Ministério da Educação
Universidade Tecnológica Federal do Paraná
Programa de Pós-Graduação em Formação Educacional, Científica e Tecnológica
Metodologia de Pesquisa em Ensino de Ciências e Matemática

Orientador: Professor Doutor Álvaro Emílio Leite

Material de Apoio Didático a ser aplicado no CE. Humberto de Alencar Castelo Branco – Pinhais Professor da Disciplina de Física e estudante do PPGFCET: Guido Valmor Buss

Introdução ao estudo de lógica e linguagem de programação para estudantes do Ensino Médio na disciplina de Física, com enfoque na aplicação de projetos com Arduino.

Dezembro de 2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

GUIDO VALMOR BUSS

PROGRAMAÇÃO E FÍSICA: POSSIBILIDADES DO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL UTILIZANDO O ARDUINO

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Ensino De Ciências E Matemática da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Ensino, Aprendizagem E Mediações.

Data de aprovação: 16 de dezembro de 2021

Prof Álvaro Emilio Leite, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Alisson Antonio Martins, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Marcos Rocha, Doutorado - Secretaria de Educação do Estado do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 16/12/2021.

Sumário

INTRODUÇÃO

Estando sempre na busca por novas ferramentas que auxiliem o trabalho em sala de aula e que proporcionem aos estudantes um dinamismo e aprofundamento em novas oportunidades de contato com experimentos relacionados à tecnologia da informação e comunicação, este material didático de apoio tem o objetivo de trazer uma inovação à escola com aulas diferenciadas de informática aos estudantes do ensino médio, mostrando o valor introduzido pelo conhecimento do pensamento computacional que será aplicado com uma linguagem de programação, neste caso a linguagem “C”, aliada ao entusiasmo dos estudantes em querer conhecer e trabalhar com a robótica. Será usada a tecnologia criada pelo professor italiano da disciplina de design de interação, Massimo Banzi, que começa por volta de 2002 no *Interaction Design Institute Ivrea (IDII)* – uma antiga escola de design italiana. Ele precisava encontrar uma maneira fácil e barata para que os seus estudantes, na maioria designers e artistas, pudessem criar dispositivos eletrônicos que reagissem fisicamente conforme fossem estimulados. O que ele e seus colegas criaram revolucionaram o universo *maker,* criaram a placa de prototipagem chamada Arduino, um componente tecnológico que oferecia uma maior relação entre humanos e máquinas, assim surgiu uma resposta à questão de como permitir que estudantes sem conhecimento técnico em eletrônica pudessem criar artefatos tecnológicos. A utilização da placa de prototipagem Arduino permite neste processo de aprendizagem, dar inteligência às coisas, onde isto pode ser propiciado através de sistemas computacionais interativos que sejam capazes de perceber eventos do mundo físico com a utilização de sensores e também de reagir a eles por meio de atuadores.

Com este pensamento computacional instaurado nas aulas iniciais, a lógica será também utilizada para testar e resolver questões e problemas de Física dos mais variados conteúdos através da linguagem de programação em “C”, veremos como ocorre o comportamento de aprendizagem neste novo ambiente, como será desenvolvido o maior interesse pelos estudos na disciplina de Física e se isto pode ter maior amplitude avançando nas outras disciplinas, como Matemática, Química e Biologia.

1 Lógica de programação:

A lógica de programação serve para desenvolver uma série de sequências lógicas onde é utilizada uma técnica para desenvolver algoritmos, onde estes algoritmos são compostos por uma sequência de instruções executadas até que uma determinada condição seja alcançada. Atingir determinados objetivos dentro de certas regras baseadas na lógica matemática e outras teorias básicas da Ciência da Computação e que depois são adaptadas para a Linguagem de Programação utilizada pelo estudante, como por exemplo, desenvolver um algoritmo que seja capaz de calcular o número PI com 30 algarismos, outro que seja utilizado para calcular o fatorial de um número natural. Em aplicações na Física, por exemplo, utilizando-se das equações da Cinemática, criar um algoritmo para calcular a velocidade de um objeto acelerado; na Dinâmica, um algoritmo para calcular a resultante de forças e descobrir se estas forças que atuam sobre um corpo o fazem permanecer em repouso ou em movimento; na Eletrostática, criar um algoritmo que calcule Força de Interação de Coulomb entre duas partículas eletricamente carregadas. Estes exemplos são alguns dos exercícios que serão propostos e desenvolvidos pelos estudantes assim estudando inicialmente a lógica de programação.

1.1 - Exemplos de Lógica de programação:

1.1.1 - Estrutura e algoritmo para somar dois inteiros e mostrar na tela o valor:

Para iniciar o estudo de lógica de programação começaremos com um algoritmo simples que soma dois valores **a** e **b** e guarda estes valores em um terceiro chamado **soma**, depois mostra o resultado com a função imprime:

```
1.  // declaração de variáveis
2.  inteiro a;
3.  inteiro b;
4.  inteiro soma;
5.
6.  // início do algoritmo
7.
8.  início
9.      a=10;
10.     b=20;
```

```
11.     soma = a+b;
12.     imprime(soma);
13. fim
```

No começo deste exemplo, são declaradas as variáveis e os **tipos de variáveis** como inteiros, quando for iniciado o assunto de Linguagem de programação em “C”, veremos os outros tipos de variáveis que serão utilizadas nos códigos fontes em “C” e para a programação que será elaborada para a placa Arduino. **Comentários** durante o programa são muito úteis, isso ocorre na linha 1 e na linha 6, quando usa-se barras duplas “//”, isto indica que o interpretador da linguagem de programação irá pular esta linha, sabendo que não corresponde a uma linha de comando, os comentários podem ser inseridos em qualquer linha ou até mesmo ao final de cada linha logo após o ponto-e-vírgula, mas nunca no meio de uma linha de comando. Caso faça isto, tudo após a barra dupla será interpretado como um comentário.

Logo após a declaração de variáveis temos que dizer quando começa efetivamente o programa e a atribuição dos valores, isso é realizado na linha 8. Em todas as linguagens de programação isto ocorre, deve-se dizer quando será iniciada a declaração de uma função ou o próprio corpo principal do programa. Em muitos casos temos as palavras **begin** e **end** para indicar o começo e o fim da parte principal de um programa ou os símbolos matemáticos de chaves, no caso da programação para Arduino que seria **{** e **}** indicando o começo e o fim do programa. Outro cuidado a ser tomado é o ponto-e-vírgula ao final de cada linha, indicando o final do comando daquela linha e que uma nova linha será caracterizada por outro comando.

1.1.2 - Estrutura e algoritmo para calcular as raízes de uma equação de segundo grau:

Um exemplo bem prático de matemática utilizado com certa frequência em muitas situações nas disciplinas de exatas, a equação que será utilizada será **$y = 2.x^2 + 3.x + 5$**.

Lembrando que esta equação fornece os valores de **a = 2**, **b = 3** e **c = 5**, mais para frente não teremos valores fixos e sim valores que podem ser inseridos como entrada do algoritmo ou programa que esteja elaborando, sendo isto possível com comandos como **scanf()** em “C” e comandos de saída ou

impressão na tela como **printf()** que no exemplo anterior foi dado como imprime();

```
1.  // declaração de variáveis
2.  inteiro a;
3.  inteiro b;
4.  inteiro c;
5.  inteiro delta;
6.  inteiro x1,x2;
7.
8.  // início do algoritmo
9.  início
10.     a=2;
11.     b=3;
12.     c=5;
13.     delta = b*b - 4*a*c;        //  em matemática seria b² - 4ac
14.     x1 = (-b+raiz(delta))/(2*a);
15.     x2 = (-b-raiz(delta))/(2*a);
16.     imprime("o valor de x1 = ",x1);
17.     imprime("o valor de x2 = ",x2);
18. fim
```

Inicia-se novamente na linha 1 com um comentário com as barras duplas, logo após com a declaração de variáveis a, b, c, delta, x1 e x2 como sendo inteiros, neste caso x1 e x2 são os resultados do cálculo das raízes da equação de segundo grau. Na linha 9 inicia-se o corpo principal do algoritmo, note também que neste caso a partir da linha 10 usa-se **tabulação**, é uma técnica de programação que serve para visualizar que neste espaço está sendo executado o escopo principal do algoritmo. Este método é considerado como padrão tanto para elaboração de algoritmos como para programação em Linguagem “C”. Nas linhas 10,11 e 12 atribuem-se os valores das variáveis a=2, b=3 e c=5, como foi estipulado pela equação **y = 2.x² + 3.x + 5** e logo abaixo na linha 13 é calculado o valor de delta da equação de segundo grau que possui cálculo delta = b² - 4ac, o valor de b² é representado como sendo b*b, o asterisco (*) representa a multiplicação e a barra (/) a divisão. Nas linhas 14 e 15 foram calculadas as raízes da equação, tomando cuidado com os parênteses, sempre deve-se pensar nas equações como um conjunto de operações e separadas pelos parênteses. Por exemplo, caso **não** tivesse sido posto parênteses logo após a divisão por (2*a), a equação ficaria modificada da seguinte maneira: $x1 =$

$\frac{(-b+sqrt(delta))}{2} * a$, muita atenção!. Aqui usou-se a função raiz quadrada como sendo sqrt() que significa square root, ou raiz quadrada em inglês, que pode ser encontrada desta forma mesmo com uma função em “C”. E por fim, mostrando os valores com imprime("o valor de x1 = ",x1); e imprime("o valor de x1 = ",x2); aqui usou-se aspas dentro dos parênteses, podendo ser usado para mostrar na tela de saída não somente o resultado, mas algum comentário ou indicação.

1.1.3 - Exemplo de algoritmo com entrada LEIA() escrita pelo usuário e com o comando de repetição PARA – FAÇA:

Existem casos onde o usuário deverá inserir valores para realizar um cálculo específico, chamamos de valores de entrada do algoritmo que será visto em seguida, isto pode ser realizado com o comando **LEIA**.

No exemplo que segue será visto o cálculo da média da altura de **n** pessoas, veremos também a estrutura de repetição **PARA – FAÇA**, que realiza um loop de repetição de comandos até que um argumento escolhido seja finalizado.

```
1.  // declaração das variáveis
2.  inteiro x,n;
3.  real altura[40];
4.  real soma;
5.  real media;
6.
7.  // início do algoritmo
8.  inicio
9.      soma = 0;
10.     imprime("Digite o número de pessoas: ");
11.     leia(n);
12.
13.     // entrada das alturas das pessoas em um comando de repetição
14.     para x=1 até n faça
15.     inicio
16.         imprime("Digite a altura da pessoa ",x);
17.         leia(altura[x]);
18.         soma = soma + altura[x];
19.     fim;
20.
21.     // cálculo da média
```

```
22.    media = soma / n;
23.    imprime("A média das alturas é de ",media:2);
24. fim
```

No início da declaração das variáveis, **x** será usado para a incrementação do comando de repetição PARA - FAÇA e **n** será usado para delimitar o **número** de pessoas que serão feitas as medidas das alturas. Se você medir 5 pessoas então n=5, por exemplo. Mas este algoritmo possui uma limitação, na linha 3 a variável altura[40] corresponde aos valores que serão guardados em uma matriz linear com apenas 40 espaços reservados na memória. Esta variável deve ser real pois as alturas serão números com vírgula, por exemplo ao ler a primeira altura será inserida em altura[1]=1.65, a segunda em altura[2]=1.78 e assim sucessivamente.

Passando para o início do algoritmo, vemos que a variável **soma** recebe o valor zero, isto serve para depois do comando de repetição termos o valor já somado de todas as alturas, por isto ele deve ser zerado anteriormente. Logo após do comando imprime na linha 10 o comando leia(n) faz a leitura do valor digitado pelo usuário na tela e guarda na variável **n**, que corresponde ao numero de pessoas a serem medidas para realizar o cálculo da média. Na linha 13 começa o comando de repetição PARA – FAÇA, que executa um loop de 1 até n (número de pessoas), para entender melhor, caso você faça a medida de 20 pessoas, teria que escrever 20 vezes:

```
imprime("Digite a altura da pessoa ",x);
leia(altura[x]);
```

Então o comando de repetição PARA – FAÇA irá executar isso até o número **n** de pessoas, inclusive já dentro deste comando de repetição está sendo realizada a soma das alturas, isto pode ser visto na linha 17. E por fim na linha 20 é calculada a média das alturas e na linha 21 mostrado na tela com dois dígitos após a vírgula com o comando **imprime("A média das alturas é de :",media:2);** Veja também mais a seguir o comando de repetição ENQUANTO – FAÇA.

1.1.4 – Exemplo de algoritmo com comando de condição SE:

Para que tenhamos uma condição sendo obedecida por um algoritmo usamos o comando SE, como exemplo a seguir, será lido um valor **numérico**

qualquer de entrada digitado pelo usuário e este valor será interpretado com os comandos de condição, imprimindo na tela uma frase de resposta.

```
1.  // declaração da variáveis
2.  inteiro x;
3.
4.  // início do algoritmo
5.  inicio
6.      imprime("Qual a sua faixa de idade ?");
7.      imprime("1 para faixa de 1 a 12 anos");
8.      imprime("2 para faixa de 13 a 20 anos");
9.      imprime("3 para faixa de 21 a 40 anos");
10.     imprime("4 para faixa de 40 a 80 anos");
11.     imprime("5 para acima de 80 anos");
12.     leia(n);
13.     se n=1 imprime("Sua faixa etária é de 1 a 12 anos");
14.     se n=2 imprime("Sua faixa etária é de 13 a 20 anos");
15.     se n=3 imprime("Sua faixa etária é de 21 a 40 anos");
16.     se n=4 imprime("Sua faixa etária é de 40 a 80 anos");
17.     se n=5 imprime("Sua faixa etária está acima de 80 anos");
18.     se (n<1 ou n>5) imprime(“Você digitou um número errado!”);
19. fim
```

O algoritmo acima verifica cada faixa etária e irá mostrar na tela somente o valor escolhido pelo usuário, isto é verificado nos comandos de repetição se a partir da linha 13 até a linha 18. Nesta linha, o algoritmo ainda verifica se o usuário digitou um valor válido, pois caso não esteja no intervalo de 1 a 5 neste caso estaria digitando um valor inválido. Isto se chama segurança de algoritmos e sempre em códigos seguros, o programador deverá saber como o algoritmo deverá se comportar caso o usuário faça alguma ação não prevista, como digitar um número errado que não está no intervalo especificado.

Neste algoritmo o comando de condição SE executa somente uma ação, que é imprimir uma frase, caso precise executar mais linhas de comando ou analisar outra condição usa-se o comando de condição SE – SENÃO, que será visto mais adiante quando entrarmos em linguagem de programação em “C”.

Agora dedique-se a realizar os exercícios a seguir pedindo auxílio ao professor em caso de dúvidas, tente seguir todas as orientações acima, declaração de variáveis, indentações das partes de início do programa e dentro

de um comando de repetição, escolhendo corretamente e escrevendo corretamente os comandos de repetição, de condição, de leitura e de escrita.

1.2 - Exercícios de Lógica de programação:

1) Faça um algoritmo que leia as 3 notas digitadas por um estudante e calcule a média final deste estudante. Sabendo que a média deve ser igual ou superior a 7,0 escreva na tela se ele passou ou não de ano;

2) Elaborar um algoritmo que lê 3 valores a,b,c e os escreve. A seguir, encontre o maior dos 3 valores e o escreva com a mensagem: "É o maior ";

3) Elaborar um algoritmo que escreva na tela as tabuadas do 5, 6 e 7;

4) Elaborar um algoritmo que leia um número digitado pelo usuário e que verifique e imprima na tela se é “POSITIVO” ou “NEGATIVO”, tomando cuidado com o valor 0 (zero);

5) Escrever um algoritmo que leia três números, positivos ou negativos, multiplique os dois primeiros, some com o terceiro e depois faça a divisão do resultado pelo primeiro;

6) Um vendedor necessita de um algoritmo que calcule o preço total devido por um cliente. O algoritmo deve receber o código de um produto e a quantidade comprada e calcular o preço total, usando a tabela abaixo:

Código do Produto	**Preço unitário**
1001	5,32
1324	6,45
6548	2,37
0987	5,32
7623	6,45

7) Faça uma função que recebe por parâmetro o raio de uma esfera e calcula o seu volume. $(V = \frac{4}{3} . \pi . R^3)$

8) Faça uma função que recebe um valor inteiro e verifica se o valor é par ou ímpar.

1.3 - Usando os algoritmos para resolver problemas na disciplina de Física:

Unidades de medida: Na Física, algumas grandezas desempenham um papel fundamental quando falamos em medidas. Praticamente todos os fenômenos naturais que percebemos são apresentados no tempo e no espaço a partis da matéria. Torna-se natural, então, que entre as grandezas consideradas fundamentais estejam: o tempo, o comprimento e a massa. A medida de qualquer quantidade é sempre feita em comparação com uma medida-padrão: a unidade de medida. O sistema então criado pela Academia de Ciências de Paris, em 1970, foi uma forma de padronização destas unidades de medida, denominado Sistema Métrico Decimal e que gradativamente passou a ser aceito em quase todo o mundo, adotava como unidades de medida o metro (m), o quilograma (kg) e o segundo (s). O sistema de unidades utilizado hoje em dia no Brasil e na maioria dos países é denominado Sistema Internacional de Unidades, abreviadamente **SI**, derivado do antigo Sistema Métrico Decimal, sendo composto por sete unidades de base (ou fundamentais), de duas unidades suplementares, de unidades derivadas e de múltiplos e submúltiplos de todas elas. As unidades base são: comprimento: metro (m), massa: quilograma (kg), tempo: segundo (s), corrente elétrica: ampère (A), temperatura termodinâmica: kelvin (K), quantidade de matéria: mol (mol), intensidade luminosa: candela (cd).

7) Sabendo que uma polegada vale 2,54 cm, escreva um algoritmo que identifique no início o tipo de transformação, se de polegada para cm ou cm para polegada e transforme o valor de acordo com a escolha do usuário;

8) Escreva um algoritmo que leia as horas, os minutos e os segundos digitados por um usuário, podendo ser a entrada cada uma separada, e depois transforme tudo para segundos;

9) Elabore um algoritmo que faça a soma das 4 medidas dos lados internos de uma sala e retorne imprimindo o perímetro interno desta sala, também imprima a área interna da sala;

10) Faça uma função que recebe a idade de uma pessoa em anos, meses e dias e retorna essa idade expressa em dias.

2 – Linguagem de programação em “C”:

A importância do aprendizado desde cedo em uma linguagem de programação propicia o conhecimento adequado de como um computador pode ser entendido corretamente, devido às operações diretas ou indiretas de acesso e alocação de memória por exemplo, utilizar a linguagem de programação C para apresentar e discutir as implementações e os usos de diferentes estruturas de dados. A linguagem C é ubíqua, estando presente em todas as plataformas computacionais, desde supercomputadores a processadores ARM embutidos. Por isso, escrever um código padrão C é escrever um código portátil; além disso, ganha-se em versatilidade, pois pode-se programar desde sistemas embarcados em hardware simples a sistemas computacionais complexos. A linguagem C permite fazer diferentes aplicações, escritas em diferentes linguagens, interoperarem entre si. A grande maioria das linguagens de programação proveem suporte à programação de interfaces em C. Sendo, no nosso caso, utilizada para podermos estudar e controlar uma placa de prototipagem nesta linguagem de programação, assim a programação em “C” permitirá entender melhor como podemos trabalhar com o Arduino e também com a maioria dos seus sensores e atuadores, como motores para montagem de robôs ou a construção de braços robóticos.

2.1 – Operadores em C:

Inicialmente faz-se necessário conhecer quais são as formas/símbolos que devem ser utilizadas na linguagem de programação em C para que possamos realizar desde as **simples operações matemáticas** de soma, subtração, multiplicação e divisão a comparadores como maior, menor ou resto de uma divisão. A tabela a seguir deve ser entendida, estudada e sempre consultada se possível para melhor aplicação dos operadores.

Operadores	Descrição	Associatividade

`++ --`
 	
`()`
`[]`
	Incremento e decremento pós-fixo
	Parênteses (chamada de função)
	Elemento de arranjo
	Seleção de elemento por
		esquerda para direita

. ->	identificador Seleção de elemento por ponteiro	
++ -- + - ! ~ (*tipo*) * & sizeof new [] delete []	Incremento e decremento prefixo Adição e subtração unária *Não* lógico e complemento Conversão de tipo de dado Desreferência Referência (endereço de elemento) tamanho de elemento Alocação dinâmica de memória Desalocação dinâmica de memória	direita para esquerda
* / %	Multiplicação, divisão, e módulo (resto)	
+ -	Adição e subtração	
<< >>	Deslocamento de bits à esquerda e à direita	
< <= > >=	“menor que” e “menor ou igual que” “maior que” e “maior ou igual que”	
== !=	“Igual a” e “diferente de ”	
&	*E* para bits	
^	*Ou exclusivo* para bits	
|	*Ou* para bits	
&&	*E* lógico	
||	*Ou* lógico	
= += -= *= /= %= <<= >>= &= ^= |=	Atribuição Atribuição por adição ou subtração Atribuição por multiplicação, divisão ou módulo (resto) Atribuição por deslocamento de bits Atribuição por operações lógicas	

De acordo com os exemplos em linguagem C nas sessões seguintes, alguns destes **operadores** estarão sendo explicados para que sejam melhor utilizados em cada caso. Nem todos serão utilizados neste curso, muitos podem ser vistos em cursos mais avançados, como cursos específicos para programadores ou em cursos universitários.

Operadores aritméticos	
Operador	**Sintaxe**
Adição unária	`+a`
Adição	`a + b`
Incremento pré-fixado	`++a`
Incremento pós-fixado	`a++`
Atribuição por adição	`a += b`
Subtração unária	`-a`
Subtração	`a - b`
Decremento pré-fixado	`--a`
Decremento pós-fixado	`a--`
Atribuição por subtração	`a -= b`
Multiplicação	`a * b`
Atribuição por multiplicação	`a *= b`
Divisão	`a / b`
Atribuição por divisão	`a /= b`
Módulo (resto)	`a % b`
Atribuição por módulo (resto)	`a %= b`

Operadores comparativos	
Operador	**Sintaxe**
Menor que	`a < b`
Menor ou igual que	`a <= b`
Maior que	`a > b`
Maior ou igual que	`a >= b`
Diferente de	`a != b`
Igual a	`a == b`
Não lógico	`!a`
E lógico	`a && b`
Ou lógico	`a || b`
Deslocamento à esquerda	`a << b`
Atribuição de deslocamento à esquerda	`a <<= b`
Deslocamento à direita	`a >> b`
Atribuição de deslocamento à direita	`a >>= b`

2.2 – Variáveis em C:

Em um programa, existe a necessidade de se guardar valores na memória, e isso é feito através de **variáveis**, que podem ser definidas simplificadamente como nomes que se referem a lugares na memória onde são guardados valores. Ao declararmos uma variável, não apenas estamos reservando um espaço de memória, como também estamos associando um nome a ele, o **identificador**.

Em C, para utilizar uma variável, ela deve ser primeiramente **declarada**, ou seja, devemos requisitar o espaço necessário para essa variável. Após reservar um espaço na memória, o computador irá associar a ele o nome da variável. Se você não declarar uma variável e tentar utilizá-la, o compilador irá avisá-lo disso e não continuará a compilação. Genericamente, para declarar uma variável, usamos a seguinte instrução:

tipo_da_variável nome_da_variável;

Por exemplo, para declarar uma variável do tipo int com o nome *a*, podemos escrever

int a;

É sempre necessário indicar o tipo da variável, pois cada tipo tem um tamanho diferente, ou seja, ocupa mais ou menos espaço na memória do

computador. Se quisermos associar um valor a uma variável, usamos o operador = (igual):

a = 5;

Nesse caso, estamos pedindo que o computador guarde o valor 5 no espaço alocado à variável *a*.

Importante: Apesar de este operador se assemelhar ao igual da matemática, sua função é diferente. Para verificar a igualdade de dois valores, usamos o operador de comparação "==" (dois iguais). É possível também atribuir um valor a uma variável ao mesmo tempo que ela é declarada, o que é chamado de *inicializar* a variável. Por exemplo:

int a = 5;

É possível também declarar mais de uma variável de um mesmo tipo em uma única instrução, separando os nomes por vírgulas. Também é possível inicializar as variáveis dessa maneira:

int a, b, c, d;

int e = 5, f = 6;

int g, h = 2, i = 7, j;

Como o próprio nome já diz, o valor existente numa variável pode ser mudado, da mesma maneira que ele é normalmente atribuído. Se tivermos:

int a;

a = 4;

a = 7;

no final o valor da variável ***a*** será 7.

2.2.1 – Tipos de Variáveis:

Na tabela seguinte temos os tipos de variáveis **int**, como ela pode ser lida pelo comando scanf() e o tamanho que ela ocupa na memória:

Tipo	Num de bits	Formato para leitura com scanf	Intervalo Inicio	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	16	%d	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	-32.768	32.767
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	-32.768	32.767
long int	32	%li	-2.147.483.648	2.147.483.647
signed long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295

Os tipos float e double servem para guardar números de ponto flutuante. A diferença entre os dois é, além do intervalo de dados, a precisão. Geralmente, o tipo float guarda dados (com sinal positivo ou negativo) de 3,4E-38 a 3,4E+38 (além do zero). Já double suporta números tão pequenos quanto 1,7E-308 e no máximo 1,7E+308.

float	32	%f	3,4E-38	3.4E+38
double	64	%lf	1,7E-308	1,7E+308
long double	80/128	%Lf	3,4E-4932	3,4E+4932

2.2.2 – Converter um tipo de variável:

A conversão de uma variável consiste em converter o tipo de uma variável em um outro. Imagine que você esteja trabalhando com uma variável do tipo float e por alguma razão queira eliminar os números que estão depois da vírgula.

Esta operação pode ser realizada de duas maneiras.

Conversões do tipo implícita: Consiste em uma modificação do tipo de variável que é feita automaticamente pelo compilador.

```
int x;
x = 7.123;
```

Conversões do tipo explícita: Também chamada de operação **cast**, consiste em forçar a modificação do tipo de variável usando o operador cast "( )".

```
int y;
y = (int)7.123;
```

2.3 – Sequências de Escape em C:

Quando for para leitura de uma variável pelo usuário ou escrita, também chamada de saída, no caso de uma variável tipo caracter ou *string*, você pode querer declarar, por exemplo, a própria aspa simples (utilizando \'), ou um caracter de nova linha (\n). Para isso você deve utilizar sequências de escape. Todas elas vêm precedidas de uma barra invertida (\).

Sequência	Equivale a
\n	Nova linha
\t	Tabulação
\b	Backspace
\"	Aspa dupla
\'	Aspa simples
\?	Ponto de interrogação
\\	Barra invertida

2.4 – Funções printf() e scanf():

Antes de mais nada, deveremos saber como poderemos fazer nossos códigos escreverem na tela, ou lerem algum valor digitado pelo usuário, assim usaremos as funções printf() para imprimir na tela e scanf() para ler tanto valores numéricos, como caracteres ou frases.

Um exemplo inicial e simples em C, imprimir “teste de programação em C” na tela com printf():

```
1. // inclusão de bibliotecas necessárias para que o compilador C possa utilizar
2.
3. #include <stdio.h>
4.
5. // início do programa
6.
7. int main (void){
8.     printf("teste de programação em C");
```

```
9.      return(0);
10. }
```

Como foi explicado no início, observe as identações logo depois de **int main(void)** e o símbolo de “{“ e “}” no final, eles indicam literalmente “início” e “fim” do código ou função que possa ser criada. Não precisamos nos preocupar com a linha int main: ela indica o início do código principal e a palavra dentro dos parênteses (void) indica que não haverá nenhum valor de retorno, pois simplesmente queremos imprimir algo na tela. A função printf() é utilizada aqui somente para imprimir então uma frase que deve sempre estar entre aspas duplas e ao final para indicar final de linha, sempre devemos inserir ponto-e-vírgula.

Para imprimir um resultado de um cálculo, por exemplo, uma simples soma de dois números inteiros, como foi visto, deveremos declarar as variáveis antes de fazermos os cálculos e para inserir este resultado na impressão, devemos obedecer às tabelas dos tipos de variáveis que estamos trabalhando. Vamos ao exemplo de impressão na tela de uma soma de dois valores inteiros:

```
1.  // inclusão das bibliotecas básicas para o compilador C
2.  #include <stdio.h>
3.  // início do programa
4.
5.  int main (void){
6.      int a=10;
7.      int b=20;
8.      int soma;
9.
10.     soma=a+b;
11.
12.     printf("\n Resulatdo da soma de %d com %d = %d \n",a,b,soma);
13.     return(0);
14. }
```

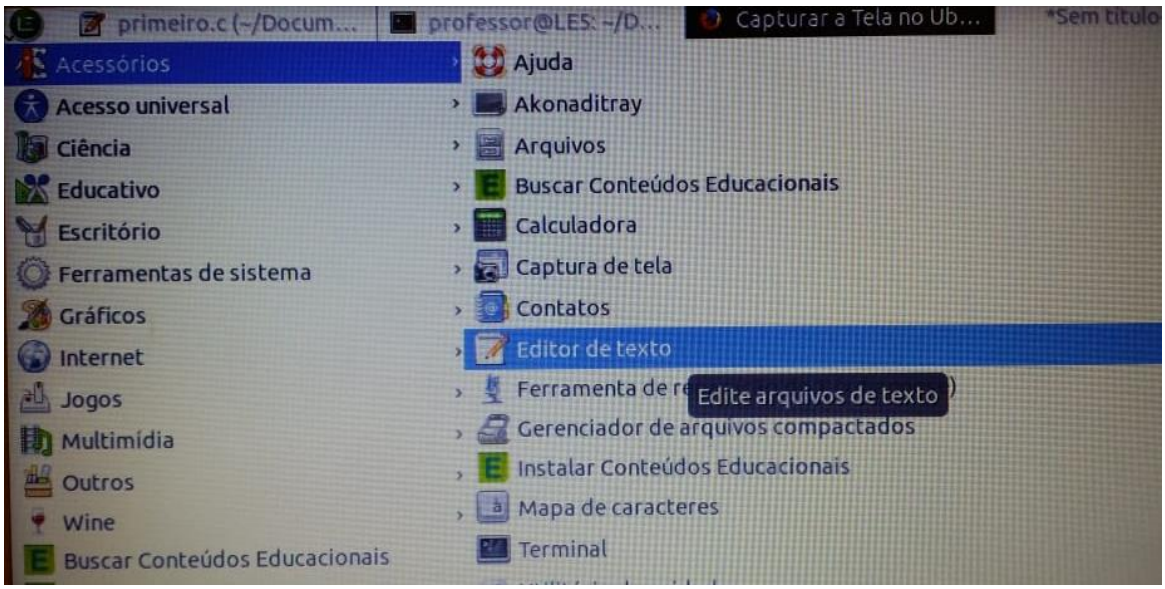
Observe que agora dentro de printf() apareceram %d três vezes, e ao final desta mesma linha foram indicadas três variáveis, que serão mostradas de acordo com a ordem que elas aparecem, então o primeiro %d, que indica a

posição de impressão de um inteiro, irá aparecer o valor de a, no segundo %d o valor de b e no terceiro valor de %d o resultado da soma.

2.5 – Como utilizar o laboratório de Informática da escola para rodar seus programas:

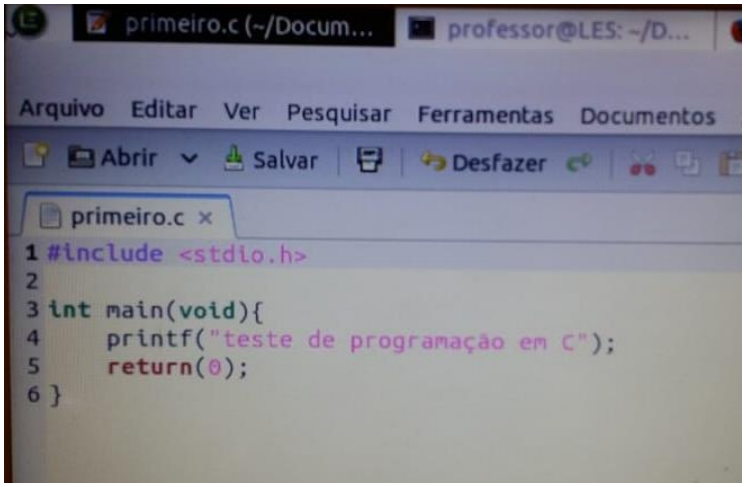
Como você está no laboratório de informática da escola e os computadores rodam uma versão do sistema operacional Linux, que é o Linux Educacional, significa que o compilador em C já está instalado nativamente, logo podemos realizar todas as compilações dos arquivos digitados em um editor de texto normal do Linux Educacional e depois compilar com o comando **gcc**, vamos ver o resultado do primeiro programa que foi digitado acima resultando na impressão em tela de “teste de programa em C”. Siga os seguintes passos:

1º) Com o computador ligado, clique no ícone verde LE no canto superior esquerdo, depois em Acessórios, depois em Editor de texto, como mostra a seguinte imagem:



Fonte: O autor (2020)

2º) Digite as linhas de programa como foi mostrado no exemplo acima,veja como fica na seguinte imagem:

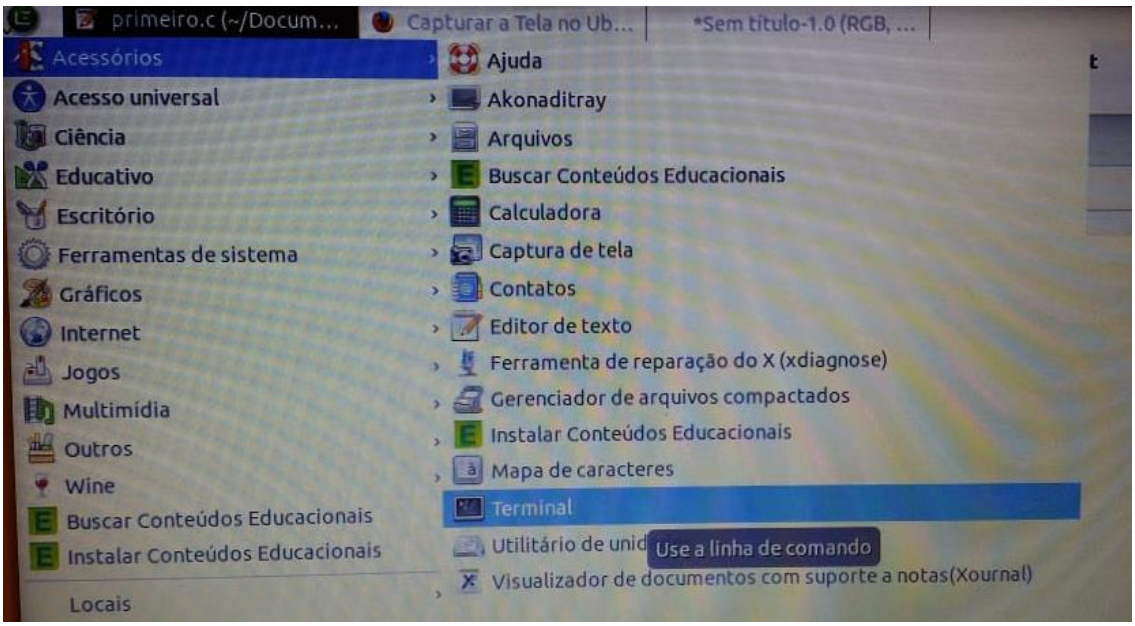


Fonte: O autor (2020)

3º) Clique na aba Arquivo e em Salvar Como:

Procure a pasta documentos e salve seu arquivo lá nomeando seu arquivo como **primeiro.c** e clique em salvar.

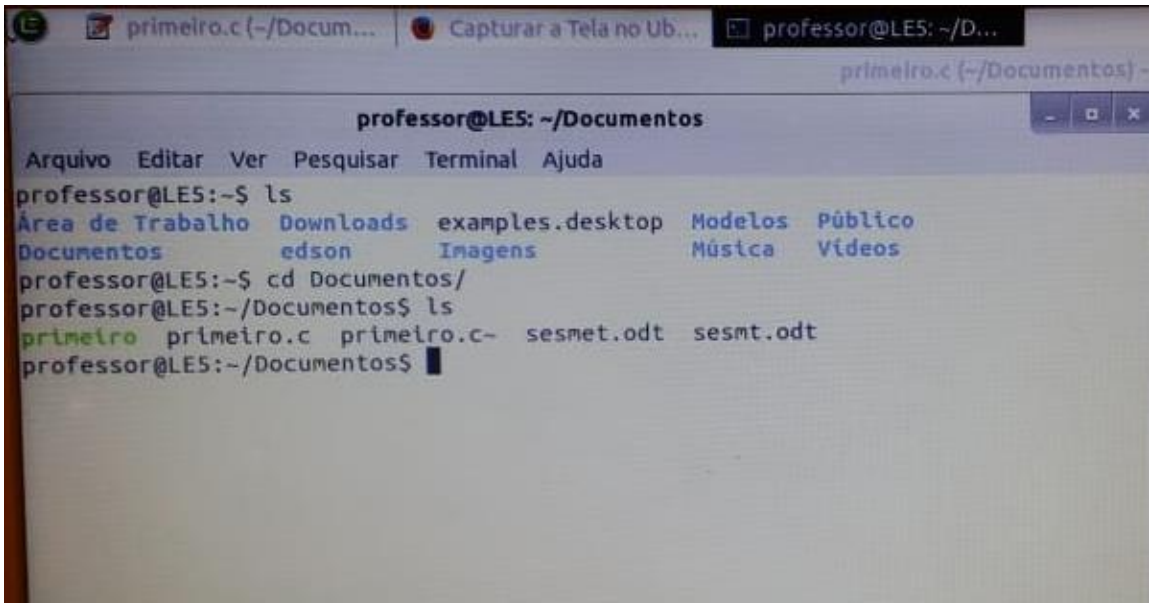
4º) Abra um terminal de comandos clicando novamente no canto superior esquerdo no ícone verde LE, depois Acessórios e depois Terminal. Veja como aparece na tela:



Fonte: O autor (2020)

5º) Na tela do terminal, veja se seu arquivo está na pasta documentos, alguns comandos em Linux deverão ser entendidos, como por exemplo o

comando **ls**, que significa **list**, irá listar todos os arquivos contidos na pasta, veja como fica na imagem abaixo:



Fonte: O autor (2020)

Veja que seu arquivo **primeiro.c** está na pasta, agora vamos compilá-lo para que possamos rodar quantas vezes for possível para fazermos os testes de programação.

6º) Para compilar o programa primeiro.c iremos utilizar o compilador gcc, na linha de comando do terminal que está na pasta Documentos digite:

gcc primeiro.c –o primeiro

Assim estamos compilando **primeiro.c** e tendo como saída o programa executável **primeiro**. Caso ocorra erro de digitação, irá aparecer uma mensagem de erro, se isto ocorrer verifique linha por linha e digite da mesma forma como o exemplo. Agora para testar o programa digite na linha de comando do terminal:

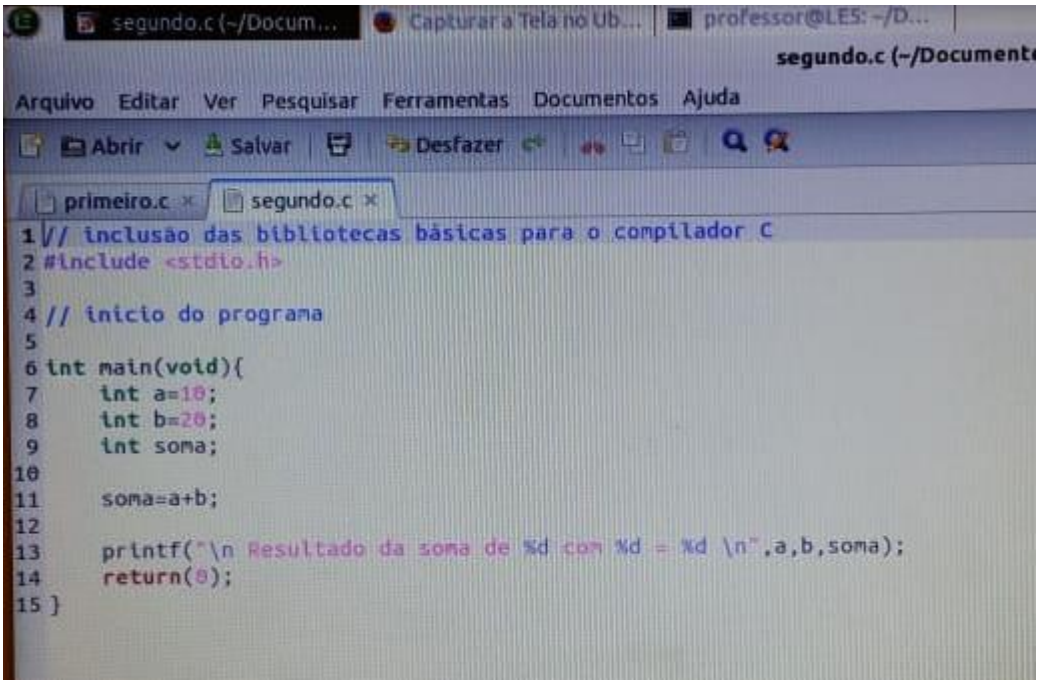
./primeiro

Veja o que ocorre, irá aparecer a frase que foi digitada dentro do programa **primeiro.c**. Note que não usamos \n para pular linha, agora volte lá no editor de texto e modifique seu programa incluindo \n dentro das aspas junto com a frase “teste de programação em C”, ficando assim “\n teste de programação em C \n” e compile no terminal de comandos com o comando abaixo para ver as modificações.

gcc primeiro.c –o primeiro

./primeiro

Vamos agora testar o segundo programa que apareceu como exemplo de utilização das funções printf(), que somou dois valores e imprimiu na tela sua soma, abra o **editor de textos** e digite como foi mostrado no exemplo, veja como deve ficar:



Fonte: O autor (2020)

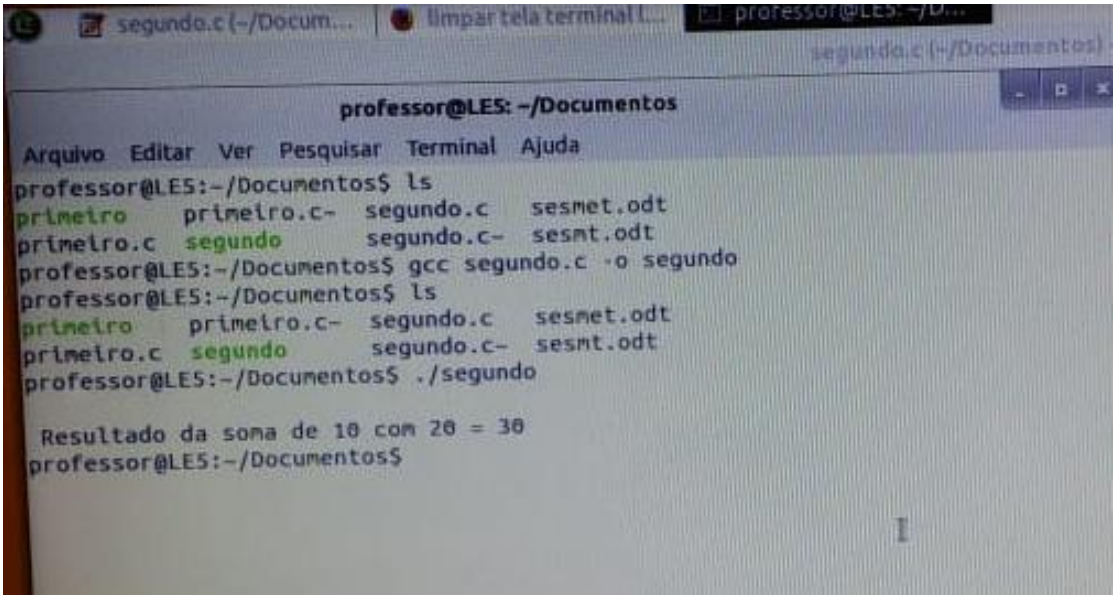
Compilando com gcc o segundo programa ficaria a linha de comando no terminal de comandos assim:

gcc segundo.c –o segundo

Após compilar, escreva ls na linha de comando para verificar se foi gerado e listado o programa **segundo** e para executar digite na linha de comando do terminal:

./segundo

Deverá aparecer o seguinte:



Fonte: O autor (2020)

Veja que aparece a frase “Resultado da soma de 10 com 20 = 30”, lembrando que na linha de comando dentro do editor de texto você digitou “Resultado da soma de %d com %d = %d”, verifique que o primeiro **%d** indica a entrada da variável inteira a que possui valor **10** como foi definido no programa, o segundo **%d** indica a entrada da segunda variável que corresponde ao valor de **b** que era **20**, e o terceiro **%d** indica a entrada do resultado da **soma**, que corresponde ao valor **30**.

Como exercício, tente modificar estes valores de **a** e **b** dentro do programa, salvar como segundo2.c e compilar novamente com:

gcc segundo2.c –o segundo2

Após isso, rodar o programa com:

./segundo2

Deverão aparecer os valores que você modificou agora na tela e um resultado diferente. Outro exercício que pode ser realizado para testarmos os tipos de variáveis seria atribuir para o valor de a um valor não inteiro, que seria um valor real que poderia ser com dois dígitos após a vírgula, como por exemplo, **a = 4.34**. Tente compilar e veja se ocorre algum erro! Caso não ocorra nenhum erro, verifique o que foi mostrado na tela, a declaração da variável **int a=4.34** está automaticamente transformando o valor de real para inteiro, por isto quando for rodado o programa, aparecerá somente 4 na tela, que corresponde à parte

inteira do número. Quando for digitar algum número real, deveremos utilizar **ponto** e não **vírgula** para a separação da parte decimal.

2.6 – Exemplo de programa em C com entrada digitada pelo usuário que soma dois valores inteiros:

Agora que você já aprendeu a digitar corretamente os programas, salvar e compilar usando o gcc nos computadores do laboratório de informática, outros exemplos surgirão para que você digite e teste mais linhas de comando, gerando um programa cada vez mais elaborado. Caso tenha dúvidas, sempre pergunte ao professor.

Neste exemplo será mostrado como são realizadas as entradas, ou seja, como você pode fazer para que o usuário participe do seu programa, interagindo com a digitação de alguns números, resultando na operação ou sequência desenvolvida por você na digitação do programa dentro do editor de textos. Abaixo veremos o uso da função **scanf()** agora para a entrada de dados, abra o editor de textos e digite o seguinte programa:

```
1.  // inclusão da biblioteca básica do compilador C
2.  #include <stdio.h>
3.
4.  // início do programa principal
5.  int main (void){
6.      //declaração de variáveis do tipo inteiro
7.      int a, b, soma;
8.
9.      printf("Digite um numero inteiro: ");
10.     scanf("%d", &a); //recebe um inteiro e armazena na variável a
11.
12.     printf("Digite um numero inteiro: ");
13.     scanf("%d", &b); //recebe um inteiro e armazena na variável b
14.     soma = a + b;     //Efetua adição de a com b e armazena na variável
    soma
15.
16.     printf("O valor da soma = %d\n", soma); //Mostra mensagem com o
    resultado
17.     return(0);
18. }
```

A função **scanf()** para receber valores deve identificar o tipo de variável que irá receber, no caso um inteiro **(%d)** e deve após a vírgula ter como

apontador o símbolo **&**. Caso esqueça este símbolo o programa irá retornar um erro de digitação!

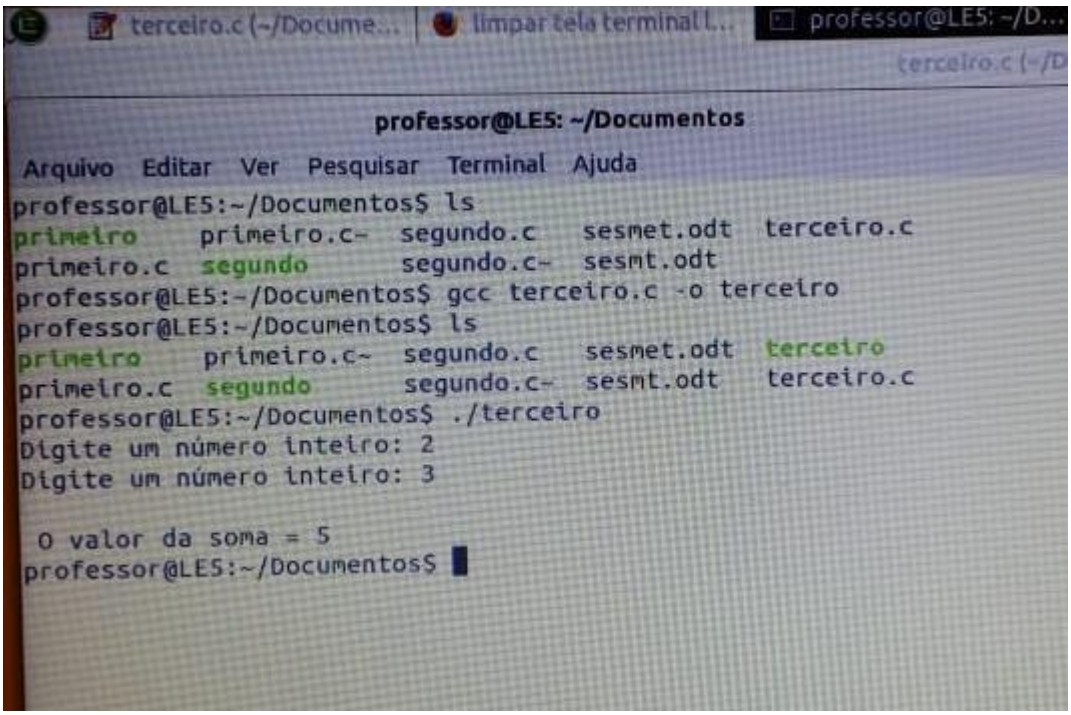
Salve o arquivo na pasta Documentos como **terceiro.c**, veja na tela do terminal de comandos com o comando ls para listar seus arquivos e verificar se foi gerado o arquivo terceiro.c. Execute gcc para compilar o programa, verificando a existência de erros ou não, caso esteja tudo correto aparecerá em verde o programa terceiro pronto para ser executado. Vamos ver como fica a sequência de comandos:

gcc terceiro.c –o terceiro

ls

./terceiro

Caso queira limpar a tela do terminal de comandos digite **clear**. Deverá aparecer a seguinte tela no terminal de comandos do Linux Educacional:



Fonte: O autor (2020)

2.7 – Comando de estrutura de repetição PARA-FAÇA em C:

Como vimos anteriormente, esta linha de comando evita fazermos repetições desnecessárias, sendo que uma linha de comando pode ser executada por inúmeras vezes, de acordo com a necessidade, como por

exemplo, repetir a frase “Estou aprendendo a linguagem de programação em C”, 20 vezes, torna-se muito simples com esta linha de comando PARA-FAÇA. Veja o exemplo:

```
1. #include <stdio.h>
2.
3. int main(void){
4.     int a;
5.     for (a=1;a<=20;a++){
6.     printf("Estou aprendendo a linguagem de programação em C\n”);
7.     }
8.     return(0);
9. }
```

Na linha 5 está o comando PARA-FAÇA, podemos ver claramente qual valor ele começa, **a=1**, até qual valor deve parar , **a<=10**, e qual incremento sofre a cada passada nesta linha de comando, **a++** que significa **a = a + 1**, vemos que neste caso a variável **a** torna-se simplesmente um contador. Ao final de cada linha é dado o comando \n que faz avançar a uma nova linha. No editor de texto do Linux Educacional, abra um **novo** arquivo e digite as linhas acima, salve o arquivo como quarto.c na pasta Documentos, compile e rode com gcc da seguinte maneira:

gcc quarto.c –o quarto

./quarto

2.8 – Comando de estrutura de repetição ENQUANTO-FAÇA em C:

Outro comando de estrutura de repetição é o ENQUANTO-FAÇA. Significa que enquanto uma comparação não retorna verdadeiro, ele continua realizando a operação dentro de um laço, como por exemplo, enquanto o professor passa a matéria no quadro, tenho que copiar. A comparação desta estrutura de repetição ocorre no início de estrutura e sua incrementação ocorre no final da estrutura, veja um exemplo:

```
1. #include <stdio.h>
2.
3. int main(void) {
4.     int contador = 1; //declarando e inicializando a variável de controle
5.
6.     while (contador <= 10) // Testando a condição
```

```
7.      {
8.          printf("%d ", contador); //Executando um comando dentro do laço
9.
10.         contador++; //atualizando a variável de controle
11.     }
12.
13.     return 0;
14. }
```

Vamos ver o que o programa realiza, começa declarando um contador inteiro e atribuindo a ele o valor 1, depois na linha 6 começa o comando de estrutura de repetição ENQUANTO – FAÇA com a seguinte comparação, enquanto o contador for <= 10 (menor ou igual a 10), faça a impressão do contador, então tome cuidado pois neste comando de repetição deve-se ao final do laço incrementar ou decrementar o contador, é o que ocorre na linha 10 com **contador ++**, senão torna-se um looping infinito e o programa não termina nunca. No editor de texto do Linux Educacional, abra um **novo** arquivo e digite as linhas acima, salve o arquivo como quinto.c na pasta Documentos, compile e rode com gcc da seguinte maneira:

gcc quinto.c –o quinto

./quinto

2.9 – Comando condicional de decisão SE-SENÃO:

Uma estrutura de decisão examina uma ou mais condições e decide quais instruções serão executadas dependendo se a condição foi ou não foi. O comando if é uma estrutura de decisão muito utilizada e sua outra forma é o SE-SENÃO, que será visto como IF – ELSE. Vamos examinar o seguinte exemplo abaixo, onde uma estrutura de decisão é utilizada para mostrar se a soma de dois valores passa o valor 12, caso isto ocorra será impresso na tela uma mensagem, caso não ocorra será impresso na tela outra mensagem.

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  // início do programa principal
5.  int main(void) {
6.      int intA, intB, ResSoma; // declaração das variáveis que são contadores
e o resultado
```

```
7.
8.      printf("Digite um numero inteiro: ");
9.      scanf("%d", &intA);                  // entrada do primeiro número
10.     printf("Digite um numero inteiro: ");
11.     scanf("%d", &intB);
12.     ResSoma = intA + intB;
13.     printf ("O Valor da soma = %d", ResSoma);
14.
15.     // estrutura de condição e decisão
16.     if(ResSoma > 12)  {
17.        printf("O valor da soma eh maior que 12\n");
18.     }
19.     else {
20.     printf("O valor é menor do que 12\n");
21.     }
22.     system ("PAUSE");
23.     return(0);
24. }
```

Caso a linha 16 for verdadeira o **printf()** será executado, senão, a linha 20 será executada, imprimindo outra tomada de decisão. No editor de texto do Linux Educacional, abra um **novo** arquivo e digite as linhas acima, salve o arquivo como sexto.c na pasta Documentos, compile e rode com gcc da seguinte maneira:

gcc sexto.c –o sexto

./sexto

Agora depois de passado por todas as estruturas necessárias, vamos exercitar, os mesmos exercícios propostos no começo que deveriam somente escrever a lógica de programação, agora deverão ser implementadas em linguagem C de programação.

2.10 – Exercícios com lógica e linguagem de programação em C:

1) Faça um algoritmo que leia as 3 notas digitadas por um estudante e calcule a média final deste estudante. Sabendo que a média deve ser igual ou superior a 7,0 escreva na tela se ele passou ou não de ano;

2) Elaborar um algoritmo que leia 3 valores a,b,c e os escreva na tela. A seguir, encontre o maior dos 3 valores e o escreva com a mensagem: "É o maior ";

3) Elaborar um algoritmo que escreva na tela as tabuadas do 5, 6 e 7;

4) Elaborar um algoritmo que leia um número digitado pelo usuário e que verifique e imprima na tela se é “POSITIVO” ou “NEGATIVO”, tomando cuidado com o valor 0 (zero);

5) Escrever um algoritmo que leia três números, positivos ou negativos, multiplique os dois primeiros, some com o terceiro e depois faça a divisão do resultado pelo primeiro;

6) Faça uma função que recebe por parâmetro o raio de uma esfera e calcula o seu volume. ($V = \frac{4}{3} . \pi . R^3$)

7) Faça uma função que recebe um valor inteiro e verifica se o valor é par ou ímpar.

2.11 – Lógica e linguagem de programação em C resolvendo exercícios de Física:

Avançando com seu conhecimento, iremos resolver alguns exercícios com programação em C na disciplina de Física, lembrando que todos os exercícios que possa imaginar podem ser resolvidos com a programação em qualquer linguagem. Antes de começar a escrever o programa no editor de textos, faça um rascunho do seu algoritmo para que haja um resultado final mais significativo e completo, pesquise antes quais as definições e Leis utilizadas pela Física, suas equações em cada caso, assim você estará estudando lógica de programação e os conteúdos de Física. Nos exercícios abaixo após escrever o programa, compile com o comando **gcc** já visto anteriormente e verifique se ocorreram erros na sua digitação ou erros de lógica de programação.

Lista de exercício trata sobre as conversões de unidades de comprimento, área, volume e intervalos de tempo mais úteis em Física.

1) Ao estudar a planta de uma construção, um engenheiro deparou-se com unidades de área dadas em cm². Certo cômodo dessa construção

apresentava área de 120 000 cm². Escreva um programa em C que leia este valor e expresse o mesmo em m².

A unidade para velocidade média no SI é o m/s, outra unidade utilizada seria o km/h, realizando-se a transformação entre elas pelo fator 3,6.

2) Um veículo desloca-se com velocidade de 216 km/h. Sua velocidade, em **metros por segundo (m/s),** é expressa por qual valor? Escreva um programa em C que transforme leia do usuário um valor em km/h e transforme para m/s e que faça também a conversão de m/s para km/h.

Lista de exercício trata sobre o estudo de Cinemática, como cálculo de deslocamento, velocidade, aceleração:

Acredita-se que Aristóteles do o primeiro filósofo a fazer estudo sistemático do que ele entendia por **movimento** (*kinesis*, em grego), na sua obra denominada Física (*Physika*, em grego), lançando assim, as bases para uma ciência independente que estudava a natureza e o mundo. Hoje dizemos que um corpo está em **movimento** em relação a determinado referencial quando sua posição, em relação a esse referencial, varia com o passar do tempo. Em Mecânica, é importante conhecer a rapidez com que um móvel sofre uma mudança de posição. A grandeza física que indica a tal rapidez é denominada **velocidade escalar média**, usado uma expressão para calcular a velocidade escalar média $\mathbf{v_m}$ para uma variação de espaço qualquer: $v_m = \frac{\Delta S}{\Delta t}$

3) Um veículo trafega em uma rodovia com velocidade média de 80 km/h. Sabendo que a viagem teve uma duração de 1 hora e 30 minutos (1,5 h), qual foi a distância percorrida pelo veículo? Escreva um programa em C específico para este exercício, fixando então as variáveis dentro do programa com os valores acima, utilizando a equação para cálculo da distância e imprima na tela o enunciado do exercício e seu resultado.

A aceleração indica a rapidez com que ocorre determinada variação na velocidade instantânea de um móvel. É muito comum dizer que um corpo que se

movimenta com velocidade variável tem aceleração. A **aceleração escalar média**, representada por $\mathbf{a_m}$ é dada por: $a_m = \frac{\Delta v}{\Delta t}$

Ao sofrer iguais variações de velocidade em iguais intervalos de tempo, um móvel estaria com **movimento uniformemente variado (MUV)**, para calcular sua posição utilizamos a função horária do espaço: $S = S_o + v_o.t + \frac{at^2}{2}$

4) Um veículo automotivo acelera a 2,0 m/s², durante 5,0 s, a partir de uma velocidade inicial de 2,0 m/s. A distância percorrida (ou deslocamento) por esse veículo, durante esse intervalo de tempo, dará qual resultado? Escreva um programa em C específico para este exercício, agora lendo do usuário três variáveis, **a aceleração, o tempo decorrido e a velocidade inicial**, utilizando a equação para cálculo de distância e imprima na tela o seu resultado.

Outra expressão utilizada no MUV é a equação de Torricelli, relacionando diretamente a velocidade escalar v, em um dado instante t, com o correspondente espaço S sem a necessidade de calcular o valor de t. Esta relação é chamada de **equação de Torricelli**, em homenagem ao cientista Evangelista Torricelli (1608 – 1647): $v^2 = {v_o}^2 + 2.a.\Delta S$

5) Um motorista dirigia a 30 m/s quando avista um buraco na pista e pisa no freio. Os freios produziram uma desaceleração de 2,0 m/s², até que o carro **para completamente**. O espaço percorrido pelo veículo até o final da frenagem foi de? Escreva um programa em C específico para este exercício, agora lendo do usuário duas variáveis, **a desaceleração e a velocidade inicial**, utilizando a equação para cálculo de espaço percorrido e imprima na tela o seu resultado.

6) Um motorista dirigindo seu carro a 60 km/h avista um semáforo fechado a sua frente, pisando no freio e desacelerando 1,5 m/s². Caso a distância inicial entre o carro e o semáforo seja de 100 m, escreva um algoritmo para verificar se este motorista irá ser multado ou não! Considere os parâmetros de entrada: a velocidade inicial de 60 km/h, sua desaceleração de 1,5 m/s² e sua distância ao semáforo de 100 m.

Lista de exercício trata sobre o estudo de Hidrostática, como cálculo de densidade, volume, pressão:

Arquimedes já sabia que, para massas iguais, o volume do bloco de prata é maior que o do bloco de ouro. Existe, portanto, uma característica que depende da massa e do volume e que distingue o ouro da prata e os materiais entre sí, de um modo geral. A essa característica dos materiais damos o nome de **densidade (d)**, expressa pela razão entre a massa m do objeto e o seu volume V, ou seja: $d = \frac{m}{V}$

6) Em um copo foi derramado um líquido com massa específica de 1,25 g/cm³, faça um programa em C que leia do usuário o valor da massa específica de outro líquido ou sólido e imprima na tela se ele irá flutuar ou afundar.

A unidade no **SI** para a densidade é o quilograma por metro cúbico (kg/m³), são também usuais as unidades grama por centímetro cúbico (g/cm³) e o quilograma por litro (kg/L). Um exemplo de transformação entre elas pode ser o fator de conversão 10³, ou seja, 1 g/cm³ equivale a 10³ kg/m³.

Quando a densidade se refere a um corpo homogêneo, líquido, gasoso ou sólido, usa-se também o termo **massa específica**, em vez de densidade. Geralmente a massa específica é representada pela letra grega μ. Assim, por exemplo, a densidade de uma esfera maciça de chumbo é 11,3 g/cm³ e coincide com sua massa específica.

7) A massa específica da glicerina tem um valor de 1,26 g/cm³. Escreva um programa em C que calcule o peso de 2 litros de glicerina ou outro valor qualquer. Considere g = 10m/s².

Podemos definir **pressão (p)** como a razão entre a intensidade da força que age perpendicularmente sobre uma superfície a área dessa superfície na qual a força se distribui, verifica-se que para uma mesma força, quanto menor a área da superfície, maior será a pressão, temos a relação inversa, logo: $p = \frac{F}{A}$

Sendo a unidade no **SI** da força em newton (N) e a área de uma superfície em metro quadrado (m²), a pressão tem como unidade o newton por metro quadrado (N/m²), essa unidade também pode ser chamada de pascal (**Pa**). Outras unidades de pressão, denominadas unidades práticas podem ser

utilizadas, como: milímetro de mercúrio (mmHg), atmosfera (atm), psi (forma abreviada do inglês pound force per square inch, cujo símbolo também pode ser escrito como lbf/in²) ou **libra**-força por **polegada quadrada.** Possuindo as seguintes relações de transformação: 1 atm = 760 mmHg = 14,7 psi = 101325 Pa = 101325 N/m^2.

Do mesmo modo que líquidos exercem pressão, os gases também o fazem, então, qualquer corpo imerso em um gás está sujeito à pressão por ele exercida, logo, se estamos imersos no ar de todos os lados vivemos em um oceano de ar, a **pressão terrestre.** A pressão a que qualquer objeto fica submetido em consequência disso é denominada **pressão atmosférica.**

8) A pressão atmosférica em determinada região da Terra é igual a 780 mmHg. Escreva um programa em C que calcule o valor da pressão atmosférica local em atm. Imprima na tela o resultado indicando qual o valor da pressão de 1 atm e o resultado do cálculo acima.

9) Uma força de 200 N é aplicada sobre uma área de 0,05 m². A pressão exercida sobre essa área, em Pa, terá qual valor? Escreva um programa em linguagem C de programação e imprima na tela o resultado.

No teorema de Stevin é estabelecido que a pressão hidrostática exercida por uma coluna de líquido em equilíbrio usamos a expressão: $p = d.g.h$. Para calcular a pressão total exercida em um ponto no interior de um líquido em equilíbrio, devemos acrescentar à pressão da coluna líquida a pressão exercida na superfície do líquido, que usualmente corresponde à pressão atmosférica (p_{atm}), quando este líquido está exposto ao ar. A expressão para este caso fica:

$$p = p_{atm} + d.g.h$$

10) A cada 10 m de profundidade de água, aumenta-se, aproximadamente, 1 atm. Após mergulhar em um lago com 20 metros de profundidade, um mergulhador estará sujeito a uma pressão, em mmHg, igual a? Escreva um programa em linguagem C de programação e imprima na tela o resultado.

Dados: 1 atm = 760 mmHg

3 – Arduino, praticidade e facilidade de programação:

Iniciando agora com o objetivo principal deste curso que é o conhecimento e utilização de uma placa chamada de prototipagem, onde poderemos controlar todos os tipos de motores e sensores para construirmos desde um simples sinalizador com led, até um robô que consegue seguir uma linha desenhada em uma superfície, ou um braço robótico utilizado para realizar tarefas repetidas, como por exemplo, empilhar objetos.

O Arduino foi criado em 2005 por um grupo de 5 pesquisadores italianos, que ensinavam e trabalhavam principalmente com design: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além disso, foi adotado o conceito de hardware livre, o que significa que qualquer um pode montar, modificar, melhorar e personalizar o Arduino, partindo do mesmo hardware básico.

Assim, foi criada uma placa composta por um **microcontrolador Atmel**, circuitos de entrada/saída e que pode ser facilmente conectada à um computador e programada via **IDE (*Integrated Development Environment*, ou *Ambiente de Desenvolvimento Integrado*)** utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB.

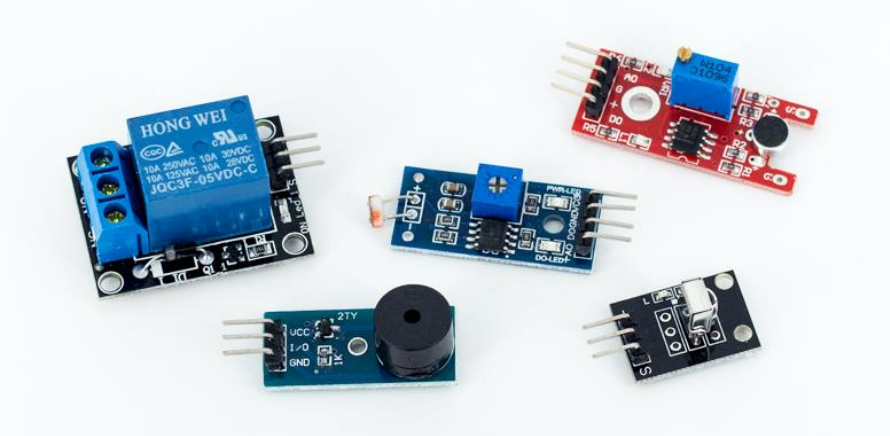
Fonte: https://uploads.filipeflop.com

Depois de programado, o microcontrolador pode ser usado de forma independente, ou seja, você pode colocá-lo para controlar um robô, uma lixeira, um ventilador, as luzes da sua casa, a temperatura do ar condicionado, pode utilizá-lo como um aparelho de medição ou qualquer outro projeto que vier à cabeça.

3.1 – O que você pode fazer com o Arduino:

A lista de possibilidades é praticamente infinita. Você pode automatizar sua casa, seu carro, seu escritório, criar um novo brinquedo, um novo equipamento ou melhorar um já existente. Tudo vai depender da sua criatividade.

Para isso, o Arduino possui uma quantidade enorme de sensores e componentes que você pode utilizar nos seus projetos. Grande parte do material utilizado está disponível em módulos, que são pequenas placas que contém os sensores e outros componentes auxiliares como resistores, capacitores e leds.



Fonte: https://uploads.filipeflop.com

Existem também os chamados Shields, que são placas que você encaixa no Arduino para expandir suas funcionalidades. A imagem abaixo mostra um **Arduino Ethernet Shield** encaixado no Arduino Mega 2560. Ao mesmo tempo que permite o acesso à uma rede ou até mesmo à internet, mantém os demais pinos disponíveis para utilização, assim você consegue, por exemplo, utilizar os pinos para receber dados de temperatura e umidade de um ambiente, e consultar esses dados de qualquer lugar do planeta.

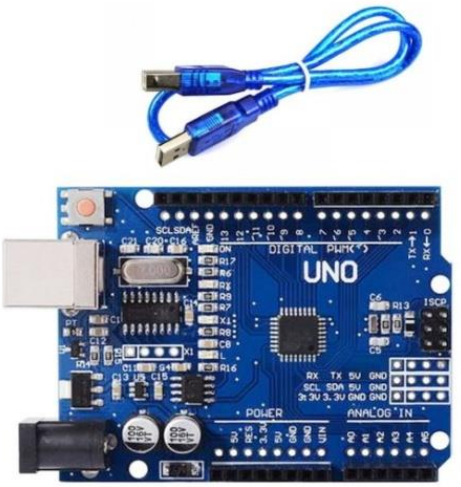
Fonte: https://uploads.filipeflop.com

3.2 – Modelos de Placas Arduino:

O tipo de placa que você vai utilizar depende muito do projeto a ser desenvolvido e o número de portas necessárias. As opções vão das mais comuns, como o **Arduino Nano**, que é o menor de todos, cada um dos 14 pinos digitais no Nano pode ser usado como uma entrada ou uma saída, usando as funções de pinMode(), digitalWrite(), e digitalRead(). Eles operam a 5 volts. Em adição alguns pinos possuem funções especializadas.

Fonte: https://uploads.filipeflop.com

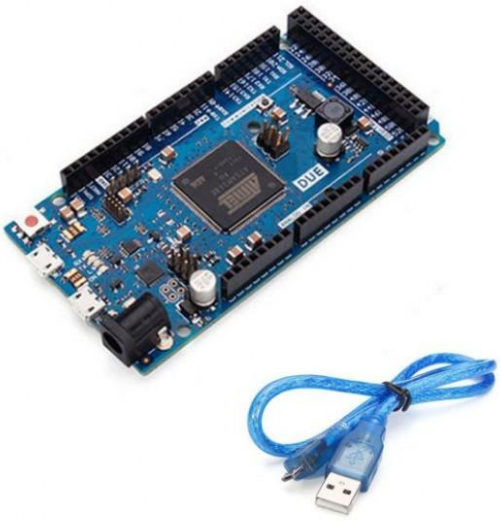
O **Arduino Uno** e suas 14 portas digitais e 6 analógicas, com seu cabo de conexão para porta USB.



Fonte: https://uploads.filipeflop.com

Passando por placas com maior poder de processamento, como o Arduino Mega, com microcontrolador ATmega2560 e 54 portas digitais, e Arduino Due, baseado em processador ARM de 32 bits e 512 Kbytes de memória.

Fonte: https://uploads.filipeflop.com



Fonte: https://uploads.filipeflop.com

Nos experimentos em laboratório para práticas de programação e robótica, iremos utilizar principalmente o Arduino Nano e o Arduino Uno.

3.3 – Estrutura de um programa em Arduino – Teste do primeiro programa:

Utilizando o seu conhecimento de lógica de programação visto no início do curso e também sobre programação em C, escrever um programa em Arduino é muito simples. Tudo o que você precisa é conectá-lo ao computador por meio de um cabo USB e utilizar um ambiente de programação chamado **IDE**, antes para programação em C era utilizado um editor de texto simples, mas agora para realizar a programação para o Arduino será utilizado um ambiente todo completo específico onde você digita o programa, faz os testes para encontrar eventuais erros e transfere o programa para o dispositivo com um cabo USB.

Na imagem abaixo temos a IDE já com um programa carregado. No site oficial do Arduino (www.arduino.cc) você pode fazer o download da IDE gratuitamente:

Fonte: o autor (2020).

IDE - ***Ambiente de Desenvolvimento Integrado,*** neste editor de texto específico para desenvolvimento de programas para Arduino, iremos escrever todos os códigos de programação, compilar e testar enviando para as placas Arduino.

No caso deste curso ser realizado no laboratório de informática da escola, deverá ser instalado a IDE Arduino para Linux, veja como proceder em:

https://www.arduino.cc/en/guide/linux.

IMPORTANTE: Caso tenha alguma dúvida sobre alguma função ou variável que será utilizada nos programa seguintes, acesse o Guia de Referência do site oficial do Arduino que mostrará todas estas informações:

https://www.arduino.cc/reference/pt/

Você não precisa ser expert em linguagem C para programá-lo, mas ter o conhecimento básico da linguagem, como usar os comandos de repetição e de condição ajudam bastante. Além da grande quantidade de exemplos que você encontra na internet devido à grande difusão e utilização hoje em dia desta tecnologia, digitando “exemplos de programação em Arduino” aparecem muitos blogs e vídeos já prontos ensinando desde o básico ao avançado, ou você pode começar um programa utilizando a estrutura básica do Arduino, que é composta por duas partes, ou dois blocos:

setup() – É nessa parte do programa que você configura as opções iniciais do seu programa: os valores iniciais de uma variável, se uma porta será utilizada como entrada ou saída, mensagens para o usuário, etc.

loop() – Essa parte do programa repete uma estrutura de comandos de forma contínua ou até que algum comando de “parar” seja enviado ao Arduino.

Vamos então ao primeiro programa em Arduino e ver exatamente como isso funciona, levando em consideração o programa abaixo, que acende e apaga o led embutido na placa em intervalos de 1 segundo:

```
1. //Primeiro Programa em Arduino: Pisca Led
2.
3. void setup()
```

```
4.  {
5.      //Define a porta do led como saida
6.      pinMode(13, OUTPUT);
7.  }
8.  void loop()
9.  {
10.     //Acende o led com o comando HIGH
11.     digitalWrite(13, HIGH);
12.     //Aguarda o intervalo de tempo especificado, aqui significa 1s
13.     delay(1000);
14.     //Apaga o led com o comando LOW
15.     digitalWrite(13, LOW);
16.     //Aguarda o intervalo especificado novamente de 1s
17.     delay(1000);
18. }
```

Comece uma linha com barras duplas (//) e tudo o que vier depois dessa linha, como ocorre na programação em C, será tratado como um comentário. Uma das boas práticas de programação é documentar o seu código por meio das linhas de comentário. Com elas, você pode inserir observações sobre como determinada parte do programa funciona ou o que significa aquela variável ResPontoA, que você criou. Isso será útil não só para você, se precisar alterar o código depois de algum tempo, como também para outras pessoas que utilizarão o seu programa.

Após os comentários, vem a estrutura do SETUP. É nela que definimos que o pino 13 do Arduino será utilizado como saída.

```
4  void setup()
5  {
6    //Define a porta do led como saida
7    pinMode(13, OUTPUT);
8  }
```

Fonte: o autor (2020).

Por último, temos o comando em loop, que contém as instruções para acender e apagar o led, e também o intervalo entre essas ações, este intervalo sempre será escrito em milissegundos (ms), logo 1000 ms corresponde a 1s, 2000 ms correspondem a 2s e assim sucessivamente.

```
10  void loop()
11  {
12    //Acende o led
13    digitalWrite(13, HIGH);
14
15    //Aguarda o intervalo especificado
16    delay(1000);
17
18    //Apaga o led
19    digitalWrite(13, LOW);
20
21    //Aguarda o intervalo especificado
22    delay(1000);
23  }
```

Fonte: o autor (2020).

A linha do código contendo digitalWrite(13, HIGH) coloca a porta 13 em nível alto (HIGH, ou 1), acendendo o led embutido na placa. O comando delay(1000), especifica o intervalo, em milisegundos, no qual o programa fica parado antes de avançar para a próxima linha.

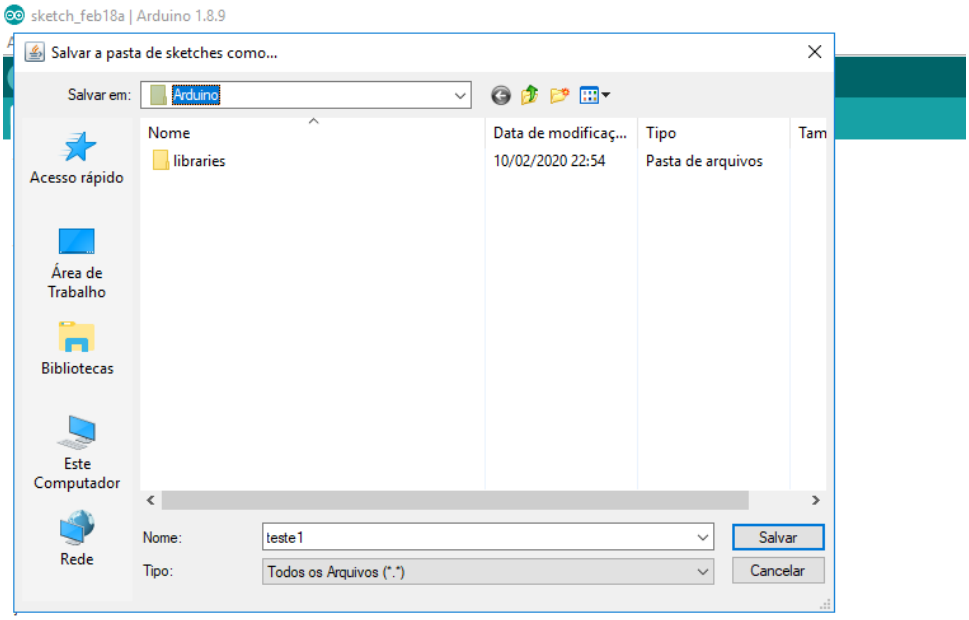
O comando digitalWrite(13, LOW), apaga o led, colocando a porta em nível baixo (LOW, ou 0), e depois ocorre uma nova parada no programa, e o processo é então reiniciado dentro do loop definido pelo programador.

Fonte: o autor (2020).

Para testar este programa, abra o IDE – Arduino e digite de acordo com o exemplo, linha por linha. Veja como ficaria na figura abaixo:

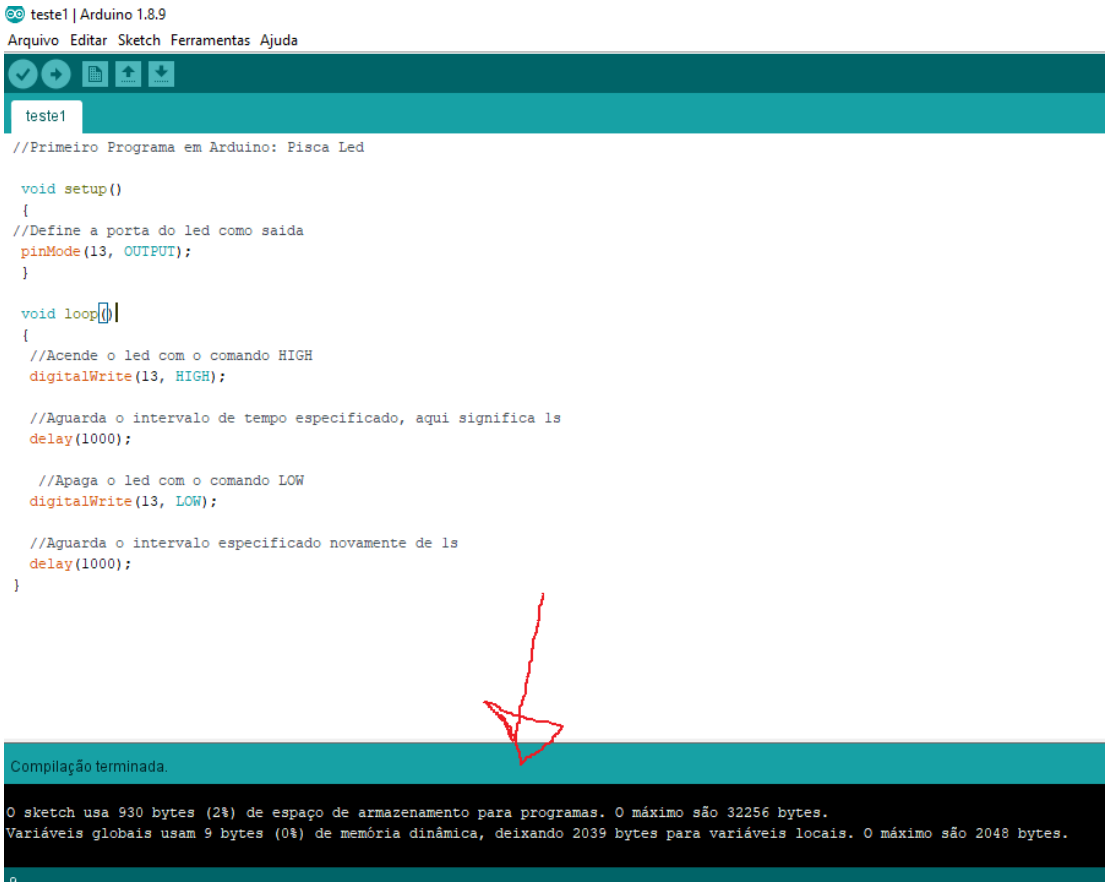
Fonte: O autor (2020)

Após digitar clique no primeiro ícone acima para compilar o seu programa, irá abrir uma aba para salvar o arquivo, nomeie como **teste1**, veja a seguinte tela. Abrirá aba para salvar:



Fonte: O autor (2020)

Caso sua digitação estiver correta será compilado e abaixo não acusará nenhum erro:

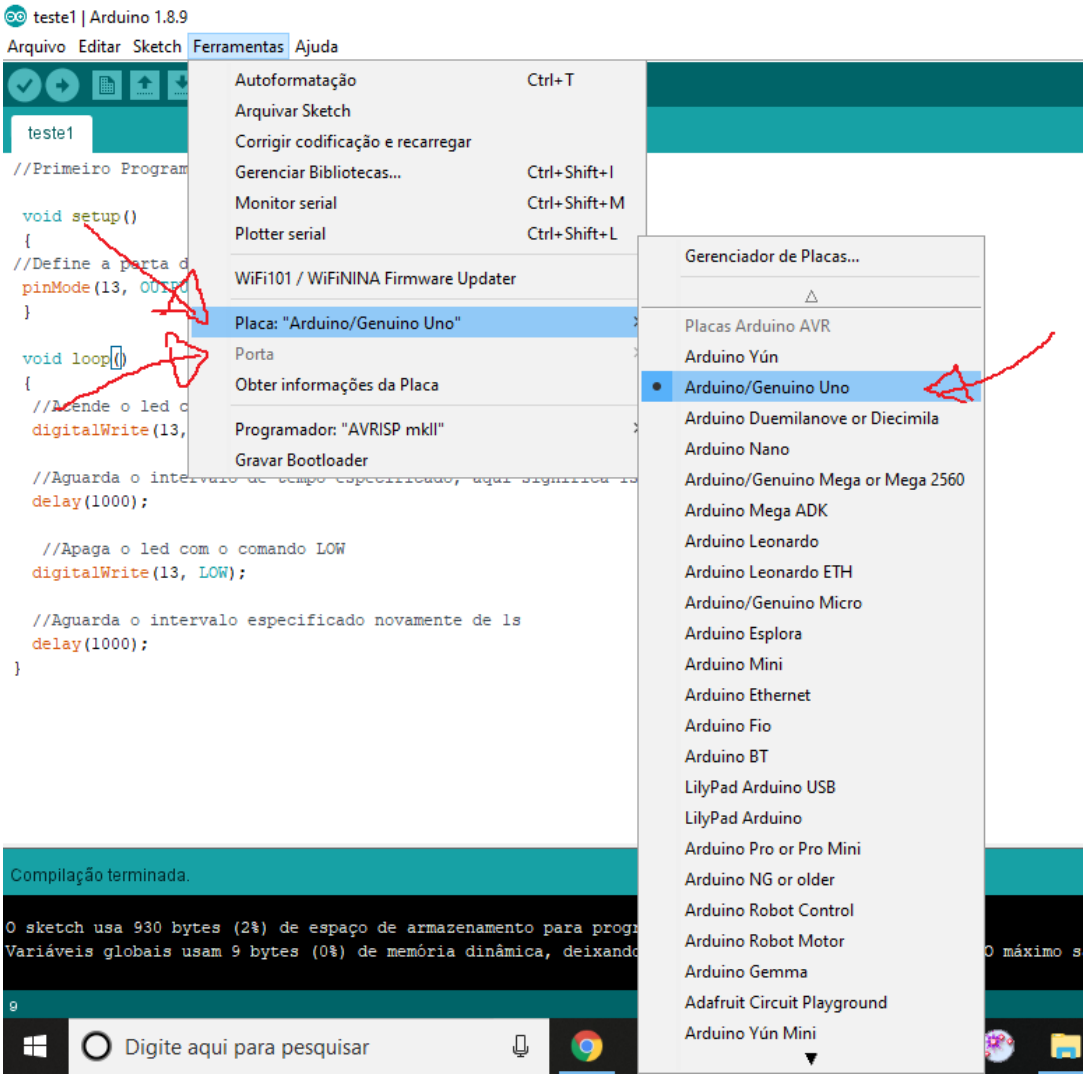


Fonte: O autor (2020)

Agora precisamos conectar o Arduino Uno via cabo USB no PC ou notebook e **enviar** este código para a placa, isto significa que irá salvar uma cópia do programa na memória e esta cópia será executada na placa.

Fonte: https://www.eletronica24h.net.br

Para enviar o programa para a placa, depois de conectada, deve-se tomar alguns cuidados de configuração, como por exemplo, escolher o tipo adequado de Arduino que estamos trabalhando e a porta onde foi conectado ao PC ou notebook; isto deve ser modificado no menu Ferramentas:



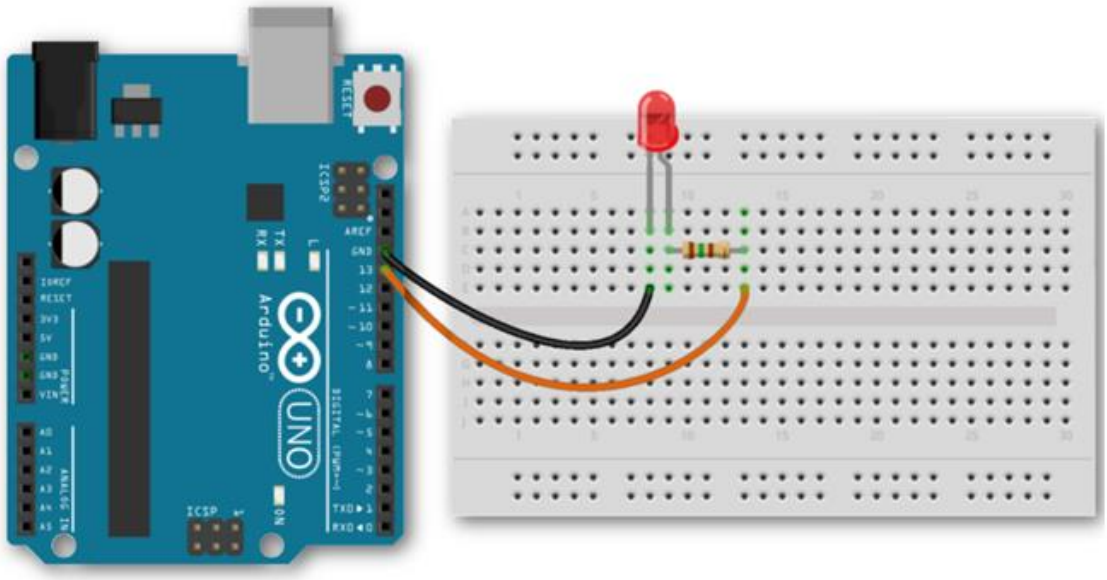
Fonte: O autor (2020)

A porta de conexão aparecerá logo abaixo desta linha Placa:”Arduino/Genuino Uno”, deverá estar setada como COM4, COM5, COM6 ou algo assim. Se não aparecer, houve algum erro de instalação de drivers para comunicação com a placa e isto deve ser resolvido procurando e instalando os drivers para comunicação entre o Arduino e o PC ou notebook.

Caso esteja tudo certo, a placa irá piscar duas vezes e isto indica que foi carregado o programa, observe que neste caso o objetivo era fazer o led da placa ficar piscando em intervalos de 1 segundo, note isto na placa Arduino.

3.4 – Segundo programa em Arduino – testando as portas de saída da placa:

Segundo os idealizadores da placa de prototipagem, um dos objetivos era instigar a intuição dos programadores aliando com uma certa facilidade com a parte da programação. Vamos ver se conseguimos fazer um led acender e apagar fora da placa. Para isto usaremos um apoio para conectar as peças e os fios de conexão chamado protoboard. Veja na seguinte imagem como deveremos ligar o led externo com uma pequena resistência que serve para não queimar o led na saída da porta 13 do Arduino Uno. O led possui polaridade, logo um fio será ligado no GND (ground) da placa Arduino e o outro na saída da porta 13 ao resistor que pode ser de 100 Ω que está ligado ao pino positivo do led. Com o **mesmo programa** que já foi carregado na placa, conecte ao PC ou notebook com o cabo USB para a alimentação da placa e veja o resultado:



Placa Arduino Placa protoboard

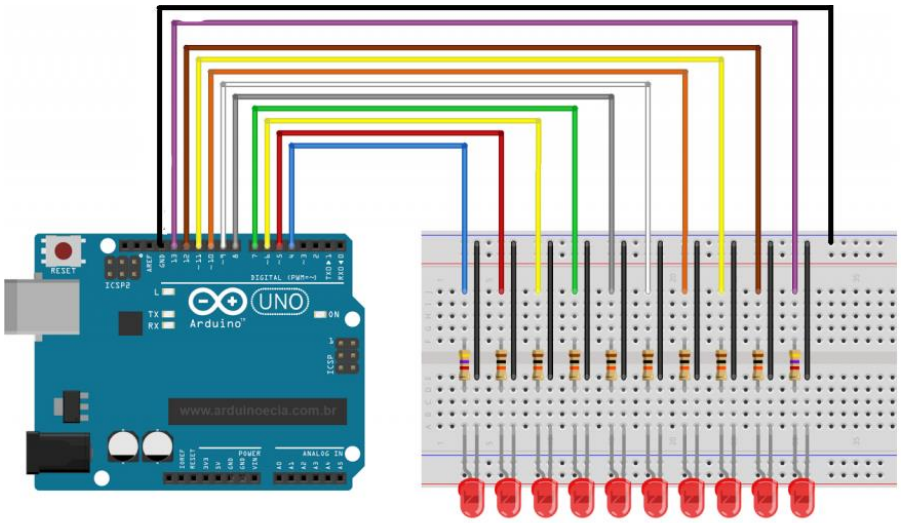
Fonte: O autor (2020)

3.5 – Terceiro programa em Arduino – sequencial de leds:

Neste exemplo será montado na placa protoboard um sequencial de 10 leds onde o Arduino irá controlar todos eles apagando um e acendendo o outro de acordo com o tempo estabelecido no programa. Abra o IDE Arduino e digite o seguinte código abaixo:

```
1.  // projeto 5 - efeito de iluminação sequencial de leds
2.
3.  //Cria um array para os pinos dos Leds
4.  byte ledPin[] = { 4,5,6,7,8,9,10,11,12,13 };
5.  int tempo = 75;   //variável que define o tempo para as alterações
6.  int ledDelay(tempo);  //intervalo entre as alterações
7.  int direction = 1;  //Direção do "movimento"
8.  int currentLED = 0;
9.  int elimina = 9;
10. unsigned long changeTime;
11.
12. void setup() {
13.     Serial.begin(9600);  //Inicializa a comunicacao serial
14.     for (int x=0; x<10; x++){   //Loop que define todos os pinos como saída
15.         pinMode(ledPin[x],OUTPUT);
16.     }
17.     changeTime = millis();
18. }
19.
20. void loop() {
21.     if ((millis() - changeTime) > ledDelay){     //verifica se já decorreram
    ledDelay em ms
22.         changeLED();                              desde a última alteração
23.         changeTime = millis();
24.     }
25. }
26.
27. void changeLED(){
28.     for (int x=0;x<10;x++){
29.         digitalWrite(ledPin[x],LOW);
30.     }
31.     digitalWrite(ledPin[currentLED],HIGH);
32.     currentLED += direction;
33.     Serial.println(currentLED);
34.     if (currentLED == 9) {
35.         direction = -1;
36.         ledDelay = 40;
37.     }
38.     if (currentLED == 0) {
39.         direction = 1;
40.         ledDelay = 100;
41.     }
42. }
```

Utilizando um Arduino Uno, uma placa protoboard, 10 leds e 10 resistores de 1KΩ e fios para conexão, faça a montagem de acordo com o seguinte esquema:



Fonte: O autor (2020)

Compile o programa e depois carregue para a placa. Veja o que ocorre.

Da linha 4 a linha 10 são declaradas as variáveis que serão utilizadas dentro do setup() e do loop(), na linha 12 é iniciado a comunicação serial que é utilizada para verificar a saída em uma tela no IDE – Arduino, para ver esta tela aperte ao mesmo tempo Ctrl+Shift+M, note a mudança dos números e entenda o significado destas mudanças. Na linha 14 o comando de repetição **FOR** inicializa todas as portas de saída com **pinMode(ledPin[x],OUTPUT);** utilizando a array ledPin[x] que começa na posição 0 (zero) como pino 4 e vai até a posição 9 do array como pino 13, pois o array foi definido no começo como **byte ledPin[] = { 4,5,6,7,8,9,10,11,12,13 };**.

Dentro da função loop() ocorre uma chamada de uma outra função externa que é a changeLED(), isso pode ser feito sempre que você necessitar realizar uma função em qualquer parte do programa, sem a necessidade de toda vez ficar digitando a função, ou seja repetindo a mesma função sendo que ela executa a mesma finalidade. Ainda dentro de loop(), se transcorreram ao menos ledDelay milissegundos desde a última alteração nos LEDs; em caso afirmativo, o código passa o controle para sua função. O motivo de você passar o controle para a função changeLED() dessa maneira, em vez de utilizar comandos delay(),

e para permitir que, se necessário, outro código seja executado no loop principal do programa (desde que esse código demore menos que ledDelay para ser executado).

Explicando a função externa changeLED(), inicialmente apaga todos os leds muito rapidamente que você nem percebe e acender o led atual, verificando com os comandos if se chegou ao final do curso. Caso isto tenha ocorrido, será invertido a sequência.

Exercícios: Pesquise na internet dois casos e tente implementá-los:

1) como mudar o tempo de delay de um led apertando um botão.

2) como mudar o tempo de delay de um led com um potenciômetro giratório.

3.6 – Usando comandos de entrada de dados para controlar motores e sensores no Arduino

Uma das grandes vantagens do trabalho com o Arduino são os controles dos sensores e motores que podem ser realizados por portas analógicas, estas podem receber variações de valores recebidos, como por exemplo, por potenciômetros que nada mais são do que resistores variáveis. O exemplo seguinte utiliza o controlador de motor L293D que é um circuito integrado chamado de ponte H que pode controlar dois motores de corrente contínua ao mesmo tempo. Com isto poderia ser implementado um robô de controle remoto, ou um braço mecânico com dois movimentos.

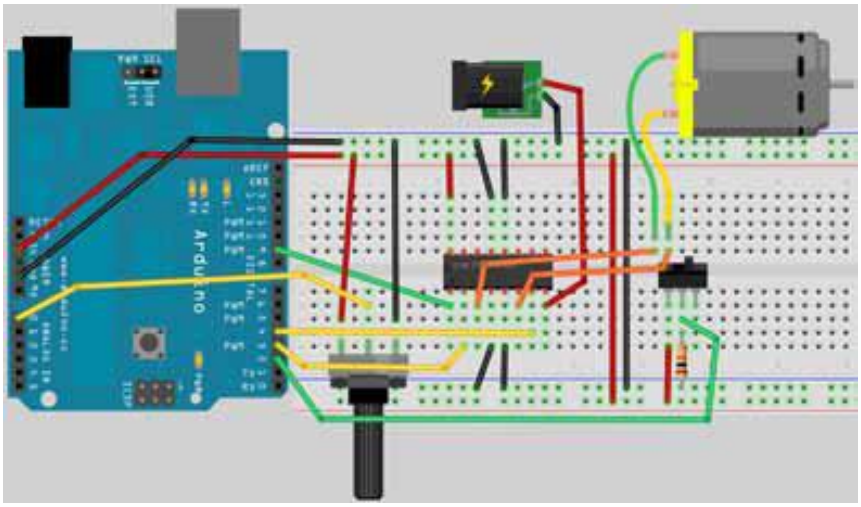
CI controlador de motor l293D

Fonte: O autor (2020)

Antes de ir mais a fundo, iremos simplesmente controlar um motor com um potenciômetro, mudando a velocidade de rotação do motor e também mudando o sentido de rotação. Digite o código no IDE – Arduino e compile para verificar possíveis erros, depois faça o upload para a placa com o cabo USB:

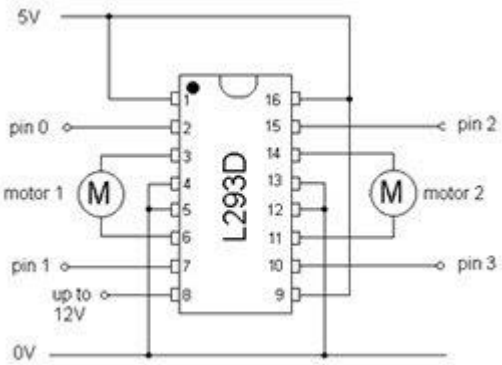
```
1. // Implementação de código usando CI controlador de motor (ponte H) L293D
2. // definições iniciais de constantes que não mudarão durante o programa
3. #define chavePino 2 // entrada da chave
4. #define motorPino1 3 // entrada 1 do L293D
5. #define motorPino2 4 // entrada 2 do L293D
6. #define velocPino 9 // pino de ativação 1 do L293D
7. #define potencPino 0 // ligação do potenciômetro no pino analógico 0
8. int atualVeloc // variável para armazenar o valor da atual velocidade
9. // início da função setup()
10. void setup() {
11.     // define o pino chave como INPUT
12.     pinMode(chavePino,INPUT);
13.     // define os pinos restantes como saída
14.     pinMode(motorPino1,OUTPUT);
15.     pinMode(motorPino2,OUTPUT);
16.     pinMode(velocPino,OUTPUT);
17. }
18. void loop(){
19.     atualVeloc = analogRead(potencPino)/4;  //lê o valor de velocidade pelo potenciômetro
20.     analogWrite(velocPino,atualVeloc);  //escreve a velocidade para o pino de ativação 1
21.     If (digitalRead(chavePino){    //se a chave estiver HIGH, gire o motor em sentido horário
22.         digitalWrite(motorPino1,LOW);
23.         digitalWrite(motorPino2,HIGH);
24.     }
25.     else{
26.         digitalWrite(motorpino1,HIGH);
27.         digitalWrite(motorpino2,LOW);
28.     }
29. }
```

Figura com o esquema de ligação do motor, do potenciômetro, da chave e da bateria.



Fonte: o autor (2020).

Alguns cuidados que deverão ser tomados antes de ligar a bateria e ver este projeto funcionando: o CI L293D possui um aquecimento que deve ser controlado colando-se sobre ele um dissipador de calor. Sua conexão com a placa protoboard deve seguir a orientação de seu pino 1 ligado na porta 9 do Arduino, veja o esquema interno do CI controlador de motor L293D:



Fonte: O autor (2020)

Este CI L293 pode ser adquirido em placa já pronta como sendo um shield para somente conectar o motor, como mostra a figura seguinte:

Fonte: https://uploads.filipeflop.com

3.7 – Material de apoio: Livro “Arduino Básico” – Michael McRoberts

À partir de agora poderemos implementar e construir qualquer tipo de projeto usando o Arduino Uno e alguns componentes para controlarmos leds, motores, sensores e servo motores. O livro do Michael McRoberts pode ser baixado gratuitamente em pdf pelo site da USP:

https://edisciplinas.usp.br/pluginfile.php/4287597/mod_resource/content/2/Ardu%C3%ADno%20B%C3%A1sico%20-%20Michael%20McRoberts.pdf

Faça o download do livro e veja o projeto 25 que ensina a controlar um servo motor, o projeto 27 que utiliza um joystick e o projeto 28 que controla um motor de passo e até mesmo o Arduino pode conectar-se à internet, veja o projeto 46.