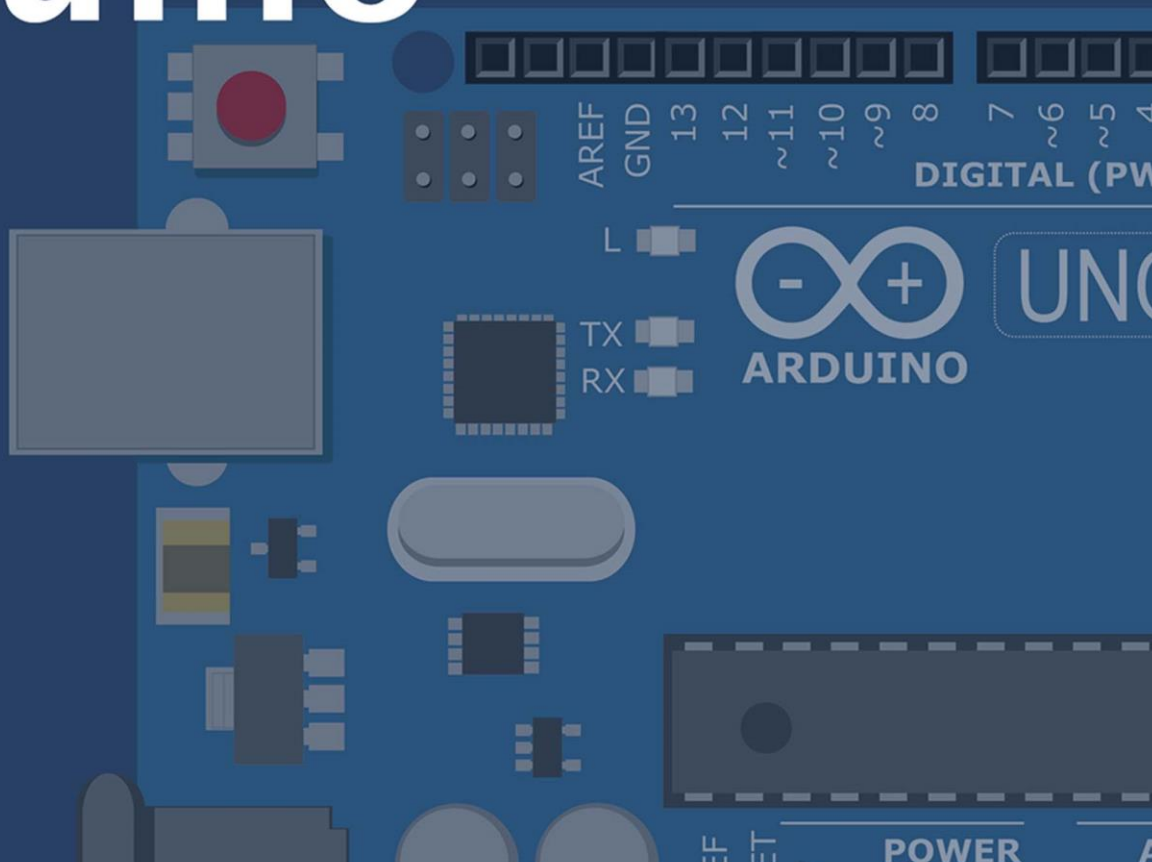
Por Luan Silver

Guia Rápido Para Aprender A Programar Arduino



Um pouco mais sobre meu trabalho

Blog sobre Robótica e afins:

GuiaRobotica.com

Perfil No Instagram:

Instagram/Guiarobotica

Canal No Youtube:

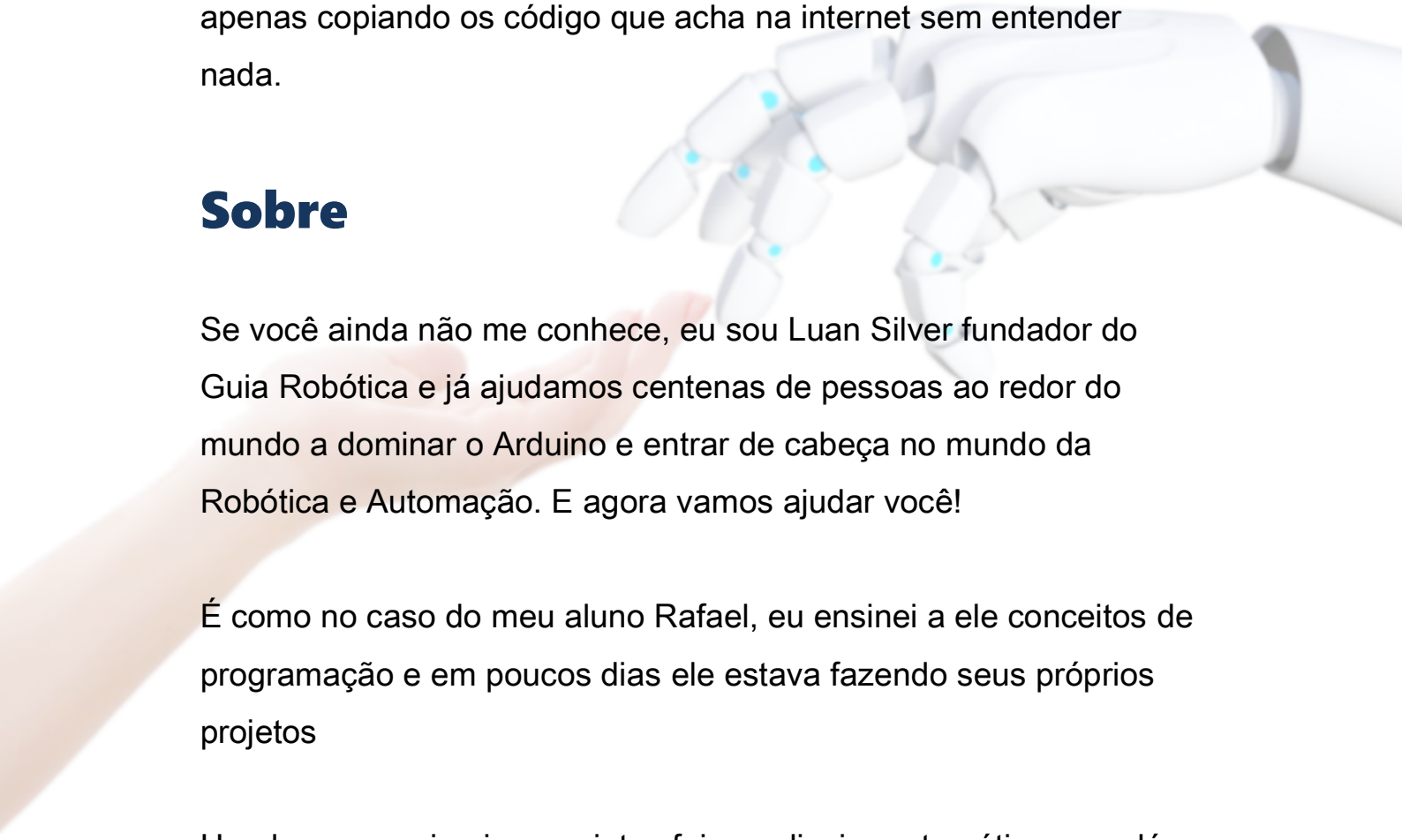
Guia Robótica

Página no Facebook:

Facebook.com/GuiaRobotica

Sumário

Nesse eBook eu vou te ensinar a dar seus primeiros passos rumo ao domínio da programação de Arduino, você vai parar de ficar apenas copiando os código que acha na internet sem entender nada.



Sobre

Se você ainda não me conhece, eu sou Luan Silver fundador do Guia Robótica e já ajudamos centenas de pessoas ao redor do mundo a dominar o Arduino e entrar de cabeça no mundo da Robótica e Automação. E agora vamos ajudar você!

É como no caso do meu aluno Rafael, eu ensinei a ele conceitos de programação e em poucos dias ele estava fazendo seus próprios projetos

Um dos seus primeiros projetos foi uma lixeira automática que além de abrir e fechar quando alguém se aproxima, ela também aciona um LED vermelho indicando que a lixeira está cheia.

Outro caso é o do Lucas, dei mentoria para ele e a pouco tempo ele terminou a construção de uma impressora 3D caseira, e agora está focando em modelagem 3D.

Alguns dos maiores erros dos iniciantes

Alguns dos problemas que a maioria dos iniciantes passa, é o fato de não seguirem uma linha lógica de aprendizagem, e SIM, eu também já passei por isso.

Uma boa lógica de aprendizagem, seria começar com conceitos teóricos e projetos práticos básicos, para além de aprender a teoria, ver funcionando ao mesmo tempo e aos poucos ir conhecendo os sensores, Shields e componentes eletrônicos separadamente, para somente depois utilizar em conjunto em seus projetos.

Mas o que acontece é que a maioria simplesmente procura projetos prontos na internet e apenas replicam sem aprender quase nada, isso quando funciona.

Em muitos casos quando não funcionam não sabem nem resolver, pois não aprendeu, apenas replicou.

O que estou falando aqui é sobre te mostrar um caminho rápido para já iniciar na programação de Arduino e alcançar seus objetivos.

Sabe, quando comecei eu pensava que teria que ser um verdadeiro expert em tecnologia, mas errei feio. Para aprender sobre Arduino, robótica e automação é muito mais fácil, basta seguir uma lógica de aprendizagem.

Inclusive, é até engraçada a historia de como eu comecei a desenvolver essa linha de aprendizagem para alunos. Deixa eu te contar para você entender...

A história por trás do que vou te ensinar

Eu sempre fui apaixonado por tecnologia desde criança, desmontava eletrônicos e computadores. O primeiro computador que eu desmontei tinha 256MB de memoria RAM e o famoso Windows XP. Eu tinha 14 anos e já tinha uma fama na família de querer desmontar as coisas mesmo sendo novas, e vou confessar para você, as vezes eu desmontava mesmo.

Tudo que eu achava legal eu queria ver por dentro, entender o funcionamento, mas teve um acontecimento decisivo na minha infância que me inspirou de um jeito que mudaria minha vida inteira, consegue imaginar o que foi?

O filme do homem de ferro em 2008, depois que eu vi aquele filme com Robert Downey jr. aquela armadura vermelha e amarela... E ele interagindo e construindo Robôs, eu sabia que queria algo semelhante àquilo na minha vida e pesquisando muito descobri que o nome daquilo tudo era robótica, fiquei fascinado.

Tão fascinado que alguns anos depois comecei a cursar engenharia de controle e automação, e nas minhas pesquisas me deparei com uma plaquinha chamada Arduino que prometia facilitar minha aprendizagem de robótica e me ajudar a fazer meus próprios projetos.

Então comecei a estudar sua programação, componentes, sensores, eletrônica, tudo que eu encontrava.

Eu encontrava projetos aleatórios na internet, explicações soltas, e estava muito difícil seguir em frente, então desisti naquele momento.

Mas uma semanada depois o professor montou com a turma um projetinho com um sensor LDR e achei incrível, quando ele apagava as luzes e ficava completamente escuro automaticamente o sensor detectava essa variação de luminosidade e acendia uma lâmpada.

Então chegando em casa naquela noite volto para minha pesquisa e não parei mais, até consegui fazer alguns projetinhos baseado nos que eu achava.

E cheguei em um momento que eu tive que criar algo do absoluto zero, eu pensei “eu sou capaz né... Já fiz alguns, não deve ser tão difícil assim”.

Então comecei a programar, eu tentava e tentava e não conseguia, aparecia sempre algum erro, quando eu ia ver era apenas um “;”

que estava faltando, ou uma letra maiúscula, um jumper que não funcionava, inúmeros erros que nem sempre eu descobria o que era.

Eu via as pessoas montando drones, plantas de fabricas, robôs, casas automatizadas, já eu nem conseguia tirar meus projetos do papel.

Mas depois me dei conta que até aquele momentos todos meus projetos tinham dado certo, porque eu não estava criando apenas estava replicando o que eu tinha visto, o famoso “CTRL+ C” “CTRL + v” tanto na programação como como na ligação dos componentes.

Então antes de montar meus projetos eu devia entender tudo que estava fazendo, para que servia cada comando, linha de código, elemento, shield, cada pequeno componente eletrônico.

Depois de entender todos esses conceitos, programar e montar projetos ficaram bem mais fáceis.

Eu descobri que **replicar não é aprender**, que a linguagem de programação é menos importante que lógica de programação, que temos que acostumar a digitar os códigos e não apenas sair copiando tudo sem entender nada.
Hoje ensino o máximo de pessoas que eu consigo com meu projeto, Guia Robótica.
Então é isso **vamos aos estudos**!

O que é um ALGORITMO

O algoritmo é o item principal que você deve entender para programar em qualquer tipo de linguagem de programação, ela resume-se em desenvolver sequências de passos para atingir um determinado objetivo.

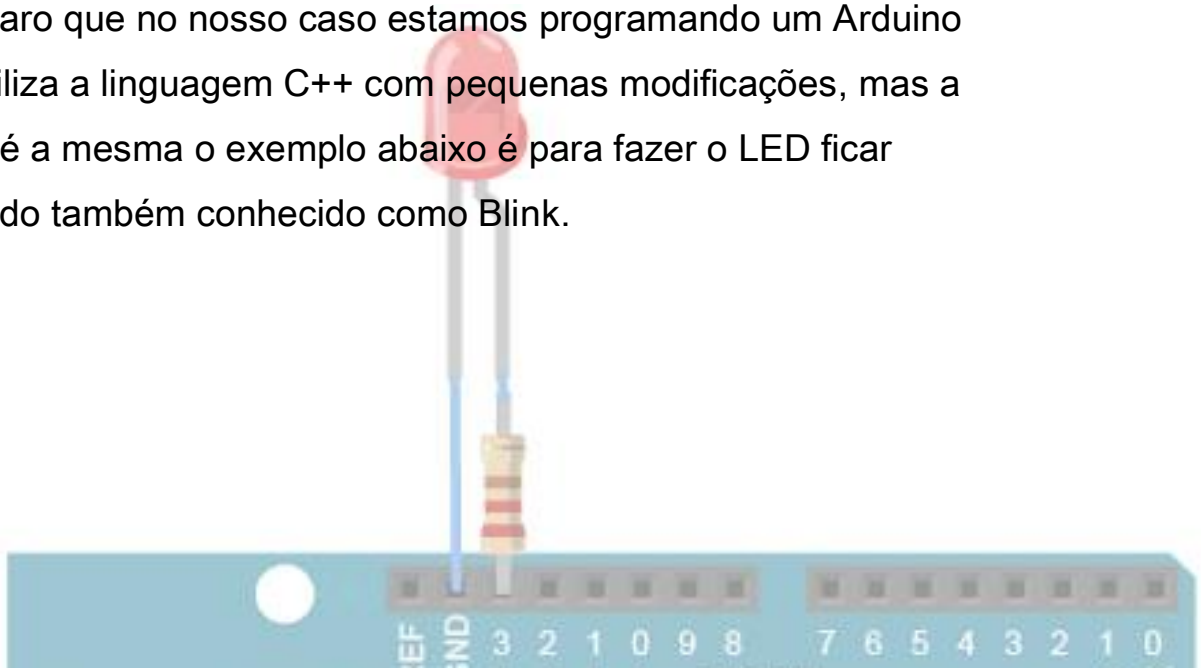
Você utiliza algoritmos o tempo todo e talvez não percebeu, vou te mostrar um que provavelmente já viu.

1. Bata no liquidificador todos os ingredientes
2. Acrescentando a farinha aos poucos.
3. Depois unte e enfarinhe uma forma e despeje a massa nela.
4. Asse em forno médio por cerca de 40 minutos.
5. Tire do forno
6. Espere amornar e desenforme.

Um exemplo de algoritmo do dia a dia uma receita de bolo.

Mas claro que no nosso caso estamos programando um Arduino que utiliza a linguagem C++ com pequenas modificações, mas a logica é a mesma o exemplo abaixo é para fazer o LED ficar piscando também conhecido como Blink.

.



```
1  int ledPin = 13;
2  void setup()
3  {
4    pinMode(ledPin, OUTPUT);
5  }
6  void loop()
7  {
8    digitalWrite(ledPin, HIGH);
9    delay(1000);
10   digitalWrite(ledPin, LOW);
11   delay(1000);
12 }
```

Explicação de cada comando.

```
int ledPin = 13;
```

Declara uma variável de números inteiro chamada ledPin que é atribuído o valor 13 referente a porta do arduino.

```
void setup()
```

Esta função é executada somente uma vez quando o arduino é ligado ou reiniciado.

```
pinMode(ledPin, OUTPUT);
```

Diz que a variável ledPin é uma saída de sinal, OUTPUT (saída).

```
void loop()
```

Função que executa infinitas vezes enquanto o arduino estiver ligado.

```
digitalWrite(ledPin, HIGH);
```

Manda um sinal alto (1) para a variável ledPin ligando o led, HIGH (alto)

```
delay(1000);
```

Espera 1000 milissegundos equivalentes a 1 segundo

```
digitalWrite(ledPin, LOW);
```

Manda um sinal baixo (0) para a variável pedPin desligando o led, LOW (baixo)

```
delay(1000);
```

Espera 1000 milissegundos equivalentes a 1 segundo

Variáveis e seus tipos dados

As variáveis servem para guardar informação no arduino, para cada tipo de informação temos um tipo de variável diferente que se encaixa melhor, vamos listar alguns desses tipos:

char: São utilizados para armazenar caracteres e ocupam um byte.

byte: Podem armazenar um número entre 0 e 255.

int: Ocupam 2 bytes (16 bits) e armazena números entre -32,768 e 32,767.

unsigned int: Também ocupam 2 bytes, mas como não possuem sinal armazena números entre 0 e 65,535.

long: ocupa 32 bits (4 bytes), armazena números de -2,147,483,648 até 2,147,483,647.

unsigned long: Ocupa também 32 bits (4 bytes) como não possuem sinal armazena números de 0 a 4.294.967.296.

float: Números decimais que ocupam 32 bits (4 bytes). Podem tomar valores entre -3.4028235E+38 e +3.4028235E+38.

double: Também armazena números decimais, mas possuem 8-bytes (64 bit).

string: Sequência de caracteres.

Exemplos

```
1 int ledPin = 13;
2 long Microsegundos = 0;
```

1. Foi criada uma variável do tipo inteiro podendo ser armazenado valores entre -32,768 e 32,767 cujo nome é ledPin e o valor atribuído foi 13.

2. Criação de variável do tipo long podendo ser armazenado valores entre -2,147,483,648 até 2,147,483,647 cujo nome é Microsegundos e o valor atribuído foi 0;

Atribuição

Atribuir um valor a uma variável significa armazenar na mesma alguma informação para uso posterior, no arduino é utilizado o símbolo de `=` para a atribuição.

Exemplos

```
int ledPin = 13;
long Microsegundos = 0;
float DistanciaemCM = 0;
```

Nem sempre quer dizer que o valor atribuído é a porta que está conectada no arduino, a variável referente à porta deve ser declarada no setup como entrada ou saída de dados.

Operadores

Um operador é um conjunto de um ou mais caracteres que serve para operar sobre uma ou mais variáveis, vamos ver alguns exemplos para ficar mais fácil de entender.

Exemplos

```
x = 7-2;
y = 7+5;
h = 5*5;
sensor = h/x;
```

1. O valor atribuído em x será de 5
2. O valor atribuído em y será de 12;
3. O valor atribuído em h será de 25

Também podemos usar operações baseadas em variáveis no caso a variável **sensor** receberá o valor de h/x que é igual a 5.

Os operadores mais utilizados

Operadores aritméticos

```
+ adição
- subtração
* multiplicação
/ divisão
```

Operadores lógicos

```
&& conjunção ("e")
|| disjunção ("ou")
== igualdade ("igual a")
!= desigualdade ("diferente de")
! negação ("não")
> "maior que"
< "menor que"
>= "maior ou igual a"
<= "menor ou igual a"
```

Comentário

Os comentários são trechos de texto que servem para explicação ou desabilitar parte da programação, em um programa com varias linhas de código às vezes podemos nos confundir em determinados momentos daí a necessidade de fazer alguns comentários para não nos confundirmos.

No arduino podemos utilizar 2 tipos de comentários:

- Comentário de linha: Utiliza // e torna toda a linha um comentário.
- Comentário de bloco: Utiliza /* e termina com os caracteres */ Tudo que estiver dentro destes caracteres será um comentário independente da quantidade de linhas.

Entenda melhor com o exemplo abaixo.

Código comentado

```
/*
Guia róbotica
Programa: Blink
*/

/*
Declara uma variável de números inteiro chamada ledPin
e é atribuido o valor 13.
*/
int ledPin = 13;

/*
Declaração da função setup()
Esta função é executada somente uma vez quando o arduino é ligado
*/
void setup() {

  pinMode(ledPin, OUTPUT); // Diz que a variável ledPin é uma saída de sinal
}

/*
Declaração da função loop()
Função que executa infinitas vezes enquanto o arduino estiver ligado
*/
void loop() {

  digitalWrite(ledPin, HIGH); // Manda um sinal alto (1) para ledPin ligando o led
  delay(1000);              // Espera 1000 milissegundos equivalentes a 1 segundo
  digitalWrite(ledPin, LOW);  // Manda um sinal baixo (0) para pedPin desligando o led
  delay(1000);              // Espera 1000 milissegundos equivalentes a 1 segundo

}
```

Estruturas de controle

Estrutura de controle (ou fluxo de controle) é um bloco de programação que analisa variáveis e escolhe uma direção para seguir baseado nos parâmetros pré-definidos.

While

O loop `while` (enquanto) irá se repetir continuamente, e infinitamente, até a expressão dentro dos parênteses ( ), se torne falsa.

Algo deve mudar a variável testada, ou o loop nunca irá encerrar. Isso pode ser no seu código, por exemplo, uma variável incrementada, ou uma condição externa, como a leitura de um sensor.

Sintaxe

```
while (condição) {
  // código a ser executado repetidamente
}
```

Exemplo

```
var = 0;
while (var < 100) {
  // faz algo repetitivo 100 vezes
  var++;
}
```

O que acontece neste código é que o comando while vai ser executado enquanto a variável **var** for menor que 100, no final do código temos **var++** que significa **var= var+1**, esta sendo utilizada como contador, quando ela atingir o valor de 100 esta condição se torna falso assim pulando para a próxima linha de código fora da estrutura while.

For

O comando **for** (para) é usado para repetir um bloco, ele funciona semelhante ao while necessitando de um contador e de um incremento que é utilizado para romper o loop.

O comando **for** é utilizado para qualquer operação repetitiva, e é usado frequentemente com vetores para operar em coleções de dados ou pinos.

Sintaxe

```
for (inicialização; condição; incremento) {
  //comando(s);
}
```

A inicialização ocorre antes e uma única vez. A cada repetição do loop, a condição é testada; se for verdadeira, o bloco de comandos e o incremento são executados. Quando a condição se torna falsa, o loop termina.

Exemplo

```
for(int i = 0; i < 3; i++) {
   digitalWrite(led, HIGH);
   delay(1000);
   digitalWrite(led, LOW);
   delay(1000);
 }
```

Para o comando **for** começamos criando uma variável e atribuindo um valor de inicio como no exemplo int i=0, a condição é o loop vai acontecer até o contador **i** for menor que 3 e o incremento é **i++** que é o a mesma coisa que **i = i+1**.
O led irá acender e apagar 3 vezes.

If

O comando **if** (se) checa uma condição e executas o comando a seguir ou um bloco de comandos delimitados por chaves, se a condição for verdadeira.

Pode-se usar vários comandos **if** um em sequência.

Exemplo

```
if (Bluetooth = "A") {
  digitalWrite(ledA, HIGH);
}
if (Bluetooth = "B") {
  digitalWrite(ledB, HIGH);
}

if (Bluetooth = "C") {
  digitalWrite(ledC, HIGH);
}
```

Temos aqui um exemplo de código com conexão bluetooth que envia caracteres para o arduino, no caso o arduino recebendo o caractere A irá ligar o ledA, caractere B liga o ledB e o caractere C liga o ledC.

If-Else

Agora vamos adicionar o comando **else** (senão) como complemento, ele checa se a condição if é verdadeira como já vimos no exemplo anterior e se ela for falsa executa os comandos dentro da condição else.

sintaxe

```
if (condição) {
  // faz coisa A
}
else {
  // faz coisa B
}
```

Exemplo:

```
if (temperatura >= 70) {
  digitalWrite(luzvermelha, HIGH);
}

else {
  digitalWrite(luzverde, HIGH);
}
```

Neste exemplo temos um sensor de temperatura, se a temperatura for maior que 70 °C ligamos uma luz vermelha indicando perigo, se a temperatura não for maior que 70 °C ligamos uma luz verde.

Conclusão

Algoritmo

Resume-se em desenvolver sequências de passos para atingir um determinado objetivo.

Variáveis e seus tipos de dados

As variáveis servem para guardar informação no arduino, para cada tipo de informação temos um tipo de variável diferente que se encaixa melhor.

Atribuição

Atribuir um valor a uma variável significa armazenar na mesma alguma informação para uso posterior.

Operadores

Um operador é um conjunto de um ou mais caracteres que serve para operar sobre uma ou mais variáveis, podendo ser operadores aritméticos ou lógicos.

Comentários

Os comentários são trechos de texto que servem para explicação ou desabilitar parte da programação

Estrutura de controle

While

O loop while (enquanto) irá se repetir continuamente, e infinitamente, até a expressão dentro dos parênteses ( ) se torne falsa.

For

O comando for (para) e usado para repetir um bloco, ele funciona semelhante ao while necessitando de um contador e de um incremento que é utilizado para romper o loop.

If

O comando if (se) checa uma condição e executa o comando a seguir ou um bloco de comandos delimitados por chaves, se a condição for verdadeira.

If-else

Primeiro testa a condição if, se ela for falsa executa os comandos dentro do bloco else (senão).

Mais uma coisa...

Entretanto, Por mais que você aplique tudo isso que aprendeu, você não terá tanto **resultado** como se tivesse adquirido um eBook mais completo sobre programação para se **aprofundar em seus**

estudos, de modo rápido e efetivo, pois seguir uma linha de aprendizagem muda tudo na hora de estudar.

E com isso, também posso te ajudar...
Sempre indico um **eBook incrível** de Arduino de um amigo meu para quem tem problemas assim como os seus...

Este amigo é o Mateus Dias, que já ajudou centenas de pessoas ao redor do mundo a **dominar o Arduino**. E você será a próxima!

Depois de anos de estudo e mais de 140 mil reais investido em conhecimento, ele compactou tudo que sabe sobre Arduino e colocou em um eBook que você está tendo a chance de adquirir.

Dentro dele você vai aprender:

- Como funcionam os ciclos básicos do Arduíno.
- Criar comunicações e armazenar dados.
- Primeiros projetos, seguindo linha de aprendizagem
- Aprender a trabalhar com bibliotecas e muito mais...

O melhor de tudo é que você **não precisa se preocupar com a implementação**, o Mateus explica tudo em um passo a passo, então mesmo que você seja completamente iniciante, vai conseguir aplicar o que ele ensina lá dentro sem nenhuma dificuldade.

Lembrando que nós preocupamos completamente com a sua satisfação e caso você se arrependa de ter adquirido o eBook, tem **7 dias de garantia absoluta de satisfação**.

Ou seja, basta enviar um e-mail que recebe 100% do seu dinheiro de volta!

Esses eBooks são **sensacionais** porque vão te **poupar milhares de horas** de suor desnecessário e milhares de reais que não irá precisar investir igual o Mateus investiu.

Já pensou se você fosse precisar investir tão alto como ele precisou?

Não precisa!

Basta pagar 5 x de R$ 9,98 e acessar imediatamente todos os **segredos e técnicas mais avançadas** que ele já aprendeu até hoje sobre o Arduino.

Se você parar para pensar, esse é o valor que você gasta comendo um hambúrguer...

Eai, ta afim de deixar um hambúrguer de lado e começar a investir em algo que pode realmente mudar sua vida?!