DESENVOLVIMENTO DE UM SEMÁFORO UTILIZANDO A PLATAFORMA ARDUINO

Erick Fonseca Boaventura

MANUAL DO PROFESSOR

INSTITUTO FEDERAL
Minas Gerais

Erick Fonseca Boaventura

DESENVOLVIMENTO DE UM SEMÁFORO UTILIZANDO A PLATAFORMA ARDUINO

1ª Edição

Belo Horizonte

Instituto Federal de Minas Gerais

2022

Pró-reitor de Pesquisa	Fernando Gomes Braga
Coordenador de Inovação	Norimar de Melo Verticchio
Autor do curso	Erick Fonseca Boaventura
Arte gráfica	Ângela Bacon
Diagramação	Eduardo dos Santos Oliveira

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP)

Índice para catálogo sistemático:

1. Não preencher

Não preencher

2022

Direitos exclusivos cedidos à

Instituto Federal de Minas Gerais

Avenida Mário Werneck, 2590,

CEP: 30575-180, Buritis, Belo Horizonte – MG,

Telefone: (31) 2513-5157

Sobre o material

Este curso é autoexplicativo e não possui tutoria. O material didático, incluindo suas videoaulas, foi projetado para que você consiga desenvolver e ministrar esse curso de forma autônoma e suficiente.

Caso opte por imprimir este *e-book*, você não perderá a possiblidade de acessar os materiais multimídia e complementares. Os *links* podem ser acessados usando o seu celular, por meio do glossário de Códigos QR disponível no fim deste manual.

Embora o material passe por revisão, somos gratos em receber suas sugestões para possíveis correções (erros ortográficos, conceituais, *links* inativos etc.). A sua participação é muito importante para a nossa constante melhoria. Para relatar qualquer erro ou sugestão de melhoria, envie um e-mail para nit@ifmg.edu.br, com o seguinte assunto: MELHORIA NO MATERIAL DO LIC

Para saber mais sobre a Inovação do IFMG acesse

www.ifmg.edu.br

Apresentação dos Ícones

Os ícones são elementos gráficos para facilitar os estudos, fique atento quando eles aparecem no texto. Veja aqui o seu significado:

Atenção: indica pontos de maior importância na atividade e que exige mais atenção do aluno e instrutor.

Dica do professor: novas informações ou curiosidades relacionadas ao tema em estudo.

Mídia digital: sugestão de recursos audiovisuais para enriquecer a aprendizagem.

Para que os circuitos eletrônicos, que são controlados pela plataforma Arduino, possam funcionar, tem-se duas partes essenciais:

1 – O circuito eletrônico montado fisicamente e conectado a placa Arduino (*hardware*);

2 – A programação do microcontrolador da placa Arduino (*software*).

1.1. O circuito eletrônico

Para a realização da montagem do *hardware*, será necessário explicar conhecimentos básicos de montagem de circuitos eletrônicos. Mais especificamente será necessário a explicação de como funciona a placa Arduino, o protoboard, a função dos jumpers, dos botões, resistores e LED's.

1.1.1. Placa Arduino

A placa Arduino possui um microcontrolador que pode ser programado. O microcontrolador é um componente eletrônico que funciona como um 'cérebro' do sistema, pois capta as informações que chegam para ele, processa essas informações de acordo com a sua programação e realiza algo, como resultado do seu processamento.

A Figura 1 mostra a placa Arduino, com destaque para:

- Botão de *reset*: possui a função de reiniciar a programação;

- Pinos digitais de 0 a 13: podem ser utilizados para receber um sinal (entrada) ou enviar um sinal (saída) de 5V;

- Entrada para conexão do cabo USB: entrada na qual será ligado o cabo USB entre o notebook/computador e a placa Arduino, para a gravação do código a ser executado pelo microcontrolador;

- Microcontrolador: receberá o código a ser executado;

- Pinos de alimentação: disponibilizam 5V (+) e GND (-), para serem utilizados pelos circuitos que serão conectados a placa Arduino.

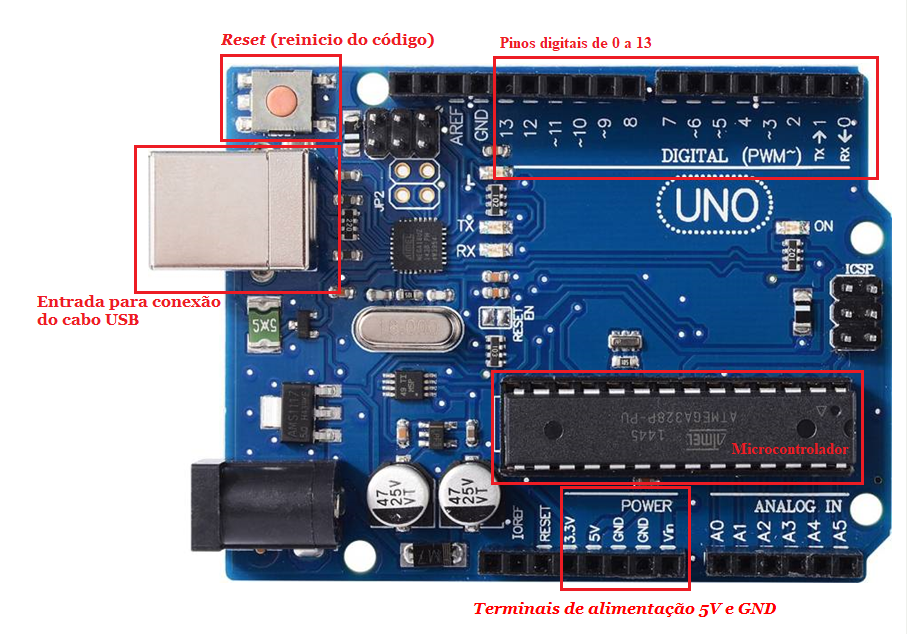


Figura 1 – Placa Arduino.
Fonte: Próprio autor (2022).

1.1.2. Protoboard/Matriz de contatos

O protoboard ou matriz de contatos, é um equipamento que nos permite realizar as conexões elétricas entre os componentes sem a necessidade de soldá-los, facilitando o desenvolvimento de projetos e testes.

O protoboard possui furos organizados em grupos de 5 (vertical) na parte central, ou vários grupos de 5 furos (horizontal) na parte de cima e de baixo, como mostra a Figura 2. A interligação elétrica existe somente entre os furos pertencentes ao mesmo grupo. Ou seja, na Figura 2, os furos estão interligados em grupos de 5 nas colunas, e interligados em vários grupos de 5 furos nas linhas.

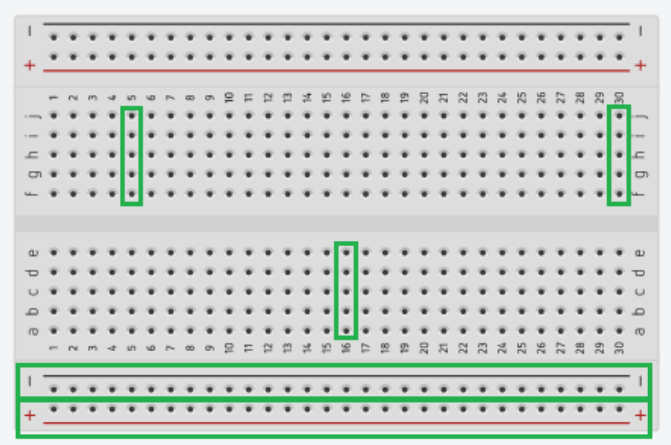


Figura 2 – Ligação interna protoboard/matriz de contatos.

Fonte: Próprio autor (2022).

1.1.3. LED's, resistores e jumpers

O LED (Diodo Emissor de Luz) é um componente eletrônico feito de um material semicondutor, que quando é percorrido por uma corrente elétrica, emite luz.

O Resistor é um componente eletrônico, que tem a finalidade de limitar a corrente elétrica em um circuito.

Os jumpers servem para realizar as conexões elétricas entre os componentes, facilitando o desenvolvimento de projetos e testes.

Para que os LED's não queimem por excesso de corrente, é necessário liga-los em série com um resistor. Para a conexão dos LED's ao Arduino, deve-se fazer uma ligação em série entre um resistor e o LED, no qual o negativo do LED ficará ligado ao GND da placa do Arduino e o resistor a um pino de 0 a 13 da placa Arduino, conforme demonstrado na Figura 3.

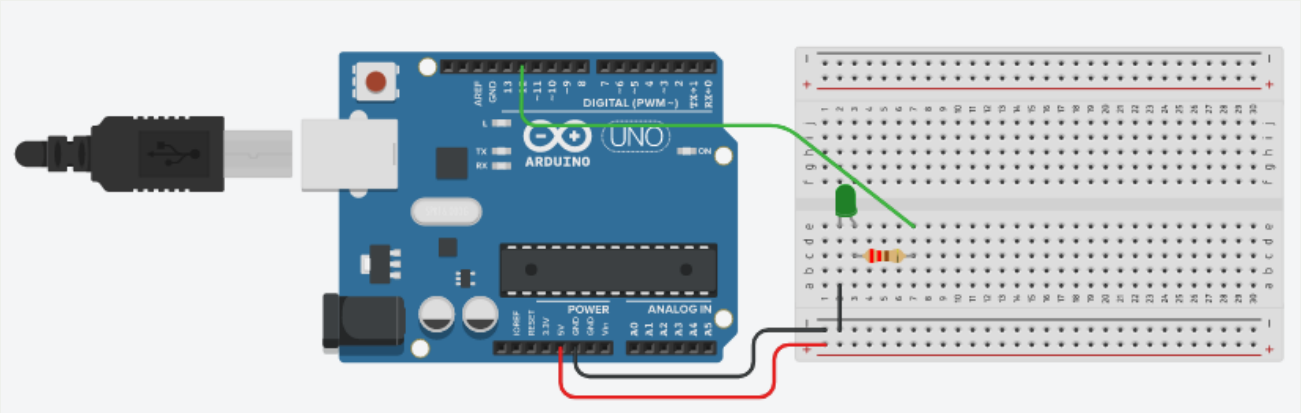


Figura 3 – Montagem no protoboard de um LED conectado ao Arduino.
Fonte: Próprio autor (2022).

1.1.4. Botões, resistores e jumpers

O botão tem a função de deixar passar ou não corrente elétrica por ele, ou seja, ele abre ou fecha um contato, que permite ou não a passagem da corrente elétrica em um circuito.

Para a conexão dos botões ao Arduino, sugere-se a ligação padrão (nível lógico baixo, enquanto o botão estiver SOLTO, para isto conectamos um resistor entre um pino de 0 a 13 da placa do Arduino e o GND. E nível lógico alto, enquanto o botão estiver PRESSIONADO, para isto, devemos conectar o botão entre o pino de 0 a 13 escolhido na placa do Arduino e o 5V), conforme demonstrado na Figura 4.

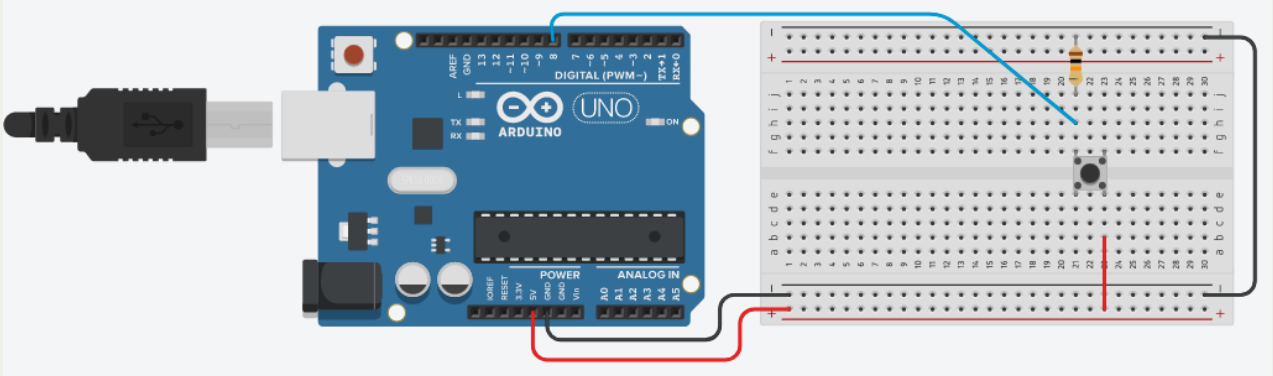


Figura 4 – Montagem no protoboard de um botão conectado ao Arduino.
Fonte: Próprio autor (2022).

1.2. A programação

Para o desenvolvimento da parte de *software*, será necessário explicar conhecimentos básicos da linguagem C e comandos específicos da plataforma Arduino. Mais especificamente para o desenvolvimento do semáforo para carros será necessário a explicação da sintaxe da linguagem de programação, dos operadores matemáticos e lógicos, de variáveis, dos métodos padrões *void loop* e *void setup*, dos comandos *if, while, pinMode, digitalRead, digitalWrite* e *delay*. Também será necessário a utilização de um notebook/computador com o programa instalado (Arduino) para que seja possível criar a programação e gravar na placa Arduino.

1.2.1. Sintaxe da linguagem de programação

Como podemos ter mais de um comando na mesma linha, temos que usar um caractere para informar para o compilador onde o nosso comando terminou. Este caractere é o ‘;’ (ponto e vírgula), sendo obrigatório em todos os comandos, menos nos controladores de fluxos ou métodos, ou seja, NÃO é usado onde iniciamos e finalizamos com ‘{}’ (chaves), como no *void loop*() e *void setup*().

O compilador diferencia letras maiúsculas e minúsculas (*case-sensitive*), então devemos prestar atenção na forma correta de cada comando.

Podemos colocar um comentário, ou seja, um texto qualquer que faz sentido ou serve apenas para informar algo relevante sobre o programa apenas para a pessoa que está programando ou lendo o programa, sendo que NÃO exerce nenhuma influência sobre o código propriamente dito.

Para adicionar um comentário de APENAS UMA linha, usamos ‘//’ e para MAIS DE UMA linha usamos ‘/*’ para iniciar e ‘*/’ para finalizar.

Exemplos:

```
//Este é um comentário de APENAS uma linha.

/* Aqui começa um comentário de MAIS DE UMA linha.
Aqui continua...
Aqui termina o comentário de MAIS DE UMA linha*/
```

1.2.2. Operadores matemáticos e lógicos

A seguir apresenta-se os principais operadores matemáticos e lógicos:

‘+’ – Realiza a soma (A = B + C);

‘-’ – Realiza a subtração (A = B - C);

‘*’ – Realiza a multiplicação (A = B * C);

‘/’ – Realiza a divisão (A = B / C);

‘%’ – Resto da divisão (A = A % B);

‘=’ – Atribui valor, ou seja, faz ficar igual (A = B);

‘<’ – Compara se é menor que (A < B);

‘>’ – Compara se é maior que (A > B);

‘<=’ – Compara se é menor ou igual à (A <= B);

‘>=’ – Compara se é maior ou igual à (A >= B);

‘==’ – Compara se é igual à (A == B).

1.2.3. Variáveis

Variáveis são espaços de memória RAM que damos um nome qualquer (sugestivo) e gravamos qualquer dado que seja necessário. O dado gravado em uma variável pode ser alterado a qualquer momento.

As variáveis são usadas normalmente para gravar informações temporárias durante a execução do programa, como por exemplo: O número de vezes que um LED piscou, quantas vezes o botão foi pressionado, um caractere de uma senha durante a digitação da senha, o número de um pino de entrada/saída, entre outros.

Estrutura:

Tipo nome = valor inicial

Tipo: boolean (1 bit, 1 ou 0), byte (8 bits, 0 a 255), int (16 bits, -32768 a 32767), unsigned int (16 bits, 0 a 65535), etc.

Nome: Nome sugestivo qualquer.

Valor inicial: Valor que a variável deve iniciar, normalmente 0 (zero).

Exemplo: byte LED = 13; //Cria uma variável do tipo byte, de nome LED e grava nela o valor 13. Na prática o LED ficaria ligado fisicamente ao pino 13 da placa Arduino.

byte BT = 5; //Cria uma variável do tipo byte, de nome BT e grava nela o valor 5. Na prática o botão BT ficaria ligado fisicamente ao pino 5 da placa Arduino.

1.2.4. Métodos padrões *void loop* e *void setup*

Os métodos padrões para se programar o Arduino são *void setup()* e *void loop()*.

A palavra '*void*' é inglês e significa vazio, a palavra '*setup*' significa configuração e a palavra '*loop*' significa infinito (execução infinita). Dentro do *void setup()* são colocados os comandos de configuração do código e no *void loop ()* normalmente é feita a resolução do problema.

Exemplo:

```
void setup()
{


}
void loop()
{


}
```

1.2.5. Comandos *if, while, pinMode, digitalRead, digitalWrite* e *delay*

if(condição) //SE – Testa se a condição é verdadeira, e caso seja, executa o Código.

{

//Código.

}

Exemplo:

if(contador == X) //Executa o código se a variável 'contador' for IGUAL a variável ‘X’.

{

X = 10; //Código a ser executado SE a condição for VERDADEIRA.

}

while(condição) //ENQUANTO.

{

//Código executado ENQUANTO a condição for VERDADEIRA.

}

Exemplo:

while(contador < X) //Executa o código enquanto a variável 'contador' for MENOR que a variável ‘X’.

{

contador = contador + 1; //Código a ser repetido ENQUANTO a condição for VERDADEIRA.

}

pinMode(pino,modo); //MODO DO PINO – Se será entrada ou saída.

Pino: Qualquer pino digital (0 a 13).

Modo: **INPUT** (entrada) ou **OUTPUT** (saída) //Tudo maiúsculo.

Definimos como entrada todos os dispositivos que enviam um sinal para a placa Arduino. Exemplo: botão, sensor, contato de saída de um relé, outro circuito eletrônico, etc.

Definimos como saída todos os dispositivos que receberão um sinal (que serão comandados) da placa Arduino. Exemplo: LED, motor, base de um transistor, outro circuito eletrônico, etc.

Exemplo:

pinMode(LED, OUTPUT); // Define o LED como saída.

pinMode(botao, INPUT); // Define o botão como entrada.

digitalRead(pino); //TESTA A ENTRADA – Retorna o nível lógico (1 – verdadeiro ou 0 – falso) de um pino digital (0 a 13), normalmente usado em conjunto com outro comando.

Exemplo:

```
if(digitalRead(BT)==1) //Testa se o botão BT foi apertado/pressionado.
{
        //Executa este código, caso o botão BT seja apertado/pressionado.
}
while(digitalRead(BT)==0) //Testa se o botão BT não foi apertado/pressionado.
{
        //Executa este código enquanto o botão BT não seja apertado/pressionado.
}
```

digitalWrite(pino,nível); //ESCREVE DIGITAL – Liga ou desliga um pino de saída digital.

Pino: Qualquer pino digital (0 a 13) ou nome dado ao pino.

Nível: **HIGH** ou **1** (Ligado, nível Alto), **LOW** ou **0** (Desligado, nível Baixo).

Exemplo: digitalWrite(LED,HIGH); // Liga o LED.

digitalWrite(LED,LOW); // Desliga o LED.

delay(tempo); //ESPERA – Aguarda um tempo em milisegundos (ms) com limite de (32 bits).

Exemplo: delay(1000); // Espera 1s = 1000ms para depois continuar o código.

1.3. Situação problema

Luiz desde pequeno sempre foi empolgado com carros. Ao completar 18 anos, logo iniciou o curso para tirar a sua carteira de habilitação para carros. Em uma das aulas do curso de legislação, o professor Bruno estava explicando sobre a importância de se respeitar as cores do semáforo.

Bruno explicou que um semáforo para carros possui 3 lâmpadas, sendo uma da cor vermelha, outra amarela e uma verde e que cada cor tem um significado importante:

- Vermelha: parar;

- Amarela: atenção, mostrando a iminência da parada obrigatória;

- Verde: seguir em frente.

Bruno ainda explicou que em um semáforo para carros a lâmpada verde fica ligada por um tempo e assim que esta lâmpada apaga, a lâmpada amarela acende no mesmo instante. Em seguida a lâmpada amarela fica acesa por um tempo e assim que ela apaga, a lâmpada vermelha acende. Ao acender a lâmpada vermelha, ela fica ligada por um tempo e assim que esta lâmpada apaga, a lâmpada verde acende novamente reiniciando o ciclo. Assim, em um semáforo para carros sempre uma lâmpada estará acesa e a outras duas estarão apagadas, para não confundir os motoristas sobre a atitude a ser tomada.

Luiz ficou tão empolgado com a aula, que após o seu término, ficou imaginando como o semáforo funcionava internamente, para que cada lâmpada acendesse e apagasse no momento exato e com tempos diferentes. Então, correu para a casa do seu avô Francisco, que é Técnico em Eletrônica e Engenheiro de Controle e Automação formado pelo IFMG, que lhe explicou nos mínimos detalhes tudo que estava por trás do funcionamento do semáforo para carros.

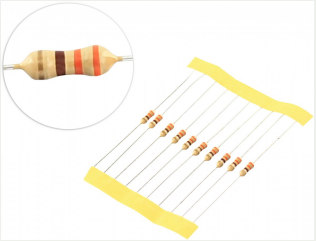
1.4. Questão motriz

Como fazer um semáforo para carros utilizando a plataforma Arduino?

Materiais necessários

O Quadro 1 apresenta a lista de materiais necessária para a atividade.

Item	Nome do item	Descrição	Quant.	Foto	Substituição
1	Notebook	Notebook para a programação do microcontrolador da plataforma Arduino	1		Computador de mesa
2	Arduino + cabo USB	Placa Arduino + cabo USB que vem junto da placa	1		Não há substituto
3	Protoboard	Protoboard/Matriz de contatos	1		Não há substituto
4	LED verde	LED monocromático na cor verde	1		Não há substituto

5	LED amarelo	LED monocromático na cor amarela	1		Não há substituto
6	LED vermelho	LED monocromático na cor vermelha	1		Não há substituto
7	Resistor LED	Resistor de 180Ω à 560Ω - 1/8W	3		Não há substituto
8	Botão	Chave táctil de 2 ou 4 pinos	2		Não há substituto
9	Resistor Botão	Resistor de 1KΩ à 10KΩ - 1/8W	2		Não há substituto

10	Jumpers	Jumper macho-macho	15		Não há substituto

Quadro 1 – Lista de materiais.
Fonte: Próprio autor (2022).

Na atividade a ser realizada, o semáforo para carros será montado com a ajuda do protoboard e demais componentes, sendo as lâmpadas do semáforo representadas pelos LED's vermelho, amarelo e verde. Sugere-se que o circuito funcione inicialmente da seguinte forma:

• Após um botão (botão 1) ser apertado:

1. O sinal verde (LED verde) acende por 3s e apaga;

2. Instantaneamente o sinal amarelo (LED amarelo) liga, fica aceso por 1s, apagando após este tempo;

3. Assim que o sinal amarelo apaga, o sinal vermelho (LED vermelho) liga instantaneamente, fica aceso por 3s e apaga;

4. Após o sinal vermelho apagar, o processo será reiniciado (item 1), ou seja, o semáforo irá ficar repetindo o ciclo representado pela Figura 5, até que um outro botão (botão 2) seja pressionado até o final do ciclo e apague todos os LED's.

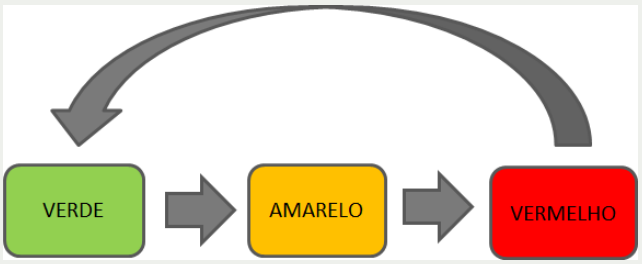


Figura 5 – Ciclo do semáforo de carros.
Fonte: Próprio autor (2022).

3.1 O circuito eletrônico (Etapa 1)

Inicialmente distribua todos os materiais necessários para os alunos.

Oriente os alunos para conectar o pino 5V da placa Arduino em uma das trilhas da parte externa do protoboard (das que são conectadas internamente na horizontal), através de um jumper vermelho. Oriente os alunos sobre a padronização de cores: vermelha para 5V e preta para GND. Demais cores usadas de forma livre.

Peça aos alunos que conecte o pino GND da placa Arduino em uma das trilhas da parte externa do protoboard (das que são conectadas internamente na horizontal) diferente da trilha de ligação do 5V, através de um jumper preto. A Figura 6 ilustra a ligação.

O objetivo desta parte é deixar disponível o sinal de 5V (nível lógico 1) ou GND (nível lógico 0) de fácil acesso no protoboard.

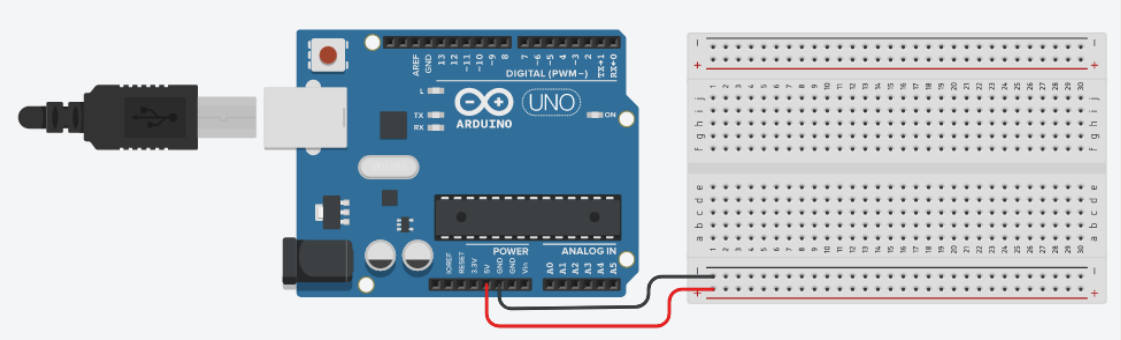


Figura 6 – Ligação dos pinos 5V e GND da placa Arduino ao protoboard.
Fonte: Próprio autor (2022).

Solicite que os alunos iniciem a montagem do circuito pelos LED's (verde, amarelo e vermelho) e seus respectivos resistores, por demandarem menos ligações. Oriente os alunos para que montem os LED's no protoboard na sequência lógica do semáforo (cores verde, amarela e vermelha, nesta ordem especificamente), para ficar mais fácil verificar o funcionamento posteriormente.

Atenção: Observar a polaridade dos LED's.

O terminal negativo (terminal do LED menor ou lado chanfrado) de cada um dos LED's deve ser conectado a trilha do protoboard que se conectou o GND inicialmente, através dos jumpers preto. O outro terminal (positivo) de cada LED, deve ser conectado com um dos terminais do seu respectivo resistor, utilizando uma trilha qualquer do protoboard, para cada conexão separadamente. O outro lado do resistor que ainda não foi utilizado, deve ser conectado com um jumper de qualquer cor, através de qualquer trilha do protoboard. O outro lado do jumper deve ser levado aos pinos digitais do Arduino. Padronize com os alunos a ligação do jumper do resistor do LED verde no pino 12, do jumper do resistor do LED amarelo no pino 11 e do jumper do resistor do LED vermelho no pino 10. A Figura 7 traz um exemplo de como os LED's podem ser montados no protoboard e como podem ser conectados a placa Arduino.

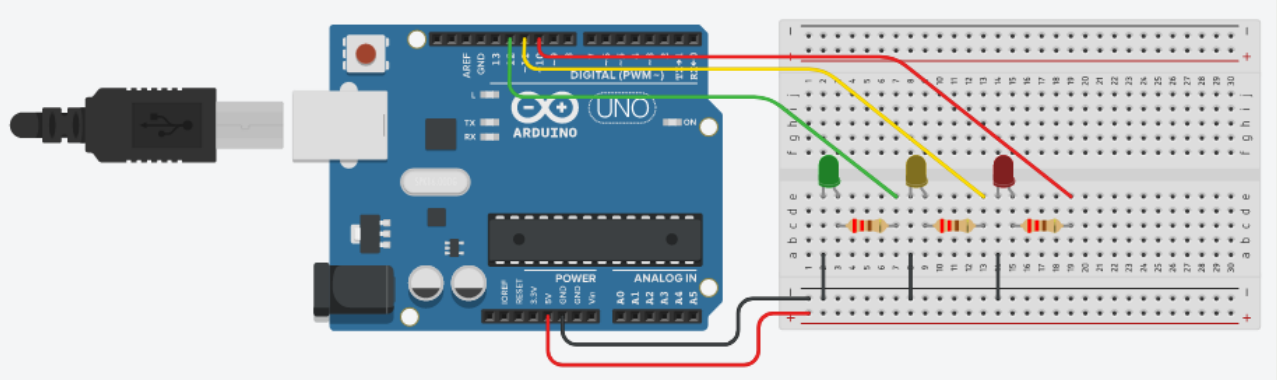


Figura 7 – Montagem dos LED's e seus respectivos resistores no protoboard.
Fonte: Próprio autor (2022).

Agora oriente os alunos para montarem os botões, com os seus respectivos resistores no protoboard. Um lado de cada botão, deve ser conectado a trilha do protoboard que já se conectou o 5V inicialmente. O outro lado de cada botão que ainda não foi utilizado (no caso do botão de 4 terminais, o outro lado é o lado oposto na diagonal da primeira ligação), deve ser conectado a um dos terminais de um jumper e a um dos terminais do resistor do botão, utilizando uma trilha qualquer do protoboard, para cada conexão separadamente. O outro terminal de cada resistor de cada botão deve ser conectado a trilha do GND e o outro terminal de cada jumper que foi ligado junto ao botão e ao resistor do botão, deve ser conectado aos pinos digitais do Arduino. Padronize com os alunos a ligação do jumper do botão 1 no pino 8 e do botão 2 no pino 9.

A Figura 8 demonstra como os botões podem ser montados no protoboard e como podem ser conectados a placa Arduino.

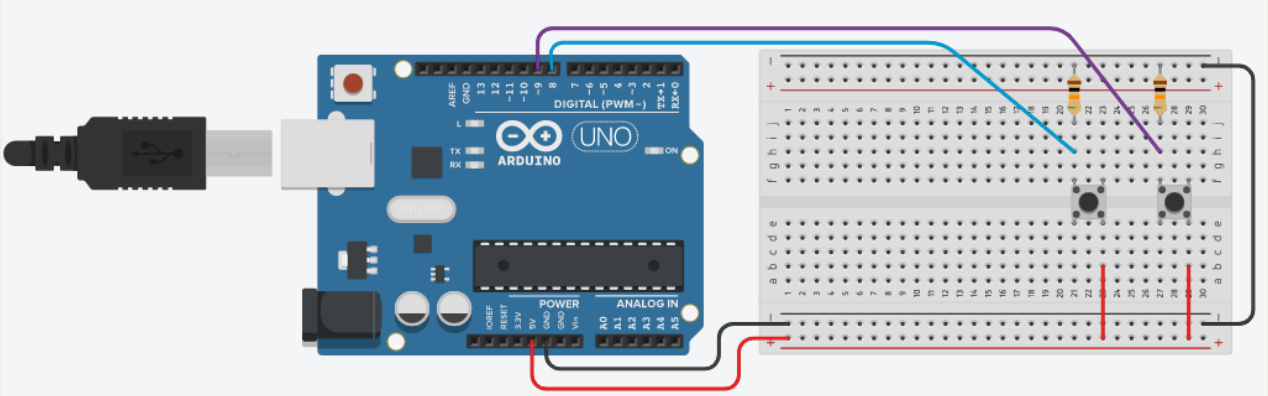


Figura 8 – Montagem dos botões e seus respectivos resistores no protoboard.
Fonte: Próprio autor (2022).

Enfim, a Figura 9 exemplifica a montagem completa do circuito do semáforo no protoboard.

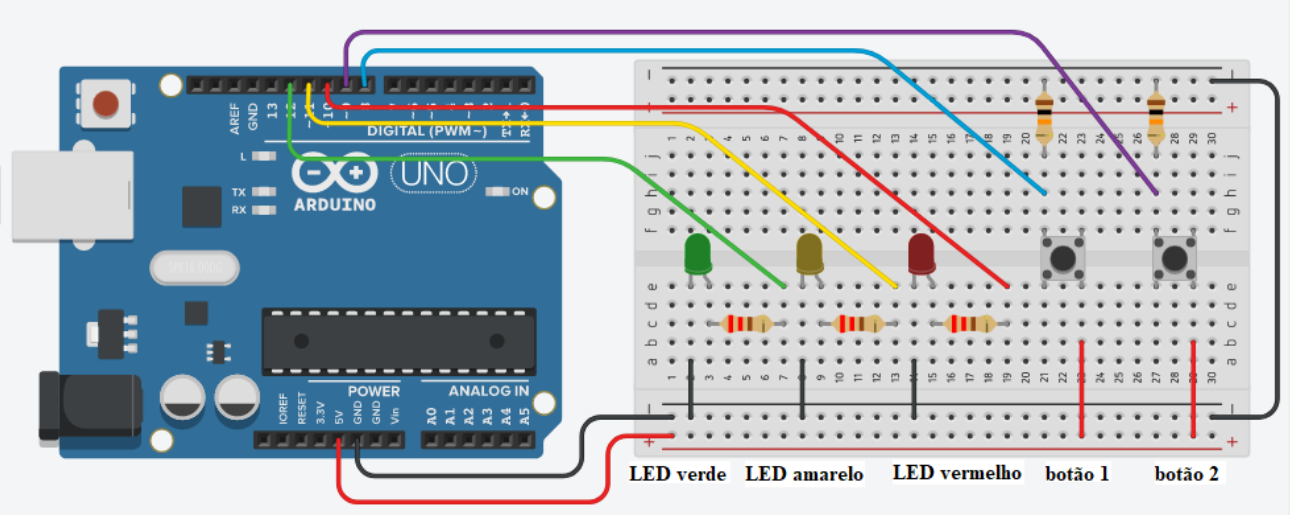


Figura 9 – Montagem completa do circuito do semáforo no protoboard.
Fonte: Próprio autor (2022).

3.2 A programação (Etapa 2)

Inicialmente solicite aos alunos para ligarem o notebook/computador e abrirem o programa do Arduino. A Figura 10 apresenta a tela inicial do programa Arduino.

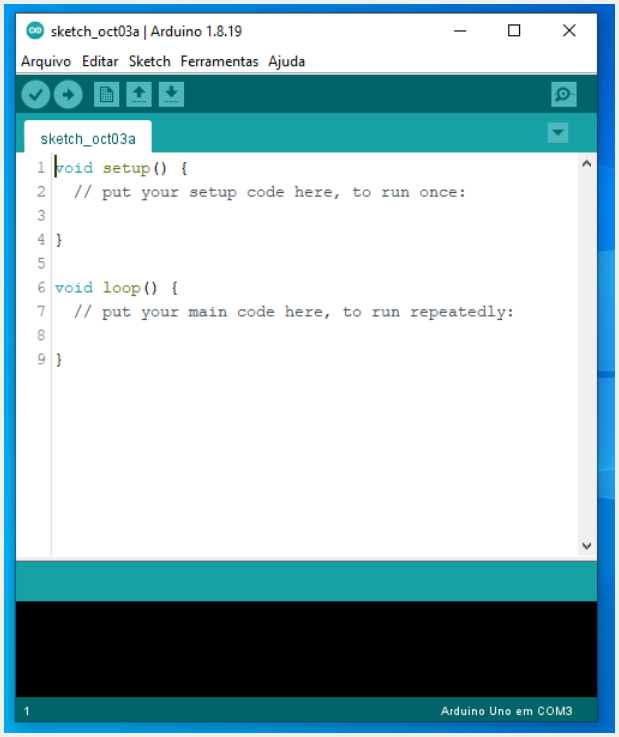


Figura 10 – Tela inicial do programa Arduino.
Fonte: Próprio autor (2022).

Em seguida, oriente os alunos a começarem a programação pela declaração de variáveis (que deve ser feita antes do *void setup*), definindo em qual dos pinos digitais do Arduino ficará ligado cada componente (LED's e botões). Padronize com os alunos a ligação do botão 1 no pino 8, do botão 2 no pino 9, do LED vermelho no pino 10, do LED amarelo no pino 11 e do LED verde no pino 12.

Atenção: É necessário que os pinos definidos na programação coincidam com os pinos em que foram ligados os componentes eletrônicos fisicamente na placa Arduino.

Agora os alunos devem configurar dentro das '{ }' (chaves) do *void setup* os botões como dispositivos de entrada e os LED's como dispositivos de saída.

Oriente os alunos a pensarem sobre a resolução do problema, que deve ficar dentro das '{ }' (chaves) do *void loop*. Lembre-se que não existe uma única solução e que cada aluno pode encontrar uma maneira diferente de resolver o problema.

O primeiro passo para a resolução do problema consiste em testar se o botão 1 foi apertado, caso seja (==1) o semáforo deve ficar funcionando (verde □ amarelo □

vermelho) até que o botão 2 seja pressionado até o final do ciclo, ou seja, enquanto o botão 2 for (==0) o semáforo fica funcionando, mas quando o botão 2 é pressionado a condição se torna falsa e o semáforo para de funcionar.

Em seguida é necessário mandar ligar e desligar o conjunto de LED's (verde, amarelo e vermelho) de acordo com o proposto na atividade e respeitando os intervalos de tempo pré-definidos.

Por fim, deve-se lembrar de mandar desligar todos os LED's antes de sair do *loop* correspondente ao botão 1, para que quando o teste do botão 2 se torne falso o semáforo apague e pare de funcionar.

A Figura 11 traz um exemplo de um código funcional para o semáforo proposto na atividade. O código está comentado em todas as suas linhas, facilitando o seu entendimento.

semaforo_LIC | Arduino 1.8.19

Arquivo Editar Sketch Ferramentas Ajuda

semaforo_LIC

```
// Declaração de variáveis e definições de pinos utilizados
byte botao_1 = 8; // Define o pino do Arduino que o botão 1 será conectado
byte botao_2 = 9; // Define o pino do Arduino que o botão 2 será conectado
byte LED_vermelho = 10; // Define o pino do Arduino que o LED vermelho será conectado
byte LED_amarelo = 11; // Define o pino do Arduino que o LED amarelo será conectado
byte LED_verde = 12; // Define o pino do Arduino que o LED verde será conectado

void setup () // Configuração
{
 pinMode(botao_1, INPUT); // Define o pino como entrada
 pinMode(botao_2, INPUT); // Define o pino como entrada
 pinMode(LED_verde, OUTPUT); // Define o pino como saída
 pinMode(LED_amarelo, OUTPUT); // Define o pino como saída
 pinMode(LED_vermelho, OUTPUT); // Define o pino como saída
}

void loop () // Execução infinita
{
 if(digitalRead (botao_1)==1) // Se o botão 1 for apertado torna-se verdadeiro e executa o código dentro das {}
  {
   while (digitalRead (botao_2)==0) // Enquanto o botão 2 não é pressionado fica executando infinitamenteo o semáforo
    {
    digitalWrite (LED_verde, 1); // Liga o led verde
    digitalWrite (LED_amarelo, 0); // Desliga o led amarelo
    digitalWrite (LED_vermelho, 0); // Desliga o led vermelho
    delay (3000); // Espera 3 segundos
    digitalWrite (LED_verde, 0); // Desliga o led verde
    digitalWrite (LED_amarelo, 1); // Liga o led amarelo
    digitalWrite (LED_vermelho, 0); // Desliga o led vermelho
    delay (1000); // Espera 1 segundo
    digitalWrite (LED_verde, 0); // Desliga o led verde
    digitalWrite (LED_amarelo, 0); // Desliga o led amarelo
    digitalWrite (LED_vermelho, 1); // Liga o led vermelho
    delay (3000); // Espera 3 segundos
    }
   digitalWrite (LED_verde, 0); // Desliga o led verde
   digitalWrite (LED_amarelo, 0); // Desliga o led amarelo
   digitalWrite (LED_vermelho, 0); // Desliga o led vermelho
  }
```

Figura 11 – Código do semáforo para carros no programa Arduino.
Fonte: Próprio autor (2022).

Após finalizado o código, para testar o funcionamento do semáforo é necessário enviar o código para o microcontrolador da placa Arduino. Primeiramente conecte a placa Arduino em alguma entrada USB do notebook/computador, com o cabo USB que acompanha a placa Arduino. Após feito isso, vá no menu superior do programa, em **Ferramentas (1°)**, **Porta (2°)** e selecione a porta serial - **COM (3°)**, na qual aparece o nome do Arduino entre parênteses, conforme demonstrado na Figura 12.

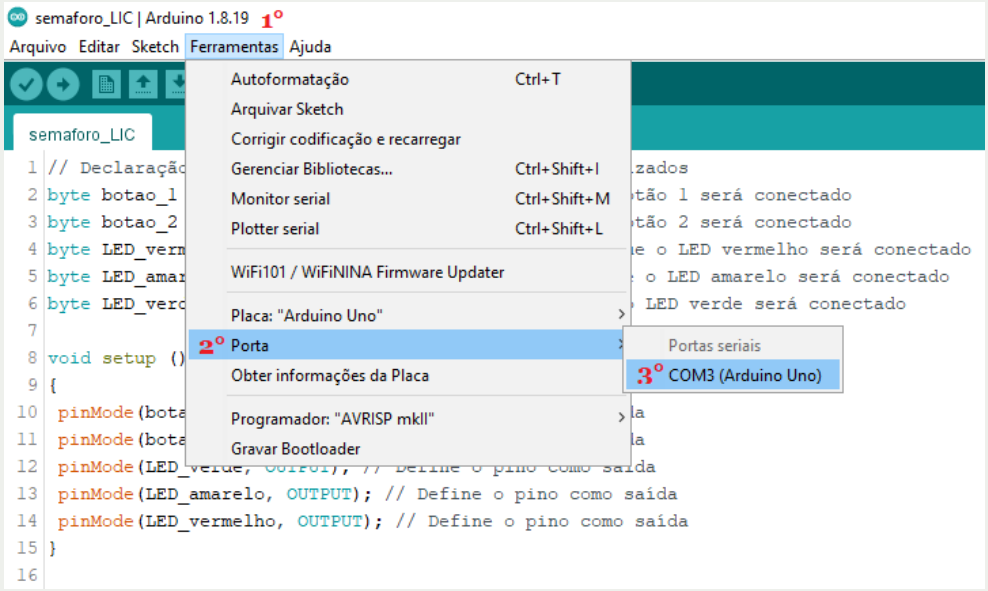


Figura 12 – Procedimento para selecionar a porta serial (USB), na qual a placa Arduino está conectada.
Fonte: Próprio autor (2022).

Após selecionada a porta serial corretamente, para enviar o código, basta clicar na **seta que aponta para a direita**, no menu superior. Assim que terminar de gravar o código, o programa mostrará a mensagem “**carregado**”, no menu inferior, do lado esquerdo do programa, conforme exemplifica a Figura 13.

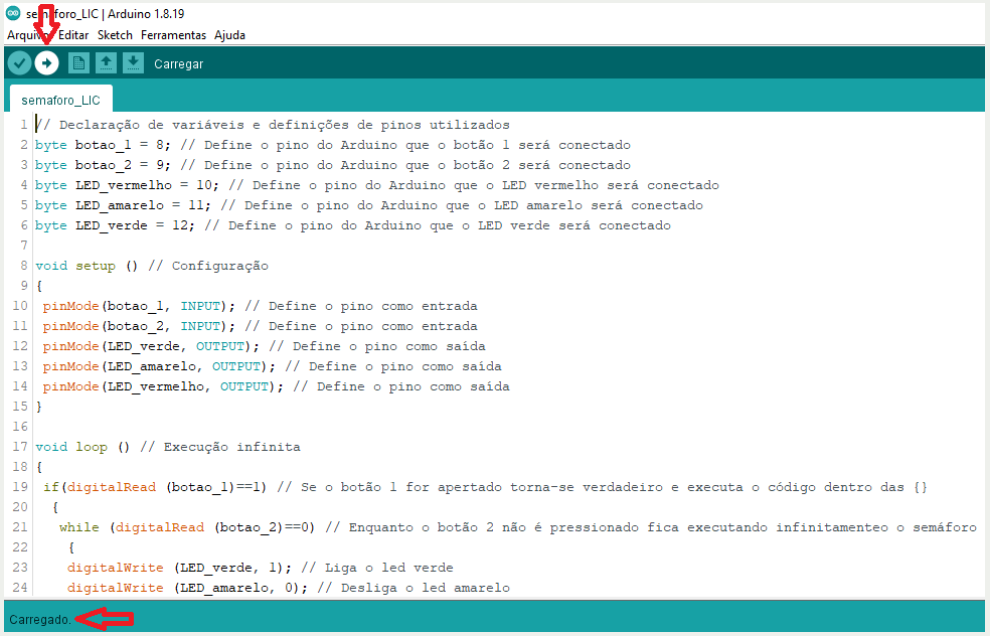


Figura 13 – Procedimento para enviar o código para o microcontrolador da placa Arduino.
Fonte: Próprio autor (2022).

Assim, com a placa conectada ao notebook/computador via cabo USB e com o código já carregado, pode-se testar o semáforo, que deve funcionar de acordo com o que foi proposto.

Respostas do questionário para os alunos:

1) Ao apertar o botão 1, o que ocorreu?

O semáforo para carros funcionou corretamente, repetindo o ciclo (verde, amarelo e vermelho).

2) Preencha o quadro a seguir com o tempo que cada LED acendeu:

Pergunta	Tempo
LED verde	3s
LED amarelo	1s
LED vermelho	3s

3) Ao pressionar o botão 2 até o final do ciclo, o que ocorreu?

Todos os LED's apagaram.

4) Qual a importância da definição correta dos pinos de entrada/saída entre o *hardware* e o *software*?

Caso os pinos não coincidam entre o *hardware* e *software*, o circuito não funcionará.

5) Qual a relevância da programação para o funcionamento correto do circuito?

A programação é o que define como o circuito físico irá funcionar, ou seja, a partir das informações que chegam para a placa Arduino (microcontrolador), ele processa o que será feito.

6) Desafio: Aumente o tempo de sinal aberto (LED verde) e diminua o tempo de sinal fechado (LED vermelho).

3.1 O circuito Eletrônico (Etapa 1)

O Quadro 2 lista os problemas que podem ocorrer durante a etapa de teste/ montagem do circuito eletrônico e as possíveis soluções para corrigir o problema.

Problema	Possíveis soluções
LED não acende	- Verificar a polaridade; - Verificar a posição no protoboard; - Verificar se utilizou o mesmo pino da placa Arduino fisicamente e na programação; - Verificar se declarou adequadamente o pino como saída na programação; - Verificar se utilizou o resistor de valor adequado; - Verificar se o jumper está quebrado; - Verificar se está queimado; - Verificar se a trilha do protoboard está estragada; - Verificar escrita/lógica da programação.
Botão não funciona	- Verificar a posição no protoboard; - Verificar se utilizou o mesmo pino da placa Arduino fisicamente e na programação; - Verificar se declarou adequadamente o pino como entrada na programação; - Verificar se utilizou o resistor de valor adequado; - Verificar se o jumper está quebrado; - Verificar se está estragado; - Verificar se a trilha do protoboard está estragada; - Verificar a escrita/lógica da programação.

Quadro 2 – Solução de problemas (Etapa 1).
Fonte: Próprio autor (2022).

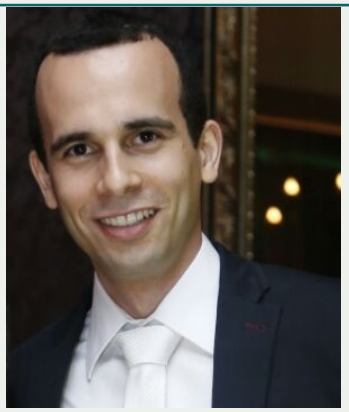
3.2 A programação (Etapa 2)

O Quadro 3 lista os problemas que podem ocorrer durante a etapa de programação/teste e as possíveis soluções para corrigir o problema.

Problema	Possíveis soluções
Arduino não recebe o código	- Verificar se o cabo USB está bem encaixado na placa Arduino e no notebook/computador; - Verificar se foi selecionado a Porta COM adequadamente; - Verificar a escrita da programação; - Verificar se tem alguma ligação errada no protoboard fechando curto-circuito na placa Arduino; - Verificar se a placa Arduino está queimada.

Quadro 3 – Solução de problemas (Etapa 2).
Fonte: Próprio autor (2022).

Currículo do autor



É Professor efetivo da área de Controle e Processos Industriais do Instituto Federal de Minas Gerais (IFMG) - Campus Sabará, desde 2016. Atua principalmente nos cursos: Engenharia de Controle e Automação e Técnico em Eletrônica. Já atuou como Professor/Técnico Trainee no SENAI MG (2009 a 2016) e como Coordenador do curso Técnico em Eletrônica no IFMG - Campus Sabará (2017 a 2020). Possui mestrado em Educação Tecnológica pelo CEFET-MG (2021); pós-graduação lato sensu em Docência na Educação Profissional e Tecnológica pelo SENAI CETIQT (2015), em Engenharia de Segurança do Trabalho pelo IFMG - Campus Governador Valadares (2016) e em Engenharia Elétrica pela Universidade Cândido Mendes (2015). É graduado em Engenharia de Produção pelo IFMG - Campus Governador Valadares (2014) e é Técnico em Eletroeletrônica (2009) e Eletrotécnica (2015) pelo SENAI MG.

Currículo Lattes: http://lattes.cnpq.br/4000149005069235

Feito por (professor-autor)	Data	Revisão de *layout*	Data	Versão
Erick Fonseca Boaventura	17/10/2022	Desginado pela proex	xx/xx/xxxx	1.0

Instituto Federal de Minas Gerais
NIT – Núcleo de Inovação Tecnológica