



ARDUINO **KIT** INICIANTE **V3.0**

Parabéns por adquirir o Kit Arduino Iniciante da RoboCore!

Este material é composto por 10 experimentos, que são intitulados módulos e projetos. O intuito principal é que o usuário que está começando a entender o fascinante mundo da eletrônica, ou mesmo o usuário que já tem boas noções, possa começar a construir protótipos utilizando sua placa Arduino. Para efeitos de explicação, chamamos de **módulos** os experimentos que, por si só, não apresentam grande efeito e, quando juntados 2 ou mais módulos, podemos fazer um **projeto**, que consiste em algo útil ou ao menos agradável de ser apreciado.

Abaixo segue a lista de módulos e projetos que podem ser construídos com o auxílio deste material:

▪ **Módulo 1**

pág. 10

Componentes: 1 Botão + 1 Led

Descrição: Conforme você pressiona um pushbutton, um led é aceso.

Dificuldade: 

▪ **Módulo 2**

pág. 12

Componentes: 3 Botões + 3 Leds

Descrição: Conforme você pressiona qualquer um dos botões, leds de diferentes cores são acesos.

Dificuldade: 

▪ **Projeto Piano**

pág. 14

Componentes: 3 Botões + 3 Leds + Buzzer

Descrição: Cada botão toca uma nota musical diferente e acende um led. É expansível – por conta do usuário – para mais uma nota musical com o botão (e o led) reserva

Dificuldade: 

▪ **Módulo 3**

pág. 16

Componentes: 1 Sensor de Temperatura NTC

Descrição: Com o auxílio da porta serial e do monitor serial, o usuário irá fazer a leitura e calibração do sensor de temperatura para fazer o projeto.

Dificuldade: 

▪ **Projeto Alarme**

pág. 20

Componentes: 1 Sensor de Temperatura NTC + 1 buzzer

Descrição: A partir dos valores colhidos no módulo 3, o usuário poderá montar um alarme que, se a temperatura de onde o sensor estiver localizado for maior, ou menor, ele soará.

Dificuldade: 

▪ **Projeto Termômetro**

pág. 22

Componentes: 2 Leds Verdes + 2 Leds Amarelos + 2 Leds Vermelhos + Buzzer + 1 Sensor de Temperatura NTC

Descrição: Conforme a temperatura do ambiente onde o sensor NTC está localizado aumenta, os leds coloridos acendem, como um termômetro. Se por algum motivo todos os 6 Leds forem acesos, um alarme intermitente deverá soar.

Dificuldade: 

▪ Módulo 4*pág. 26*

Componentes: 1 Potenciômetro + 1 Led

Descrição: Conforme o valor do potenciômetro é alterado, o led pisca de forma mais rápida ou mais lenta

Dificuldade: **▪ Projeto Dimmer***pág. 28*


Componentes: 1 Potenciômetro + 1 Led Alto Brilho

Descrição: Conforme o valor do potenciômetro é alterado, o led fica mais claro ou mais escuro graças ao PWM

Dificuldade: **▪ Projeto Iluminação Automatizada***pág. 32*

Componentes: 1 Led Alto Brilho + 1 Sensor de Luminosidade LDR

Descrição: Se a iluminação ambiente, por qualquer motivo, diminuir ou apagar completamente, um led de alto brilho acende gradativamente

Dificuldade: **▪ Projeto Alarme Multipropósito***pág. 36*

Componentes: 2 Leds Verdes + 2 Leds Amarelos + 2 Leds Vermelhos + 1 Sensor de Luminosidade LDR + 1 Sensor de Temperatura NTC + 1 Led Alto Brilho + 1 buzzer

Descrição: Temos 2 *bargraphs*, ou seja, dois indicadores: um de luminosidade e outro de temperatura, através das cores dos leds. Se a temperatura estiver alta e acender os 3 Leds que a corresponde, um alarme deverá soar. De maneira análoga, se os 3 Leds correspondentes à luminosidade estiverem apagados – indicando uma falta total de luminosidade no ambiente - um alarme deverá soar e um led de alto brilho irá acender.Dificuldade: 

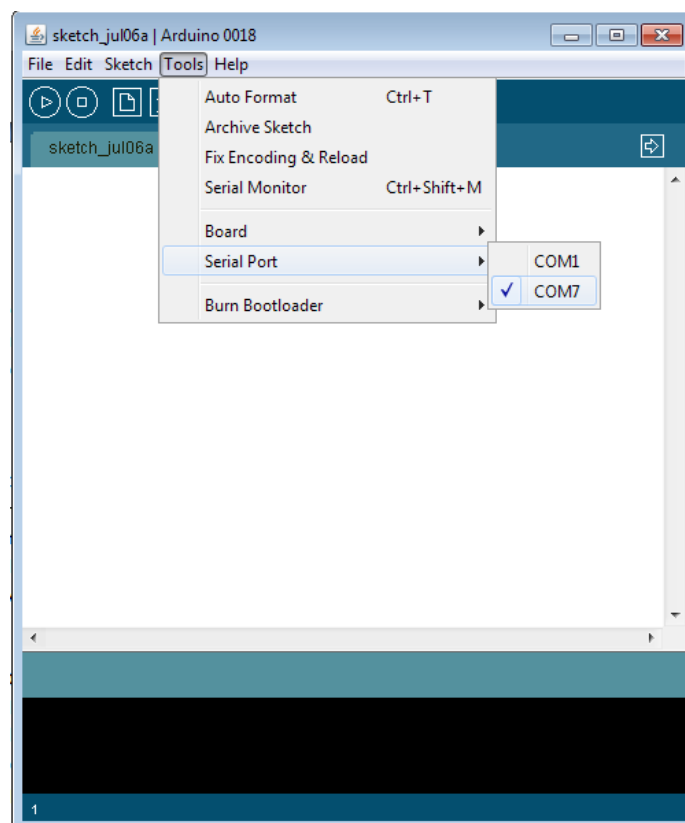
INSTALAÇÃO DO SOFTWARE ARDUINO

Como você já deve ter percebido o ambiente de desenvolvimento do Arduino não precisa ser instalado. Uma vez baixado em seu computador, ele pode ser rodado diretamente por se tratar de um aplicativo feito em Java. Também por este motivo, o ambiente de desenvolvimento pode ser rodado nos mais diversos sistemas operacionais (Windows, Linux, MAC), bastando apenas o download do ambiente que corresponde ao seu sistema operacional. O download pode ser feito na página: http://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=70

INSTALAÇÃO DO DRIVER ARDUINO

O dispositivo Arduino é totalmente Plug & Play. Uma vez rodando o ambiente de desenvolvimento, insira o cabo USB AB no Arduino e depois no computador. Seu computador deverá reconhecer automaticamente o Arduino e uma nova porta COM (no caso de sistema operacional Windows Vista, Windows 7 ou Linux). Caso o sistema operacional não reconheça a placa automaticamente, os drivers podem ser localizados na pasta “**arduino-0018\drivers\FTDI USB Drivers**”.

Para selecionar esta nova porta COM onde o Arduino está localizado, abra o ambiente de desenvolvimento, então clique em **TOOLS > SERIAL PORT > COM X** (onde X é o número da porta que o Arduino foi instalado automaticamente). Na imagem a seguir temos um exemplo do que você deverá ver:



Note que o número da porta COM não é necessariamente 7 como na imagem ao lado. Cada computador poderá mostrar um número de porta diferente.

UMA BREVE DESCRIÇÃO DOS COMPONENTES

Para montar os experimentos deste kit não é necessário nenhum tipo de curso anterior de eletrônica. Mas, para você identificar cada um dos componentes e deixar a compreensão do circuito um pouco mais fácil, aqui detalharemos um pouco cada um deles.

• RESISTOR

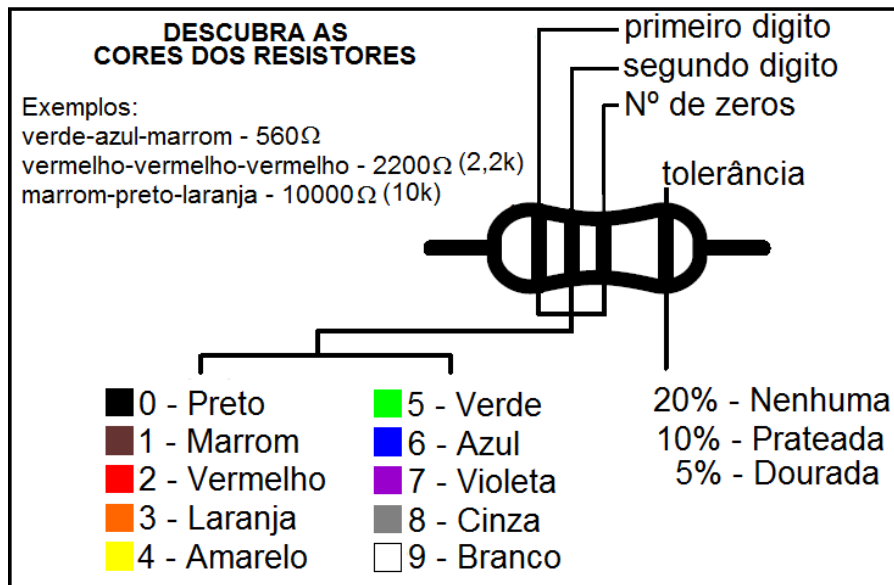


O que isto faz: Limita a corrente elétrica que passa pelo circuito. Para limitar mais ou menos corrente, o valor deste componente pode variar.

Número de pinos: 2 pinos do mesmo comprimento

+ **Detalhes:** <http://pt.wikipedia.org/wiki/Resistor>

Para saber o valor de cada resistor, basta seguir o esquema abaixo:



• BUZZER



O que isto faz: Quando uma corrente elétrica passa por ele, ele emite um som. Também conhecido como sensor piezoelétrico.

Número de pinos: 2 pinos (este componente tem polaridade, portanto fique atento na hora de ligá-lo.)

+ **Detalhes:** http://pt.wikipedia.org/wiki/Sensor_piezoel%C3%A9trico

- **CHAVE MOMENTÂNEA**

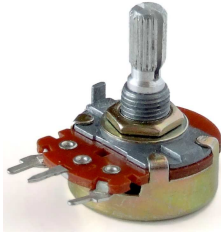


O que isto faz: Quando o botão é apertado, os contatos entre os terminais de cada lado são ligados.

Número de pinos: 4 pinos (os 2 pinos de cada lado já estão em contato normalmente. Quando o botão é apertado os 4 entram em contato)

+ Detalhes: http://en.wikipedia.org/wiki/Push_button (em inglês)

- **POTENCIÔMETRO**

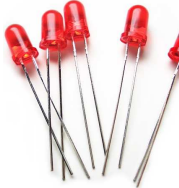


O que isto faz: Varia a resistência dos terminais conforme a haste superior é girada

Número de pinos: 3 pinos (a resistência varia entre um dos pinos mais da extremidade para com o do centro)

+ Detalhes: <http://pt.wikipedia.org/wiki/Potenci%C3%B4metro>

- **LED**



O que isto faz: Emite uma luz quando uma pequena corrente o excita (apenas em uma direção, do pino mais longo para o pino mais curto)

Número de pinos: 2 pinos (um mais longo e outro mais curto)

+ Detalhes: http://pt.wikipedia.org/wiki/Diodo_emissor_de_luz

- **SENSOR DE TEMPERATURA NTC**



O que isto faz: É uma resistência que varia conforme a temperatura a que é submetido

Número de pinos: 2 pinos do mesmo comprimento

+ Detalhes: http://pt.wikipedia.org/wiki/Negative_Temperature_Coefficient

- **SENSOR DE LUMINOSIDADE LDR**



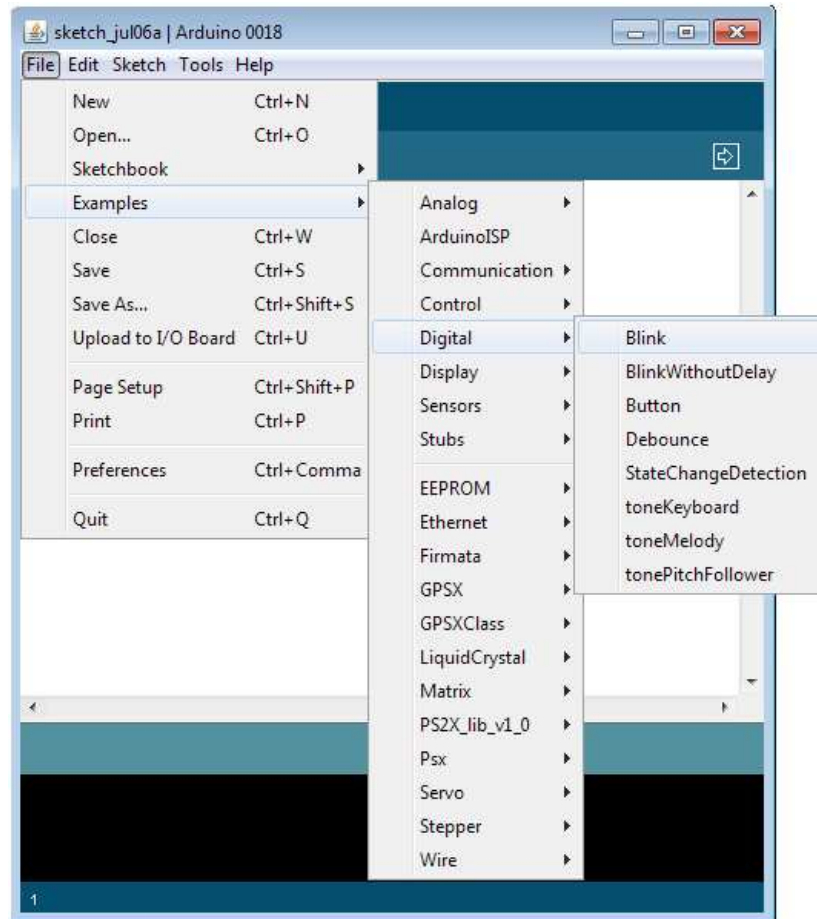
O que isto faz: É uma resistência que varia conforme a luminosidade se altera sobre ele

Número de pinos: 2 pinos do mesmo comprimento

+ Detalhes: <http://pt.wikipedia.org/wiki/Ldr>

INTRODUÇÃO

Para entender como funciona o arduino, vamos começar com o mais básico, o exemplo BLINK que está pronto no software de compilamento do Arduino. Para acessá-lo clique em **FILE > EXAMPLES > DIGITAL > BLINK** como mostrado na figura abaixo:



Feito isto, o código do programa irá aparecer na tela do ambiente de desenvolvimento. É interessante que você analise o programa para tentar compreendê-lo. Para tanto iremos colocar abaixo todo o programa, assim como você deve estar vendo na tela do ambiente de desenvolvimento, para analisá-lo com você:

Código:

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
The circuit:
* LED connected from digital pin 13 to ground.
* Note: On most Arduino boards, there is already an LED on the board
connected to pin 13, so you don't need any extra components for this example. Created 1 June
2005
By David Cuartielles
http://arduino.cc/en/Tutorial/Blink
based on an original by H. Barragan for the Wiring i/o board
*/

int ledPin = 13; // LED connected to digital pin 13
// The setup() method runs once, when the sketch starts
void setup() {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}
// the loop() method runs over and over again,
// as long as the Arduino has power
void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(ledPin, LOW);  // set the LED off
  delay(1000);                // wait for a second
}
```

Para iniciar o entendimento do código, devemos observar o que são e como são feitos os comentários em um código de linguagem C.

Para fazer um comentário quer irá se desenvolver por mais de 1 linha, devemos usar os caracteres:

`/*` para começar um comentário de mais de 1 linha

`*/` para finalizar os comentários que foram feitos anteriormente

Para fazer um comentário em 1 linha apenas, podemos utilizar:

`//` para fazer um comentário de apenas 1 linha

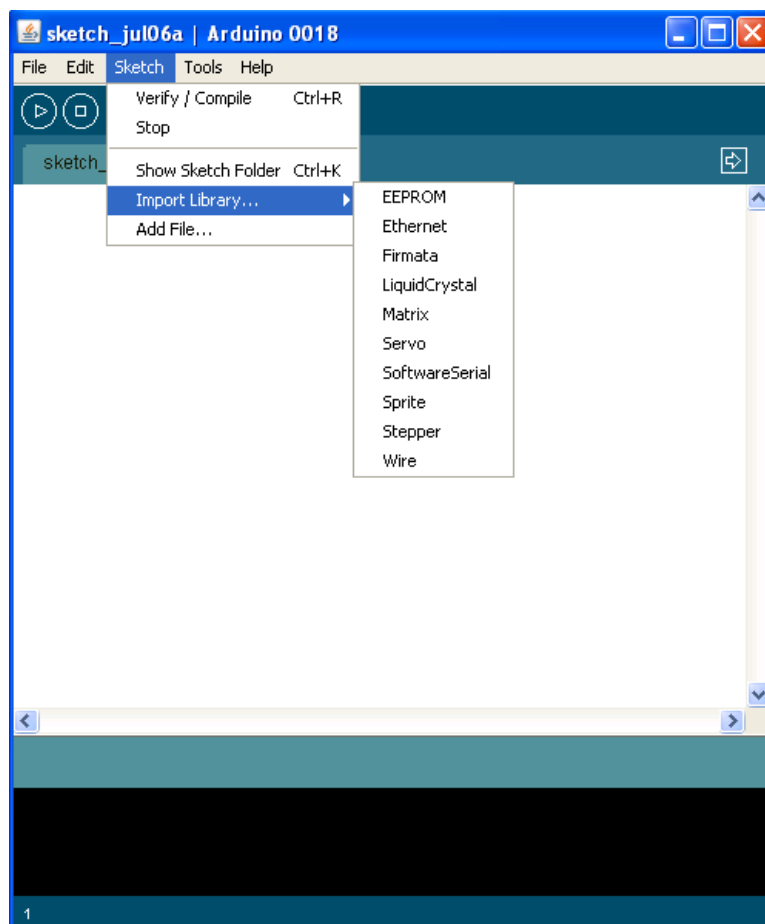
Entendido isto, e se olharmos o código do BLINK mais afundo, veremos que o código está escrito em apenas 11 linhas. Veja se você consegue identificar quais são estas 11 linhas.

Vamos agora entender a estrutura dos programas. No início de todos os programas uma ordem deve ser respeitada:

1. Estrutura de Inclusão de Bibliotecas
2. Estrutura de Declaração de Variáveis
3. Estrutura Setup
4. Estrutura Loop
5. Demais estruturas de funções

O que são estas 5 estruturas citadas a cima?

O diferencial de uma placa como o Arduino está profundamente ligada à estrutura de numero 1 citada a cima. Quando você estiver pensando em fazer algum projeto mirabolante, você pode ter certeza que há 90% de chances de alguém já o ter feito. Desta forma, quando alguém já o fez, é bem provável que este alguém, em qualquer parte do mundo, já tenha escrito toda uma biblioteca para fazer o tal projeto. Por exemplo, digamos que em um sábado a noite dê uma vontade louca de fazer um carrinho de controle remoto controlado por um controle de PlayStation®. Você não faz a menor ideia de como começar a pensar em como programar este carrinho controlado por este controle tão comum no seu dia-a-dia. Então, a primeira coisa que você deve fazer é ir ao Google e perguntá-lo se alguém já desenvolveu uma biblioteca para você utilizar um controle de PlayStation® com seu Arduino. Digite no Google: **ARDUINO PLAYSTATION CONTROLLER LIBRARY**. Na data que este documento está sendo redigido, você encontra “Aproximadamente 18.000 resultados”. E sim, um deles, pelo menos, é a biblioteca que você está precisando para desenvolver seu carrinho de controle remoto com controle de PlayStation®. Portanto, o que são Bibliotecas? São conjuntos de funções desenvolvidas para uma aplicação particular. Seu ambiente de desenvolvimento Arduino já vem com algumas bibliotecas instaladas. Para vê-las, simule que você quer importar uma biblioteca (apenas simule, não precisa clicar em nenhuma para importar). Para tanto, clique em **SKETCH > IMPORT LIBRARY...** e veja quantas bibliotecas prontas para seu uso já existem:



Neste momento não iremos utilizar nenhuma das bibliotecas mostradas à cima, mesmo porque nosso programa BLINK não necessita de uma biblioteca para funcionar, pois é um programa muito básico e utiliza apenas escritas digitais e delays, funções que já estão inserida em todos os programas feitos no ambiente de desenvolvimento Arduino. Por este motivo você pode notar que o programa BLINK, após os comentários iniciais, começa com a declaração de variáveis:

```
int ledPin = 13; // LED connected to digital pin 13
```

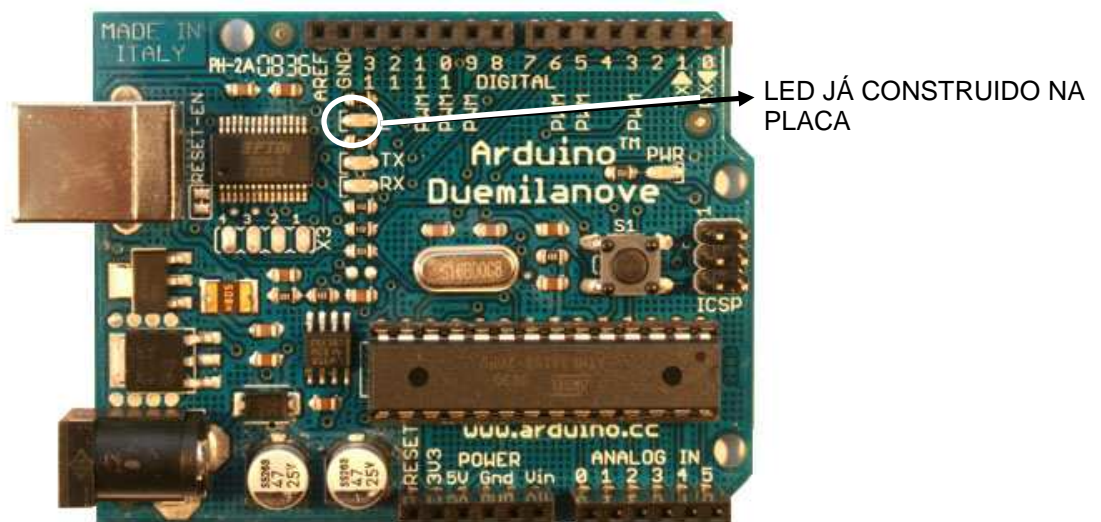
A linha anterior quer dizer o seguinte:

int : variável do tipo inteira

ledPin = 13; : nome da variável. Neste caso, como o próprio nome diz, temos que a variável PINO DO LED vale 13.

// LED connected to digital pin 13 : comentário dizendo que existe um LED conectado ao pino digital de número 13.

Agora nós lhe convidamos a olhar seu Arduino mais de perto. Se você notar, verá que logo abaixo do pino 13 digital existe um LED SMD, ou seja, um microled, já colocado na placa, como mostra a figura abaixo:



Vamos agora olhar a estrutura de Setup do programa:

```
void setup() {  
  // initialize the digital pin as an output:  
  pinMode(ledPin, OUTPUT);  
}
```

void setup() { : Declaração que irá começar o Setup do programa. Sempre aberto com uma “{” e fechada, no fim da declaração, por uma “}”.

// initialize the digital pin as an output: : Comentário dizendo que o pino digital será inicializado como uma saída

pinMode(ledPin, OUTPUT); : Escolha do modo do pino, se é entrada (INPUT) ou saída (OUTPUT).

Como neste caso queremos acender um led, a corrente elétrica irá sair do pino e não entrar. Logo, setamos o ledPin (que tinha o valor 13, por causa do pino digital 13) como saída.

Por fim, neste programa, iremos analisar a estrutura Loop:

```
void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(ledPin, LOW);  // set the LED off
  delay(1000);                // wait for a second
}
```

void loop() : De modo análogo ao setup, com o comando ao lado dizemos que irá começar o loop do programa, ou seja, o programa principal que ficará rodando por tempo indeterminado. Também é aberto com uma “{” e fechado com uma “}”.


digitalWrite(ledPin, HIGH); // set the LED on : Escrita digital. Por tratar-se de um pino digital, ou você terá nível lógico 1 ou terá nível lógico 0, no caso de um led, ou teremos led acesso (1) ou teremos led apagado (0). O comando então liga o led, ou seja, envia 1 para o pino 13

delay(1000); // wait for a second : Delay é mais uma função pronta de seu arduino. O numero que for inserido entre os parêntesis será o valor, em milissegundos, que o Arduino irá esperar para seguir para a próxima instrução. No caso, temos um delay de 1000 milissegundos, ou seja, uma espera de 1 segundo para executar a próxima instrução.


digitalWrite(ledPin, LOW); // set the LED off

delay(1000); // wait for a second : Estes dois comandos são análogos aos dois vistos anteriormente, com a única diferença que a escrita digital escreverá um 0 no pino do led, ou seja, um nível lógico baixo: o led apagará e o Arduino espera 1 segundo para fazer a próxima instrução que, no caso, volta a ser o **digitalWrite(ledPin, HIGH);** .

Se este programa está 100% entendido, já podemos compilar o mesmo e fazer o upload para nossa placa Arduino. Para compilar o programa devemos clicar no botão Verify do ambiente de

desenvolvimento, para ver se não existe nenhum erro de código. O botão é o seguinte: 

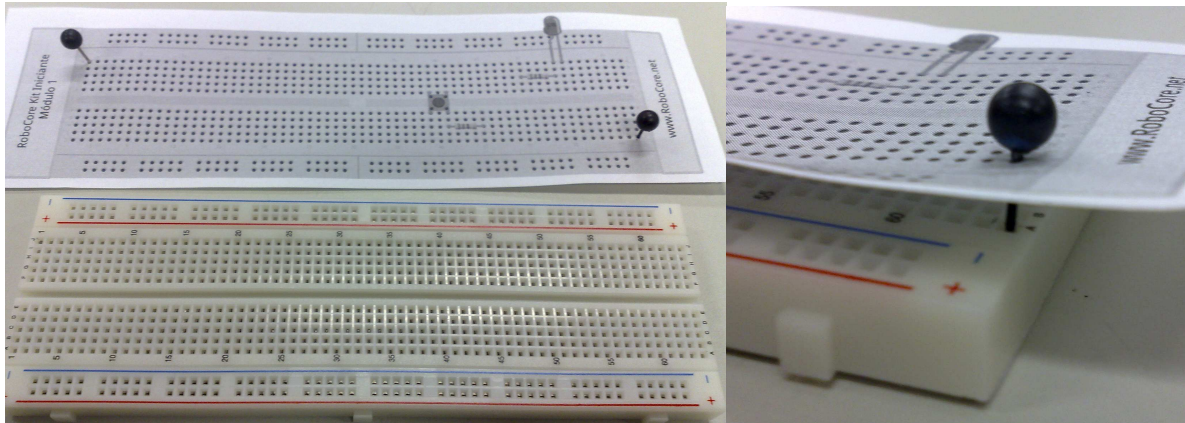
Se na barra inferior aparecer a mensagem: Done Compiling, o programa está pronto para ser

enviado ao Arduino. Para tanto, basta clicar no botão Upload que é o seguinte: . Espere então o upload ser completado e pronto. Você deverá ver o led da placa piscando com intervalos de 1 segundo.

ATENÇÃO

Para todos os experimentos, existe um gabarito. O modelo encaixa perfeitamente sobre a protoboard e você pode furar o mesmo com os terminais dos componentes que for utilizar para que nada saia errado em sua experiência.

Utilize os pinos de fixação para prender o papel à protoboard da seguinte forma:



Visto toda esta explicação agora sim pode começar a estudar o primeiro módulo deste material.

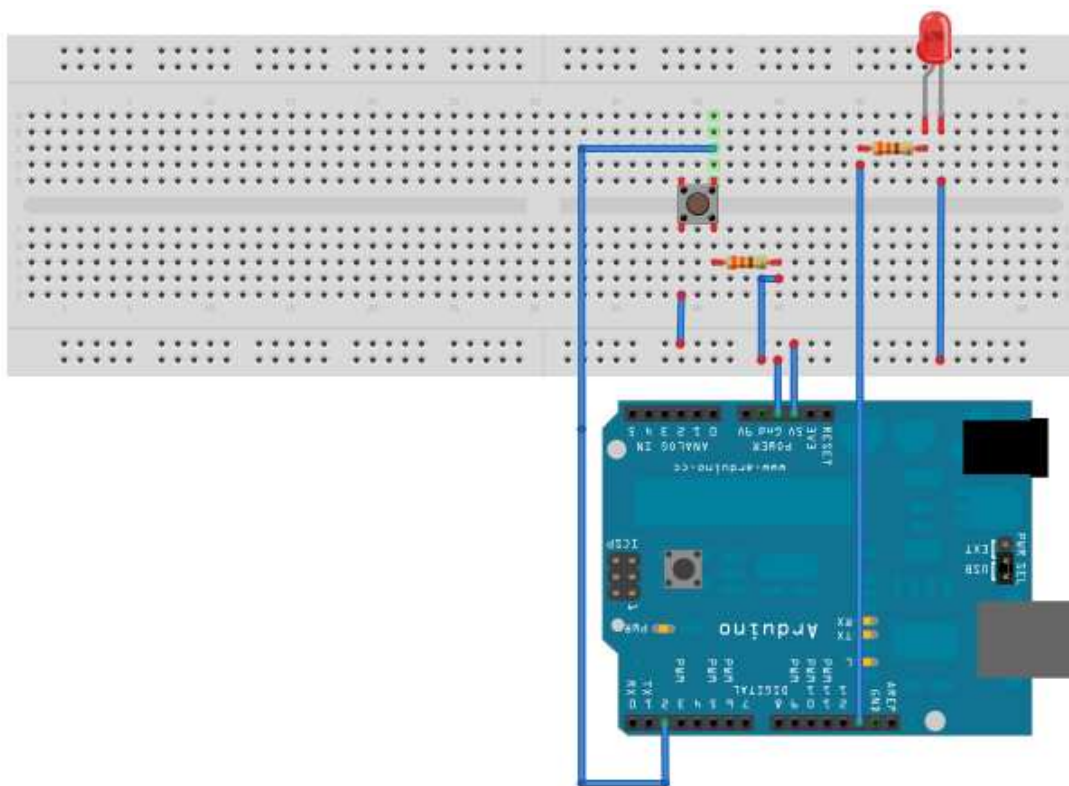
▪ **Módulo 1**

Componentes: 1 Botão + 1 Led

Descrição: Conforme você pressiona um pushbutton, um led é aceso

Dificuldade: ⚡⚡⚡⚡⚡

Trata-se de fazer um botão acender um led quando pressionado e, quando solto, o led deverá apagar. Para tanto, utilize o PaperSketch denominado Módulo1, que veio junto com o Arduino Kit Iniciante. Coloque os componentes como está sendo mostrado no PaperSketch. As ligações dos componentes podem ser as feitas na figura a seguir:



Código:

```
/******\
**      ROBOCORE ARDUINO KIT INICIANTE      **
**                                          **
**              Módulo 1                  **
\*****/
int ledPin = 13; //led no pino 13
int Botao = 2; //botao no pino 2
int EstadoBotao = 0; //Variavel para ler o status do pushbutton
void setup(){
  pinMode(ledPin, OUTPUT); //Pino do led será saída
  pinMode(Botao, INPUT); //Pino com botão será entrada
}
void loop(){
  EstadoBotao = digitalRead(Botao); /*novo estado do botão vai ser igual ao que
                                     Arduino ler no pino onde está o botão.
                                     Poderá ser ALTO (HIGH)se o botão estiver
                                     Pressionado, ou BAIXO (LOW),se o botão
                                     estiver solto */
  if (EstadoBotao == HIGH){ //Se botão estiver pressionado (HIGH)
    digitalWrite(ledPin, HIGH); // acende o led do pino 13.
  }
  else{ //se não estiver pressionado
    digitalWrite(ledPin, LOW); //deixa o led do pino 13 apagado
  }
}
```

Recomendamos que você tente entender passo a passo o programa anterior, para não ter problemas quando os códigos começarem a ficar mais complexos.

Após compilar o código e fazer o upload na sua placa, você já deve poder apertar o botão e o led da protoboard acender e, quando soltar, o led deve apagar.

Se houve algum problema, procure seu erro e tente consertá-lo. Se não, parabéns! Você concluiu o primeiro módulo RoboCore Arduino Kit Iniciante. Agora você está pronto para começar o módulo de número 2.

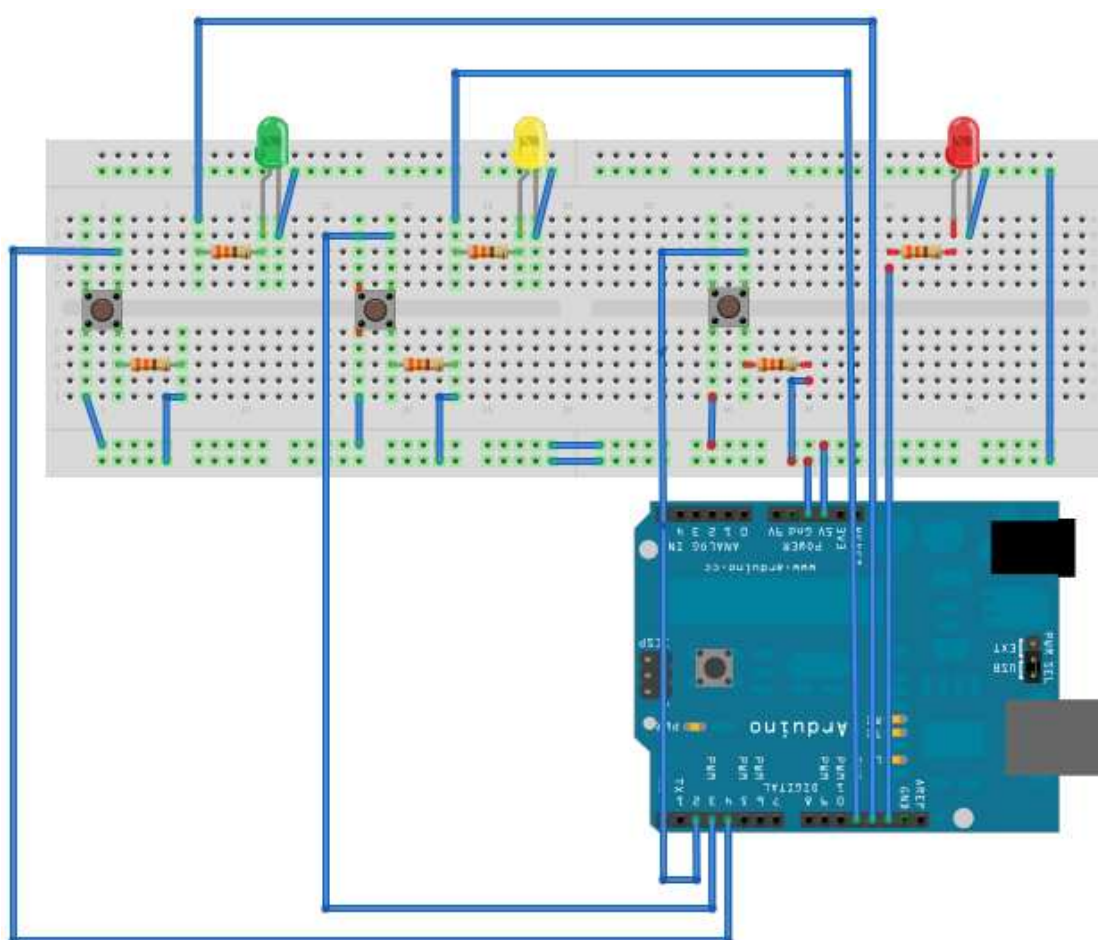
▪ Módulo 2

Componentes: 3 Botões + 3 Leds

Descrição: Conforme você pressiona qualquer um dos botões, leds de diferentes cores são acesos

Dificuldade: 

Este módulo é uma expansão do módulo anterior. A diferença deste com o módulo 1, é que neste teremos mais 2 botões e mais 2 leds de cores diferentes. Você pode tentar montar sozinho o novo circuito, ou utilizar o modelo para fazer a montagem. Um modelo de ligações pode ser o que segue:



Neste ponto você já tem autonomia para desenvolver o resto do programa, mas, se preferir, um código para utilizar os 3 botões pode ser o seguinte:

Código:

```
/*
*****
**      ROBOCORE ARDUINO KIT INICIANTE      **
**                                          **
**              Módulo 2                  **
*****
\
int ledPin1 = 13;
int ledPin2 = 12;
int ledPin3 = 11;
int Botao1 = 2;
int Botao2 = 3;
int Botao3 = 4;
int EstadoBotao1 = 0;
int EstadoBotao2 = 0;
int EstadoBotao3 = 0;
void setup(){
  pinMode(ledPin1, OUTPUT);
  pinMode(Botao1, INPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(Botao2, INPUT);
  pinMode(ledPin3, OUTPUT);
  pinMode(Botao3, INPUT);
}
void loop(){
  EstadoBotao1 = digitalRead(Botao1);
  EstadoBotao2 = digitalRead(Botao2);
  EstadoBotao3 = digitalRead(Botao3);
  if (EstadoBotao1 == HIGH){
    digitalWrite(ledPin1, HIGH);
  }
  else{
    digitalWrite(ledPin1, LOW);
  }
  if (EstadoBotao2 == HIGH){
    digitalWrite(ledPin2, HIGH);
  }
  else{
    digitalWrite(ledPin2, LOW);
  }
  if (EstadoBotao3 == HIGH){
    digitalWrite(ledPin3, HIGH);
  }
  else{
    digitalWrite(ledPin3, LOW);
  }
}
```

Lembrete: Nunca se esqueça dos ponto e vírgula (;) no final dos comandos em seu programa em C.

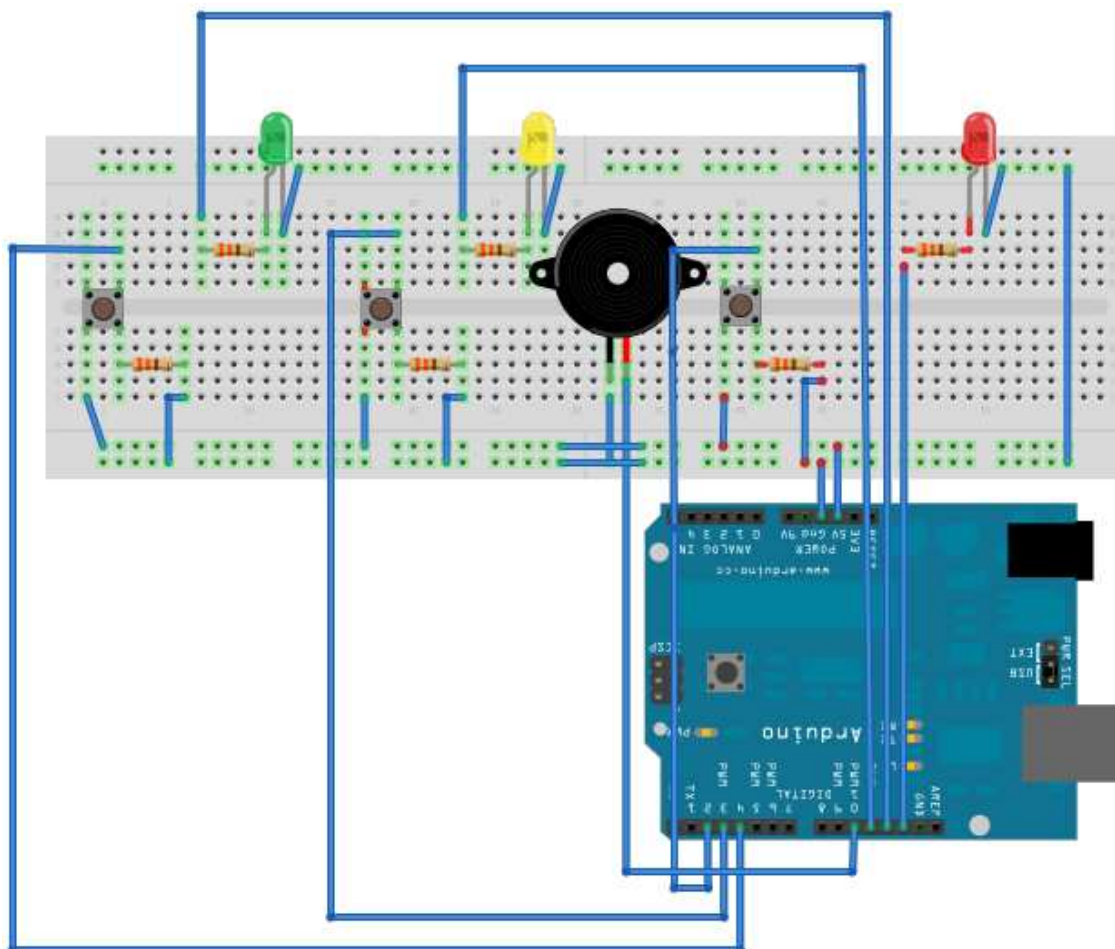
▪ Projeto Piano

Componentes: 3 Botões + 3 Leds + Buzzer

Descrição: Cada botão toca uma nota musical diferente e acende um led. É expansível – por conta do usuário – para mais uma nota musical com o botão (e o led) reserva

Dificuldade: 

Utilizando os conceitos aprendidos nos módulos 1 e 2, podemos agora montar o primeiro projeto: o Projeto Piano. Neste projeto cada um dos 3 botões tocará uma nota musical diferente. Para montar o projeto usaremos um novo componente: o Buzzer. Um Buzzer nada mais é do que um pequeno autofalante. Obviamente que ele não consegue tocar musicas, mas consegue fazer apitos soarem, como sirenes ou alarmes. A maioria dos alarmes de pequenos equipamentos eletrônicos é feito através de um buzzer. Ele funciona da seguinte maneira: quando alimentado por uma fonte, componentes metálicos internos vibram da frequência da fonte, produzindo assim um som. Para este experimento, você também pode utilizar um pequeno auto-falante (o som sai mais puro e a diferença entre as notas musicais é mais nitida). Para fazer a montagem, o modelo a seguir pode ser seguido:



Veja que a única diferença entre este projeto e o módulo 2 é a inserção de um Buzzer.

Código:

```

/*****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                           *
**      Projeto Piano                      **
\*****/

const int ledPin1 = 13;
const int ledPin2 = 12;
const int ledPin3 = 11;
const int Botao1 = 2;
const int Botao2 = 3;
const int Botao3 = 4;
const int Buzzer = 10; //O buzzer está colocado no pino 10
int EstadoBotao1 = 0;
int EstadoBotao2 = 0;
int EstadoBotao3 = 0;
int Tom = 0; //Variavel para armazenar a nota musical

void setup() {
  pinMode(Buzzer, OUTPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(Botao1, INPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(Botao2, INPUT);
  pinMode(ledPin3, OUTPUT);
  pinMode(Botao3, INPUT);
}

void loop(){
  EstadoBotao1 = digitalRead(Botao1);
  EstadoBotao2 = digitalRead(Botao2);
  EstadoBotao3 = digitalRead(Botao3);
  if(EstadoBotao1 && !EstadoBotao2 && !EstadoBotao3) {
    Tom = 100;
    digitalWrite(ledPin1, HIGH);
  }
  if(EstadoBotao2 && !EstadoBotao1 && !EstadoBotao3) {
    Tom = 200;
    digitalWrite(ledPin2, HIGH);
  }
  if(EstadoBotao3 && !EstadoBotao2 && !EstadoBotao1) {
    Tom = 500;
    digitalWrite(ledPin3, HIGH);
  }
  while(Tom > 0) { //enquanto Tom for maior que zero faça o que esta descrit o
baixo:
    digitalWrite(Buzzer, HIGH); // Liga buzzer
    delayMicroseconds(Tom); // Espera o tempo proporcional ao comprimento de
onda da nota musical em milisegundos
    digitalWrite(Buzzer, LOW); // Desliga buzzer
    delayMicroseconds(Tom); // Espera o tempo proporcional ao comprimento de
onda da nota musical em milisegundos
    Tom = 0; // Reseta o Tom para zero, para sair do loop while e nao tocar o
som constantemente
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
  }
}

```

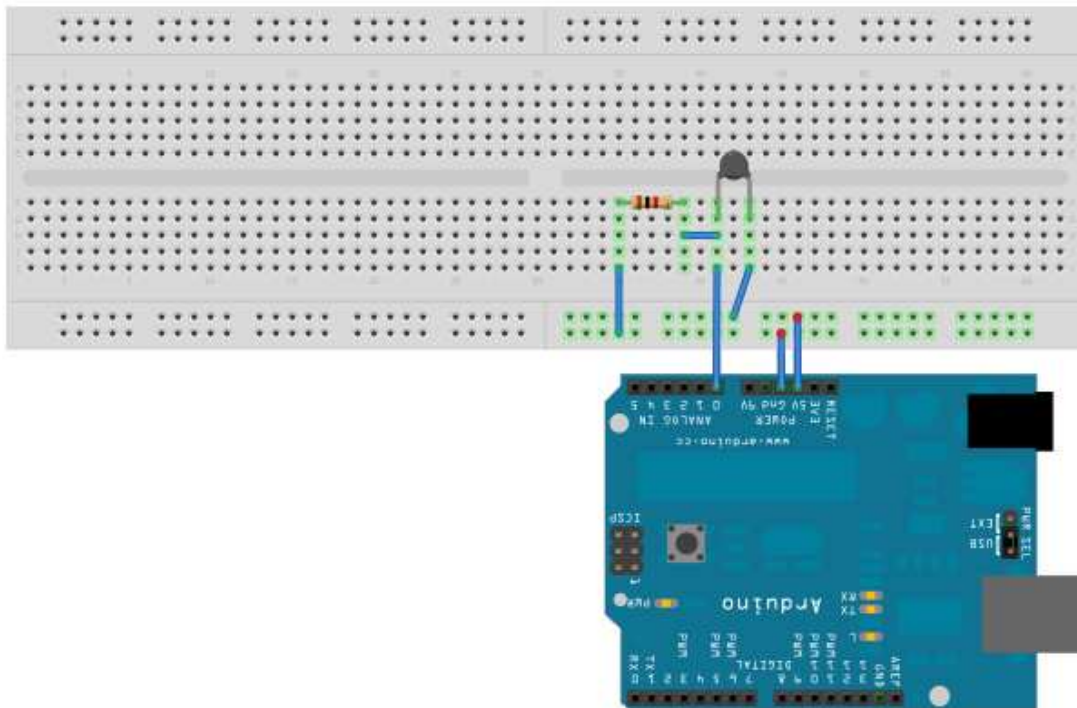
▪ Módulo 3

Componentes: 1 Sensor de Temperatura NTC

Descrição: Com o auxílio da porta serial e do monitor serial, o usuário irá fazer a leitura e calibração do sensor de temperatura para fazer o projeto.

Dificuldade: 

Este experimento é muito simples, mas tem um valor agregado muito grande. Iremos aqui fazer a primeira aquisição de dados do mundo externo pra dentro do Arduino. Usaremos para tanto um sensor de temperatura ligado a uma das entradas analógicas da placa. O circuito a ser montado é o seguinte:



Código:

```

/*****
**      ROBOCORE ARDUINO KIT INICIANTE      **
**                                          **
**              Módulo 3                  **
*****/

const int PinoSensor = 0;//pino Analógico de Entrada 0
int valorSensor = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){
  ValorSensor = analogRead(PinoSensor);
  Serial.print("valor do sensor = ");
  Serial.println(ValorSensor);
  delay(500);
}

```

Vamos entender este programa. Como já foi dito, no começo do programa colocamos as bibliotecas usadas para fazer o projeto. Novamente, não temos nenhuma biblioteca por enquanto. O próximo conjunto de instruções são as variáveis e a declaração das mesmas:

const int PinoSensor = 0; //Significa que PinoSensor é uma **CONSTANTE INTEIRA** – por isso o “CONST INT”. É uma constante porque, a posição do sensor não mudará: ficará sempre na entrada analógica 0.

int ValorSensor = 0; //ValorSensor é uma variável do tipo **INTEIRA**.

Seguindo com o programa vamos a parte do **setup**:

```
void setup(){  
  Serial.begin(9600);  
}
```

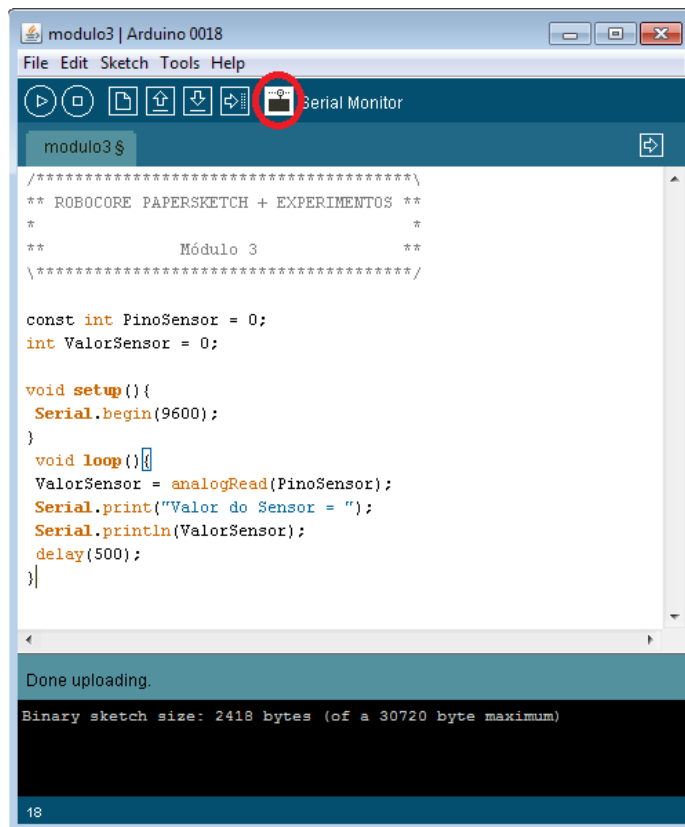
O comando Serial.begin serve para dizer ao Arduino que você irá coletar ou escrever dados no Arduino utilizando a porta serial, ou seja, através do cabo USB AB você vai ler ou escrever valores no mundo externo. O número entre os parênteses trata-se da taxa de dados com que você vai fazer esta leitura ou escrita. Neste caso usaremos 9600kbps.

Quanto ao loop principal:

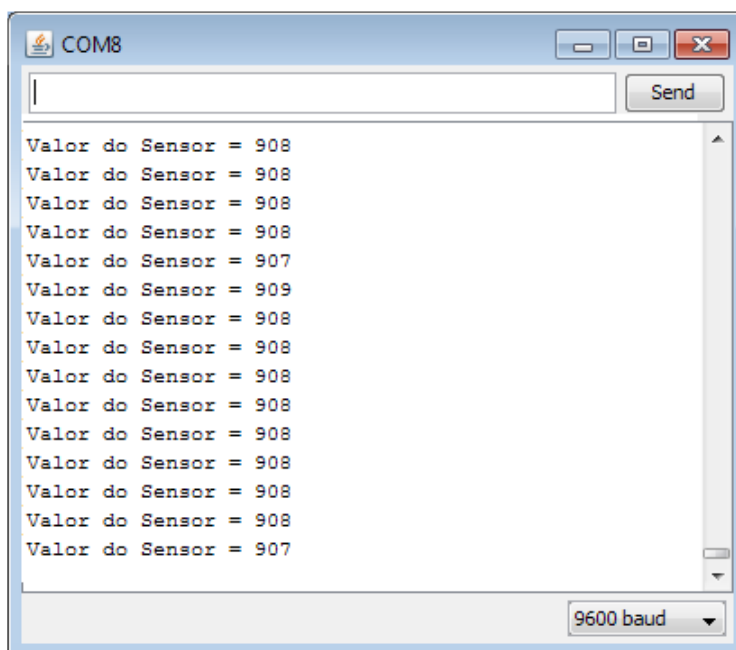
```
void loop(){  
  ValorSensor = analogRead(PinoSensor);  
  Serial.print("Valor do Sensor = ");  
  Serial.println(ValorSensor);  
  delay(500);  
}
```

O loop é muito simples. Na primeira linha o Arduino irá assimilar o valor lido na entrada analógica 0 (que é nossa constante PinoSensor) à variável ValorSensor. Após isto, escreverá no MONITOR SERIAL o “Valor do Sensor = (valor lido)”. Espera 0,5 segundos para fazer uma nova leitura.

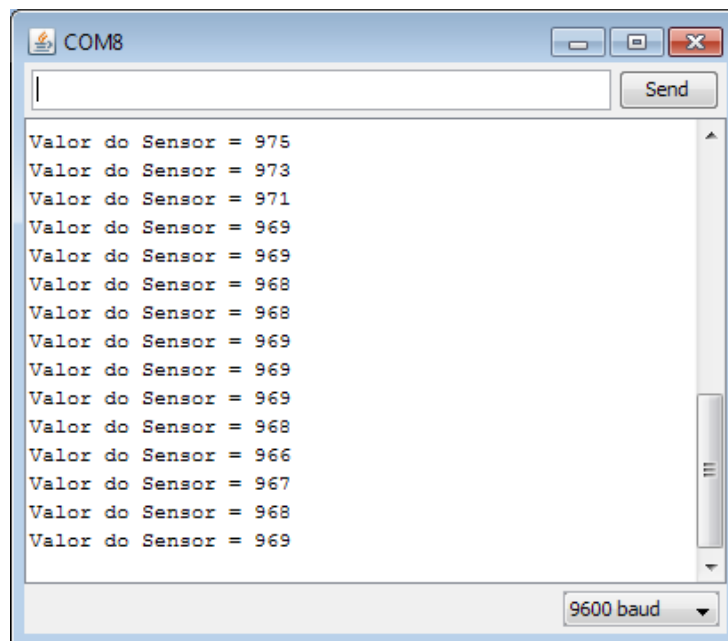
Para ver os dados no monitor serial, basta clicar no seguinte botão no ambiente de desenvolvimento do Arduino:



Depois de feito o compilamento e o upload do programa para sua placa Arduino, e após abrir o monitor serial pelo botão indicado anteriormente, você deverá ver algo parecido com:



Vale lembrar que a porta COM não é necessariamente 8, como está no topo da imagem anterior. Cada computador tem sua numeração de portas. Veja que no canto inferior direito temos selecionado 9600 baud. Isto tem de ser selecionado conforme a configuração do parâmetro Serial.begin do **setup** de seu programa. Também é bom ressaltar que, como os componentes eletrônicos não são totalmente iguais e que a temperatura ambiente em cada ponto do mundo é diferente, você não necessariamente vai ler valores como 908. Esta é a temperatura ambiente lida pelo sensor no local onde este material foi desenvolvido. Para fazer um teste com o sensor de temperatura, podemos utilizar um ferro de solda, ou um ferro de passar roupas, ou um secador de cabelo (qualquer coisa que esquite rapidamente) bem como seus dedos, visto que a temperatura deles é maior do que a ambiente. Quando aproximamos um ferro de solda ao sensor de temperatura, fazemos leituras como as seguintes:




Deste modo, podemos fazer uma calibração do nosso sensor e mapear os valores para uma escala real em, por exemplo, graus Celsius. Agora estamos prontos para fazer nosso próximo projeto, o alarme de temperatura.

▪ Projeto Alarme

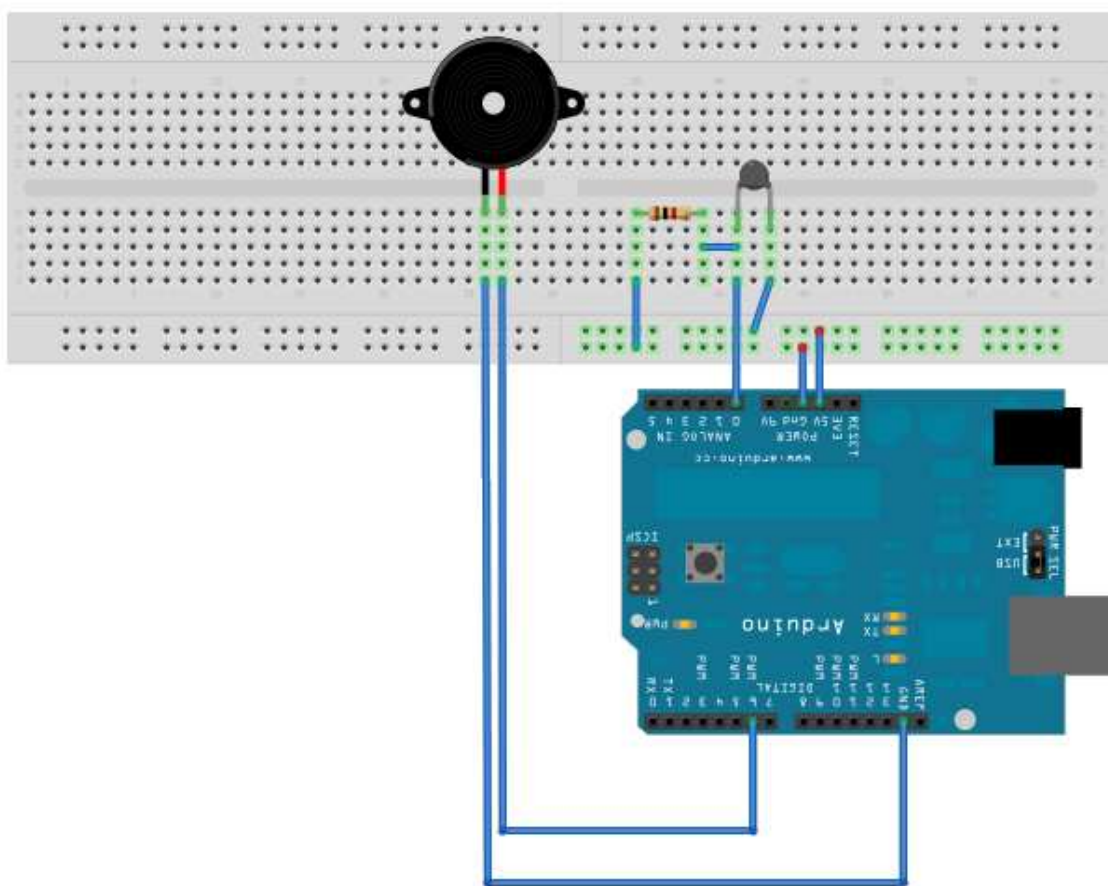
Componentes: 1 Sensor de Temperatura NTC + 1 buzzer

Descrição: A partir dos valores colhidos no módulo 3, o usuário poderá montar um alarme que, se a temperatura de onde o sensor estiver localizado for maior, ou menor, ele soará.

Dificuldade: 

O intuito é muito simples: quando a temperatura for maior que um valor, escolhido por você, o buzzer começará a soar até que a temperatura volte ao estado perfeito.

O circuito é o seguinte:



Código:

```
/* ****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                           *
**      Projeto Alarme                      **
\ *****/

const int PinoSensor = 0;
const int Buzzer = 6;
int ValorSensor = 0;

void setup(){
  pinMode(Buzzer, OUTPUT);
  Serial.begin(9600);
}
void loop(){
  ValorSensor = analogRead(PinoSensor);
  Serial.print("Valor do Sensor = ");
  Serial.println(ValorSensor);
  if (ValorSensor > 912){
    digitalWrite(Buzzer, HIGH);
  }
  else{
    digitalWrite(Buzzer, LOW);
  }
}
```

A única diferença deste para o código do último módulo é que, se a temperatura lida for maior do que 912 (valor hipotético, apenas para teste. Você pode e deve mudar este valor para o que achar mais adequado em seu caso) o buzzer é ligado e só desliga quando a temperatura for menor que 912.

Agora vamos fazer um projeto um pouco mais visual: o projeto Termômetro.

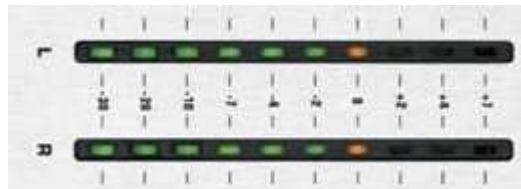
▪ Projeto Termômetro

Componentes: 2 Leds Verdes + 2 Leds Amarelos + 2 Leds Vermelhos + Buzzer + 1 Sensor de Temperatura NTC

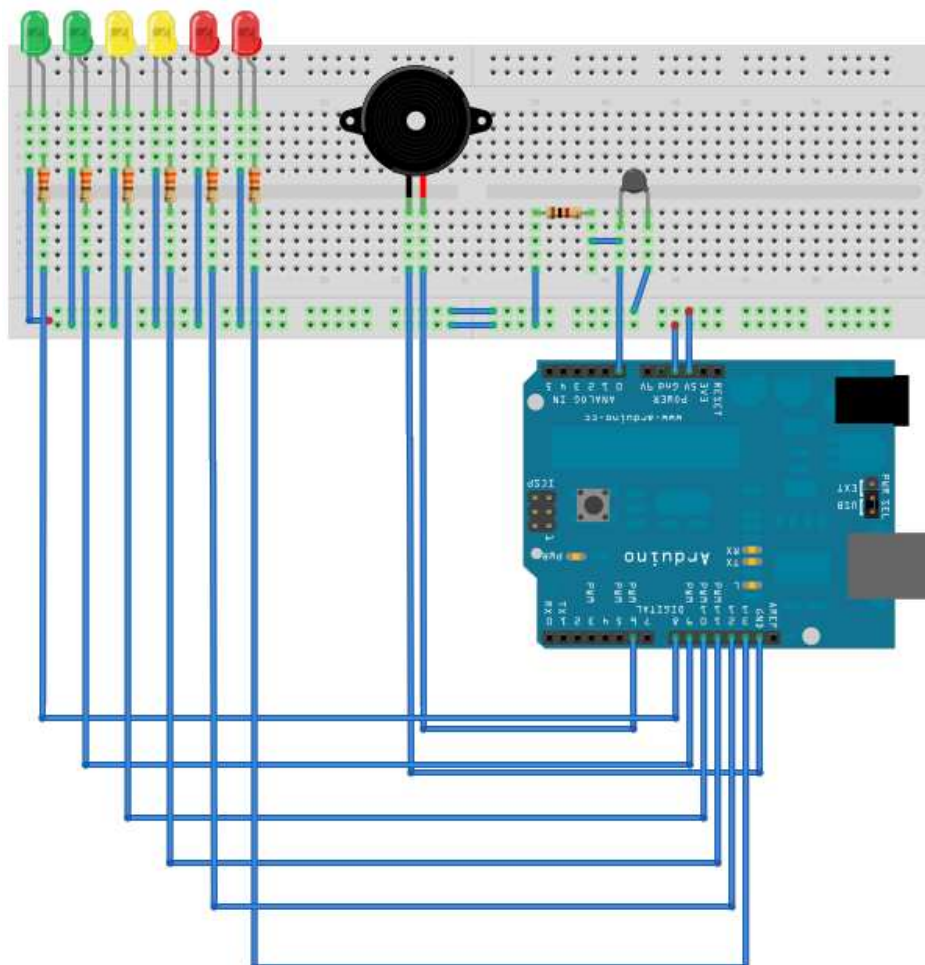
Descrição: Conforme a temperatura do ambiente onde o sensor NTC está localizado aumenta, os leds coloridos acendem, como um termômetro. Se por algum motivo todos os 6 Leds forem acesos, um alarme intermitente deverá soar.

Dificuldade: ⚡⚡⚡⚡⚡

Este projeto é, sem dúvida, muito bonito para os olhos tanto dos aficionados em eletrônica quanto às pessoas comuns. Implicto neste projeto estará o conceito de um *bargraf*, que nada mais é do que uma barra de leds que acendem conforme algum parâmetro. *Bargrafs* muito conhecidos são os de equipamentos de som. Quando o som está alto, ou com os graves altos, as luzes acendem do verde até o vermelho, como na figura a seguir:



Exemplo de um típico *bargraf* na horizontal



Tanto o esquema de ligações quanto o código parecem ser mais complexos, portanto tenha muita calma e atenção para montar o esquema. Revise o circuito algumas vezes antes de ligá-lo.

Código:

```
/*
*****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                              *
**      Projeto Termômetro                  **
\*****/

const int PinoSensor = 0;
const int Buzzer = 6;
const int led1 = 8;
const int led2 = 9;
const int led3 = 10;
const int led4 = 11;
const int led5 = 12;
const int led6 = 13;

int ValorSensor = 0;

void setup(){
  pinMode(Buzzer, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  ValorSensor = analogRead(PinoSensor);
  Serial.print("Valor do Sensor = ");
  Serial.println(ValorSensor);

  if (ValorSensor > 0){
    digitalWrite(led1, HIGH);
  }
  else{
    digitalWrite(led1, LOW);
  }

  if (ValorSensor > 915){
    digitalWrite(led2, HIGH);
  }
  else{
    digitalWrite(led2, LOW);
  }
}
```

```
if (ValorSensor > 920){
    digitalWrite(led3, HIGH);
}
else{
    digitalWrite(led3, LOW);
}

if (ValorSensor > 930){
    digitalWrite(led4, HIGH);
}
else{
    digitalWrite(led4, LOW);
}

if (ValorSensor > 935){
    digitalWrite(led5, HIGH);
}
else{
    digitalWrite(led5, LOW);
}

if (ValorSensor > 940){
    digitalWrite(led6, HIGH);
    digitalWrite(Buzzer, HIGH);
}
else{
    digitalWrite(led6, LOW);
    digitalWrite(Buzzer, LOW);
}
}
```

NÃO SE ASSUTE! O código é grande, mas é completamente entendível. O que mudou deste código para o do Projeto Alarme foi que adicionamos 6 leds. Desta forma, no começo do código tivemos que declarar onde estes leds estavam colocados:

```
const int led1 = 8;  
const int led2 = 9;  
const int led3 = 10;  
const int led4 = 11;  
const int led5 = 12;  
const int led6 = 13;
```

Colocamos os leds verdes nos pinos 8 e 9 e os chamamos de "led1" e "led2";
Colocamos os leds amarelos nos pinos 10 e 11 e os chamamos de "led3" e "led4";
Colocamos os leds verdes nos pinos 12 e 13 e os chamamos de "led5" e "led6";

Depois de declarado onde estariam os leds, era hora de declarar que todos os pinos (de 8 a 13) eram saídas. Fizemos isso da seguinte maneira no **setup**:

```
pinMode(led1, OUTPUT);  
pinMode(led2, OUTPUT);  
pinMode(led3, OUTPUT);  
pinMode(led4, OUTPUT);  
pinMode(led5, OUTPUT);  
pinMode(led6, OUTPUT);
```

Agora vem a parte da programação. Veja que o código possui agora diversas estruturas IF, como por exemplo a seguinte correspondente ao led3:

```
if (ValorSensor > 920){  
    digitalWrite(led3, HIGH);  
}  
else{  
    digitalWrite(led3, LOW);  
}
```

O que estamos dizendo com esta estrutura?

SE o valor lido no sensor for **MAIOR** que **920** faça:

ACENDA o led3

SE NÃO faça:

APAGUE o led3

Fazendo este tipo de estrutura para cada led com determinados valores crescentes de temperatura, iremos ver que, quanto maior a temperatura, mais leds são acesos e, se o último led vermelho for aceso (simulando uma situação crítica de altíssima temperatura), um alarme soa.

Vamos agora mudar um pouco o foco dos projetos. Vamos aprender a fazer outro tipo de leitura analógica, utilizando um potenciômetro.

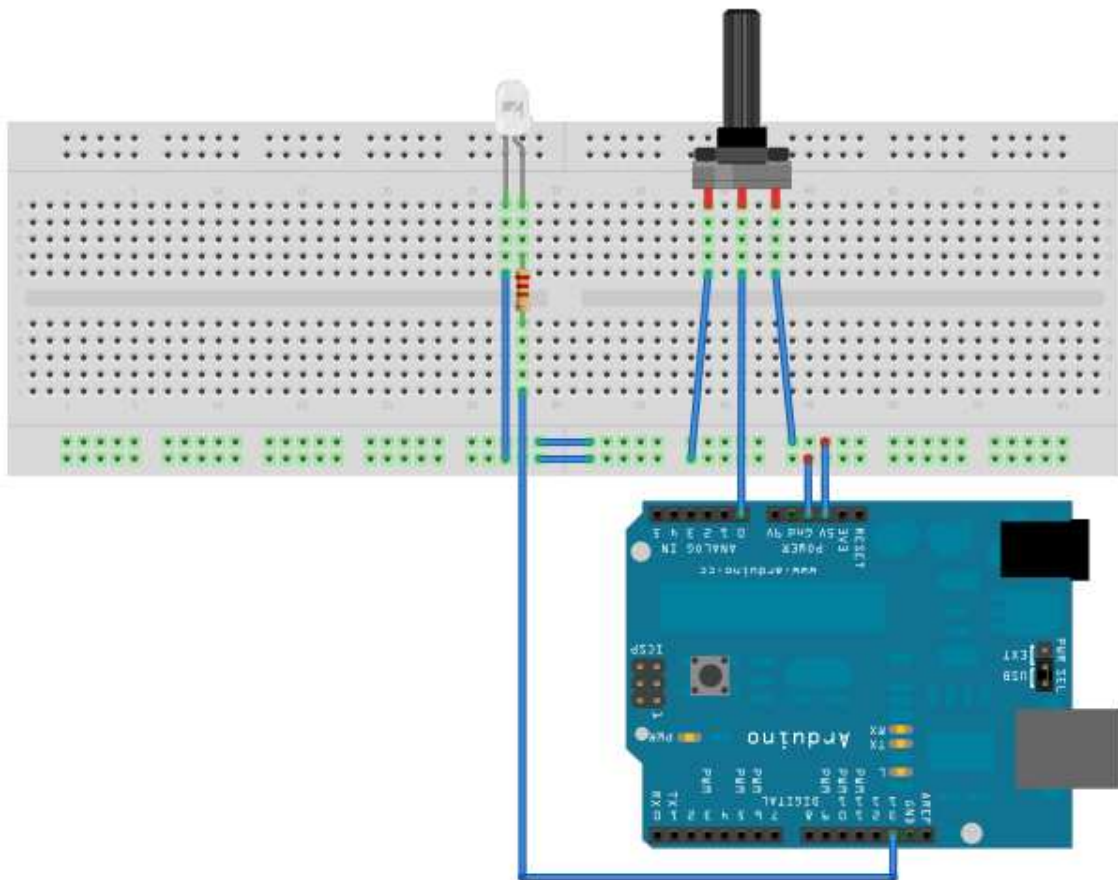
▪ **Módulo 4**

Componentes: 1 Potenciômetro + 1 Led

Descrição: Conforme o valor do potenciômetro é alterado, o led pisca de forma mais rápida ou mais lenta

Dificuldade: 

Vamos voltar aos circuitos simples. Neste módulo faremos com que um led pisque mais rápido ou mais devagar conforme os parâmetros de um potenciômetro.



Dica: Você consegue usar os próprios pinos do potenciômetro para ligar ele à sua protoboard.

Código:

```
/******\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                           *
**              Módulo 4                  **
\*****/

const int PinoPotenciometro = 0;
const int Led = 13;
int ValorPot = 0;

void setup() {
  pinMode(Led, OUTPUT);
}

void loop() {
  ValorPot = analogRead(PinoPotenciometro);
  digitalWrite(Led, HIGH);
  delay(ValorPot);
  digitalWrite(Led, LOW);
  delay(ValorPot);
}
```

Este código deve ser de fácil entendimento. Primeiro declaramos que o pino do potenciômetro será o Analógico 0 e será constante:

const int PinoPotenciometro = 0;

Depois dizemos que teremos um led no pino13 e também será constante:

const int Led = 13;

Então declaramos uma variável do tipo inteira para armazenar os valores do potenciômetro. Veja que esta variável irá de 0 a 1023, pois estes são os valores que a entrada analógica pode variar por termos uma resolução de 10 bits.

Declaramos o pino do Led como saída no **setup**, como já feito na maioria dos nossos programas e então vem o **loop** principal:

```
void loop() {
  ValorPot = analogRead(PinoPotenciometro);
  digitalWrite(Led, HIGH);
  delay(ValorPot);
  digitalWrite(Led, LOW);
  delay(ValorPot);
}
```


Primeiramente, como já não tem mais segredo para nós, assimilamos o valor lido no Pino do Potenciômetro à variável ValorPot, ou seja, ao valor do potenciômetro. Então ligamos o Led. Esperamos um tempo, que varia de 0 a 1023 porque são esses valores que nosso potenciômetro pode ter, e desligamos o led. Novamente esperamos o tempo e voltamos para a primeira instrução do **loop** para fazer a nova leitura de uma nova posição do potenciômetro.

Agora já estamos aptos a fazer um projeto muito comum nas iluminações residenciais hoje, o Dimmer.

▪ Projeto Dimmer

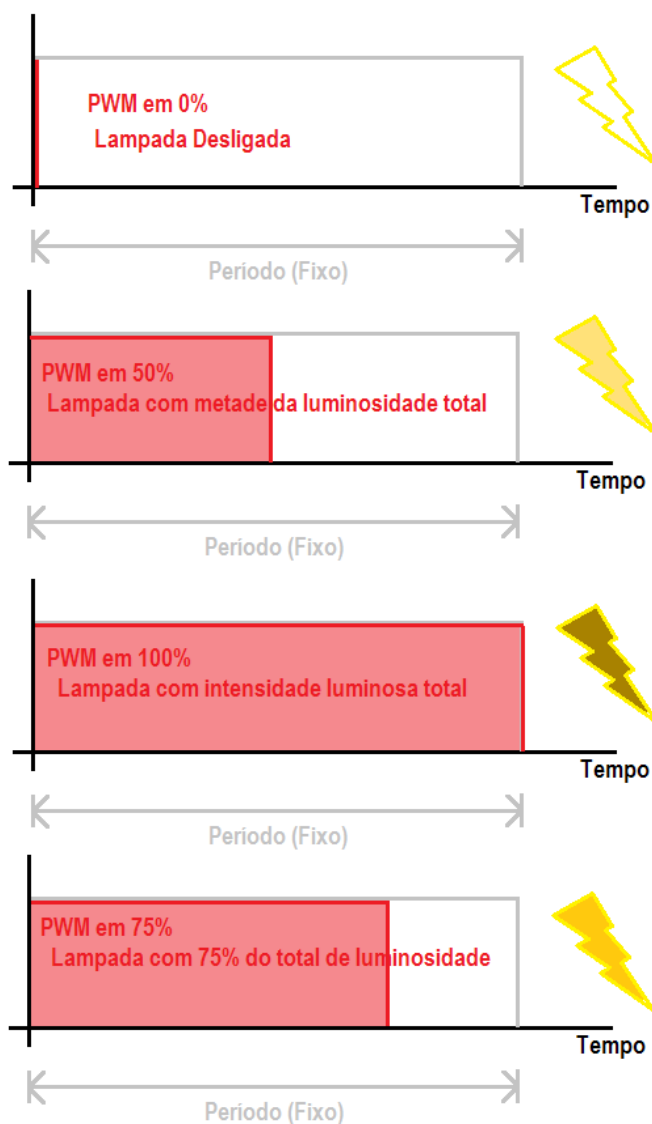
Componentes: 1 Potenciômetro + 1 Led Alto Brilho

Descrição: Conforme o valor do potenciômetro é alterado, o led fica mais claro ou mais escuro graças ao PWM

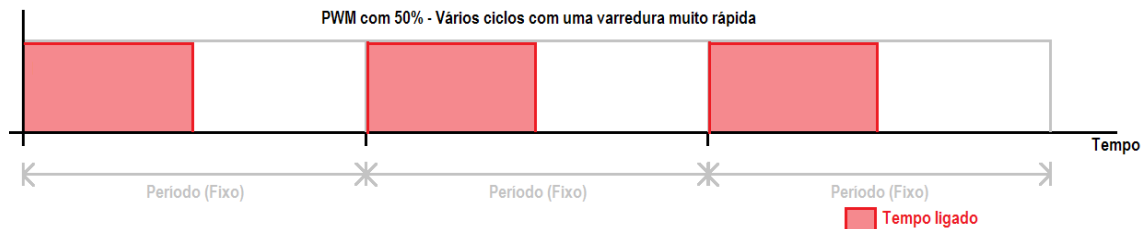
Dificuldade: 

Este projeto é muito simples, mas é outro que dirá respeito um conceito importantíssimo na eletrônica: o PWM. Esta sigla significa Pulse Width Modulation, ou seja, modulação por largura de pulso. De uma maneira bem simples, esta técnica pode ser explicada como: utilizando bases de tempo, conseguimos ligar e desligar uma porta tão rapidamente que para nossos olhos parece estar sempre ligado, e o que muda é a intensidade com a qual a porta está ligada.

A figura a seguir ilustra esta ideia:



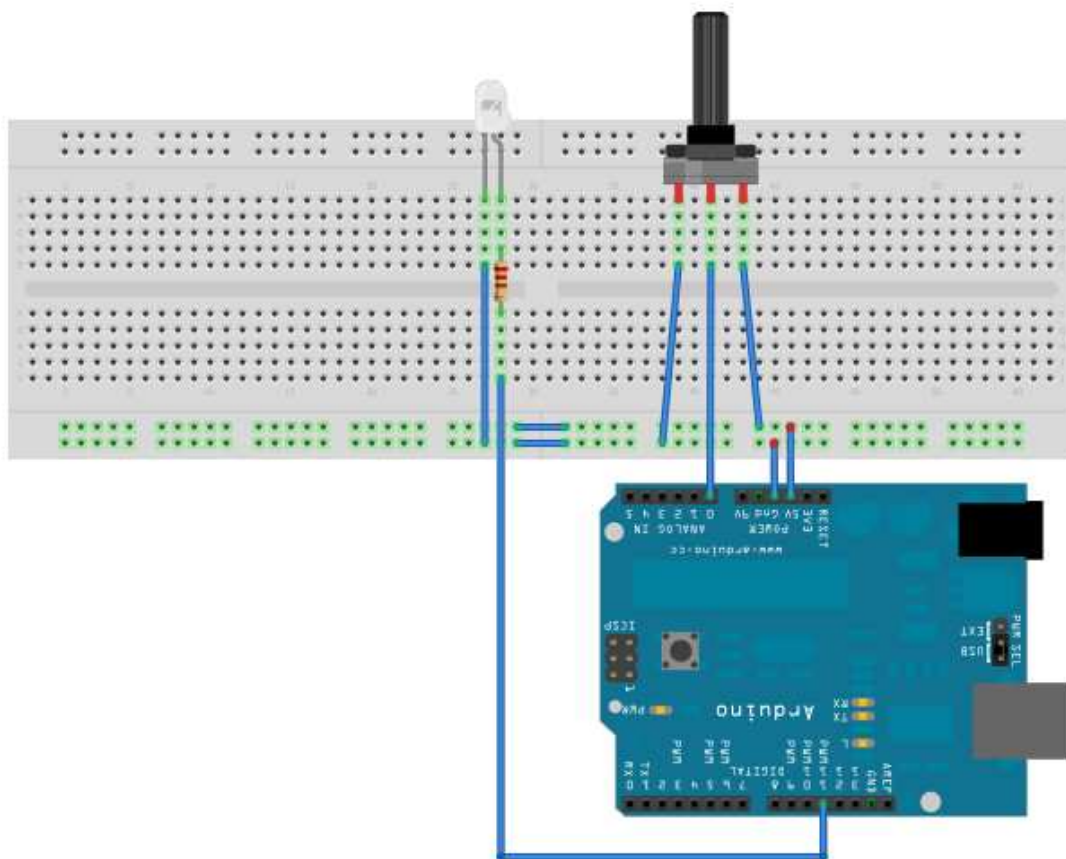
O período é fixo. Por exemplo, se nosso período forem 10 milissegundos e ligarmos o PWM em 50%, ou seja, 5 milissegundos ligado e 5 milissegundos desligado, veremos a lâmpada (ilustrada pelo raio amarelo) acesa com metade da intensidade luminosa que teríamos se deixássemos a lâmpada ligada os 10 milissegundos. Acontece que existe um tempo de varredura, que quando um período chega ao fim, outro começa instantaneamente e a lâmpada fica ligada, como podemos ver na figura a seguir:



Como o tempo em que isso ocorre é muito rápido, não enxergamos a lâmpada ligar e desligar. Mas, se tivéssemos um período de 10 segundos, e deixássemos a lâmpada acesa 5 segundos e apagada outros 5 segundos veríamos o tempo aceso e apagado, pois nossos olhos conseguem distinguir o que são 5 segundos, mas não o que são 5 milissegundos.

No Arduino 2009, temos 6 saídas digitais que podem ser utilizadas como PWM. Neste projeto, vamos utilizar 1 delas, no caso o pino 11.

O esquema a ser montado é muito semelhante ao anterior, apenas agora teremos que mudar o pino onde colocamos o led anteriormente.



Código:

```
/* ****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                           *
**              Projeto Dimmer              **
\*****/

const int PinoPotenciometro = 0;
const int Led = 11;
int ValorPot = 0;
int pwm = 0;

void setup() {
  pinMode(Led, OUTPUT);
}

void loop() {
  ValorPot = analogRead(PinoPotenciometro);
  pwm = map(ValorPot, 0, 1023, 0, 255);
  analogWrite(Led, pwm);
}
```

Este código possui alguns elementos ainda não vistos neste material. Vamos estudá-lo passo a passo novamente:

```
const int PinoPotenciometro = 0;
const int Led = 11;
int ValorPot = 0;
int pwm = 0;
```

Declaração de constantes e variáveis: os pinos onde estão o Potenciômetro e o Led (agora no pino 11) são constantes, ValorPot e pwm são variáveis.

```
void setup() {
  pinMode(Led, OUTPUT);
}
```

No **setup** setamos o pino do Led como saída.

```
void loop() {
  ValorPot = analogRead(PinoPotenciometro);
  pwm = map(ValorPot, 0, 1023, 0, 255);
  analogWrite(Led, pwm);
}
```

Aqui estão 2 estruturas ainda não vistas. Na primeira linha assimilamos o valor lido no Pino do Potenciômetro à variável ValorPot. Logo depois fazemos um mapeamento de uma variável. Isto significa que vamos redefinir os limites de valores de uma variável. Antes de continuar a mostrar como funciona este mapeamento, vamos estudar a estrutura **analogWrite**.

Como o próprio nome diz, esta estrutura faz uma escrita analógica, ou seja, faz aquela escrita variável para fazermos nosso PWM. No Arduino está predefinido que para ter 0% de PWM, basta você escrever: **analogWrite(pino a ser escrito, 0)**; do mesmo modo que, para escrever 100% de PWM, basta você escrever: **analogWrite(pino a ser escrito, 255)**, ou seja, na estrutura que o Arduino entende como PWM, os valores vão de 0 (mínimo, ou seja, 0%) até 255 (máximo, ou seja, 100%). Voltamos então para o mapeamento. Vamos entender esta estrutura:

```
pwm = map(ValorPot, 0, 1023, 0, 255);
```

Com isto, queremos dizer que a variável “pwm” irá receber valores mapeados da seguinte forma:

Variável Recebedora = map(Valor Lido, Mínimo do Potenciômetro, Máximo do Potenciômetro, Novo Mínimo definido por você, Novo Máximo definido por você)

Portanto,

Valor Lido vale ValorPot: é o valor lido anteriormente pela função **analogRead**;

Mínimo do Potenciômetro vale 0;

Máximo do Potenciômetro vale 1023;

Novo Mínimo definido por você vale 0;

Novo Máximo definido por você vale 255.

Tente ver se você entendeu que os valores da variável **pwm** irão variar de 0 a 255, conforme o potenciômetro varia de 0 a 1023.

Após compilar e fazer o upload deste projeto, você terá um Dimmer de leds. Use sua imaginação e conhecimentos para utilizar isto da maneira mais adequada a suas necessidades ou projetos.

▪ **Projeto Iluminação Automatizada**

Componentes: 1 Led Alto Brilho + 1 Sensor de Luminosidade LDR

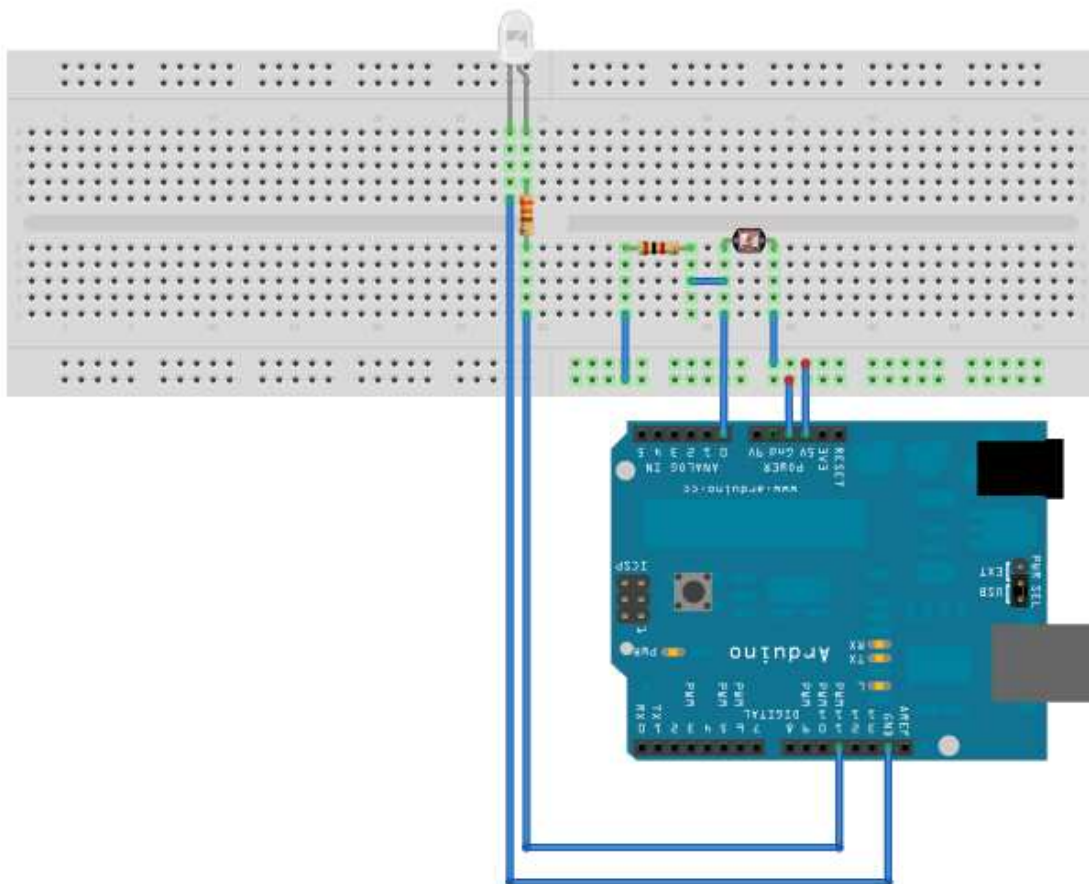
Descrição: Se a iluminação ambiente, por qualquer motivo, diminuir ou apagar completamente, um led de alto brilho acende gradativamente

Dificuldade: 

Com conceitos de entradas analógicas, sensores e pwm, já podemos pensar em um projeto de automação. Projeto semelhante é utilizado em postes de luz, onde as lâmpadas acendem sozinhas, conforme a luminosidade do dia - ou você acha que todo dia uma pessoa responsável liga e desliga todas as luzes de todos os postes de todas as ruas?

Para este projeto, usaremos um LDR. LDR nada mais é do que uma resistência que varia conforme a luminosidade: é um sensor de luminosidade.

Monte o seguinte circuito:



Veja que o led está colocado no pino 11 digital e o sensor de luminosidade no pino 0 analógico.

Antes de qualquer coisa, temos que calibrar o sensor.

Código:

```

/*****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                              *
**              Calibrar LDR              **
\*****/

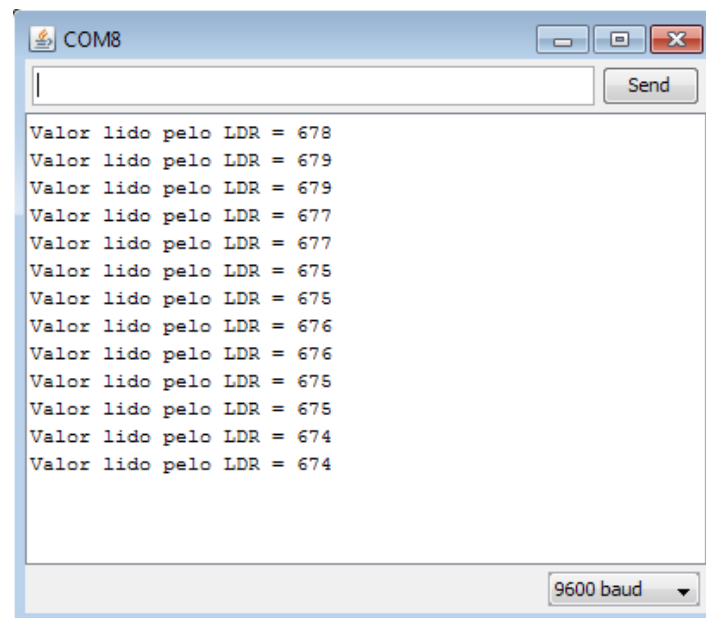
const int LDR = 0;
int valorLido = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  valorLido = analogRead(LDR);
  Serial.print("Valor lido pelo LDR = ");
  Serial.println(valorLido);
  delay(500);
}

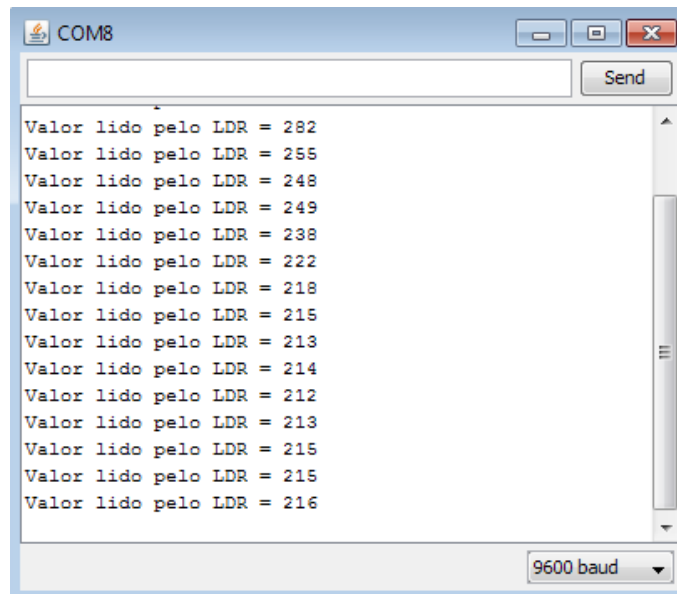
```

O código acima já é conhecido. Ele mostra no monitor serial os valores que o LDR está lendo. No caso da iluminação no local onde este material foi escrito, os valores lidos pelo LDR são os seguintes:



Você deve estar vendo em seu monitor algo parecido com esta figura a cima. Se você está lendo um valor fixo de 1023, certifique-se que os componentes estão bem colocados e na posição correta. Este é um erro muito comum neste tipo de experimento.

Coloque agora a palma da sua mão, ou qualquer outro material que tampe a luz ambiente, sobre o sensor tampando a luz e fazendo o sensor ficar na sombra. Você deve ler valores como os seguintes:



Como deve ter ficado subentendido:

Quanto mais luz o LDR receber, mais alto será o valor lido.

Quanto menos luz o LDR receber, menor será o valor lido.

Agora já temos os valores para calibrar nosso sensor. Vamos supor que você queira fazer com que um led acenda quando o valor lido é de 500 (uma sombra moderada sobre o LDR). Podemos então utilizar o seguinte código para fazer este projeto:

```

/*****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
**                                           **
**      Projeto Iluminação Automatizada      **
\*****/

const int LDR = 0;
const int Led = 11;
int valorLido = 0;
int pwm = 0;

void setup() {
  pinMode(Led, OUTPUT);
}

void loop() {
  valorLido = analogRead(LDR);

  if (valorLido < 500){
    analogwrite(Led, pwm);
    pwm++;
  }
  else{
    digitalWrite(Led, LOW);
  }

  if(pwm > 255){
    pwm=255;
  }
}

```

A maior parte dos elementos deste código já foi estudada. Vamos para a parte que merece nossa atenção no **loop**:

```
void loop() {  
  ValorLido = analogRead(LDR);  
  
  if (ValorLido < 500){  
    analogWrite(Led, pwm);  
    pwm++;  
  }  
  else{  
    digitalWrite(Led, LOW);  
  }  
  
  if(pwm > 255){  
    pwm=255;  
  }  
}
```

Primeiramente assimilamos o valor lido pelo LDR com a variável ValorLido. Depois disso fazemos as seguintes condições:

SE a variável **ValorLido** for **MENOR** que 500 (uma leve sombra), **FAÇA:**
Escreva de uma maneira **ANALÓGICA**, ou seja, PWM no **Led** e
Some **1** na variável **pwm** (na linguagem C, colocar uma variável seguida de dois sinais de positivo significa somar 1 a esta variável);

SE NÃO (ou seja, Se ValorLido for **MAIOR** que 500), **FAÇA:**
Escreva de uma maneira **DIGITAL**, ou seja, alto ou baixo no **Led** e o apague (**LOW**)

A próxima condição serve apenas para garantir que a variável pwm não ultrapasse 255, pois, como já visto, para fazer escritas analógicas com pwm podemos usar valores indo de 0 a 255.

SE a variável **pwm** for **MAIOR** que 255, **FAÇA:**
pwm é **IGUAL** a 255 (desta forma garantimos que pwm nunca passará dos 255).

Pronto. Compile e faça o upload deste código juntamente com o circuito montado e veja que circuito útil você tem agora em mãos.

Agora podemos ir para o último projeto, e sem dúvida o mais complexo de todos.

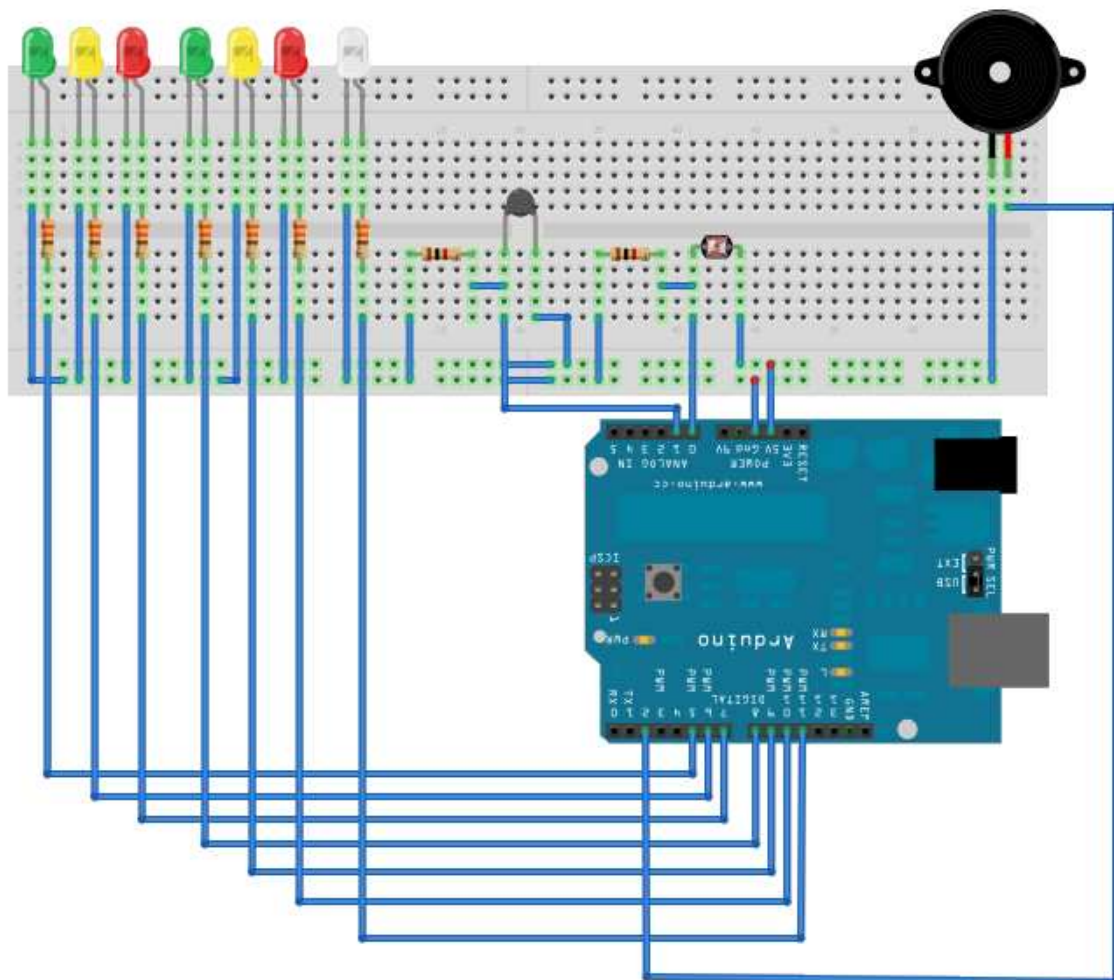
Se você não teve dúvidas até agora está pronto para desenvolvê-lo. Se algo não saiu como os conformes, refaça quantas vezes for necessário o experimento.

▪ **Projeto Alarme Multipropósito**

Componentes: 2 Leds Verdes + 2 Leds Amarelos + 2 Leds Vermelhos + 1 Sensor de Luminosidade LDR + 1 Sensor de Temperatura NTC + 1 Led Alto Brilho + 1 buzzer
Descrição: Temos 2 *bargraphs*, ou seja, dois indicadores: um de luminosidade e outro de temperatura, através das cores dos leds. Se a temperatura estiver alta e acender os 3 Leds que a corresponde, um alarme deverá soar. De maneira análoga, se os 3 Leds correspondentes à luminosidade estiverem apagados – indicando uma falta total de luminosidade no ambiente - um alarme deverá soar e um led de alto brilho irá acender.

Dificuldade: ⚡⚡⚡⚡⚡

Para este complexo sistema, o circuito montado deve parecer com o seguinte:



Dica: tente usar as próprias “pernas” dos componentes para fazer as ligações, desse modo utilizando a menor quantidade de fios possível.

Cuidado: preste muita atenção em cada ligação para tudo dar certo no fim do experimento.

Código:

```
/* ****\
**      ROBOCORE ARDUINO KIT INICIANTE      **
*                                           *
**      Projeto Alarme Multipropósito        **
\*****/

const int LDR = 0;
const int NTC = 1;
const int Buzzer = 2;
const int led1 = 5;
const int led2 = 6;
const int led3 = 7;
const int led4 = 8;
const int led5 = 9;
const int led6 = 10;
const int ledAB = 11;

int ValorLDR = 0;
int ValorNTC = 0;
int pwm = 0;

void setup(){
  pinMode(Buzzer, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(ledAB, OUTPUT);
}

void loop(){
  ValorLDR = analogRead(LDR);
  ValorNTC = analogRead(NTC);

  if (ValorNTC > 0){
    digitalWrite(led1, HIGH);
  }
  else{
    digitalWrite(led1, LOW);
  }

  if (ValorNTC > 935){
    digitalWrite(led2, HIGH);
  }
  else{
    digitalWrite(led2, LOW);
  }

  if (ValorNTC > 945){
    digitalWrite(led3, HIGH);
    digitalWrite(Buzzer, HIGH);
  }
  else{
    digitalWrite(led3, LOW);
    digitalWrite(Buzzer, LOW);
  }
}
```



```
if (ValorLDR > 600){  
  digitalWrite(led6, HIGH);  
}  
else{  
  digitalWrite(led6, LOW);  
}  
  
if (ValorLDR > 500){  
  digitalWrite(led5, HIGH);  
}  
else{  
  digitalWrite(led5, LOW);  
}  
  
if (ValorLDR > 450){  
  digitalWrite(led4, HIGH);  
  digitalWrite(ledAB, LOW);  
}  
else{  
  digitalWrite(led4, LOW);  
  digitalWrite(ledAB, HIGH);  
}  
  
}
```

Neste ponto você já é capaz de entender perfeitamente o que se passa neste projeto, mesmo porque ele é apenas uma junção de alguns módulos vistos nesta apostila com alguns projetos.

Se os valores das condições não estão satisfatórios para seu projeto, sinta-se a vontade para alterá-los como quiser. Uma dica é sempre fazer a calibragem dos sensores que você for utilizar, com o código proposto no Projeto Iluminação Automatizada.

Para mais informações sobre tecnologia, eletrônica e robótica acesse www.RoboCore.net e fique por dentro de tudo que acontece no mundo tecnológico. Use também o fórum do site para discutir novos projetos e também falar sobre as 10 experiências propostas aqui.

Esperamos que esta apostila tenha-lhe sido válida e que você seja mais um **Arduinizador do Mundo**.

Arduinize o Mundo
RoboCore PaperSketch + Experimentos 2010
WWW.ROBOCORE.NET