpo de alumínio e um encaixe em tecnil. Esses eixos foram acoplados aos potenciômetros e aos motores de passo, como mostrado na Figura 10.

Figura 10 – Acoplamento entre motor de passo e potenciômetro.

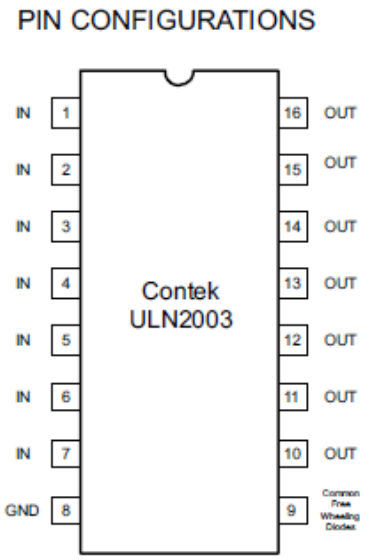
Os eixos de acoplamento foram projetados nas seguintes medidas constantes da Tabela 6:

Tabela 6 – Dimensões dos eixos.

Eixo 1	Comprimento	120mm
	Espessura	12.4mm
	Espessura de encaixe no potenciômetro	24.8mm
	Diâmetro do eixo do motor	9mm
Eixo 2	Comprimento	120mm
	Espessura	12.4mm
	Espessura de encaixe no potenciômetro	24.8mm
	Diâmetro do eixo do motor	9mm
Eixo 3	Comprimento	120mm
	Espessura	12.4mm
	Espessura de encaixe no potenciômetro	14.5mm
	Diâmetro do eixo do motor	9mm

Como a placa Arduino não possui potência suficiente para acionar os motores, foi criado um circuito que utiliza CI's (circuitos integrados) de potência alimentados por uma fonte externa. Nesse caso em específico, foi utilizada uma fonte chaveada de 5V e dois CI's modelo ULN2003, sendo um para cada motor. De acordo com o *Datasheet* do fabricante, este CI pode fornecer até 50V de tensão de saída e corrente máxima de 500mA em cada terminal, sendo que sua potência de saída depende diretamente da potência fornecida pela fonte de alimentação conectada a ele. Segue abaixo, o esquema de pinagem do CI.

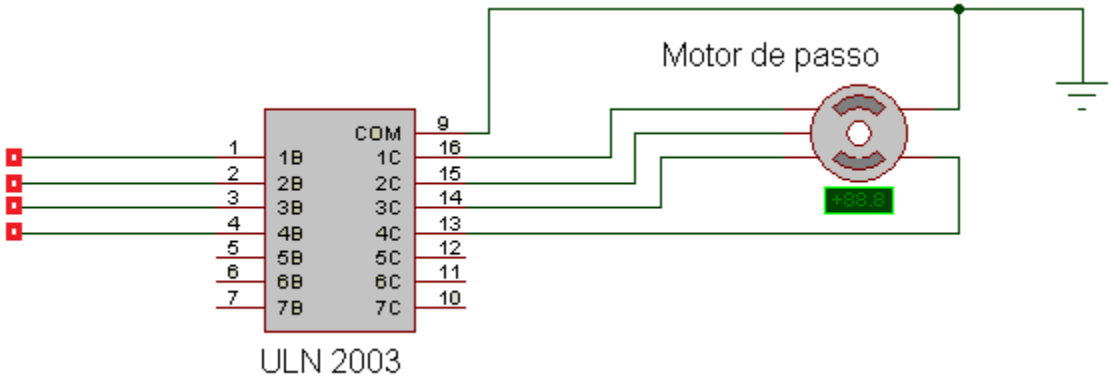
Figura 11 – Pinagem do CI ULN2003.



Fonte: Datasheet do fabricante.

Para acionar este CI, a placa Arduino aplica um sinal digital de nível lógico 1 em um dos sete terminais de entrada (IN), fazendo com que o terminal de saída (OUT) correspondente seja acionado.

Figura 12 – Circuito de acionamento do motor de passo utilizando CI ULN2003.



7. Motores de passo

O motor de passo é um dispositivo eletromagnético que possui um rotor com movimento preciso, controlado digitalmente por um hardware ou software específico (MESSIAS, 2014). Essa precisão é determinada pela sua configuração de fábrica. Os motores de passo possuem três estados:

Tabela 7 – Estados de um motor de passo.

Desligado	Não há alimentação suprindo o motor. Nesse caso não existe consumo de energia, e todas as bobinas estão desligadas. Na maioria dos circuitos este estado ocorre quando a fonte de alimentação é desligada.
Parado	Pelo menos uma das bobinas fica energizada e o motor permanece estático num determinado sentido. Nesse caso há consumo de energia, mas em compensação o motor se mantém alinhado numa posição fixa.
Rodando	As bobinas são energizadas em intervalos de tempos determinados, impulsionando o motor a girar numa direção.

Fonte: (Messias).

Para obter maior precisão no controle dos motores foi utilizada a configuração em *Half step* (meio passo), que por sua vez, proporciona um aumento no número de passos disponíveis pelo motor. Assim, se o motor gira originalmente em 7,5° a cada passo, em uma configuração de meio passo passará a girar em 3,75° por passo. A tabela abaixo demonstra como ocorre o acionamento das bobinas no modo *Half Step*.

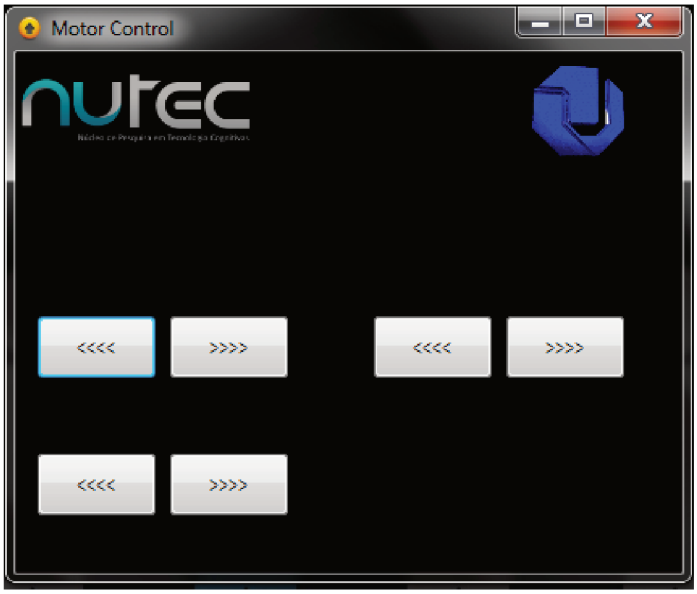
Tabela 8 – Acionamento do motor de passo em modo *Half Step.*

Nº do passo	B3	B2	B1	B0	Decimal
1	1	0	0	0	8
2	1	1	0	0	12
3	0	1	0	0	4
4	0	1	1	0	6
5	0	0	1	0	2
6	0	0	1	1	3
7	0	0	0	1	1
8	1	0	0	1	9

Fonte: (BRITES; SANTOS).

Durante a fase inicial do projeto, foi desenvolvido um software simples, de acesso local para realização de testes sobre o controle e funcionamento dos motores de passo. Criado em Visual Studio na linguagem C#, o programa roda em sistemas operacionais Windows e já foi testado com sucesso nas versões XP, W7 (Windows Sete) e W8 (Windows Oito). O código deste programa está disponível no Apêndice II deste trabalho. Segue abaixo, uma imagem do Layout deste programa.

Figura 13 – Software local para controle dos motores de passo.



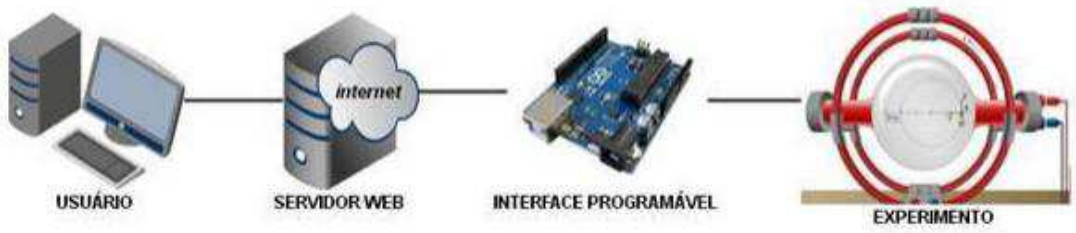
8. Funcionamento do sistema

Para realizar o controle do experimento foi construída uma página web que possibilitasse o maior grau de interação possível do usuário com o experimento. Três *webcams* estrategicamente posicionadas transmitem em tempo real o feixe de elétrons no interior do bulbo e as medidas de tensão e corrente registradas pelos multímetros digitais conectados ao circuito do experimento.

Figura 14 – Página Web.

Ao clicar em um dos botões responsáveis pelo controle dos motores (Azul, verde e amarelo) o usuário automaticamente gera um pacote de instruções no qual é indicado qual motor foi acionado e qual será o sentido de rotação. Esses comandos são enviados para o servidor Web que está instalado no local onde o experimento foi montado. O servidor, por sua vez, transfere essas instruções para a placa Arduino através de comunicação serial (conexão USB). A placa interpreta o sinal e aciona as portas digitais correspondentes ao motor e ao sentido em que o rotor deve se mover. O reposicionamento do motor é diretamente transferido ao potenciômetro da fonte ao qual ele está conectado através do eixo.

Figura 15 – Esquema de funcionamento do sistema.



Fonte: (Takahashi, 2013)

As alterações nos valores de tensão e corrente do experimento geram uma modificação no raio do feixe de elétrons. Essas alterações são visualizadas pelo usuário através das webcams e através dos valores medidos no amperímetro e no voltímetro é possível calcular o valor numérico da relação carga-massa do elétron.

9. Conclusão

Através da utilização da plataforma Arduino, que é um equipamento de hardware e software livre é possível construir um sistema de experimentação remota de baixo custo e que pode ser facilmente reproduzido e adaptado para outros experimentos, possibilitando assim, a montagem de um laboratório de experimentação remota, controlado através da internet (Weblab). Além disso, em função da facilidade de programação e montagem de circuitos eletrônicos, o Arduino pode ser utilizado como recurso de ensino/aprendizagem no curso de Física do ensino superior, na forma de estudo por projeto, viabilizando aliar a teoria à prática no desenvolvimento de dispositivos simples que trabalham conhecimentos de programação, eletrônica básica e mecânica.

Desta forma, a utilização didática do Arduino na construção de dispositivos simples pode contribuir para o desenvolvimento do raciocínio lógico, viabilizado pela programação computacional, além de estimular a aprendizagem significativa de conceitos da eletrônica e da mecânica.

REFERÊNCIAS BIBLIOGRÁFICAS

ARDUINO. 2014. Disponível em: <arduino.cc>. Acesso em: 04 Ago.2014.

BANZI. M. **Getting Started with Arduino**. Sebastopol: O'Reilly, 2011.

BRITES, F. G.; SANTOS , V.P. DE A. **Motor de Passo**. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf>. Acesso em: 17 Ago. 2014

CARDOSO, D. C.; TAKAHASHI, E. K. Experimentação remota em atividades de ensino formal: um estudo a partir de periódicos Qualis A. **Revista Brasileira de Pesquisa em Educação em Ciências**, v. 11, n. 3, 2011.

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

CONTEK. **ULN2003 LINEAR INTEGRATED CIRCUIT**. Disponível em: <http://pdf.datasheetcatalog.com/datasheet_pdf/contek-microelectronics/ULN2003.pdf>. Acesso em 12 Ago. 2014.

LOJA VIRTUAL ROBOCORE. Disponível em: <https://www.robocore.net/modules.php?name=GR_LojaVirtual&categoria=13>. Acesso em: 17 Ago. 2014.

MESSIAS, A.R. **CONTROLE DE MOTOR DE PASSO ATRAVÉS DA PORTA PARALELA**. Rogercom. 2014. Disponível em: <http://www.rogercom.com/pparalela/IntroMotorPasso.htm>. Acesso em: 11 Ago. 2014.

NUTEC. Disponível em: <nutec.ufu.br>. Acesso em 18 Ago. 2014.

SILVA, L. C. M.; SANTOS, W. M. S.; DIAS, P. M. C. A carga específica do elétron. Um enfoque histórico e experimental. **Revista Brasileira de Ensino de Física**, v. 33, n. 1, 2011.

TAKAHASHI, E. K; CARVALHO, D. C.; GEDRAITE, R.; NERI, H. G.; MOURA, R. M.; DANTAS, A. C. A weblab for teaching physics. In: ICPE-EPEC 2013: THE INTERNATIONAL CONFERENCE ON PHYSICS EDUCATION (ICPE), 2013, Praga. Disponível em: http://www.icpe2013.org/uploads/BookOfAbstracts_ICPE_EPEC_2013.pdf. Acesso em 18 Ago; 2014.

APÊNDICES

Apêndice I – Código do projeto utilizado na placa Arduino

Segue abaixo o código geral do sistema, responsável pelo controle dos três motores de passo, do acionamento da iluminação dos multímetros e ligação das fontes de alimentação do experimento. Para facilitar sua análise e utilização em outros projetos, o mesmo está comentado e indentado.

```
//-----pinagem motor 1----
int motorPin1 = 2;
int motorPin2 = 3;
int motorPin3 = 4;
int motorPin4 = 5;
//-----pinagem motor 2----
int motorPin5 = 6;
int motorPin6 = 7;
int motorPin7 = 8;
int motorPin8 = 9;
//-----pinagem motor 3----
int motorPin9 = 10;
int motorPin10 = 11;
int motorPin11 = 12;
int motorPin12 = 13;
//------pinagem Relés e Leds-----
int      rele1 = A0;
int      rele2 = A2;
int       led1 = A3;
int       led2 = A4;

int b=0, c=0;
```

```
char bytes_lidos[1];

void setup() {
  // initialize serial communication:
  Serial.begin(9600);

  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
  pinMode(motorPin5, OUTPUT);
  pinMode(motorPin6, OUTPUT);
  pinMode(motorPin7, OUTPUT);
  pinMode(motorPin8, OUTPUT);
  pinMode(motorPin9, OUTPUT);
  pinMode(motorPin10, OUTPUT);
  pinMode(motorPin11, OUTPUT);
  pinMode(motorPin12, OUTPUT);
  pinMode(  rele1   , OUTPUT);
  pinMode(  rele2   , OUTPUT);
  pinMode(  led1    , OUTPUT);
  pinMode(  led2    , OUTPUT);

}

void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    bytes_lidos[0]=Serial.read(); //comando para que a porta serial faça a leitura
    //Serial.readBytes(bytes_lidos,2);
```

```
    //Obs.: bytes_lidos[0] == 1 é o "nome" da variável

//-------------------Relés e Leds-----------------------------//
     if(bytes_lidos[0] == 11)
    {   digitalWrite (rele1, HIGH);
        digitalWrite (led1, HIGH);
        digitalWrite (led2, HIGH);
    }
    if(bytes_lidos[0] == 10)
    {   digitalWrite (rele1, LOW);
        digitalWrite (led1, LOW);
        digitalWrite (led2, LOW);}
   //************************************
    if(bytes_lidos[0] == 13)
    {   digitalWrite (rele2, HIGH);}
    if(bytes_lidos[0] == 12)
    {   digitalWrite (rele2, LOW);}
   //************************************

//------------------------Motor 1-----------------------------//

    if(b==0)
    {  if(bytes_lidos[0] == 1){b=7;
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);//passo [4]
       }
       if(bytes_lidos[0] == 0){b++;
        digitalWrite(motorPin1, HIGH);
```

```
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, LOW);//passo [8]

   }
}
else if(b==1)
{  if(bytes_lidos[0] == 0){b++;
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, HIGH);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, LOW);//passo [12]
   }
   if(bytes_lidos[0] == 1){b--;
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, HIGH);
    digitalWrite(motorPin3, HIGH);
    digitalWrite(motorPin4, LOW);//passo [6]
   }

}
else if(b==2)
{  if(bytes_lidos[0] == 0){b++;
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, HIGH);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, LOW);//passo [4]
   }

   if(bytes_lidos[0] == 1){b--;
    digitalWrite(motorPin1, LOW);
```

```
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, HIGH);
    digitalWrite(motorPin4, LOW);//passo [2]
   }

}
else if(b==3)
{  if(bytes_lidos[0] == 0){b++;
      digitalWrite(motorPin1, LOW);
      digitalWrite(motorPin2, HIGH);
      digitalWrite(motorPin3, HIGH);
      digitalWrite(motorPin4, LOW);//passo [6]
}
   if(bytes_lidos[0] == 1){b--;
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, HIGH);
    digitalWrite(motorPin4, HIGH);//passo [3]
   }

}
else if(b==4)
{  if(bytes_lidos[0] == 0){b++;
      digitalWrite(motorPin1, LOW);
      digitalWrite(motorPin2, LOW);
      digitalWrite(motorPin3, HIGH);
      digitalWrite(motorPin4, LOW);//passo [2]
}
   if(bytes_lidos[0] == 1){b--;
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
```

```
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, HIGH);//passo [1]
   }

}
else if(b==5)
{  if(bytes_lidos[0] == 0){b++;
      digitalWrite(motorPin1, LOW);
      digitalWrite(motorPin2, LOW);
      digitalWrite(motorPin3, HIGH);
      digitalWrite(motorPin4, HIGH);//passo [3]
}
   if(bytes_lidos[0] == 1){b--;
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, HIGH);//passo [9]
   }

}
else if(b==6)
{  if(bytes_lidos[0] == 0){b++;
      digitalWrite(motorPin1, LOW);
      digitalWrite(motorPin2, LOW);
      digitalWrite(motorPin3, LOW);
      digitalWrite(motorPin4, HIGH);//passo [1]
}
   if(bytes_lidos[0] == 1){b--;
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, LOW);
```

```
      digitalWrite(motorPin4, LOW);//passo [8]
     }

  }
  else if(b==7)
  {  if(bytes_lidos[0] == 0){b=0;
       digitalWrite(motorPin1, HIGH);
       digitalWrite(motorPin2, LOW);
       digitalWrite(motorPin3, LOW);
       digitalWrite(motorPin4, HIGH);//passo [9]
  }
     if(bytes_lidos[0] == 1){b--;
      digitalWrite(motorPin1, HIGH);
      digitalWrite(motorPin2, HIGH);
      digitalWrite(motorPin3, LOW);
      digitalWrite(motorPin4, LOW);//passo [12]
     }

  }

//------------------------Motor 2-----------------------------//

  if(c==0)
  {  if(bytes_lidos[0] == 2){c=3;
      digitalWrite(motorPin5, LOW);
      digitalWrite(motorPin6, LOW);
      digitalWrite(motorPin7, HIGH);
      digitalWrite(motorPin8, LOW);//passo [2]
     }
     if(bytes_lidos[0] == 3){c++;
      digitalWrite(motorPin5, HIGH);
```

```
    digitalWrite(motorPin6, LOW);
    digitalWrite(motorPin7, LOW);
    digitalWrite(motorPin8, LOW);//passo [8]

  }
}
else if(c==1)
{  if(bytes_lidos[0] == 3){c++;
    digitalWrite(motorPin5, LOW);
    digitalWrite(motorPin6, HIGH);
    digitalWrite(motorPin7, LOW);
    digitalWrite(motorPin8, LOW);//passo [4]
  }
  if(bytes_lidos[0] == 2){c--;
    digitalWrite(motorPin5, LOW);
    digitalWrite(motorPin6, LOW);
    digitalWrite(motorPin7, LOW);
    digitalWrite(motorPin8, HIGH);//passo [1]
  }

}
else if(c==2)

{  if(bytes_lidos[0] == 3){c++;
    digitalWrite(motorPin5, LOW);
    digitalWrite(motorPin6, LOW);
    digitalWrite(motorPin7, HIGH);
    digitalWrite(motorPin8, LOW);//passo [2]
  }

  if(bytes_lidos[0] == 2){c--;
```

```
      digitalWrite(motorPin5, HIGH);
      digitalWrite(motorPin6, LOW);
      digitalWrite(motorPin7, LOW);
      digitalWrite(motorPin8, LOW);//passo [8]
     }

  }
  else if(c==3)
  {  if(bytes_lidos[0] == 3){c=0;
       digitalWrite(motorPin5, LOW);
       digitalWrite(motorPin6, LOW);
       digitalWrite(motorPin7, LOW);
       digitalWrite(motorPin8, HIGH);//passo [1]
  }
     if(bytes_lidos[0] == 2){c--;
      digitalWrite(motorPin5, LOW);
      digitalWrite(motorPin6, HIGH);
      digitalWrite(motorPin7, LOW);
      digitalWrite(motorPin8, LOW);//passo [4]
     }

  }
//------------------------Motor 3-----------------------------//

  if(b==0)
  {  if(bytes_lidos[0] == 4){b=3;
      digitalWrite(motorPin9, LOW);
      digitalWrite(motorPin10, LOW);
      digitalWrite(motorPin11, HIGH);
      digitalWrite(motorPin12, LOW);//passo [2]
     }
```

```
   if(bytes_lidos[0] == 5){b++;
    digitalWrite(motorPin9, HIGH);
    digitalWrite(motorPin10, LOW);
    digitalWrite(motorPin11, LOW);
    digitalWrite(motorPin12, LOW);//passo [8]

   }
}
else if(b==1)
{  if(bytes_lidos[0] == 5){b++;
    digitalWrite(motorPin9, LOW);
    digitalWrite(motorPin10, HIGH);
    digitalWrite(motorPin11, LOW);
    digitalWrite(motorPin12, LOW);//passo [4]
   }
   if(bytes_lidos[0] == 4){b--;
    digitalWrite(motorPin9, LOW);
    digitalWrite(motorPin10, LOW);
    digitalWrite(motorPin11, LOW);
    digitalWrite(motorPin12, HIGH);//passo [1]
   }

}
else if(b==2)

{  if(bytes_lidos[0] == 5){b++;
    digitalWrite(motorPin9, LOW);
    digitalWrite(motorPin10, LOW);
    digitalWrite(motorPin11, HIGH);
    digitalWrite(motorPin12, LOW);//passo [2]
   }
```

```
      if(bytes_lidos[0] == 4){b--;
       digitalWrite(motorPin9, HIGH);
       digitalWrite(motorPin10, LOW);
       digitalWrite(motorPin11, LOW);
       digitalWrite(motorPin12, LOW);//passo [8]
      }

   }
   else if(b==3)
   {  if(bytes_lidos[0] == 5){b=0;
        digitalWrite(motorPin9, LOW);
        digitalWrite(motorPin10, LOW);
        digitalWrite(motorPin11, LOW);
        digitalWrite(motorPin12, HIGH);//passo [1]
   }
      if(bytes_lidos[0] == 4){b--;
       digitalWrite(motorPin9, LOW);
       digitalWrite(motorPin10, HIGH);
       digitalWrite(motorPin11, LOW);
       digitalWrite(motorPin12, LOW);//passo [4]
      }
   }
  }//end Loop
}
```

Apêndice II – Código do software local

Segue abaixo o código do programa local montado para realização de testes nos motores. Este programa foi construído no ambiente Visual Studio. Para facilitar sua análise e utilização em outros projetos, o mesmo está comentado e indentado.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        SerialPort porta1 = new SerialPort("COM3", 9600, Parity.None, 8, StopBits.One);
//Configura a porta serial e sua velocidade (9600)
        private void send_Click(object sender, EventArgs e)//ESQUERDA (1)
        {
            byte[] buff = new byte[1]; //O valor dentro das chaves indica o tamanho do vetor.
            buff[0] = (byte) 1;//envia valor (1). Esse valor é um número inteiro, ou seja, é um
valor de variável
```

```
            porta1.Open(); //abre a porta serial
            porta1.Write(buff, 0, 1); //começa a enviar o valor do 0 ao 1 [do primeiro
termo(0) ao último(1)]

            porta1.Close(); //fecha porta serial
        }
        private void read_Click(object sender, EventArgs e)//DIREITA (0)
        {
            byte[] buff = new byte[1];
            buff[0] = (byte)0; //envia valor (0)
            porta1.Open();
            porta1.Write(buff, 0, 1);
            porta1.Close();
         }

        private void button1_Click(object sender, EventArgs e)  //MOTOR 2__
ESQUERDA(2)
        {
            byte[] buff = new byte[1];
            buff[0] = (byte)2;
            porta1.Open();
            porta1.Write(buff, 0, 1);
            porta1.Close();
        }

        private void button2_Click(object sender, EventArgs e)  //MOTOR 2__
ESQUERDA(3)
        {
            byte[] buff = new byte[1];
            buff[0] = (byte)3;
```

```
            porta1.Open();
            porta1.Write(buff, 0, 1);
            porta1.Close();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            byte[] buff = new byte[1];
            buff[0] = (byte)4;
            porta1.Open();
            porta1.Write(buff, 0, 1);
            porta1.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            byte[] buff = new byte[1];
            buff[0] = (byte)5;
            porta1.Open();
            porta1.Write(buff, 0, 1);
            porta1.Close();
        }
    }
}
```