Apostila didática: Edubot

Nícolas Rocha[‡] - nicolas.rocha@aluno.unb.br,
Roberto Baptista[§] - robertobaptista@unb.br,
João França[*] - 241009815@aluno.unb.br,
Miguel Scardini[*] - 241009851@aluno.unb.br,
Renato Ferreira[*] - 2401009860@aluno.unb.br,
[*] Engenharia Mecatrônica - Faculdade de Tecnologia, Universidade de Brasília
[‡] Engenharia Aeroespacial - Faculdade de Ciências e Tecnologias em Engenharia, Universidade de Brasília
[§] Professor Doutor - Faculdade de Ciências e Tecnologias em Engenharia, Universidade de Brasília

***Resumo*—Essa apostila foi criada pela equipe do Edubot para te ajudar a desvendar o *Sparki*, esse robozinho simpático muito bem equipado, produzido pela Arcbotics. E, com isso, mostrar que programar todos os sensores, motores, tela, luzes e acessórios que ele possui pode ser bem simples e divertido, e não um monstro de sete cabeças como a maioria das pessoas pensa.**

I. Quem somos nós?

Acreditando na diferença que a programação e a robótica podem fazer na trajetória escolar e profissional de uma pessoa, o Edubot é um Projeto de Extensão de Ação Contínua da Universidade de Brasília (UnB) e apoiado pelo capítulo estudantil de Robótica e Automação, da sigla em inglês 'RAS' (Robotics and Automation Society), do ramo estudantil da IEEE (Institute of Electrical and Eletronic Engineers) na UnB, que tem como principal objetivo o incentivo à formação de jovens nas áreas de Tecnologia e Engenharia através da realização de oficinas de robótica como atividades extracurriculares em escolas e instituições públicas de nível médio do Distrito Federal.

Para saber mais, acesse nosso Instagram <@projetoedubot> e nosso website <sites.google.com/view/edubotunb>.

II. Introdução

O Sparki vem pronto, com sensor de distância e de luz, acelerador, tela LCD, LED RGB, campainha, controle remoto, dois motores e a mesma IDE do Arduino. Nessa apostila, vamos aprender a programar tudo isso, além de poder ver na prática alguns assuntos que vemos em Matemática e Física durante o Ensino Médio.

Além disso, aprender a programar pode ser bem útil na sua vida pessoal, dentre as vantagens, cito aqui algumas: o pensamento computacional traz uma grande metodologia para solucionar problemas, partindo de questões menores para depois resolver o todo, esse mesmo pensamento computacional pode melhorar até sua organização pessoal, além de desenvolver seu raciocínio lógico, aumentando a clareza, a rapidez e a fluidez de seus pensamentos. Ah! Também será ótimo para aprender um pouco de inglês.

E aí?! Pronto pra começar?

III. Afinal, o que é um robô?

O desejo de solucionar problemas do dia-a-dia de maneira mais eficiente e simples, foi - e ainda é - um dos maiores promotores para o desenvolvimento tecnológico das mais diversas sociedades. De acordo com a necessidade de cada período, cientistas (engenheiros, matemáticos, físicos, etc) de diferentes povos propunham maquinários e ferramentas que pudessem operar em conjunto com outros seres vivos para **modificar** a forma de trabalho, melhorando a qualidade de vida dos trabalhadores e das suas famílias. Com isso, impulsionou-se a busca por maiores níveis de **automação**, consequentemente levando às tentativas de aplicar esse conceito no cotidiano, sendo esse o patamar que alcançamos e presenciamos nos dias atuais.

É nesse contexto que a ideia da utilização de robôs para a realização de diversas tarefas acabou se popularizando, colocar uma máquina para completar uma tarefa muito trabalhosa ou difícil e assim poupar trabalho humano é um dos focos da robótica. O robozinho utilizado como material de estudo no nosso curso é o *Sparki*, ele é uma dessas máquinas que tem como objetivo faciliar a nossa vida, nesse caso o ensino de programação, mas o que difere ele e outros robôs de um ar condicionado ou um projetor?

A. Para que estudar a definição de robô?

Como descrito anteriormente, a robótica está presente em diversos aspectos do nosso contidiano. Dentre as mais diversas aplicações existentes, podemos listar algumas:

- Culinárias: robôs que auxiliam na preparação de pratos e na entrega dos mesmos em restaurantes (segue-linha);
- Robôs industriais: como carregados (segue-linha), robôs montadores de peças em indústrias;
- Robôs cirurgicos: como o *Star* (tecidos moles mais delicados), o *PRECEYES* (áreas delicadas, como olhos), o *CorPath* (operações cirúrgicas a distância através do WiFi), *The Monarch Platform* (broncospia, i.e. inteverção cirúrgica nos brônquis), *Mako Rio* (auxiliar em implante em joelhos e costela), *Versius* (cirurgias não invasivas);
- Robôs para comunicação com crianças;
- Robôs para exploração espacial, como o *Rover*, em marte;
- Robô para inspecionar tubulações;

- Robô para serviços de casa (Seja de superfícies aquáticas, ou terrestres). Exemplos: *Row-bot*, *Ro-Boat*, *Roomba*.

B. Definindo um robô

Para uma máquina ser considerada um robô, a primeira coisa que ela deve ser capaz de fazer é raciocinar de alguma maneira.

Como assim, um robô tem que ser capaz de pensar assim como nós (pessoas)? Não necessariamente, algo assim é muito complexo e difícil de acontecer... Quando dizemos que um robô deve raciocinar, queremos dizer que ele deve ter a capacidade de analisar o mundo ao seu redor e tomar decisões a partir do que ele analisou, ele precisa agir em função do seu ambiente. Um robô segue linha, por exemplo, necessita de algum sensor que permita a ele encontrar a localização da linha e a partir dai decidir como irá se movimentar sobre ela e o que caso perca a linha de vista. De maneira geral, a ação realizada pelos robôs normalmente é algum tipo de movimento, alguma atividade mecânica como andar ou mover algo, um semáforo que acende a luz vermelha para carros para possibilitar a passagem de pedestres após ter seu botão apertado não é considerado um robô, uma porta automática se aproximaria mais do que é um robô.

Qual a relevância deste assunto no mundo? Afinal, para que mesmo eu estou lendo esta apostila? Eu sei que eu vou ganhar um diploma do curso, mas... E aí? O que mais que isso aqui pode me acrescentar?

Nosso curso busca ajudar os alunos a entender de forma simples como que um robô, um computador ou outros equipamentos raciocinam, como que nós pessoas somos capazes de criar linhas de pensamento para coisas não pensantes. Nós mostraremos que a programação não é um bicho de sete cabeças e fornecemos a base necessária para que vocês cheguem mais preparados e entusiasmados em algum curso superior, técnico ou trabalho que aborde programação.

Definição

Um robô é uma máquina **autônoma**, que existe no **mundo físico**, possui **sensores** ara perceber o ambiente e consegue agir sobre o meio para alcançar um ou mais objetos definidos.

Nós já falamos sobre os sensores e conseguir agir sobre o meio, mas e os outros requisitos? Existir no mundo físico nada mais é que algo que somos capazes de tocar ou manipular, uma bola de futebol, por exemplo, existe no mundo físico, os sonhos que temos ao dormir não existem no mundo físico, pois não somos capazes de tocá-lo. Uma máquina autônoma ou automática, é uma máquina que após ligada ela funciona sozinha sem a constante necessidade de interferência externa, ela ainda pode aceitar o aperto de botões ou o recebimento de outros tipos de comandos, mas a ideia é que ela seja capaz de realizar tarefas sozinha quando for necessário.

IV. O QUE É UM ALGORITMO?

Diariamente utilizamos algoritmos sem mesmo perceber. Quando cozinhamos uma macarrão, assamos um bolo, montamos um móvel, um brinquedo ou mesmo quando escovamos os dentes estamos utilizando este conceito para completar essas tarefas.

Isto porque tomamos essas decisões nos baseando em instruções claras para chegar a um objetivo. Então, se estamos preparando um bolo, não podemos simplesmente colocar os ingredientes no forno sem antes misturá-los conforme a receita. Não teríamos um bolo, mas sim um Frankestein! Da mesma forma, não poderíamos esperar que ele fique pronto ao ligarmos o forno, mas não colocássemos a massa do bolo nele.

Definição

Algoritmo é um conjunto finito de instruções sequenciais lógicas, bem definidas, não ambíguas que levam à solução de um problema.

Mas afinal, o que esse bando de palavras complicadas querem dizer? Eu não entendi foi nada.

O que elas querem dizer é que, em outras palavras, um algoritmo indica um conjunto de instruções para realizar uma tarefa qualquer como, por exemplo, fazer pipoca. Seguimos uma receita que nos diz bem direitinho o que fazer para atingir o objetivo desejado, que no caso é aquela pipoca bem quentinha e crocante no final das contas! Assim ficou mais fácil de entender, né?

V. LINGUAGEM DE PROGRAMAÇÃO

Se fosse dado o comando abaixo para o *Sparki* você acha que ele entenderia?

"Sparki, vai pra frente! Sparki, vai pra trás!"

Quem pensou que ele não entenderia, acertou! O *Sparki* não iria entender nada do que foi dito.

Então como ele entende os comandos que pedimos pra fazer?

Para nos comunicarmos com o *Sparki*, é preciso passar os comandos para o computador através de uma linguagem de programação, que irá converter o que escrevemos para 0's e 1's e depois enviará para *Sparki*, e assim ele será capaz de entender e executar o comando.

Definição

Uma linguagem de programação é um método padronizado para comunicar instruções para um computador.

De maneira mais simples, linguagem de programação é a língua que usamos para nos comunicar com o computador assim como a língua portuguesa é a língua que usamos para nos comunicar entre nós. Assim como existem várias línguas como inglês, alemão e japonês, existem várias linguagens de programação, cada uma com sua peculiaridades, mas aqui no nosso curso iremos focar a linguagem que o *Sparki* usa.

Como mencionado anteriormente, o que escrevemos para o *Sparki* é antes transformado em 0's e 1's, isso porque os computadores utilização notação binaria. onde os números são representados apenas utilizando-se 0's e 1's. Podemos concluir então que, na verdade, os computadores não entendem a linguagem de programação!! Tudo o que escrevemos para

eles em linguagem de programação é depois transformado em 0's e 1's para que finalmente possamos nos comunicar.

VI. Funções importantes void setup() e void loop()

O comando `#include <Sparki.h>` e as funções `#void loop()` são as funções mais importantes na hora de começar a programar o *Sparki*, pois entender o que elas significam e como funcionam é essencial para saber onde escrever determinada parte do código em relação ao tipo de aplicação.

```
1  #include <Sparki.h>
2
3  void setup()
4  {
5  }
6
7  void loop()
8  {
9  }
```

- Biblioteca`sparki.h`: esta biblioteca guarda todas as funções relacionadas ao *Sparki*, e é necessário inserir ela na primeira linha do código, para que estas funções sejam habilitadas para a comunicação entre o computador e o robô funcionar.
- `void setup`: esta função é a primeira a rodar no código, e tudo que está dentro de suas chaves é executado apenas uma vez. A função é bastante útil para inserir configurações iniciais, como por exemplo limpar a tela LCD, ou fazer algum movimento inicial que não irá se repetir no resto do programa.
- `void loop()`: esta função é usada para colocar códigos que irão ficar se repetindo no programa, ou seja, sempre ficará rodando até que seja carregado algum outro programa no *Sparki*. Um exemplo para esta função é deixar o *Sparki* sempre com o LED de uma cor ligado, ou sempre se movimentando em uma determinada direção.

A. Exercícios

1) Em suas palavras, defina o que é um algoritmo e dê 2 exemplos não citados no capítulo.

2) O objetivo de uma linguagem de programação é:
a) Fornecer um conjunto de regras bem definidas que permita escrever programas de computador de forma mais amigável evitando ambiguidades.
b) Estabelecer regras para a comunicação entre seres humanos.
c) Dar a liberdade para cada programador escrever programas de computador a partir do seu próprio conjunto de regras, facilitando o processo de programação.
d) Definir um padrão para programadores utilizarem os comandos binários intrísecos a cada arquitetura de processador.

VII. Tela LCD (Liquid Crystal Display)

A. Introdução

O *Sparki* possui um tela LCD onde é possível desenhar, escrever e visualizar dados e sensores em funcionamento. Ao olhar bem de perto, pode-se perceber que a tela é formada por pequenos pontos que podem ser programados para estarem em um de dois estados, ligado ou desligado. Nós chamamos esses pontos de *pixel*. Eles também estão presentes em TVs, monitores de computador, celulares e eles são ligados ou desligados em uma disposição na qual seja possível formar uma imagem na tela. No caso do *Sparki*, ele possui 128 *pixels* na horizontal e 64 *pixels* na vertical, e para identificar cada pixel deve ser fornecido a coordenada horizontal, vertical e se ele deverá ficar aceso ou apagado.

B. Funções

Para imprimir algo na tela LCD, você deve seguir 4 passos importantíssimos:

- Limpar o LCD -> `sparki.clearLCD()`
- Informar o que você quer que apareça no LCD ->

```
1  sparki.drawLine(x, y, 127, 63);
2  sparki.drawRect(5, 5, 30, 10);
3  sparki.drawRectFilled(15, 17, 30,
10);
4  sparki.drawCircle(20, 45, 12);
5  sparki.drawFilled(15, 17, 30, 10);
6  sparki.print("não pula linha");
7  sparki.println("pula linha
depois");
```

- Atualizar a tela -> `sparki.updateLCD()`
- Criar um intervalo de tempo -> `delay()`

1) **Limpar o LCD**: O primeiro passo para mostrar algo no LCD é limpar todas as informações que estão armazenadas no *Sparki* e, para isso, utilizamos a função `sparki.clearLCD()`. Esta função não precisa de parâmetros, então não precisamos escrever nada dentro dos parâmetros.

Para não esquecer!

"Clear"traduzido para o português significa "limpar".

2) **Informar ao LCD**:

- **sparki.drawLine(xi, yi, xf, yf)**: Esta função cria uma linha de acordo com o ponto de início e do fim do traço. Então xi e yi são as coordenadas do ponto de início do traço.
- **sparki.drawRect(xi, yi, xf, yf)**: Esta função desenha um retângulo, sendo xi e yi coordenadas do canto superior esquerdo do retângulo e xf e yf coordenadas do canto inferior direito do retângulo.
- **sparki.drawRectFilled(xi, yi, xf, yf)**: Esta função é semelhante à `sparki.drawRect(xi, yi,xf, yf)`, a única diferença é que ele desenha um círculo preenchido.

- **sparki.drawCircle(xc, yc, raio)**: Esta função desenha um círculo de acordo com as coordenadas xc e yc indicam onde deve ficar o centro do círculo e o parâmetro do `raio` indica o tamanho do raio.
- **sparki.drawCircleFilled(xc, yc, raio)**: Esta função é semelhante à `sparki.drawCircle(xc, tc, raio)`, a única diferença é que ele desenha um círculo preenchido.
- **sparki.print()**: Com esta função, é possível imprimir caracteres e números na tela, basta escrever a mensagem a ser impressa dentro dos parênteses e das aspas. Essa função também é bastante utilizada para imprimir valores recebidos dos sensores, mas, para essa utilidade, o parâmetro deverá ser uma variável, e o nome não será escrito entre aspas.
- **sparki.println**: Assim como a função anterior, ela imprime uma mensagem na tela. A diferença é que essa função gera uma quebra de linha ao fim dessa mensagem.

3) **Atualizar a tela LCD**: a função `sparki.updateLCD()` é importante para atualizar a tela, informando ao Sparki que ele já pode mostrar no LCD as informações que foram passadas.
4) **Criar um intervalo de tempo**: para criar um intervalo de tempo, devemos utilizar a função `delay()`. Ela não precisa de parâmetros.
Não entendi o porquê de precisarmos criar um intervalo de tempo. Não faz sentido!

   O intervalo de tempo é necessário para que a mensagem/imagem permaneça na tela por tempo suficiente para que possamos visualizá-la. Caso criemos um código para o LCD sem incluir esse passo, não conseguiremos visualizar a tela. Ficou confuso? Basta pensar que estamos limpando e atualizando o LCD com novas informações o tempo todo, então, os *pixels* estão sendo ligados e desligados a cada vez que o `void loop()` é lido.

Para não esquecer!

"Draw"traduzido para o português significa "desenhar".
"Line"traduzido para o português significa "linha".
"Rect"é uma abreviação de "rectangle", que traduzido para o português significa "retângulo".
"Circle"traduzido para o português significa "círculo".
"Filled"significa "preenchido".
"Print"significa "imprimir".
"Update"traduzido para o português significa "atualizar".

VIII. VARIÁVEIS

Provavelmente você já ouviu o seu professor de matemática ou de física falando dessa tal de variável e que você também já usou muitas variáveis enquanto estudava, mas o que na verdade é essa tal de variável?

Existem várias maneiras de definir o que é uma variável, por exemplo, quando seu professor de matemática falou que ela era aquela letrinha 'x' que vinha acompanhada de uma função e que, de fato, varia com o valor de entrada dessa função. No nosso caso, vamos olhar para as variáveis de uma maneira um pouquinho diferente.

Pense em que um armário bem organizado, em que todas as roupas são separadas de acordo com o seu tipo, por exemplo, camisas em uma gaveta, calças e shorts em outras e assim por diante. Cada divisória do armário, ou seja, cada gaveta, deverá conter apenas um tipo de roupa específico para que o armário continue organizado.

Pensei! Mas o que isso tem a ver com o conceito de variáveis?

Tem tudo a ver! Assim como cada gaveta guarda um tipo específico de roupa, **cada variável guarda um tipo de específico de números ou caracteres**, dentre eles:números inteiros, números reais, letras,...

Definição

Variável é um local reservado na memória para armazenar um tipo de dado.

Ao escrever um código, não conheceremos o endereço onde a variável será armazenada. Dessa forma, para fazer referência à variável usamos o nome da mesma e o computador se encarrega do resto. Toda variável deve um nome e esse nome não pode ser iniciado com um número. Além de ter um nome, a variável também precisa ter um tipo. O tipo de dado de uma variável determina o que ela é capaz de armazenar.

A. Tipagem

Imagine que você acabou de se mudar para uma casa nova e agora você e sua família precisam tirar toda a mudança de dentro de várias caixas.

Pronto, imaginei! Mas ainda não entendi como coisas e variáveis podem ser parecidas.

Pode não parecer, mas caixas e variáveis são conceitos muito parecidos! Vamos lá, vou listar algumas características que as duas têm em comum. Ambas são usadas para:

- Guardar coisas;
- Separar itens;
- Organizar um ambiente.

Viu como são conceitos parecidos? A principal diferença é que usamos caixas no mundo físico e variáveis usamos no mundo virtual.

Agora que você conseguiu entender a relação entre caixas e variáveis, vamos voltar ao exemplo que pedi para você imaginar. Antes da mudança ter sido transportada para sua nova casa, as coisas da sua antiga casa foram guardadas dentro de caixas que foram nomeadas da seguinte forma: "quarto 1", "quarto 2", "quarto 3", "sala"e "cozinha".

Você e sua família agora precisam organizar a nova casa, e como toda mudança foi bem organizada, não deve ser uma tarefa muito difícil, né?! Se vocês quiserem organizar a cozinha, por exemplo, é só procurar as caixas que foram nomeadas como "cozinha".

Variáveis funcionam exatamente como as caixas da sua mudança. Na linguagem C, **para cada tipo de informação,**

um tipo de variável deve ser utilizada para armazená-la. Assim como em uma mudança não podemos guardar os utensílios da cozinha na mesma caixa em que guardamos nossas roupas, também não podemos guardar tipos diferentes de informações em uma mesma variável.

Na linguagem C existem:

- Variáveis do tipo **`int`** que guardam números inteiros: `int` idade = 16;
- Variáveis do tipo **`char`** que guardam caracteres: `char` letra = 'a';
- Variáveis do tipo **`float`** que guardam números com ponto decimal: `float` altura = 1.70;

ATENÇÃO!

Nada nos impede de armazenar um número inteiro em uma variável do tipo `float`, afinal, um número inteiro também está dentro do conjunto dos números racionais (números com vírgula).

Da mesma forma, também podemos guardar um número decimal em uma variável do tipo `int`, mas nesse caso, apenas o número inteiro será armazenado, a parte decimal será ignorada. Por fim, também é possível armazenar números inteiros e números com vírgula em uma variável do tipo `char`, porém, o valor armazenado será considerado como caractere, e não como número.

Mas como assim? Agora fiquei confuso! Antes você disse que só era possível guardar valores de acordo com o tipo de variável. Números inteiros em variáveis do tipo `int`, números com casa decimal em variáveis do tipo `float` e caracteres em variáveis do tipo `char`. E agora você me diz que eu posso guardar qualquer coisa dentro de uma variável do tipo `char`.

Pois é, meu caro amigo! Essa é uma das pegadinhas da linguagem C. Respondendo a sua pergunta, sim, é possível guardar números em uma variável do tipo `char`, mas isso não significa que você podera fazer operações matemáticas com valores armazenados em uma variável do tipo `char`, porque ela será estendida como um caractere. Mas isso é assunto para a nossa próxima seção.

B. Declaração de Variáveis

Agora que entendemos que cada variável armazena um tipo de dado, devemos aprender a declarar uma variável de acordo com o tipo dela. Esse é o primeiro passo para utilizarmos uma variável em nosso código, sem ele, a compilação apresentará falhas.

Lembrando: compilação é a fase em que o computador olha o seu código e verifica se não existem erros de sintaxe em relação à linguagem que você escreveu.

EXEMPLO I)

Neste exemplo iremos declarar uma variável do tipo `int` chamada 'distancia'.

```
1  #include <Sparki.h>
2
3  int distancia; //declarando a variavel
distancia do tipo int
4
5  void setup()
6  {
7  }
8
9  void loop()
10  {
11    distancia=10; //atribuindo o valor
de 10 (int) para a variavel distancia
12  }
```

EXEMPLO II)

Neste exemplo iremos fazer o código diferente ao anterior mas que faz a mesma coisa.

```
1  #include <Sparki.h>
2
3  int distancia=10; //declarando a
variavel distancia como int e atribuindo o
valor 10 a ela
4
5  void setup()
6  {
7  }
8
9  void loop()
10  {
11  }
```

Geralmente, optamos por declarar por uma variável fora do `void setup()` e do `void loop()`, pois, assim, podemos utilizá-la em qualquer lugar do código, mas também é possível declará-la dentro do `void setup()` ou `void loop()`. A diferença é que, caso você declare uma variável dentro do `void loop()`, apenas poderá utilizá-la dentro do `void loop();`

C. Operações entre variáveis

Vejamos alguns exemplos de operações entre números inteiros.

EXEMPLO I)

```
1  #include <Sparki.h>
2
3  int valor1 = 10;
4  int valor2 = 4;
5
6  void setup()
7  {
8  }
9
```

```
10  void loop()
11  {
12    sparki.clearLCD();
13
14    sparki.print ("valor 1 = ");
15    sparki.println(valor1);
16
17    sparki.print("valor 2 = ");
18    sparki.println(valor2);
19
20    sparki.print("valor1 + valor2 = ");
21    sparki.println(valor1 + valor2);
22
23    sparki.print("valor1 - valor2)";
24    sparki.prinln(valor1 - valor2);
25
26    sparki.print("valor1 * valor2");
27    sparki.println(valor1 * valor2);
28
29    sparki.print("valor1 / valor2");
30    sparki.print(valor1 / valor2);
31
32    sparki.updateLCD();
33    delay(1000);
34  }
```

No código acima podemos observar que as variáveis foram declaradas nas **linhas 1 e 2**. Nas **linhas 14 e 17** utilizamos a função sparki.print para imprimir os textos `'valor 1 ='` e `'valor 2 ='`. Já nas **15 e 18** utilizamos a função sparki.print para imprimir os valores das variáveis na tela LCD.

Para efetuar a operação de adição entre variáveis utilizamos o operador de adição +, como mostrando na **linha 21**. Da mesma forma, para efetuar a operação de subtração utilizamos o operador de subtração −, como podemos ver na **linha 24**. Para as operações de multiplicação e divisão, utilizamos os operadores `*` e `/`, conforme as **linhas 27 e 30**.

Você deve ter notado que a última linha apresentada na tela LCD não apresenta o resultado correto da operação, né? Afinal, 10 dividido por 4 é igual a 2,5, e não 2.

Ah, disso eu lembro! Você disse na seção anterior que variáveis do tipo `int` não guardam a casa decimal do valor. Então foi por isso que a operação não mostrou o resultado correto.

Muito bem lembrado! Se a operação fosse entre variáveis do tipo `float` o valor correto teria sido apresentado na tela LCD.

Ainda existe mais uma forma de efetuar operações de adição e subtração entre uma mesma variável do tipo `int`. Imagine que desejamos somar **6** à variável `distancia` e depois disso imprimiremos o novo valor da variável na tela LCD. Podemos fazer isso de duas maneiras, vejamos o exemplo abaixo.

EXEMPLO III)

```
1   #include <Sparki.h>
2   void setup()
3   {
4   }
5
6   void loop()
7   {
8     sparki.clearLCD();
9
10     int distancia=10;
11     sparki.print("distancia=");
12     sparki.println(distancia);
13     distancia=distancia+1;
14     sparki.print("A nova distancia é:");
15     sparki.println(distancia);
16
17     int distancia=10;
18     sparki.print("distancia=");
19     sparki.println(distancia);
20     distancia++;
21     sparki.print("A nova distancia é:");
22     sparki.println(distancia);
23     sparki.updateLCD();
24     delay(1000);
25   }
```

O código apresentado acima mostra na tela LCD o mesmo resultado duas vezes, porém, a operação é feita de formas diferentes. Na **linha 13** a operação é feita de forma convencional. Na **linha 20** o atalho `variável++` é utilizado, que consiste em somar 1 ao valor anterior da variável. Ou seja, se a variável declarada armazenasse o valor **19**, após a utilização do código `variável++`, o novo valor da variável seria **20**.

Assim como no exemplo anterior, também podemos utilizar a mesma estrutura do código acima para fazer a operação que subtrai 1 da variável. Aepnas precisamos trocar o operador ++ pelo operador −. No caso da subtração, o "atalho"consiste em `variavel- -`.

EXEMPLO IV)

Neste exemplo, iremos juntar o nosso conhecimento em LCD com o de varáveis.

```
1   #include <Sparki.h>
2   void setup()
3   {
4   }
5
6   void loop()
7   {
8     int var = 10;
9     sparki.clearLCD();
10     sparki.println("Valor da variavel
var:");
11     sparki.print(var);
12     sparki.updateLCD();
```

```
13    delay(1000);
14  }
```

Note que, quando vamos imprimir o valor que uma variável está armazenando, devemos escrever o nome dela como parâmetro sem as aspas. Mas, se quisermos imprimir apenas os caracteres do jeito que está escrito nos parâmetros, devemos escrever a mensagem desejada entre aspas.

D. Exercícios

1) Ajuste o código do Exemplo I da seção de "Operações entre variáveis"para que todos os valores das operações sejam apresentados corretamente na tela LCD, ou seja, apresentando também a parte decimal do resultado.

2) Escreva um código que apresente um contador na tela LCD (contadores são ciclos de repetição de um código, no qual acumulam seu próprio valor, acrescentando 1 a cada execução do programa. Ex: 0 + **1** -> 1 + **1** -> 2 + **1** -> 3 + **1** -> 4 ...).

IX. Ultrassom

A. Introdução

No capítulo de ultrassom iremos tratar dos "olhos"do *Sparki*. Eles estão presentes na plaquinha azul, naquilo que você pode ter considerado ser a cabeça dele. Embora, à primeira vista, os dois "olhos"pareçam câmeras feitas para que nosso robozinho possua visão, na verdade eles fazem parte de um sensor bem diferente, que está mais associado com a audição.

Iremos então entender o que é esse sensor, como ele funciona e como podemos utilizá-lo facilmente, de forma que o *Sparki* seja capaz de desviar de obstáculos enquanto anda.

B. Sensor Ultrassônico

O funcionamento do sensor é muito simples. O que muitos podem ter achado se tratar de duas câmeras, na verdade são um alto-falante e um microfone. O alto-falante emite uma onda sonora de alta frequência, alta demais para que nós humanos sejamos capazes de escutá-la. Essa onda de som é refletida assim que vai ao encontro de algum objeto e retorna para o microfone.

Mas qual a utilidade de se emitir um som e depois escutá-lo de volta?

A funcionalidade não está no som em si, mas no tempo que demorou para que a onda sonora retornasse para o sensor. Essa informação de tempo é então utilizada para calcular a distância do sensor até o objeto que refletiu a onda de volta.

O sensor ultrassônico é utilizado para obter a distância entre o robô e algum objeto à sua frente. Mas nem sempre conseguimos encontrar essa distância como esperado. Isso ocorre pois, às vezes, o objeto não está bem posicionado, logo, não reflete o som de volta para o microfone, ou então ele absorve todo o som, o que pode acontecer se ele for muito macio.

C. Aplicações no Cotidiano

1) Ultrassonografia:

A Ultrassonografia, também conhecida por Ecografia e Ultrassom, é um exame de diagnóstico que serve para visualizar em tempo real qualquer órgão ou tecido do corpo. Ela é comumente usada para observar o desenvolvimento do feto durante a gravidez.

Sabe aquelas imagens cinzas borradas que ninguém além do médico é capaz de compreender? Elas são feitas utilizando o ultrassom.

2) Ecolocalização:

Embora não seja utilizada por seres humanos, a ecolocalização ou biossonar é muito importante na vida de diversos animais como morcegos, golfinhos e baleias. Eles são seres capazes de emitir ondas de alta frequência e cronometrar o tempo levado até que consigam escutar o seu próprio eco, assim, são capazes de conhecer seus arredores mesmo sem o uso da visão.

3) Sonares:

Sonares são muito utilizados por navios e submarinos para detectar a presença de outros corpos debaixo da água. Seu funcionamento é idêntico ao de um biossonar utilizado por uma baleia, por exemplo. Sonares diferem dos radares pois os radares utilizam ondas de rádio para medir a distância.

D. Função

Para podermos utilizar o sensor ultrassônico presente no *Sparki*, tudo o que precisamos fazer é escrever a função `sparki.ping()`, que ativará o sensor. Como já dito, o sensor serve para medir a distância, logo, quando o ativamos, espera-se que recebamos algum valor de volta que indique qual é essa distância. A função `sparki.ping()` já faz os cálculos necessários e nos retorna o valor da distância em centímetros.

A função então nos devolve um valor, mas como que eu vejo isso? Onde que ele vai me falar qual a distância que ele mediu? Ele escreve em algum lugar?

AHA! Acabamos de chegar na parte importante desse capítulo, as funções que retornam valores. O *Sparki* irá escrever o valor que o sensor encontrou no local que a gente determinar para isso, mas claro que isso não quer dizer que podemos simplesmente falar para ele escrever a resposta num papel que ele irá fazer (possível, mas o código para isso seria muito grande).

Uma forma interessante de resolver este conflito é a seguinte: nós criamos o local onde ele deve escrever a resposta, mas esse local não pode ser qualquer um, ele tem que ser uma variável. Sempre que criamos uma variável temos que definir também o tipo dela, e como, no nosso caso, estamos trabalhando apenas com centímetros é mais adequado criar uma variável do tipo `int`, ou seja, que armazena números inteiros.

O código que mostra como fazemos isso está escrito logo abaixo:

```
#include <Sparki.h>

int distancia_em_cm;

void setup()
{
}

void loop()
{
    distancia_em_cm = sparki.ping();
    delay(300);
}
```

Nesse código estamos dizendo que o *Sparki* deve executar a função `sparki.ping()` e salvar o valor que ela retornar dentro da nossa variável `distancia_em_cm`. Agora que temos a nossa distância salva dentro de uma variável, podemos trabalhar com ela livremente para fazermos o que quiser, como fazer ele escrever o valor numa folha de papel.

É importante notar a presença da função `delay()` no código, pois o funcionamento do sensor não é tão rápido quanto a velocidade de processamento do nosso robô, então precisamos dar um pequeno intervalo de tempo para que o sensor possa funcionar sem erros.

E. Servo motor

Se considerarmos o sensor ultrassônico como sendo a "cabeça"do robô, é muito útil que tenhamos também um "pescoço"que sirva para movimentarmos a "cabeça", então surge a necessidade de um outro servo motor no *Sparki*. Você deve se lembrar da garra apresentada no capítulo de Movimentação. Assim como o nosso "pescoço", a garra também funciona por meio da ação de um servo motor, mas afinal o que ele é?

Um servo motor é um motor que tem a capacidade de girar o seu eixo em ângulos precisos de até 90º ou 180º normalmente, sendo que o servo motor no qual o sensor ultrassônico está acoplado é capaz de girar seu eixo em até 180º. Ele é muito útil para movimentar braços robóticos, por exemplo. Nesse caso, não é necessária a realização de voltas completas e a precisão torna-se muito importante. No nosso caso, utilizamos ele para girar o sensor, e assim, podermos medir distâncias em outras direções sem ter que utilizar as rodas para girar o robô, infelizmente não temos como fazer o *Sparki* olhar para cima ou para baixo utilizando um servo motor apenas.

F. Função

Agora que sabemos da existência do servo motor, precisamos saber como utilizá-lo. Para começar, devemos ativá-lo escrevendo a função `sparki.servo()`. Esta função deve receber um valor de angulação em graus, que determina o quanto ele deve girar. Este valor deve estar entre -90º e 90º (sendo valores negativos para a esquerda e positivos para a direita) e o valor pode ser um número quebrado, isto é, o argumento da função é do tipo `float`.

Para facilitar a nossa vida, a função `sparki.servo()` já vem com 3 valores predefinidos dos ângulos mais utilizados: `SERVO_LEFT` (-90º), `SERVO_CENTER` (0º) e `SERVO_RIGHT` (90º). Assim, tudo o que precisamos fazer é escrever `SERVO_[]` com o sentido desejado como argumento que o *Sparki* se encarrega de colocar o sensor na posição correta.

Abaixo temos um código que faz com que o *Sparki* meça a distância dele para os objetos na sua esquerda, frente, direita e diagonais esquerda e direita:

```
#include <Sparki.h>

int distancia_esquerda;
int distancia_direita;
int distancia_frente;
int distancia_diagonal_esquerda;
int distancia_diagonal_direita;

void setup()
{
    sparki.servo(SERVO_LEFT); //gira ára
a posição -90 graus
    delay(500);
    distancia_esqueda = sparki.ping();
    sparki.servo(-45); //gira ára a
posição -45 graus
    delay(500);
    distancia_diagonal_esqueda =
sparki.ping();
    delay(500);
    sparki.servo(SERVO_CENTER); //gira
ára a posição 0 graus
    delay(500);
    distancia_frente = sparki.ping();
    delay(500);
    sparki.servo(45); //gira ára a
posição 45 graus
    delay(500);
    distancia_diagonal_direita =
sparki.ping();
    delay(500);
    sparki.servo(SERVO_RIGHT); //gira
ára a posição 90 graus
    delay(500);
    distancia_direita = sparki.ping();
    delay(500);
}

void loop()
{
}
```

X. If e else

A. Introdução

Neste capítulo aprofundaremos nosso estudo de programação por meio de comandos que permitem a tomada de decisão, ou seja, estruturas auxiliares que estabelecem uma condição para a execução de um comando principal. Ficou confuso? Não se preocupe, vou explicar a seguir!

B. Estruturas de decisão

Desde o momento em que acordamos, estamos tomando decisões, às vezes até sem perceber! Por exemplo, optar por ficar alguns minutos a mais na cama e adiar o despertador ou decidir levantar no instante em que ele tocar. Vamos supor que amanhã você tem uma prova muito importante às 8 horas da manhã, o fato de escolher ignorar o despertador e dormir mais pode resultar em consequências, como perder aquela prova de final de semestre e, possivelmente, acabar ficando com nota baixa. Como não é isso que queremos, é sempre bom acordar cedo em dias que tivermos provas pela manhã.

Estabelecemos uma condição para acordar cedo, você percebeu? A condição de acordar cedo é se houver uma prova pela manhã... É exatamente isso que faremos neste capítulo! Estabeleceremos condições para que o *Sparki* realize determinada ação!

Para não esquecer!

As estruturas de decisão que utilizaremos são "if"e "else", elas significam "se"e "senão"em inglês, respectivamente.

C. Como utilizar "if"e "else"?

A seguir temos exemplos de uso dessas estruturas:

EXEMPLO I)

Exemplo de comparação de variáveis.

```
1  if(Condição 1)
2  {
3    Ação 1;
4  }else
5  {
6    Ação 2;
7  }
```

Esse exemplo significa que, se a "Condição 1"for verdadeira, o *Sparki* executará a "Ação 1", senão, ele executará a "Ação 2". Utilizar o `else` como uma opção alternativa ao `if()` é optativo, ou seja, podem existir expressões condicionais apenas com `if()`.

Reflita: A “Ação 1” e a “Ação 2” poderiam ser executadas sequencialmente, em apenas uma leitura desse código?

A seguir temos uma tabela com todas as opções de execução do código desse primeiro exemplo: (Tente entender a tabela, apenas decorar pode te deixar confuso mais para frente)

EXEMPLO II)

Condição 1	Ação 1	Ação 2
Verdadeira	Executa	Não executa
Falsa	Não executa	Executa

Exemplo da explicação. (Apenas para fins didáticos)

```
1  if(For dia de prova)
2  {
3    Acordar mais cedo;
4  }else
5  {
6    Acordar no horário normal;
7  }
```

Neste caso, se for o dia da prova, devemos acordar mais cedo e, se não for o dia da prova, devemos acordar no horário normal. A segunda ação apenas ocorre se a primeira não ocorrer, logo, essas duas ações não poderiam ser executadas sequencialmente, ou seja, em uma mesma verificação da condição de ser ou não dia de prova.

EXEMPLO III)

Exemplo com “else if”. (Apenas para fins didáticos)

```
1  if(Condição 1)
2  {
3    Ação 1;
4  }else if(Condição 2)
5  {
6    Ação 2;
4  }else
5  {
6    Ação 3;
7  }
```

Agora apareceram `else` e `if` em uma mesma linha, o que isso significa?

Para entender os dois termos juntos, vamos revisar o significado deles separados. `else` significa: "se as condições anteriores forem falsas, executar o comando dentro das chaves". `if()` significa: "se a condição dentro dos parênteses for verdadeira, executar o comando dentro das chaves". Consequentemente, a expressão `else if()` implica na execução do comando apenas se as condições anteriores forem falsas e a condição dentro do parênteses for verdadeira.

Condição 1	Ação 1	Condição 2	Ação 2	Ação 3
Verdadeira	Executa	Falsa	Não executa	Não executa
Verdadeira	Executa	Verdadeira	Não executa	Não executa
Falsa	Não executa	Verdadeira	Executa	Não executa
Falsa	Não executa	Falsa	Não executa	Executa

EXEMPLO IV)

Imagine uma situação hipotética em que existam 3 compromissos na sua agenda, no mesmo dia e horário, e você

terá que escolher apenas um, de acordo com a previsão do tempo. (Apenas para fins didáticos)

```
if (Estiver fazendo muito calor)
{
  Ir ao clube;
}else if (Estiver chovendo)
{
  Ver um filme em casa;
}else if (Estiver nevando)
{
  Ir esquiar;
 }else
 {
   Ir ao cinema;
 }
```

Isshh... Agora ficou complicado! E se a previsão do tempo afirmar que vai chover e nevar nesse dia? Qual compromisso eu escolho?

Essa é fácil! É necessário apenas observar qual das condições será lida primeiro nesse código, ou seja, o que vier primeiro! Como a condição "Estiver chovendo"aparece primeiro, o compromisso seria "Ver um filme em casa".

E qual seria a condição para "Ir ao cinema"?

A condição seria a negação das anteriores, neste caso, se não estivesse fazendo muito calor, nem chovendo, nem nevando.

Uma curiosidade dessa estrutura condicional é que se escrevermos 1 dentro dos parênteses do `if()`, ele será verdadeiro e o que estiver dentro das chaves será executado. Assim, se escrevermos 0 dentro dos parênteses, o `if()` será falso e o que estiver dentro das chaves não será executado. Isso se dá porque a programação possui uma base binária, ou seja, ela é baseada em vários 1's e 0's, sendo 1 verdadeiro e 0 falso.

D. Operações de comparação

Iremos aprender nesta seção os tipos de operadores que podemos utilizar para comparar dois valores ou incógnitas. Lembrando que eles devem ser utilizados apenas dentro do parênteses do `if()`. São eles:

- == para verificar se os dois valores são iguais.
- != para verificar se os dois valores são diferentes.
- > para verificar se o primeiro é maior que o segundo.
- >= para verificar se o primeiro é maior que o segundo ou igual a ele.
- < para verificar se o primeiro é menor que o segundo.
- <= para verificar se o primeiro é menor que o segundo ou igual a ele.

Agora que você já entendeu como funcionam as expressões condicionais e os operadores de comparação, vamos para um exemplo de verdade!

EXEMPLO 1)

Atribuiremos o valor 2042385 à variável `x` e queremos que o *Sparki* responda se esse número é ímpar ou par. Para isso, utilizaremos a informação da sessão anterior sobre os 1's e os 0's.

```
#include <Sparki.h>

void setup()
{
  sparki.clearLCD();
  if (x % 2)
  {
    sparki.print("O numero e impar.");
  }
   else
   {
     sparki.print("O numero e par.");
   }
   sparki.updateLCD();
   delay(1000);
 }
```

Sabemos que o número 2042385 é ímpar pela regra da divisão por 2, que afirma que um número é divisível por 2 quando o último algarismo dele for divisível por 2. Nesse código, não utilizamos essa regrinha, mas sim o método mais tradicional, fazemos o *Sparki* verificar se a divisão desse número por 2 é exata, ou seja, se o resto é 0. Assim, o *Sparki* chega a mesma conclusão que chegamos, que o número 2042381 é ímpar, e imprime essa informação no LCD.

Lembrando: O símbolo `%` significa o resto da divisão do primeiro número pelo segundo. Exemplo:

10 % 2 = 0 e 10 % 3 = 1

EXEMPLO 2)

Dessa vez, faremos com que o *Sparki* responda se o número 2042385 é divisível por 3 e por 5.

```
#include <Sparki.h>

void setup()
{
  sparki.clearLCD();
  if((x % 3) == 0)
  {
    sparki.println("O numero e
divisivel por 3.");
   }else
   {
     sparki.println("O numero nao e
divisivel por 3.");
   }
   if((x % 5) == 0)
   {
     sparki.print("O numero e divisivel
por 5.");
   }else
```

```
19    {
20      sparki.print("O numero nao e
divisivel por 5.");
21    }
22    sparki.updateLCD();
23    delay(1000);
24  }
```

Esse exemplo é parecido com o anterior, mas ele está aqui para mostrar que podemos inserir duas estruturas condicionais independentes em um mesmo código. As duas comparações para saber se o número é divisível por 3 e por 5 não dependem uma da outra, pois um número pode ser divisível por 3 e por 5 ao mesmo tempo, por isso que existem dois `if()` e dois `else`. Consequentemente, o *Sparki* imprimirá duas mensagens na tela, na primeira linha: "O numero e divisivel por 3"e na segunda: "O numero e divisivel por 5".

EXEMPLO 3)

Exemplo de comparação de variáveis.

```
1  #include <Sparki.h>
2  void setup()
3  {
4  }
5  void loop()
6  {
7    int x = 10;
8    int y = 100;
9    sparki.clearLCD();
10    if(x >= y) {
11      sparki.print("Condicao 1");
12    } else if(y != (x * x)) {
13      sparki.print("Condicao 2");
14    } else if((y / x) >= x) {
15      sparki.print("Condicao 3");
16    } else if((y - 100) <= x) {
17      sparki.print("Condicao 4");
18    }
19    sparki.updateLCD();
20    delay(1000);
21  }
```

Você reparou que a disposição das chaves no código está diferente das utilizadas anteriormente? Os dois modos são válidos, sendo as chaves coladas no `if` ou em outra linha, mas é sempre bom manter um padrão, por isso, antes de começar um código, decida qual será a convenção utilizada. Então, já sabe o que o *Sparki* fará ao ler este código? Vamos ver passo a passo:

- **Condição 1)** `x >= y`
  10 >= 100
  Essa afirmação é FALSA, logo, o *Sparki* não executará a ação entre as chaves.
- **Condição 2)** `y != (x * x)`
  100 != (10 * 10)
  100 != 100
  Essa afirmação também é FALSA, por isso a ação não será executada.
- **Condição 3)** `(y / x) >= x`
  (100 / 10) >= 10
  10 >= 10
  Essa afirmação é VERDADEIRA, logo, o *Sparki* executará a ação entre as chaves, imprimirá no LCD "Condicao 3".
- **Condição 4)** Essa condição não será nem lida, pois ela só poderia ser executada se todos os `if()` anteriores, dentro dessa estrutura condicional, fossem falsos.

Lembrando: Sempre devemos executar a operação dentro dos parênteses antes dos operadores de comparação.

E. Operadores lógicos

Nesta seção, iremos aprender a lidar com comparações que envolvem mais de um fator a ser analisado e, para isso, utilizaremos o início da teoria da Álgebra Booleana, criada por George Boole. Na Álgebra de Boole, existem 3 tipos de operadores lógicos: E, OU e NÃO, cada um com uma funcionalidade própria. Eles devem ser usados dentro de condicionais `if()` para ligar duas comparações ou para negar uma ou mais comparações.

1) E (AND):

Este operador depende de pelo menos dois "resultados"de afirmações, por exemplo, vamos supor que você quer ir à piscina, mas você apenas irá se estiver fazendo sol E se a piscina estiver limpa. Então, a 1ª afirmação é: se estiver fazendo sol e a 2ª afirmação é: se a piscina estiver limpa. O operador AND vai funcionar como uma "ponte"de ligação entre essas duas afirmações.

A condição estabelecida pelo `if()`, com o AND entre as afirmações, será VERDADEIRA apenas quando as afirmações forem ambas VERDADEIRAS.

Então o AND vai ser uma comparação entre duas comparações?? Que confuso!

Mais ou menos... É mais certo dizer que o AND irá realizar uma operação entre as duas comparações/afirmações. Utilizando o exemplo anterior, as duas afirmações têm que ser verdadeiras, ou seja, 1) tem que estar fazendo sol e 2) a piscina tem que estar limpa para que a comparação seja verdadeira e você possa ir à piscina.

EXEMPLO 0)

```
1  if(Fizer sol AND Piscina estiver limpa)
2  {
3    Ir à piscina();
1  }else
2  {
3    Não ir à piscina();
2  }
```

Se a comparação 1 der VERDADEIRA e a comparação 2 der FALSA, o `if()` não será realizado e você não irá à piscina. Porém, se as duas comparações derem VERDADEIRAS, o `if()` será realizado e você poderá ir à piscina.

Agora, vamos fazer um esquema com todas as possibilidades de combinação entre duas comparações e ver o resultado final:

- VERDADEIRO AND VERDADEIRO -> Resultado: VERDADEIRO
- VERDADEIRO AND FALSO -> Resultado: FALSO
- FALSO AND VERDADEIRO -> Resultado: FALSO

Lembrando: Resultado: VERDADEIRO, significa que o if() será executado. Resultado: FALSO, significa que o if() não será executado.

Tá, mas como eu aplico isso na programação?

Na programação, utilizamos os caracteres && para simbolizar o AND. Aqui temos um exemplo:

EXEMPLO 1)

Neste exemplo, serão criadas 3 variáveis para armazenar a idade de 3 crianças: o Enzo, que tem 1 ano, a Valentina, que tem 2 anos, e a Sofia com 3 anos. Se a idade das 3 crianças forem iguais, o Sparki andará para frente, se a Sofia for a mais velha e o Enzo o mais novo, o Sparki andará para trás, e, por último, se as duas condições anteriores forem falsas, o Sparki ficará parado.

```
1  #include <Sparki.h>
2  void setup(){
3  }
4  void loop(){
5    int idade_enzo = 1;
6    int idade_valentina = 2;
7    int idade_sofia = 3;
8    if (idade_enzo == idade_valentina &&
 idade_enzo == idade_sofia){
9      // Se o Enzo, a Valentina e a Sofia
tiverem a mesma idade
10     // Neste caso, FALSO AND FALSO ->
Resultado: FALSO
11     sparki.moveForward();
12   }
13   else if (idade_valentina >
idade_enzo &&
 idade_sofia > idade_valentina){
14     // Se o primeiro caso "if"for
falso e a Sofia for mais velha que a
15     // Valentina, que por sua vez é
mais velha que o Enzo
16     // Neste caso, VERDADEIRO AND
VERDADEIRO -> Resultado: VERDADEIRO
17     sparki.moveBackward();
18   }
19   else{
20     // Se o primeiro e o segundo caso
forem falsos
21   }
22 }
```

O Sparki andará para trás.

- **Condição 1)**
  `idade_enzo == idade_valentina`
  1 == 2
  Essa afirmação é FALSA.

  `idade_enzo==idade_sofia`
  1 == 3
  Essa afirmação é FALSA.

  Como temos o (AND) ligando as duas afirmações, a comparação será FALSA.

- **Condição 2)**
  `idade_sofia > idade_valentina`
  3 > 2
  Essa afirmação é VERDADEIRA.

  `idade_valentina > idade_enzo`
  2 > 1
  Essa afirmação também é VERDADEIRA.

  Como as duas afirmações são verdadeiras e o && (AND) está ligando-as, a comparação será VERDADEIRA.

EXEMPLO 2)

Um exemplo com as mesmas variáveis anteriores, mas um pouco mais complicado.

```
1  #include <Sparki.h>
2  void setup()
3  {
4  }
5  void loop()
6  {
7    int idade_enzo = 1;
8    int idade_valentina = 2;
9    int idade_sofia = 3;
10    sparki.clearLCD();
11    if ((idade_sofia - idade_valentina)
== idade_enzo &&
 (idade_valentina - idade_enzo) == 3)
12    {
13      // VERDADEIRO AND FALSO ->
Resultado: FALSO
14      sparki.drawCircleFilled(63, 32,
10);
15    }
16    else if ((idade_enzo + 1) != 2 &&
```

```
(idade_valentina - idade_enzo) !=
idade_sofia)
17    {
18      // FALSO AND VERDADEIRO ->
Resultado: FALSO
19      sparki.drawRectFilled(10, 0, 15,
30);
20    }
21    else if ((idade_enzo * 6) ==
(idade_valentina * 3))
22    {
23      // VERDADEIRO
24      sparki.drawChar(10, 1, 'a');
25    }
26    else
27    {
28      sparki.drawString(40, 4, "123");
29    }
30    sparki.updateLCD();
31    delay(1000);
32  }
```

O *Sparki* irá imprimir o caractere ”a” no LCD.

- **Condição 1)**
  `(idade_sofia - idade_valentina) == idade_enzo`
  (3 - 2) == 1
  1 == 1
  Essa afirmação é VERDADEIRA.

  `(idade_valentina - idade_enzo) == 3`
  (2 - 1) == 3
  1 == 3
  Essa afirmação é FALSA.

  Como temos o && (AND) ligando essas duas afirmações, a comparação será FALSA.

- **Condição 2)**
  `(idade_enzo + 1) != 2`
  (1 + 1) != 2
  2 != 2
  Essa afirmação é FALSA.

  `(idade_valentinaidade_enzo) != idade_sofia`
  (2 - 1) != 3
  1 != 3
  Essa afirmação é VERDADEIRA.

  Como o && (AND) está ligando essas duas afirmações, a comparação será FALSA.

- **Condição 3)**
  `(idade_enzo * 6) == (idade_valentina * 3)`
  (1 * 6) == (2 * 3)
  6 == 6

  Essa afirmação é VERDADEIRA, logo, a comparação também é.

Para não esquecer!

“And” traduzido para o português significa “e”.

F. OU (OR)

Este operador deve ser colocado entre duas ou mais afirmações, assim como o AND. Vamos supor que a sua mãe está viajando e você combinou de se encontrar com ela quando ela chegasse no aeroporto. O combinado foi o seguinte: ela mandaria mensagem OU ligaria quando estivesse embarcando no voo de volta, e você sairia de casa por volta de 30 minutos depois, para se encontrar com ela no aeroporto.

A condição estabelecida pelo if(), com o OR entre as afirmações, será VERDADEIRA se pelo menos uma das duas comparações for VERDADEIRA.

EXEMPLO 0)

Exemplo da explicação, apenas para fins didáticos.

```
1  if (Sua mãe te mandasse mensagem OR
Sua mãe te ligasse)
2  {
3    Esperar 30 minutos;
4    Sair de casa;
5  }
6  else
7  {
8    Continuar esperando a mensagem ou a
ligação;
9  }
```

Então, se a sua mãe te ligasse (afirmação VERDADEIRA) mas não te mandasse mensagem (afirmação FALSA), a condição do if() seria VERDADEIRA e você esperaria 30 minutos para sair de casa. Se as duas afirmações fossem VERDADEIRAS, ou seja, se ela te mandar mensagem e te ligar, a condição if() também será VERDADEIRA, e você executará as ações dentro das chaves do if(). A seguir temos todos os resultados possíveis para um OR entre duas afirmações:

- VERDADEIRO OR VERDADEIRO -> Resultado: VERDADEIRO
- VERDADEIRO OR FALSO -> Resultado: VERDADEIRO
- FALSO OR VERDADEIRO -> Resultado: VERDADEIRO
- FALSO OR FALSO -> Resultado: FALSO

Na programação, utilizamos os caracteres || para simbolizar o OR.

EXEMPLO 1)

```
#include <Sparki.h>;

void setup()
{
}

void loop()
{
  int idade_enzo = 1;
   int idade_valentina = 2;
   int idade_sofia = 3;
 }
 if(idade_enzo == idade_valentina ||
idade_sofia != idade_valentina)
 {
   //Neste caso, FALSO OR VERDADEIRO ->
Resultado: VERDADEIRO
   sparki.moveBackward();
 }
 else if(idade_valentina ==
(idade_sofia - 1) || (idade_valentina +
1) == idade_sofia)
 {
   //Neste caso, VERDADEIRO OR
VERDADEIRO -> Resultado: VERDADEIRO
   sparki.moveForward();
 }
 else
 {
   //Se a primeira e a segunda condição
forem falsas
   sparki.moveRight();
 }
```

O Sparki andará para trás.

- **Condição 1)**
  `idade_enzo == idade_valentina`
  1 == 2
  Essa afirmação é FALSA.

  `idade_sofia != idade_valentina`
  3 != 2
  Essa afirmação é VERDADEIRA.

  Como o `||` (OR) liga essas duas afirmações, sabemos que a comparação será VERDADEIRA.
- **Condição 2)** Essa condição não será nem lida, pois ela só poderia ser executada se todos os `if()` anteriores, dentro dessa estrutura condicional, fossem falsos.

EXEMPLO 2)

```
#include <Sparki.h>;

void setup()
{
}

void loop()
{
  int idade_enzo = 1;
   int idade_valentina = 2;
   int idade_sofia = 3;
   sparki.clearLOO();
   if (idade_valentina != 2 ||
idade_enzo != 1)
   {
     // FALSO OR FALSO -> Resultado:
FALSO
     sparki.print("Condicao 1");
   }
   else if (idade_sofia !=
(idade_valentina - 1) &&
 idade_sofia == idade_enzo)
   {
     // Código continua...
   }
 //VERDADEIRO AND FALSO -> Resultado:
FALSO
 sparki.print("Condicao 2");
 else if ((idade_enzo + 2) ==
(idade_sofia + 1) || idade_enzo == (4 -
idade_sofia))
 {
   //FALSO OR VERDADEIRO -> Resultado:
VERDADEIRO
   sparki.print("Condicao 3");
 }
 else
 {
   sparki.print("Nenhuma das anterio-
res");
 }
 sparki.updateLCD();
 delay(1000);
 }
```

O *Sparki* irá imprimir na tela "Condição 3".

- **Condição 1)**
  `idade_valentina == 2`
  `idade_enzo == 1`
  Ambas afirmações são FALSAS.

- **Condição 2)**
  `idade_sofia == idade_enzo`
  Essa afirmação é FALSA.

- **Condição 3)**
  `(idade_enzo + 2) == (idade_sofia + 1)`
  (1 + 2) == (3 + 1)

3 == 4
Essa afirmação é FALSA.

`idade_enzo == (4 - idade_sofia)`
1 == (4 - 3)
1 == 1
Essa afirmação é VERDADEIRA, logo FALSO OR VERDADEIRO -> Resultado: VERDADEIRO.

Para não esquecer!
"Or" traduzido para o português significa "ou".

G. NÃO (NOT):

- NOT(VERDADEIRO) -> Resultado: FALSO
- NOT(FALSO) -> Resultado: VERDADEIRO

```
#include <Sparki.h>

void setup()
{
}

void loop()
{
  int idade_enzo = 1;
   int idade_valentina = 2;
   int idade_sofia = 3;

   if (!(idade_enzo == 1))
   {
     //NOT(VERDADEIRO) -> Resultado:
FALSO
   }
   else if (!(idade_valentina == 3))
   {
     //NOT(FALSO) -> Resultado:
VERDADEIRO
     sparki.moveForward();
   }
   else if (!(idade_sofia != 3))
   {
     //NOT(FALSO) -> Resultado:
VERDADEIRO
     sparki.moveBackward();
   }
 }
```

O *Sparki* andará para frente, pois:

- **Condição 1)**
  `idade_enzo = 1`
  1 == 1
  Sabemos que a afirmação 1 == 1 é VERDADEIRA, mas há um caractere ! antes dela, por isso, ela acaba se tornando FALSA.

- **Condição 2)**
  `idade_valentina = 3`
  2 == 3
  A afirmação 2 == 3 é FALSA, mas o caractere ! antes dela a torna VERDADEIRA.

- **Condição 3)**
  Essa condição não será lida.

Para não esquecer!
"Not" traduzido para o português significa "não".

XI. INFRAVERMELHO

1) Introdução:

No capítulo de infravermelho, iremos falar sobre o funcionamento e utilidade que os sensores infravermelhos do Sparki possuem. As ondas infravermelhas estão mais presentes no nosso dia a dia do que você pode imaginar, na maioria das vezes elas são úteis quando utilizados receptores de luz infravermelha. Nesse capítulo iremos explicar como que o Sparki pode ser controlado pelo controle remoto e como fazê-lo seguir linhas utilizando dessas ondas.

A. Ondas Infravermelhas

Para começar, que tal falarmos do que exatamente e uma onda infravermelha? Ela, assim como a luz visível, e uma onda eletromagnética, uma onda capaz de se propagar no vácuo. A luz infravermelha e invisível aos olhos humanos, isso porque nossos fotorreceptores dentro dos nossos olhos, são incapazes de serem estimulados pela frequência das ondas infravermelhas. Chamamos essa onda de infravermelho, porque a sua frequência e menor do que a frequência da cor vermelha (o prefixo infra indica que algo e inferior). A cor vermelha é a cor com a frequência mais baixa que o ser humano e capaz de enxergar, logo se temos uma onda com frequência menor, ela se torna invisível.

Você deve estar lembrando do Capítulo de Ultrassom, onde falamos que ondas ultrassônicas são ondas sonoras que os seres humanos são incapazes de escutar porque possuem uma frequência muito alta. Essa é a mesma lógica da luz ultravioleta, presente acima na imagem 8.1. Elas são ondas eletromagnéticas com frequência mais alta do que a cor violeta, que é a cor com frequência mais alta que nós somos capazes de enxergar, logo, ela também é invisível aos nossos olhos. Alguns animais são capazes de ver essas cores, logo eles conseguem ver o mundo ainda mais colorido do que nós humanos.

Os raios infravermelhos originam-se dos corpos quentes como o Sol e o corpo de seres vivos, por exemplo. Cada cor está relacionada com uma temperatura (se você iluminar um termômetro com luzes de cores diferentes, ele irá dar temperaturas diferentes), e o vermelho é a cor mais quente. Logo, mesmo sem conseguirmos ver o infravermelho, ainda somos capazes de senti-lo porque ele aquece a nossa pele. Esse tipo de onda é muito importante para a vida na Terra. Ela é responsável pelo efeito estufa, onde parte dos raios solares

ficam presos dentro da nossa atmosfera e assim eles mantêm a Terra aquecida e propensa à vida.

B. Aplicações no cotidiano

1) Óculos de visão noturna:

Alguns óculos de visão noturna permitem que a gente enxergue no escuro captando a radiação infravermelha do ambiente e gerando um termograma, que nada mais é do que uma imagem colorida que mostra onde está mais quente e onde está mais frio. Assim, somos capazes de enxergar objetos e outras pessoas mesmo na escuridão.

2) Câmeras térmicas:

Usando a mesma tecnologia dos óculos de visão noturna mencionados acima, essas câmeras convertem frequências que estão no espectro infravermelho para o espectro visível, assim conseguimos visualizar os objetos por meio do calor que eles emitem. Essas câmeras são úteis para encontrar pessoas ou animais em meio à vegetação, destroços ou esconderijos, além de poder encontrar falhas em equipamentos dependendo de onde estiver mais quente ou mais frio do que o normal.

3) :

Quando você aperta um botão do seu controle remoto para mudar o canal da sua televisão, ele se comunica com a TV utilizando raios infravermelhos. Se você apontar a câmera do seu celular para a lâmpada na ponta do seu controle e apertar algum botão, você conseguirá enxergar os raios sendo emitidos. A seguir, iremos explicar como essa comunicação funciona.

C.

Agora, vamos começar a falar sobre como podemos controlar o Sparki utilizando o controle remoto, que vem junto com o kit. Ele funciona emitindo luz infravermelha a partir de sua lâmpada, que deve ser captada pelo sensor de infravermelho presente no Sparki.

Cada botão do controle remoto possui um padrão a ele relacionado. Esse padrão representa quantas vezes a lâmpada do controle deve piscar e por quanto tempo. O sensor recebe esse padrão, e o Sparki é responsável por traduzir esse padrão para o botão correspondente. Para facilitar o entendimento, podemos lembrar do código Morse, onde cada letra e número é associado a um conjunto de pontos e traços. O que o controle remoto faz é parecido com a tradução de uma palavra em português para o código Morse; depois, nosso robô deve saber ler essa palavra em código Morse e traduzi-la de volta para o português.

Atenção! Os sinais infravermelhos não estão em código Morse; mencionamos ele apenas como uma forma de facilitar a compreensão. No geral, controles remotos têm padrões diferentes uns dos outros.

Para facilitar a nossa vida, os criadores do Sparki atribuíram a cada botão um número inteiro.

```
1  #include <Sparki.h>;
2
3  int botao;
4
5  void setup()
6  {
7  }
8
9  void loop()
10  {
11    botao = sparki.readIR();
12  }
```

Esta função salva na nossa variável inteira o número correspondente ao último botão que foi apertado antes dela ser chamada. Caso não tenhamos apertado nenhum botão, ela irá salvar o valor '-1'. Graças ao capítulo de 'if' e 'else', podemos escrever um algoritmo que executa uma atividade diferente para cada botão que for apertado. Podemos controlar a movimentação do nosso robô da seguinte forma, por exemplo:

```
1  #include <Sparki.h>;
2
3  int botao;
4
5  void setup()
6  {
7  }
8
9  void loop()
10  {
11    botao = sparki.readIR();
12    if(botao == 70)
13    {
14      sparki.moveStop();
15      sparki.moveForward();
16    }
17    else if(botao == 67)
18    {
19      sparki.moveStop();
20      sparki.moveRight();
21    }
22    else if(botao == 68)
23    {
24      sparki.moveStop();
25      sparki.moveLeft();
26    }
27    else if(botao == 21)
28    {
29      sparki.moveStop();
30      sparki.moveBackward();
31    }
32    else if(botao == 64)
33    {
34      sparki.moveStop();
```

```
35      }
36      delay(1000);
37    }
```

D. Vetor de sensores infravermelhos

Talvez você nunca tenha reparado, mas embaixo do Sparki há cinco retângulos pretos perto da garra: três no meio e um em cada canto. Esses retângulos são nada mais nada menos do que os sensores infravermelhos responsáveis por medir a reflectância da luz do chão abaixo dele (a reflectância de um corpo é a quantidade de luz que ele reflete).

Você já deve ter ouvido falar que a cor preta é a ausência de todas as cores, enquanto o branco é quando todas elas estão presentes. Isso porque uma superfície preta absorve todas as frequências do espectro visível, enquanto o branco as reflete de volta. Já uma superfície amarela, por exemplo, absorve todas as frequências menos a do amarelo, por isso a vemos da cor amarela, porque foi a única frequência refletida para os nossos olhos. Como você já deve ter percebido, quando medimos a reflectância de uma superfície branca, certamente o valor será bem maior do que o de uma superfície preta.

Esses sensores então são utilizados para conseguirmos identificar quais são as cores que estão embaixo do Sparki? Essa é uma questão interessante! É possível diferenciar algumas cores pela sua reflectância, mas até mesmo cores bem diferentes podem representar valores iguais ou muito semelhantes. Os sensores não são tão precisos e adequados para essa situação; certamente haverão muitos enganos. A verdadeira utilização deles é conseguir diferenciar superfícies mais escuras das mais claras, o que é de extrema importância quando queremos fazer um exercício muito importante: um robô seguidor de linhas, que normalmente anda por cima das linhas pretas em um fundo branco.

É necessário certo cuidado ao utilizar esses sensores, porque uma mesma superfície pode retornar valores diferentes de reflectância. Não podemos garantir que esteja tudo pintado no exato mesmo tom ou que haja a mesma incidência de luz em todo lugar, por isso, quando mexemos com valores de reflectância, devemos adotar limiares. Limiares são intervalos de valores que, embora diferentes um do outro, consideramos como sendo a mesma coisa. Daremos como exemplo a separação das fases da vida: temos a fase infantil, adulta e terceira idade, por exemplo, e dependendo da sua idade você irá se encaixar em uma dessas três fases.

Limiares	0 a 1 ano	1 a 12 anos	12 a 18 anos	18 a 60 anos	60 anos em diante
Classificação	Bebê	Criança	Adolescente	Adulto	Idoso

A quantidade de limiares que vamos utilizar depende do problema que estamos tratando. Na situação acima, talvez fosse conveniente utilizar apenas 5 limiares, mas pode ser que, em outra situação, você tenha que incluir classificações como pré-adolescente ou ancião. Depois de determinar quais serão nossos limiares, podemos começar a coletar dados e, então, separá-los e classificá-los. Por exemplo, você tem 16 anos? Então você é um adolescente. Você tem 20 anos? Então você é um adulto. Você tem 14 anos, 7 meses, 2 semanas, 5 dias, 14 horas e meia de vida? Então você também é considerado um adolescente.

A importância do exemplo acima é fazer você entender que mais de um valor pode significar a mesma coisa, porque ao utilizar os sensores infravermelhos, você receberá muitos valores de reflectância diferentes uns dos outros, e mesmo assim todos eles podem indicar que o chão é preto ou não, ou quem sabe só um pouco claro.

1) Função:

Enfim, já está na hora de ensinarmos como você recebe esses valores de reflectância, não é mesmo? Para cada um dos 5 sensores, existe uma função diferente que o ativa. Ela nos retorna então um número inteiro, o qual devemos salvar numa variável inteira. As funções que ativam os sensores são: sparki.lineLeft(), sparki.lineRight(), sparki.lineCenter(), sparki.edgeLeft() e sparki.edgeRight().

Para não esquecer!

"Left" traduzido para o português significa "Esquerda", "Right" significa "Direita" e "Edge" significa "Borda ou Beira".

Caso estejamos interessados em saber se o Sparki está pisando em cima de uma linha ou não, podemos escrever o seguinte código (vamos considerar que uma reflectância abaixo de 300 significa que a cor é preta e que acima disso a cor é branca):

```
1  #include <Sparki.h>
2
3  int limiar = 300, esquerda, centro,
direita;
4
5  void setup()
6  {
7    sparki.clearLCD();
8    sparki.updateLCD();
9  }
10
11  void loop()
12  {
13    esquerda = sparki.lineLeft();
14    centro = sparki.lineCenter();
15    direita = sparki.lineRight();
16    delay(500);
17
18    if(esquerda < limiar || centro <
limiar || direita < limiar)
19    {
20      sparki.clearLCD();
21      sparki.print("O Sparki está em
cima de uma linha");
22      sparki.updateLCD();
23    }
```

```
24    else
25    {
26      sparki.clearLCD();
27      sparki.print("O Sparki NÃO está em
cima de uma linha");
28      sparki.updateLCD();
29    }
30
31    delay(500);
32  }
```

Nesse caso, se qualquer um dos 3 sensores do meio identificar a cor preta, estamos assumindo que o Sparki está passando em cima de uma linha.

Calma lá! Mas afinal, por que esses sensores se chamam sensores infravermelhos? O que eles têm a ver com a luz infravermelha?

Opa, desculpa! A razão do nome é porque, junto com esses sensores, debaixo do Sparki, nós temos LEDs (diodos emissores de luz) que emitem raios infravermelhos que serão refletidos pelo chão, e são esses raios que os sensores irão utilizar para medir a reflectância do chão.