Universidade Federal do Pará - UFPA
Instituto de Tecnologia – ITEC
Programa de Educação Tutorial de Engenharia Elétrica – PET EE

Minicurso Básico de Arduino.

SUMÁRIO

1) CONHECENDO A PLACA ARDUINO.

A placa Ar[illegible]

com o mundo externo. Vejamos como estão organizados os pinos da mesma:

14 pinos de entra e saída digital (pinos 0-13): Esses pinos podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto e conforme foi definido no sketch criado na IDE.

6 pinos de entradas analógicas (pinos A0 - A5): Esses pinos são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 V onde serão convertidos para valores entre 0 e 1023.

6 pinos de saídas analógicas (pinos 3, 5, 6, 9, 10 e 11): São pinos digitais que podem ser programados para ser utilizados como saídas analógicas, utilizando modulação PWM.

A alimentação da placa pode ser feita a partir da porta USB do computador ou através de um adaptador AC. Para o adaptador AC recomenda-se uma tensão de 9 a 12 volts.

2) Criando uma conta na plataforma TinkerCad

O tinkerCad é uma plataforma online que permite a simulação de variados tipos de projetos, a qual será utilizada por nós para realizar experimentos no envolvendo a placa Arduino.

Para utilizar a plataforma, você deve, primeiramente, criar uma conta acessando https://www.tinkercad.com/. Clique em “inscrever - se” :

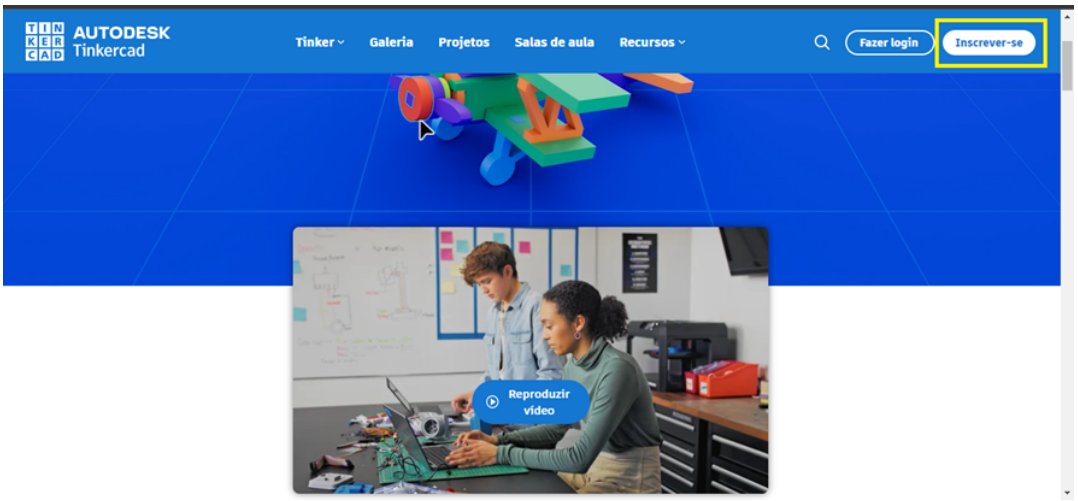


Figura 2: entrando na plataforma Tinkercad.

Em seguida, clique para criar uma conta pessoal:

Figura 3: Criando uma conta.

Após criar sua conta, seguindo os devidos passos, estamos prontos para montar

nossos primeiros projetos.

Figura 4: Criando o primeiro projeto.

3) LINGUAGEM DE PROGRAMAÇÃO.

A linguagem de programação utilizada para a gravação de programas é baseada na linguagem de programação C++. Logo, muitas características e sintaxes da linguagem que iremos utilizar são análogas as da linguagem C++. Porém, existem funções criadas especialmente para a programação em Arduino. Muitas destas funções fazem referência as portas que a placa possui e também permitem utilizar a comunicação serial para transferência de dados entre o micro controlador e o computador.

Abaixo estão listadas as funções básicas que serão necessárias em todos os códigos a serem programados em experiências futuras:

pinMode(N, XXXXXX): Função que declara o número da porta digital que será utilizada pela placa (que representamos pela letra "N") e se a porta deve operar como entrada(INPUT) ou saída(OUTPUT) de dados, que substituirão a sequência de "Xs" no comando "pinMode". Toda porta utilizada deve ser declarada.

Ex:

pinMode(3,OUTPUT); //Porta digital número 3 configurada como saída
pinMode(7,INPUT); //Porta digital número 7 configurada como entrada

digitalWrite(N,XXXX): Envia um sinal digital para uma porta de saída. Este sinal possui apenas dois valores possíveis: HIGH(1) ou LOW(0).

Ex:

digitalWrite(3,HIGH); //A porta de saída 3 enviará um valor lógico 1(este valor corresponde a 5 volts no circuito em que é aplicado)

digitalRead(N): Identifica o valor que está sendo mandado para uma porta digital de entrada. Este valor precisa ser salvo em uma variável para ser visualizado.

Ex:

int val= digitalRead(7); //O valor lido na porta 7 é armazenado na variável "val"

analogRead(N):Lê um valor de tensão que está sendo aplicado em uma porta analógica de entrada. A porta analógica representa os valores lidos (que são analógicos, portanto, podem variar entre uma longa faixa de valores) em um número inteiro que pode variar entre 0 e 1023. Este valor precisa ser salvo em uma variável para ser visualizado.

Ex:

int val= analogRead(5); //O valor lido na porta 5 é armazenado na variável "val"

analogWrite(N,XXXXX): Escreve um valor analógico (onda PWM) em um pino N. A mesma gera um sinal de onda quadrada de uma razão cíclica (duty cicle) especificada, até que uma nova chamada à função seja realizada. A frequência do sinal PWM na maioria dos pinos é de cerca de 490 Hz.

Ex:

analogWrite(3,valor); //A porta de saída 3 enviará um valor analógico.

delay(t):O compilador do programa lê e executa o código linha por linha. Ao executar a função delay**, o** programa pausa a sua leitura e execução por um tempo que é determinado como parâmetro desta função. O tempo especificado entre parênteses é dado em milissegundos.

Ex:

delay(1000); //pausa a leitura do programa por 1 segundo

.setup(): Função sem parâmetros, dentro da qual devem ser declarados todos os pinos que serão utilizados e possíveis outros comandos, de acordo com a necessidade*.

Uma função sem parâmetros não recebe valores entre seus parênteses. Adiante no curso, estudaremos como fazer nossas próprias funções, as quais muitas vezes receberão parâmetros.

Nota *: O critério para definir a “necessidade” de receber ou não comandos pode ser “Terei de executar mais de uma vez este comando?”. Caso a resposta seja negativa, coloque ele na setup. Caso contrário, coloque na loop; o motivo será visto no próximo parágrafo.

.loop(): Função sem parâmetros, dentro da qual deve ser escrito todas as linhas de código que descrevem as ações a serem executadas pelo microcontrolador repetidas vezes. Estas linhas de código serão executadas em loop, de forma que, enquanto o microcontrolador estiver executando o código, ele irá ler a função loop indefinidamente; diferente da setup, que será lida uma única vez.

A grande maioria dos nossos códigos irão se aproveitar dessa propriedade de repetição da loop. Ela é fundamental, pois por meio dela podemos ter uma atualização frequente de que dados estão sendo recebidos pelo arduino, assim como enviar dados sempre que for necessário.

4) ATIVAÇÃO DA PORTA SERIAL

A biblioteca Serial é responsável pelos comandos relacionados à comunicação serial. Os principais comandos são:

Serial.begin(): Função sem retorno que inicia a comunicação serial e tem como parâmetro a velocidade de transmissão. Ao longo do nosso curso, adotaremos 9600 como valor padrão de velocidade de transmissão.

println(): Função que recebe como argumento variáveis e imprime na tela seu valor. É possível, também, imprimir mensagens de acordo com a conveniência de cada situação.

EX:

```
int valor = 0;//declarando variável "valor" e atribuindo valor 0
```

```
void setup(){
  Serial.begin(9600);//função que permitirá o uso de comandos
                     //seriais


}

void loop(){

 Serial.print("Valor = \t");//comando que exibe a frase "valor = "
                          //e dá um espaço maior por meio do "\t"


  Serial.println(valor);//Impra a variável "valor" no monitor


  valor += 1; //A cada ciclo da loop é somado 1 à variável valor

}
```

.

EXEMPLO 1: IMPRIMIR *"HELLO,WORLD!" NA PORTA SERIAL*.

Digite o seguinte código na IDE Arduino:

```
void setup() {
  Serial.begin(9600); //ativa a comunicação serial
}
void loop() {
  Serial.println("hello,world!"); // imprime uma string na porta serial
  delay(1000);
}
```

Inicie a simulação e abra o monitor serial. Perceba que a frase “Hello, World!” se repete em intervalos de 1 segundo. a função setup() define as configurações iniciais do Arduino. Nela são declarados os padrões de comunicação, uso de pinos digitais, valores iniciais e outros parâmetros. Nesse caso, o comando Serial.begin(9600) definirá uma taxa de transmissão de 9600 bits/s.

A função loop() define os comando que se repetirão enquanto o Arduino estiver ligado. O comando Serial.println(“Hello, World!”) faz com que a String localizada entre as aspas (“) seja imprimida na porta serial, e depois efetua uma quebra de linha (\n).

O comando delay(1000) indica o tempo de atraso do Arduino até executar o próximo comando.

EXEMPLO 2: UTILIZAÇÃO DE UM PINO DIGITAL PARA PISCAR O LED.

Segue um exemplo de um código utilizado para fazer um LED piscar a cada 2 segundos e o esquemático do circuito na figura 5:

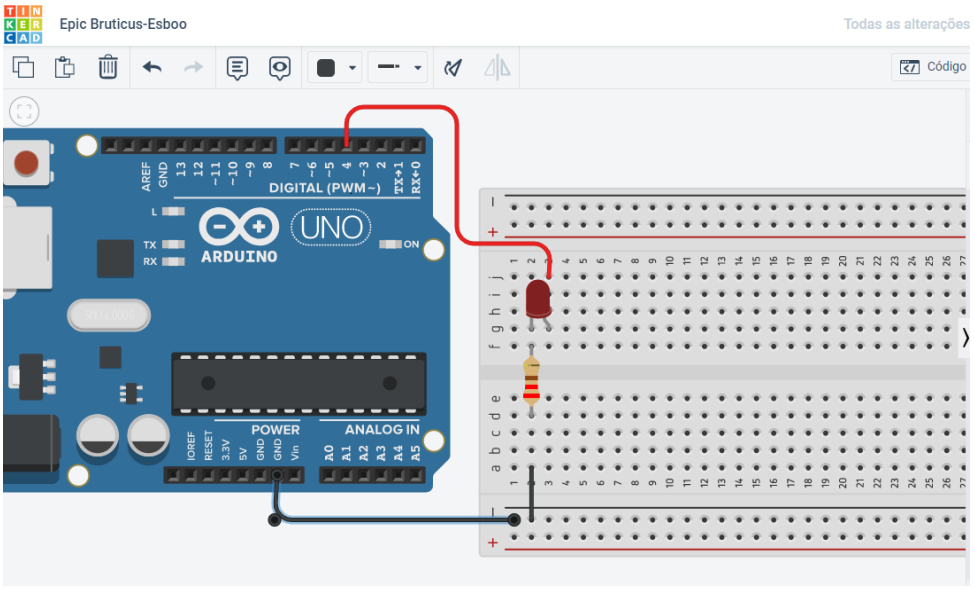


Figura 5: Esquemático do circuito

Ex:

```
void setup()
{
pinMode(4,OUTPUT);
}
 void loop()
{ digitalWrite(4,HIGH);
 delay(2000);
 digitalWrite(4,LOW);
 delay(2000);
}
```

Algo importante a se mencionar é o fato de que a linguagem utilizada na programação da placa é “Case Sensitive”, ou seja, existe a diferenciação entre letras maiúsculas e minúsculas. Todas as funções acima devem ser escritas da mesma forma que foram apresentadas aqui.

Estudo de componentes:

Resistor: Componente que fornece resistência à passagem de corrente no

circuito dissipando parte da energia sobre seus terminais. Geralmente utilizado para diminuir a tensão sobre certos dispositivos por medidas de segurança. O valor em ohms de um resistor pode ser identificado através da tabela de cores da figura 6.

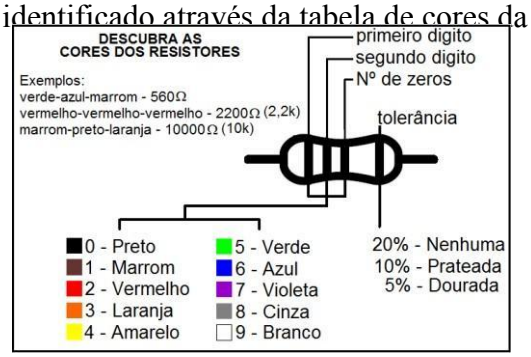


Figura 6: Tabela de Cores

Protoboard: trata-se de uma placa de plástico, cheia de pequenos furos com ligações internas, onde irão ser feitas as ligações elétricas. Os furos nas extremidades superior e inferior são ligados entre si na horizontal, enquanto que as barras do meio são ligadas na vertical. Para ilustrar isto, veja na figura 4 como são as ligações internas da protoboard.

LED: Sigla em inglês para Light Emitting Diode (diodo emissor de luz), é um diodo semicondutor que ao ser transpassado por corrente, emite luz em uma faixa de frequência estreita. O LED, assim como a maioria dos componentes que serão utilizados, possuem uma tensão ideal de funcionamento que, caso seja ultrapassada, pode danificar o componente. A seguir se encontra o exemplo de como dimensionar o resistor ideal para se utilizar com um LED de cor vermelha.

5) ENTRADAS E SAIDAS ANALÓGICAS.

Antes de iniciar o exemplo prático é necessário entender a fermenta PWM. O "Pulse Width Modulation" é um mecanismo utilizado para aproximar um sinal analógico nas saídas do microcontrolador. O Arduino, em específico, não possui saídas analógicas propriamente ditas (apesar de possuir entradas analógicas), mas utiliza os pinos digitais para simular valores intermediários de tensão. A técnica é alternar entre 5V e 0V em uma frequência tal que o sinal de saída seja uma média dos valores, sendo que quanto mais tempo em 5V, maior o valor da tensão de saída. Logo sempre que a função analogWrite() é utilizada a mesma recebe dois parâmetros, o primeiro é o pino a ser usado, necessariamente um pino PWM, indicado no Arduino pelo símbolo "~" ao lado do número do pino, e o segundo parâmetro é o valor entre 0 e 255 representando o valor de tensão aplicado na saída, onde 0 imprime o valor 0V e 255 imprime o valor 5V.

EXEMPLO 3: ILUMINAR UM LED DE ACORDO COM UM VALOR DE ENTRADA LIDO A PARTIR DE UM POTENCIÔMETRO.

Em outas palavras, vamos construir um dimmer extremamente simples. O potenciômetro será ligado ao pino 5, um pino de entrada analógica, e o LED ao pino de saída PWM 10. A variável "valor" irá armazenar o valor lido a partir do potenciômetro e seu valor será usado para definir o grau de iluminação do LED. Como programado no código a seguir:

```
int entradaPotenciometro = A5;//declara a variável inteira
                              //"entradaPotenciometro" na
                              //entrada analógica A5.

int LED = 10;//declara a variável inteira "LED" no pino PWN 10

 int valor = 0; //declara a variável "valor" igual a zero
void setup()
{
pinMode(LED, OUTPUT);// indica que o pino 10 é uma saída.

}
void loop()
{
valor = analogRead(entradaPotenciometro); //nesta linha a variável valor é atualizada
com os dados lidos do pino de entrada analógica ligado ao potenciômetro.

analogWrite(LED, valor / 4); // a variável "valor" dividida por 4 é escrita na saída
analógica ligada ao led.
```

}

Note que ao chamarmos a função analogWrite(), definimos o pino que será escrito (LED, pino 10), e o valor que será escrito é o valor lido do potenciômetro dividido por 4. Fazemos essa divisão pois a entrada recebe valores que vão de 0 a 1023, porém a saída da função somente pode entregar valores entre 0 e 255, ou seja, um quarto da resolução de leitura.

Na fugira 5 é possível observar o do circuito do exemplo estudado:

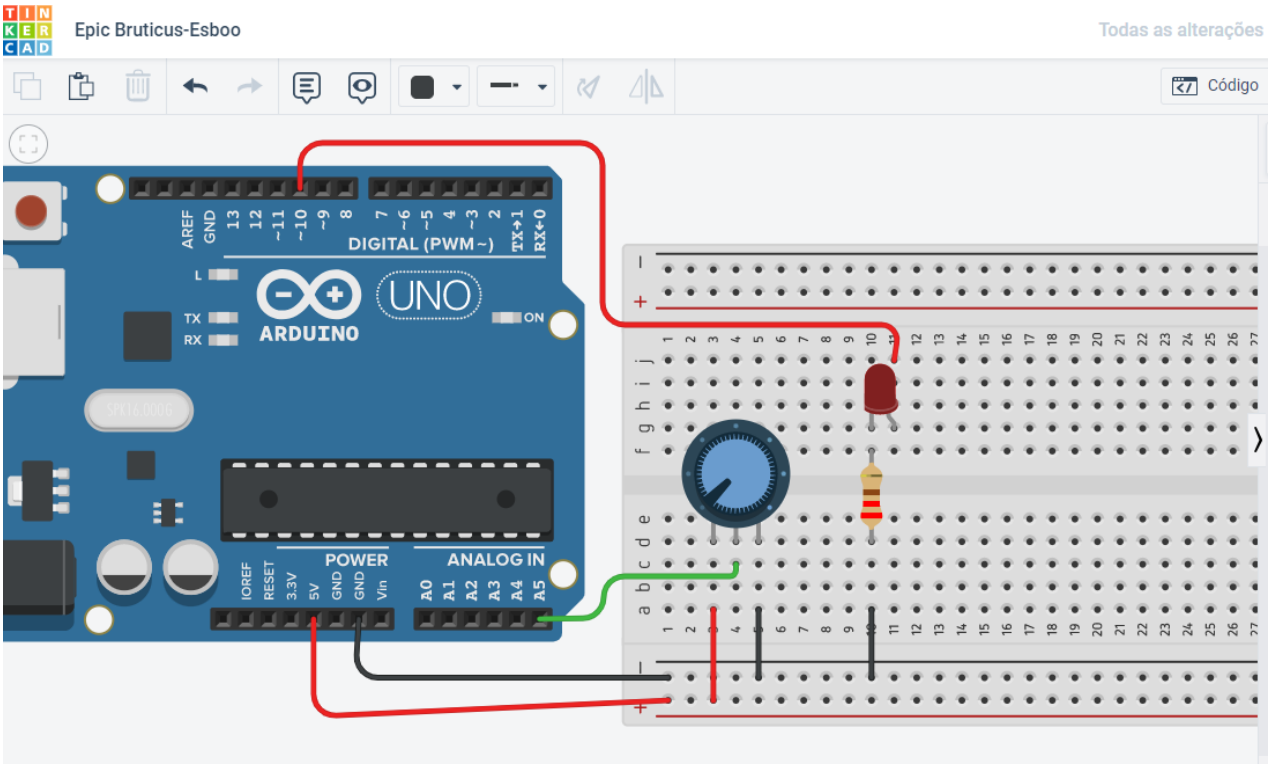


Figura 5: circuito referente ao exemplo 3.

Estudo de componentes:

Os principais componentes utilizados neste exemplo, além da placa Arduino, são o led e o protoboard (que foram descritos anteriormente) e o potenciômetro que é um componente eletrônico que possui resistência elétrica ajustável. Geralmente, é um resistor de três terminais onde a conexão central é deslizante e manipulável. Se todos os três terminais são usados, ele atua como um divisor de tensão

EXEMPLO 4: USO DO SENSOR LDR E A PORTA SERIAL.

Neste exemplo será feita a leitura do valor do sensor de luz LDR e exibiremos este valor no Serial Monitor da IDE do arduino. O sensor de luz LDR (Light Dependent Resistor) ou foto resistor é um tipo de resistor que varia a sua resistência de acordo com a intensidade de luz que recebe. O código fonte abaixo faz a leitura e exibição dos valore lidos no sensor de luz. A leitura é feita através da porta analógia A0 e a exibição dos valores lidos é feita através do serial monitor da IDE do arduino com o comando Serial.println, estudado anteriormente.

```
//Pino analógico em que o sensor está
conectado. int sensor = 0;

//variável usada para ler o valor do sensor em tempo
//real.
 int valorSensor = 0;

//função setup, executada uma vez ao ligar o
//Arduino.
void setup(){

  //Ativando o serial monitor que exibirá os
  //valores lidos no sensor.
  Serial.begin(9600);
}

//função loop, executada enquanto o Arduino estiver
```

```
//ligado.
 void loop(){

  //Lendo o valor do sensor.
  int valorSensor = analogRead(sensor);

  //Exibindo o valor do sensor no serial monitor.
  Serial.println(valorSensor);

  delay(500);
}
```

É possível perceber que este exemplo utiliza os conceitos estudados anteriormente que envolvem portas analógicas e a comunicação serial do Arduino. O circuito correspondente ao exemplo está ilustrado na figura 6:

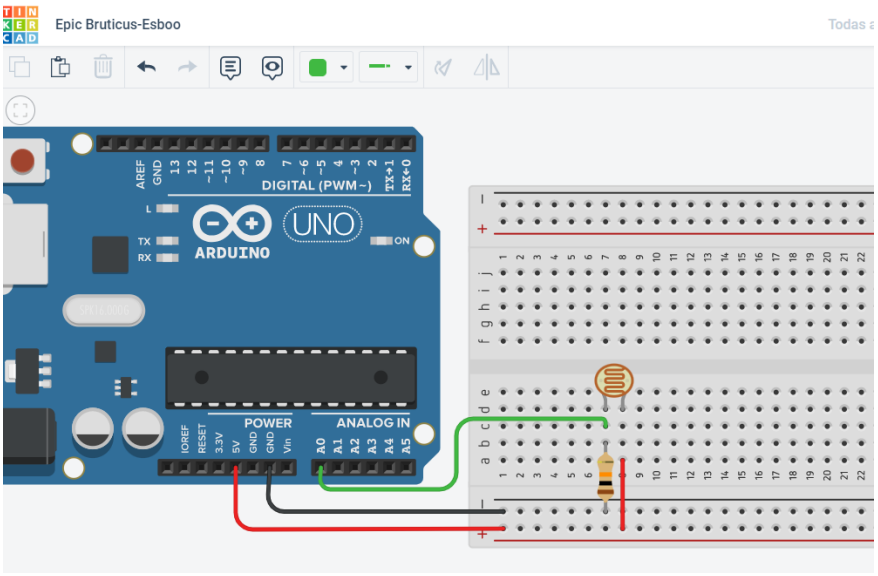


Figura 6: circuito referente ao exemplo 4.

Estudo de componentes:

Neste exemplo os componentes utilizados são, alem do Arduino, uma protoboard, um resistor de 10K ohms; e o sensor de luz LDR de 5 ou 10 mm. O único componente ainda não explicado anteriormente é o sensor de luz que é um componente eletrônico, sensível a luz, que tem por finalidade limitar a corrente elétrica que passa sobre ele, como um resistor comum, só que o grande diferencial é que ele é um resistor variável que interage com a luz. Conhecido também como LDR, light dependent resistor, este componente eletrônico tem a sua resistência alterada de acordo com a luz que incide sobre ele. Quanto mais luz menor a resistência que ele oferece e quanto menos luz

maior a resistência que ele oferece. Assim como num resistor comum o fotoresistor ou LDR não é polarizado e a sua a resistência é medida em ohms e nos casos mais comuns tem em torno de 1MΩ ohms quando exposto a luz, e pode chegar de 1,5MΩ a 2MΩ na ausência de luz, mas dependendo do seu tamanho e fabricante isso pode variar. Abaixo, em uma imagem ampliada, está o modelo de fotoresistor mais comum encontrado em lojas de componentes eletrônicos e os tamanhos são diversos, mas os mais comuns são de 10mm, 7mm e 5mm.

6)ALGUMAS ESTRUTURAS DE CONTROLE.

5.1)LAÇO DE REPETIÇÃO

O comando for permite que um certo trecho de programa seja executado um determinado número de vezes. E por isso é conhecido como laço de repetição.

A forma do comando for é:

for (comandos de inicialização; condição de teste; incremento ou decremento)

{

// comandos a serem repetidos

// comandos a serem repetidos

}

// comandos após o 'for'

Como ilustrado na figura

7:

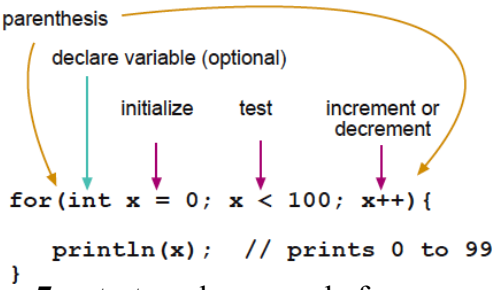


Figura 7: estrutura do comendo for.

O funcionamento é o seguinte:

1. Executa os comandos de inicialização;
2. Testa a condição;
3. Se a condição for falsa então executa o comando que está logo após o bloco subordinado ao for.
4. Se a condição for verdadeira então executa os comandos que estão subordinados ao for;
5. Executa os comandos de incremento/decremento;
6. Volta ao passo 2.

O comando for deve ser usado sempre que:

- Soubermos exatamente quantas vezes o laço deve ser repetido;
- O teste deva ser feito antes da execução de um bloco de comandos;

- Houver casos em que o laço não deva ser repetido nenhuma vez.

É válido ressaltar que os comandos de inicialização são executados apenas 1 vez; o contador é incrementado ou decrementado sempre ao final da execução do bloco e o teste é feito sempre antes do início da execução do bloco de comandos.

EXEMPLO 5: USO DO COMANDO FOR.

Neste exemplo vamos aumentar o brilho do led utilizando o laço de repetição, utilizando o código a seguir:

```
int PWMpin = 10; // LED em série com resistor 470 ohm na porta 10~

void setup()
{
  // nada a ser feito no setup
}

void loop()
{
   for (int i=0; i <= 255; i++) //a variável contadora i inicia em zero(0v) e é incrementada até chegar em 255(5v)
{
     analogWrite(PWMpin, i); // o valor atual de i é escrito na porta analógica a cada repetição
     delay(10);
   }

}
```

Na figura 8 representa o circuito a ser montado para a 4 execução do exemplo:

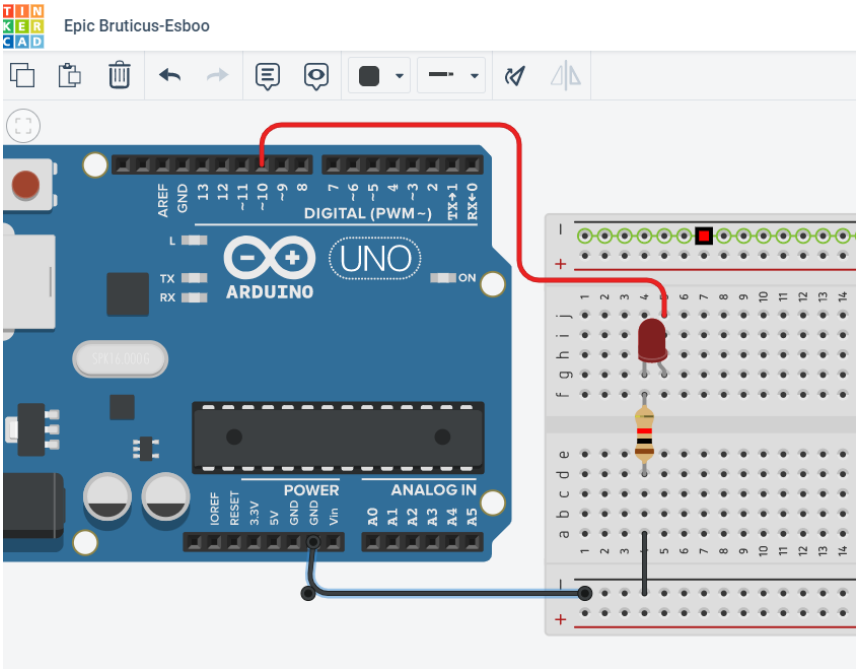


Figura 8: Circuito Para Alterar O Brilho Do Led Com O Laço De Repetição.

5.2) COMANDO CONDICIONAL

A condição na programação é definida como uma expressão que pode ser verdadeira ou falsa. A mesma é chamada de expressão lógica.

Por exemplo, (3 > 2) é uma expressão lógica que possui valor verdadeiro. Por outro lado, (4 < 1) é uma expressão lógica que possui valor falso.

Os operadores (< e >) usados nos exemplos acima são chamados de operadores relacionais pois possibilitam saber qual a relação existente entre seus dois operandos. Além destes dois existem mais 4 operadores relacionais, que podem ser vistos na tabela seguir:

Operadores Relacionais

==	Igual a
!=	Diferente
>=	Maior ou igual
>	Maior que
<	Menor que
<=	Maior ou igual

Além dos operadores relacionais, existem os chamados operadores lógicos ou "conectivos lógicos". Estes, servem para conectar duas expressões relacionais. Os operadores lógicos são apresentados na tabela a seguir:

Operadores Lógicos

||	OU lógico
&&	E lógico
!	Negação

Precedência de Operadores

!	Operador de negação	Executado Antes
-	menos unário (sinal)	
* / %	Operadores Multiplicativos	
+ -	Operadores aditivos	
< > <= >= == !=	Relacionais	
&&	AND lógico	Executado Depois
||	OR lógico	

O comando “if” é uma estrutura de decisão que permite ou não que uma sequência de comandos seja executada, dependendo do resultado de uma condição pré-estabelecida que pode utilizar os operadores listados anteriormente. Sua sintaxe é:

```
if (condição)
{
  lista de instruções
}
```

A **condição** é verificada a cada passagem pela estrutura IF. Se a condição for satisfeita (V), então a lista de instruções que se encontra entre chaves será feita. Se a condição NÃO for satisfeita (F), então serão feitas as instruções existentes logo após o fecha chaves.

Uma variação do comando if é a estrutura “if else”:

```
if (condição)
  {
   lista de instruções
   }
else
   {
    lista de instruções
    }
```

Neste caso, se a condição colocada após o IF não obtiver resultado verdadeiro, automaticamente serão feitas as instruções que estão dentro do ELSE, desconsiderando aquelas que estão abaixo do IF. Caso a condição seja VERDADEIRA, serão feitas as instruções que estão entre chaves abaixo do IF.

Quando acabar tanto a lista de instruções abaixo do IF quanto a lista de instruções referente ao ELSE, automaticamente serão desenvolvidas as instruções que estão após a lista de instruções do ELSE (....).

EXEMPLO 6: USO DO SENSOR LDR PARA A ATIVAÇÃO DE UM LED.

Dando continuidade ao exemplo 4 e utilizando os conceitos do comando condicional "if", vamos gora utilizar o valor lido pelo sensor LDR para acionar um led. O programa para este exemplo lê o valor da porta analógica (que deve estar na faixa de 0 a 1024), verificando se o valor é maior do que 800 (LDR encoberto) e consequentemente acendendo o led. Como mostra o código a seguir:

```
int portaLed = 10; //Porta a ser utilizada para ligar o led
int portaLDR = A5; //Porta analógica utilizada pelo LDR

void setup()
{
 pinMode(portaLed, OUTPUT); //Define a porta do Led como saída

 Serial.begin(9600);// inicializa a porta serial
}

void loop()
{
 int estado = analogRead(portaLDR); //Lê o valor fornecido pelo LDR

   Serial.println(estado); //Exibindo o valor do sensor no serial monitor.

 // Caso o valor lido na porta analógica seja menor que
 // 300, acende o LED
 // Ajuste o valor abaixo de acordo com o seu circuito
 if (estado < 300)
 {
  digitalWrite(portaLed, HIGH);
 }
 else //Caso contrário, apaga o led
 {
  digitalWrite(portaLed, LOW);
 }
}
```

Uilizando os mesmos componentes já descritos anteriormente que são o led, o resistor e o LDR podemos montar o circuito referente ao programa como mostra a figura 9:

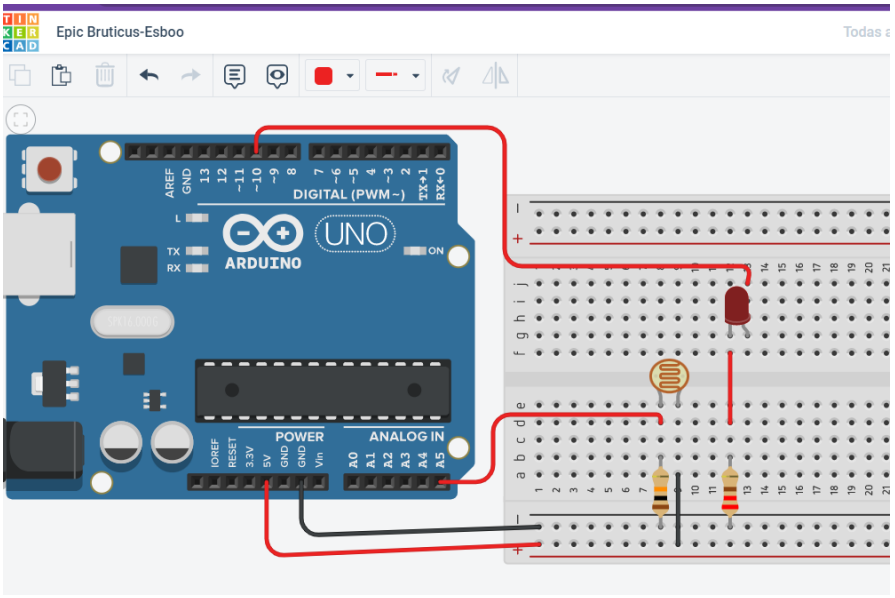
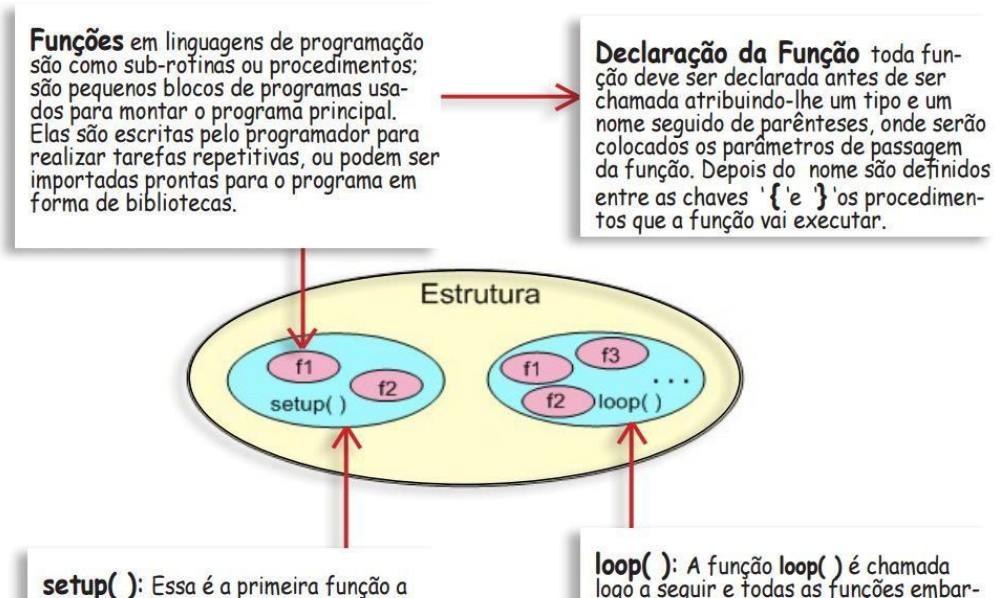


Figura 8: Circuito Para condicionar o funcionamento do led.

6) FUNÇÕES

As **funções** (**functions**), também conhecidas como sub-rotinas, são muito utilizadas em programação. Um dos grandes benefícios é não precisar copiar o código todas as vezes que precisar executar aquela operação, além de deixar a leitura do código mais intuitiva. Além das funções void setup e void loop, no Arduino pede-se criar outras funções que contenham blocos de programação que serão utilizados no decoorer do programa. a figura 9 explica um pouco melhor este conceito.

Figura



EXEMPLO 7: USO DO SENSOR LDR PARA A ATIVAÇÃO DE TRÊS LEDS.

O mesmo princípio do exemplo 5 pode ser utilizado para acendermos 3 led's em condições diferentes de luminosidade, então incluindo o conceito de funções para apagar todos os led's sempre que necessário, pode-se desenvolver o seguinte código:

```
int sensor = 0;      //Pino analógico em que o sensor está conectado.
int valorSensor = 0; //Usada para ler o valor do sensor em tempo
real.

const int ledVerde = 10;
const int ledAmarelo = 9;
const int ledVermelho = 8;
//Função setup, executado uma vez ao ligar o Arduino.
void setup(){
  //Ativando o serial monitor que exibirá os valores lidos no sensor.
  Serial.begin(9600);

  //Definindo pinos digitais dos leds como de saída.
  pinMode(ledVerde,OUTPUT);
  pinMode(ledAmarelo,OUTPUT);
  pinMode(ledVermelho,OUTPUT);
}
//Função loop, executado enquanto o Arduino estiver ligado.
void loop(){

  //Lendo o valor do sensor.
  int valorSensor = analogRead(sensor);

    if (valorSensor < 150) {//SE a Luminosidade for baixa
    apagaLeds();
    digitalWrite(ledVermelho,HIGH);//acende o led vermelho
  }
  if (valorSensor >= 150 && valorSensor <= 800) { //SE a Luminosidade for média.
    apagaLeds();
    digitalWrite(ledAmarelo,HIGH); //acende o led amarelo
  }

   if (valorSensor > 800) { //SE a Luminosidade for
    alta apagaLeds();
    digitalWrite(ledVerde,HIGH); //acende o led verde
  }
  Serial.println(valorSensor); //Exibindo o valor do sensor no serial monitor.
  delay(50);
}
void apagaLeds() {//Função criada para apagar todos os leds de uma vez.
  digitalWrite(ledVerde,LOW);
  digitalWrite(ledAmarelo,LOW);
  digitalWrite(ledVermelho,LOW);
```

}

O circuito referente ao exemplo em questão está ilustrado na figura 10, e utiliza os mesmos componentes já anteriormente descritos, led's, resistores e sensor de luz.

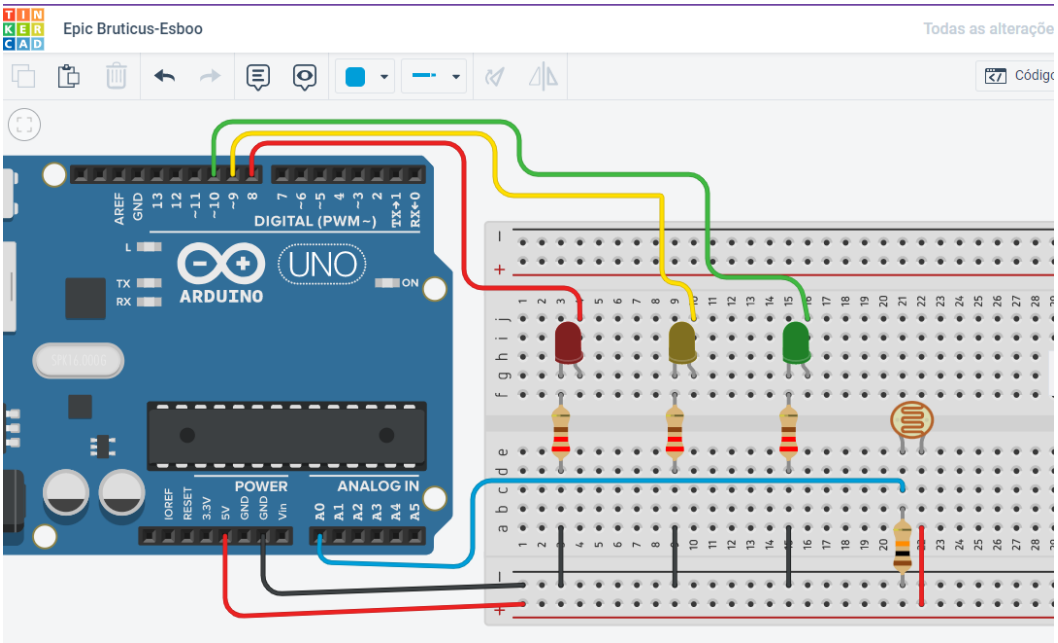


Figura 10: Circuito Referente Ao Exemplo 6.

O primeiro passo é conectar os componentes na protoboard. Conecte o sensor ldr com uma de suas pernas no 5V do arduino e a outra perna no pino analógico A0. Conecte o resistor de 10K ohms com uma perna entre o sensor ldr e o fio jumper do pino digital A0 e a outra perna no pino GND.

Conecte os três leds na protoboard e na perna maior (positivo) de cada led conecter um resistor de 100 ohms e após o resistor conecte um fio jumper. Ligue a perna menor (negativo) de cada um dos leds no GND do arduino. Os fios positivos dos led ficarão nos seguintes pinos digitais do arduino:

- fio do led vermelho no pino digital 8;
- fio do led amarelo no pino digital 9;
- fio do led verde no pino digital 10.

EXERCÍCIOS PROPOSTOS:

1) Com 4 bits, conseguimos representar 16 possibilidades, que se usássemos para a representação de números decimais, poderíamos representar números de 0 até 15. Faça uma lógica para que 4 LEDs representem cada um desses números levando em conta a representação em binário natural, que deverão ser exibidos do 0 até o 15. Nota: fica mais eficiente com a utilização de funções

2) Faça um semáforo de três cores, de forma que ele deve começar com o sinal verde, o qual ficará aberto por 90 segundos, faça a transição para o amarelo(que deverá ficar aberto por 20 segundos) e por fim fique vermelho(durante 45 segundos).

3) Aproveitando os dados do exemplo anterior, faça ainda um sinal de pedestres, de forma que o sinal verde fique aberto 5 segundos depois do semáforo ficar vermelho e o sinal fique vermelho 5 segundos antes do semáforo ficar verde.

4) Os sinais enviados pelo potenciômetro ao arduíno variam de 0 a 1023. Faça um esquemático em que divida essa faixa de valores em 4 partes iguais e monte um circuito com três LEDs. Na primeira faixa de valores os 3 LEDs devem permanecer apagados, na segunda faixa um dos LEDs deve acender, na terceira devem acender 2 e na última todos devem estar acesos. Nota: quando você fizer a transição de uma faixa maior para uma menor, os LEDs que estavam acesos na faixa maior devem apagar.

5) Crie um circuito com quatro leds e dois botões. Um botão irá somar 1 a uma variável, que valerá 0 inicialmente, o outro irá subtrair 1 dessa variável. Se a variável em questão valer 0, os quatro leds devem permanecer apagados. Se a variável valer 1, 1 Led deve acender, se valer 2, 2 leds devem acender e assim por diante. Nota: quando você fizer a transição de um valor maior para um menor da variável, os LEDs que estavam acesos no valor maior devem apagar.

6) Crie um circuito que receberá valores de um potenciômetro(pot), de um fotoresistor(foto) e de dois botões (som e sub). Um dos botões irá somar um a uma variável “var” e outro irá subtrair um. O sistema deve imprimir no monitor serial o resultado da expressão:

$$(pot + foto) / var$$

É possível divisão por 0? Envie uma mensagem no monitor se esta possibilidade ocorrer. Dica: os valores de divisão podem ser fracionários, assim, utilize variáveis do tipo float