Arduino + Brino para a robótica educacional

Gabriel Rodrigues Pacheco, Giulia Fricke Galice, Mateus Berardo de Souza Terra, Rafael Mascarenhas Dal Moro, Victor Rodrigues Pacheco.

Brasília, 2016

**Índice**

1. Introdução ...................................................................................................... 3
2. Começando .................................................................................................... 4

**Introdução**

A tecnologia está presente em todos os aspectos da vida humana, desde rastreadores de sono e outras tecnologias vestíveis até ferramentas educacionais e os mais tradicionais computadores e smartphones. Dentro desse contexto, o desenvolvimento de habilidades especificas se torna cada vez mais importante, mesmo para quem não trabalha na área de TI, a ponto de saber uma linguagem de programação ser tão fundamental quanto o inglês daqui poucos anos.

            Uma linguagem de programação é uma série de instruções, com regras sintáticas e semânticas, que são executadas por um processador. Praticamente todas as linguagens existentes são feitas utilizando o inglês como base. Observando a dificuldade de jovens e de quem não sabe inglês nós criamos uma linguagem de programação em português para o Arduino, o Brino, linguagem que almeja facilitar o entendimento da lógica de programa, sendo caracterizada como linguagem intuitiva e de simples compreensão.

O Arduino é uma placa com um microcontrolador que pode ser utilizada para prototipagem de sistemas e máquinas de forma fácil e rápida. Não apenas ao hardware, mas também um conjunto de software se refere o nome. A placa é baseada em um processador Atmel AVR e é feita em hardware livre. Portanto, é possível acessar a página do arduino e baixar os esquemáticos da placa para montar o seu próprio clone.

O arduino é atualmente utilizado em diversas áreas. Não apenas no ramo da tecnologia e desenvolvimento, pessoas adeptas do movimento maker (Faça você mesmo, do inglês DIY - Do it yourself) adotam esta placa pela curva de aprendizagem relativamente pequena. Além de hobbistas, artistas estão aproveitando as possibilidades da placa para construir obras interativas e responsivas.

A versatilidade da plataforma fez com que ela se tornasse muito popular. Nosso primeiro contato com o Arduino aconteceu no ano de 2014. Desde lá, desenvolvemos diversos projetos desde alguns voltados para IoT (Internet of things - internet das coisas), até alguns jogos e robôs autônomos. Para aprender utilizamos alguns livros e muita internet! O próprio fórum do arduino é um ótimo local para buscar auxílio assim como outros pela WEB. Sites como o instructables também é uma mina de ouro para quem busca um projeto interessante e não sabe por onde começar.

Os projetos disponíveis na internet são programados utilizando a própria linguagem do Arduino, se você está utilizando o Brino pode buscar nosso auxílio por e-mail, na página do facebook ou até utilizar o código nativo do arduino em seu rascunho (o Brino oferece suporte ao código nativo do Arduino). Tudo bem, você já leu uma folha inteira de teoria sobre o que é o Arduino, linguagens de programação, etc, etc... Se você, como nós, tem um espírito maker, já está de saco cheio de texto e quer começar a desenvolver seu projeto e fazer seu arduino ser útil, então vamos pôr a mão na massa! Antes disso, uma pequena dica: não seja levado completamente por sua impulsividade, aproveite um tempo antes de cada projeto para elaborar um bom planejamento e avaliar as possibilidades, isso evitará que você perca tempo e dinheiro.

**Começando**

Para começar você vai precisar, claro, de um Arduino. Seja um original ou um clone, a placa que você preferir provavelmente funcionará! Nessa apostila utilizaremos o arduino nano como referência. Recomendamos que o resto dos componentes seja adquirido quando você começar a desenvolver o projeto do capítulo, comprar todos juntos pode ser custoso. Algumas instruções para habilidades necessárias para o desenvolvimento de alguns projetos podem ser encontradas no final do livro, como confecção de placas de circuito impresso e soldagem.

Primeiro, vamos configurar o seu computador para que você possa programar sua placa. Entre no site do arduino [arduino.cc] e vá na aba downloads para baixar a IDE mais recente que na ocasião era a versão 1.6.7. Siga as imagens para completar a instalação:

//Imagens aqui

Depois de instalar a IDE do arduino, você está pronto para instalar o Brino. Para isso, basta acessar a página do GitHub: ratosdepc.github.io/Brino ; Na área de primeiros passos você encontra o link para a página de releases e os nossos contatos no final. Na página de releases, baixe o zip da versão mais recente e descompacte-o em algum lugar (de preferência que você possa encontrar facilmente). Depois disso, basta abrir o arquivo BrinoIDE.exe e você estará pronto para começar! E se quiser consultar algum código de exemplo, eles podem ser facilmente encontrados na pasta exemplos.

**Nota:** Quando for compilar algum código, não se esqueça de selecionar a placa com que você está trabalhando na aba de ferramentas/placa.

Depois de instalar tudo você pode começar a desenvolver o primeiro projeto, “Piscar”. Para esse projeto o programa é extremamente simples e não é necessária nenhuma experiência prévia, nem componentes além da sua placa arduino, pois ela já contém um LED que pode ser controlado. Entretanto, se você nunca programou antes, sugerimos que você leia o capítulo de Introdução à Programação, localizado na área de habilidades adicionais. Caso já tenha programado vale a pena ler os tipos de variáveis da linguagem Brino, que vamos utilizar, e as funções e métodos obrigatórios em todo rascunho.

**Projetos com o Arduino**

1. **Piscar:**

Neste capítulo trabalharemos o mais simples dos projetos de robótica: piscar um LED (como se o título não fosse autoexplicativo...). Mesmo simples, este projeto é extremamente importante, pois trabalha o básico de saídas do Arduino, estrutura fundamental de um rascunho Brino, que nada mais é do que o programa que sua placa executará, e como utilizar a IDE.

Antes de começar, vamos listar o material necessário para este projeto:

* Protoboard
* LED
* Resistor de 470 ohms (amarelo, lilás, marrom)
* Jumpers
* Arduino

Durante o capítulo, montaremos o projeto de duas formas: com o led externo e com o já soldado no Arduino. Para começar então, utilizaremos o led interno.

**O Código**

Abra a IDE do arduino e digite o código a seguir:

|  |
| --- |
| // Projeto 1 – Piscar  Numero pinoLed = 13;  Configuracao(){  Pino.definirModo(pinoLed, Saida);  }  Principal(){  Pino.ligar(Digital.pinoLed);  esperar(2000);  Pino.desligar(Digital.pinoLed);  esperar(2000);  } |

Depois de digitar, conecte seu arduino ao computador e verifique a porta serial que ele está conectado. Abra as ferramentas do IDE e selecione a porta e a placa que você está utilizando. Depois de selecionar, clique em Verificar e Carregar. Se uma janela aparecer solicitando o nome do rascunho, insira “Piscar”, clique em “OK” e espere até o log do IDE (área embaixo do editor) mostrar que o rascunho foi compilado e carregado. Caso ocorra qualquer erro, verifique o seu código, a placa e a porta serial selecionadas. Depois de carregado, observe o LED da sua placa acender por dois segundos e depois desligar também por dois segundos.

**Analisando o código**

A primeira linha do código é:

*//Projeto 1 - Piscar*

Essa linha é apenas um comentário com o nome do projeto.

A linha seguinte é:

*Numero pinoLed = 13;*

Essa linha cria uma variável do tipo Número com o nome pinoLed para armazenar o número do pino em que conectamos o LED.

Logo depois, declaramos a função Configuraçao():

*Configuracao(){*

*Pino.definirModo(pinoLed, Saida);*

*}*

A função de configuração é um dos métodos obrigatórios a todo rascunho Brino e é responsável por preparar o necessário para a execução da função Principal(). Neste caso, ela possui apenas uma instrução em seu bloco de código:

*Pino.definirModo(pinoLed, Saida);*

Essa linha define o modo do *pinoLed*, que possui o valor 13, como saída, ou seja, o arduino irá controlar esse pino como uma saída e não uma entrada. O método *definirModo* do conjunto *Pino* tem como argumentos o número do pino e o modo, que pode ser *Entrada* ou, como neste caso, *Saida*. Depois de executar a configuração o arduino inicia o método *Principal()* que no nosso caso é:

*Principal(){*

*Pino.ligar(Digital.pinoLed);*

*esperar(2000);*

*Pino.desligar(Digital.pinoLed);*

*esperar(2000);*

*}*

A função principal é o segundo método obrigatório a todo rascunho Brino e será repetido enquanto o arduino estiver ligado. A primeira linha dele é:

*Pino.ligar(Digital.pinoLed);*

O método *ligar* do conjunto *Pino* liga o pino fornecido como argumento, no caso *Digital.pinoLed.* A indicação *Digital.* Antes de nossa variável avisa para o arduino que estamos utilizando um pino digital. Quando ligamos uma porta enviamos 5 volts pra ela e quando desligamos enviamos 0 volts. A linha seguinte é:

*esperar(2000);*

A função esperar é um método arduino que faz uma “pausa” na execução do código durante o número de milissegundos indicados entre os parênteses, no caso 2000 que equivale a 2 segundos. Depois de esperar o tempo definido o arduino executa a próxima linha:

*Pino.desligar(Digital.pinoLed);*

Se a função *ligar* fazia com que o pino ligasse, o método *desligar* faz o contrário, desliga o pino. Ou seja, essa linha irá enviar 0V pelo pino *pinoLed* fazendo o LED apagar. Depois disso o código tem outra espera de dois segundos e reinicia o método *Principal()*, repetindo-o enquanto o arduino permanecer ligado.

Agora montaremos o hardware para ligar um LED externo utilizando a protoboard. Para isso basta montar o circuito abaixo, sob ele é possível ver também o diagrama ou esquemático das ligações:

//inserir circuito do fritzing e diagrama do proteus

**Introdução à Programação**

Para criar nossos próprios códigos é necessário aprender palavras-chaves e suas estruturas. Assim como um texto em língua Portuguesa ou em inglês, programas de computador utilizam linguagens com regras específicas de sintaxe para que o computador possa compreender o que queremos que ele faça.

Nesse capitulo abordaremos assuntos como:

* Variáveis;
* Comentários;
* Incrementadores;
* Instruções e laços de controle;
* Operadores lógicos;
* Funções ou Métodos.

1. **Variáveis:**

Variáveis são muito usadas na programação pois elas são capazes de armazenar dados. Uma das formas mais simples de pensar em uma variável é como uma caixa ou balde onde o computador pode armazenar ou ler dados. O uso de variáveis torna o código mais fácil de se entender e mais fácil de ser mantido. Por exemplo, se você possuir uma variável chamada *pinoLED* que define o pino 13 como uma saída para um LED e, mais a frente, decidir usar o pino 8 para isso, será fácil a troca sendo necessária apenas a troca do valor 13 pelo valor 8.

Esse exemplo poderia ser aplicado no segmento a seguir:

|  |
| --- |
| Numero pinoLED = 13 <= Aqui está a nossa variável!  Configuracao() {  Pino.definirModo(pinoLED, Saida);  }  Principal() {  Pino.ligar(pinoLED); // liga a porta digital acedendo o LED  esperar(1000); // espera por um segundo  Pino.desligar(pinoLED); // desliga a porta digital apagando o LED  esperar(1000); // espera por um segundo  } |

Variáveis podem ser locais ou globais, dependendo da parte do código em que ela for declarada. As locais são as que são declaradas dentro de uma função (Principal(), para(), if(), etc.), as globais são declaradas fora delas. As variáveis locais só podem ser usadas dentro de suas funções enquanto as globais podem ser usadas em qualquer parte do programa.

**Nota:** É possível criar duas variáveis locais com o mesmo nome em funções diferentes, mas tome cuidado para não se esquecer que elas serão independentes.

Agora que entendemos qual a utilidade das variáveis em nossos programas vamos examinar os seus tipos:

* Numero: As variáveis do tipo Numero (deve ser escrito dessa forma para ser entendido pelo compilador, sem acento e com letra maiúscula. Isso se aplica a diversas palavras que serão abordadas no capítulo!) são muito usadas pois elas são capazes de, como o próprio nome sugere, armazenar números inteiros entre

-32.768 a 32.767, ou seja, um numero de 16 bits.

Ex.: Numero minhaVariavel = 3600;

* NumeroDecimal: Esse tipo de variável é capaz de armazenar números de até 32 bits com um ponto decimal.

Ex.: NumeroDecimal raio = 3,5;

* Letra: Essa variável armazena um caractere ASCII (iremos abordar isso mais a frente), ou seja, ela é capaz de armazenar qualquer caractere (a, A, 1, &, entre outros). Operações aritméticas podem ser aplicadas sobre esses dados (também estudaremos isso mais a fundo). Seu dado deve vir entre aspas simples (‘ ‘).

Ex.: Letra nota = ‘A’;

* Palavra: Esse tipo especial de variável pode ser comparada a uma serie de caracteres. Ela é usada para armazenar palavras e frases. Seu dado deve vir entre aspas simples (‘’).

Ex.: Palavra saudacao = ‘oi’;

* Condicao: A menor variável que vamos estudar é usada para guardar apenas dois valores, Verdadeiro ou Falso, e será muito usada em operações logicas e como controle.

Ex.: Condicao chovendo = Falso;

Outro ponto importante quanto as variáveis são os nomes que elas podem receber, que pode ser quase qualquer coisa, desde que não seja nenhuma palavra palavra-chave. Mesmo com toda essa liberdade recomendamos fortemente que não usem acentos ou caracteres especiais, como ç, @, “ e coisas do tipo, e que seja dada a preferência a nomes sugestivos (não, uhsefiuhfwiu não é nada sugestivo) para que o código possa ser mais facilmente entendido pelos outros e por você mesmo. São exemplos de bons nomes: valorSensor, motorDireito, portaLED e leituraDistancia.

Caso se deseje trabalhar com uma constante a palavra-chave Constante pode ser adicionada ao início da variável, tornando-a invariável.

Ex.: Constante Numero minhaVariavel = 3600;

**Nota:** Com os tipos de números inteiros, você pode obter uma situação chamada *roll over*, em que um valor é somado ou subtraído extrapolando os limites da variável fazendo com que o fim de um intervalo role para outra extremidade. Por exemplo, temos uma variável do tipo Numero com o valor armazenado de 32.767 e somamos 1 a ela. O valor resultante dessa operação será de -32.768 e não 32.768.

1. **Comentários:**

Comentários são um recurso muito utilizados na programação que ajuda no entendimento do código por meio de “notas do autor” e uma ótima forma de manter o código organizado para o entendimento do próprio autor e de todos . Eles são ignorados na hora da compilação, ou seja, não são lidos pela máquina, apenas pelos desenvolvedores. É muito importante deixar notas pelo código explicando o que está sendo feito e qual o seu objetivo com cada bloco. Eles são importantes, principalmente, para algoritmos complexos e extensos. Imagine procurar um bug, ou consertar um erro em um código com 500 linhas (não é uma marca muito difícil de ser alcançada) sem ter a menor ideia de onde procurar...

No caso do Brino os comentários podem ser introduzidos por meio de uma barra dupla na frente da linha (//) fazendo com que essa linha em especifico seja um comentário. Outra forma de usar essa ferramenta é pelo uso de uma barra acompanhada por um asterisco (/\*) abrindo um bloco de comentário que só terminará em um asterisco seguido por uma barra (\*/).

Obs.: evite colocar acentos em um comentário e em todo o código no geral. Ao abrir o código em diferentes editores de texto, eles podem não entender um caractere acentuado da mesma forma. De forma simples, nem todos os editores de texto “escrevem” na mesma língua, por isso entendem os acentos de forma diferente.

Ex.: // Isso e um comentario.

Ex. 2: /\*

\* Isso e um comentario de bloco

\*/

1. **Incrementadores:**

No Brino existem alguns operadores que podem incrementar ou decrementar o valor deuma variável. Além daqueles com que estamos acostumados (+, -, \*, /), existem outros especiais. São eles:

|  |  |  |
| --- | --- | --- |
| Operador | Exemplo | Resultado |
| ++ | Variável++ | O valor da variável vai ser incrementado em uma unidade |
| -- | Variável-- | O valor da variável vai ser decrementado em uma unidade |
| += | Variável += n | O valor da variável será incrementado em n unidades(no caso de Palavras, o trecho n será adicionado ao final) |
| -= | Variável -= n | O valor da variável será decrementado em n unidades |
| \*= | Variável \*= n | O valor da variável será igual ao produto do seu antigo valor e n |
| /= | Variável /= n | O valor da variável será igual ao quociente do seu antigo valor e n |

**Nota:** Quando estamos falando de programação os símbolos da divisão e da multiplicação costumam ser substituídos por uma barra simples (/) e por um asterisco (\*) respectivamente.

**Nota:** Quando atribuímos o valor por meio de uma igualdade (=), o valor da direita é atribuído ao lado esquerdo.

Ex.: X = 2 \* 5

1. **Instruções e laços de controle:**

Assim como nós, as maquinas são capazes de tomarem decisões, porém de maneira muito mais simples. Os códigos fazem decisões por meio das chamadas operações booleanas. Para essas operações podem existir apenas dois resultados. Uma forma de pensar quanto a isso é usando ideias do cotidiano. Por exemplo, se estiver chovendo devo pegar o guarda-chuva, caso contrário, não preciso pegá-lo. Nesse caso a expressão booleana seria o estado de estar chovendo ou não e a instrução de controle seria pegar ou não o guarda-chuva.

Uma forma muito usada para representar essas situações é por meio de diagramas de blocos como o representado a seguir:

Agora que temos uma noção do que se trata, podemos estudar os Operadores Relacionais. Eles nada mais são do que comparadores que usaremos para analisar informações tendo uma saída de Verdadeiro ou Falso. Os operadores estão representados na tabela a seguir:

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Descrição | Exemplo | Resultado |
| > | Maior que | 2 > 1 | Verdadeiro |
| >= | Maior ou igual a | 2 >= 2 | Verdadeiro |
| < | Menor que | 2 < 1 | Falso |
| <= | Menor ou igual a | 2 <= 2 | Verdadeiro |
| == | Igualdade | A == A | Verdadeiro |
| != | Desigualdade | A != A | Falso |

Assim como podemos observar os operadores não estão limitados a comparar apenas números, mas também podem comparar Letras e outras variáveis.

**Nota:** Tome cuidado!!! Não se esqueça que o símbolo = faz uma atribuição sendo diferente do == que faz uma comparação.

Tendo uma ideia de como obter resultados Verdadeiros e Falsos podemos desenvolver um pouco mais a ideia usando os operadores condicionais. Esses são os operadores que vamos usar para tomar decisões quando associados aos comparadores. Veremos a seguir como usar os operadores se, senao e o senao se.

Eles seguem as estruturas exemplificada abaixo:

|  |
| --- |
| se(expressão booleana){  // Se a expressão for Verdadeira esse bloco será executado.  }  // Se ela for Falsa o bloco será ignorado. |

|  |
| --- |
| se(expressão booleana){  // Se a expressão for Verdadeira esse bloco será executado.  }  senao{  // Se ela for Falsa esse bloco será executado.  } |

|  |
| --- |
| se(expressão booleana){  // Se a expressão for Verdadeira esse bloco será executado.  }  senao se(outra expressão booleana){  // Se a segunda expressão for Verdadeira  // esse bloco será executado.  }  senao{  // Se nenhuma dos blocos anteriores forem executados,  // esse será.  } |

Além do *se* e do *senao,* existe a instrução enquanto(). O laço enquanto(), como o próprio nome sugere, executa um bloco de código enquanto uma condição, entre os seus parênteses, for verdadeira. Ele é usado para realizar um processo enquanto for necessário, enquanto a expressão for verdadeira, e para quando não for mais necessário, a expressão se tornar falsa.

|  |
| --- |
| Enquanto(Condicao){  // Esse bloco é repetido enquanto a condição for verdadeira.  } |

O laço para() é usado para repetir um determinado bloco de código um número determinado de vezes usando, para isso, uma variável como contador. Ao contrário dos outros anteriormente citados, o para() aceita mais parâmetros. O primeiro é a declaração das variáveis locais, sendo seguido pela expressa booleana e por uma expressão de incremento ou decremento do valor da variável.

|  |
| --- |
| para(Tipo <nome> = <valor>; <nome> <operador> <valorReferencia>; <incremento>){  // Bloco que será repetido.  } |

Ex.:

|  |
| --- |
| para(Numero x = 0; x <= 10; x++){  ...;  } |

1. **Operadores lógicos:**

Os operadores lógicos são usados quando uma expressão booleana não é o suficiente para a tomada de decisões, então, por meio deles, nós podemos ter mais do que uma expressão booleana com apenas uma saída. Um jeito interessante de pensar nesses problemas é voltando ao exemplo do guarda-chuva. Eu estou saindo de casa, se estiver ensolarado eu não irei pegar o guarda-chuva, mas caso esteja chovendo ou pareça que vai chover, devo pega-lo. A seguir estão representados os operadores:

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Descrição | Exemplo | Resultado |
| e | Se ambas forem verdadeiras | 2 > 1 e 2>=2 | Verdadeiro |
| ou | Se uma das duas for verdadeira | 2 >= 2 ou 1>3 | Verdadeiro |

Repare nos exemplos, para saber a saída, devemos fazer essa operação por etapas. No caso 2 > 1 e 2>=2 podemos observar duas expressões booleanas. Como já sabemos resolve-las podemos dizer que o problema fica Verdadeiro e Verdadeiro. Podemos observar que, ambas as expressões tem uma saída verdadeira e estão ligadas pelo operador *e*, logo o resultado será Verdadeiro.

1. **Funções ou Métodos:**

Funções ou métodos consistem basicamente em um determinado bloco de código escrito pelo desenvolvedor para evitar repetir um conjunto de instruções. O Brino requer o uso de pelo menos duas funções. Essa ferramenta é extremamente importante para reduzir o tamanho dos códigos e otimizar a utilização da memória do computador. Para se declarar uma função, é necessário dizer que tipo de dado ela nos retornará, ou seja, qual a resposta que ela nos fornece após efetuar todas as suas instruções, que pode ser qualquer tipo de variável ou *SemRetorno* se ela não responde nada. Além do retorno, precisamos declarar os dados, ou argumentos, que ela receberá para efetuar suas operações. A declaração de uma função segue o modelo:

|  |
| --- |
| TipoDeRetorno <nome>(TipoDeVariável <nome\_argumento\_1>, ..., argumento n){  // Bloco de instruções da função ou método.  } |

O Brino possui duas funções obrigatórias que iremos debater:

* Configuraçao():

Executada uma vez quando o arduino é inicializado. É responsável por preparar o hardware para a execução do loop principal.O método de Configuracao() é executado uma única vez quando o Arduino é ligado e é ignorado ate que seja reiniciado. É nele que definimos a conexão com a porta USB, os modos de operação dos pinos, entre outros parâmetros.

|  |
| --- |
| Configuracao(){  // Esse bloco é repetido apenas uma vez na inicialização.  } |

* Principal():

O método Principal() é um dos mais usados para se colocar a parte principal do programa. Ele é executado a partir do momento que o Arduino é ligado, após a execução da configuração, até o momento em que ele é desligado podendo ser repetido incontáveis vezes. Nele colocamos todas as operações necessárias para o funcionamento contínuo do projeto.

|  |
| --- |
| Principal(){  // Esse bloco é repetido continuamente.  } |

**Introdução a eletrônica básica**

Uma parte muito importante da robótica é a parte da eletrônica pois ela que é capaz de interagir com o meio externo ao robô. É por meio de atuadores (motores e servomotores) e sensores que ele será capaz de analisar e se locomover interagindo com o ambiente a sua volta. Esse é um assunto muito amplo e que pode requerer muitos cálculos, mas como esse material possui apenas caráter introdutório iremos abordar apenas as partes realmente necessárias como:

* Resistores;
* Interruptores;
* Capacitores;
* Diodos;
* LEDs.

1. **Resistores:**
2. **Interruptores:**
3. **Capacitores:**
4. **Diodos:**
5. **LEDs:**

**Materiais importantes**

//escrever aqui sobre coisas importantes q usados como jumpers, protoboard, ... (com fotos).

**Tabelas importantes**

1. **Valor resistores:**

A unidade de medida de resistência é o ohm, representado pelo símbolo grego Ômega. Para saber a resistência de um resistor basta ler as duas primeiras faixas e multiplicar esse valor pela terceira. A quarta mostra a tolerância do resistor que é o valor de sua variação em relação ao valor nominal.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cor | Valor | Multiplicador | Cor quarta faixa | Tolerância |
| Dourado | - | 0.1 | Prata | 10% |
| Preto | 0 | 1 | Ouro | 5% |
| Marrom | 1 | 10 | Amarelo | 4% |
| Vermelho | 2 | 100 | Laranja | 3% |
| Laranja | 3 | 1.000 | Vermelho | 2% |
| Amarelo | 4 | 10.000 | Marrom | 1% |
| Verde | 5 | 100.000 | - | - |
| Azul | 6 | 1000.000 | - | - |
| Violeta | 7 | Não existe | - | - |
| Cinza | 8 | Não existe | - | - |
| Branco | 9 | Não existe | - | - |

1. **ASCII:**

A memória do computador não é capaz de armazenar diretamente caracteres, tendo que os armazenas na forma de números. Cada caracter possui por conseguinte o seu equivalente em código numérico: é o **código ASCII** (*American Standard Code for Information Interchange* - Código Americano Standard para a Troca de Informações). Existem versões estendidas desse código, mas aqui trataremos da sua versão básica que possui 7 bits, ou seja, possui 128 caracteres.

Nessa tabela o código 0 a 31 não são realmente caracteres, sendo chamados de *caracteres de controle*. Os códigos 65 a 90 representam as letras maiúsculas e os códigos 97 a 122 representam as letras minúsculas. O codigo 32 representa o espaço. Infelizmente pelo tamanho dessa tabela nos iremos reprsentar apenas parte dela, mas ela pode ser facilmente encontrada na internet.

**Nota:** Bastar somar ou subtrair 32 ao código ASCII para trocar entre as letras maiúsculas e minúsculas. Isso representa a troca do 6° bit da representaçao binaria.

//Adicionar tabela (ou uma imagem dela q ocupe menos espaço).