Apostila Arduino

Exemplos de aplicação

Apostila resumida com aplicações para iniciar um trabalho na plataforma Arduino

Alexandre
03/10/2016

Apostila de Arduino

Exemplos no Arduino UNO

Desenhos utilizando Fritzing

Comunicação com PC utilizando Processing

Descrição de sensores e atuadores

Programação baseada em linguagem C com uma interface amigável

Exemplos e fotografias tirados do site original do Arduino

Alexandre Lima de Carvalho

1. Introdução

Arduino é uma plataforma de eletrônica aberta para a criação de protótipos baseada em software e hardware livres, flexíveis e fáceis de usar. Foi desenvolvida para artistas, designers, hobistas e qualquer pessoa interessada em criar objetos ou ambientes interativos.

O Arduino pode adquirir informação do ambiente através de seus pinos de entrada, para isso utilizamos uma gama de sensores (dispositivos de entrada). Por outro lado, o Arduino pode atuar no ambiente controlando luzes, motores ou outros atuadores (dispositivos de saída).

O microcontrolador da placa Arduino é programado mediante a linguagem de programação especifica, baseada em Wiring, e o ambiente de desenvolvimento (IDE) está baseado em Processing.

Os projetos desenvolvidos com Arduino podem ser executados mesmo sem a necessidade de estar conectados a um computador, apesar de que também podem ser feitos comunicando-se com diferentes tipos de software (como Processing , Delphi e Java).

As placas podem ser feitas a mão ou compradas montadas de fábrica. O download do software pode ser feito de forma gratuita e os desenhos da placa estão disponíveis sob uma licença aberta, assim você também é livre para adaptá-lo às suas necessidades.

No site oficial você encontra todas as informações necessárias ,assim como alguns exemplos de aplicação , alguns deles descritos nessa apostila . Site : www.arduino.cc .

Open Source Hardware consiste em dispositivos físicos de tecnologia concebidos e oferecidos pelo movimento de design aberto. Tanto o software livre como o open source hardware são criados sob o movimento de cultura open source e aplica este conceito a uma variedade de componentes. O termo normalmente significa que a informação sobre o hardware é facilmente reconhecida. O design no hardware (ou seja, desenhos mecânicos, esquemas, lista de materiais, dados de layout do PCB, código fonte e dados de layout de circuitos integrados), além do software livre que aciona o hardware, estão todos liberados com a abordagem livre e open source.

Processing é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado (IDE), construído para as artes eletrônicas e comunidades de projetos visuais com o objetivo de ensinar noções básicas de programação de computador em um contexto visual.

Fritzing é um programa de automação de design eletrônico open source destinado a ajudar designers e artistas a passar dos protótipos (utilizando, por exemplo, placas de teste) para os produtos finais. Foi criado sob os princípios de Processing e Arduino e permite a designers, artistas, pesquisadores e amadores documentar seu protótipo baseado em Arduino e criar diagramas de circuitos impressos para mais tarde fabricar. Além disso, tem um site complementar que ajuda a compartilhar e discutir projetos, experiências e reduzir os custos de fabricação.

2. Base Eletrônica

As entradas, ou inputs, são sensores eletrônicos ou mecânicos que tomam os sinais (em forma de temperatura, pressão, umidade, contato, luz, movimento, ph, etc.) do mundo físico e converte em sinais elétricos. Exemplos de entradas são sensores de gás, temperatura, velocidade, luz, pressão potenciômetros, sensores de movimento, chaves e outros.

As saídas, ou outputs, são atuadores, ou outros dispositivos que convertem os sinais de elétricos em sinais fisicamente úteis como movimento, luz, som,força ou rotação, entre outros. Exemplos de saídas são motores, LEDs ou sistemas de luzes , atuadores pneumáticos ou hidráulicos, som entre outros.

As entradas e saídas de um sistema eletrônico serão consideradas como sinais variáveis. Em eletrônica se trabalha com variáveis que são tomadas na forma de tensão ou corrente, que podem simplesmente ser chamados de sinais. Os sinais podem ser de dois tipos: digital ou analógico.

Variáveis digitais :também chamadas de variáveis discretas, se caracterizam por ter dois estados diferentes e portanto também podem ser chamadas de binárias (em lógica seria valores Verdadeiro (V) ou 1 e Falso (F) ou 0). Geram sinais com dois níveis de tensão (0e 5V).

São aquelas que podem tomar um número infinito de valores compreendidos entre dois limites. A maioria dos fenômenos da vida real são sinais deste tipo (som, temperatura,luminosidade, etc.). Exemplos de sinais analógicos são as ondas senoidais.

O processamento de sinal é realizado mediante circuitos conhecidos como microcontroladores. São circuitos integrados construídos para manipular, interpretar e transformar os sinais de voltagem e corrente vindos dos sensores (entradas) e ativar determinadas ações nas saídas.

Sistemas digitais:

• Sistema combinacional: Saída depende apenas dos estados das entradas.
Ex.: decodificador

• Sistema sequencial: Saídas dependem das entradas e dos estados anteriores das saída.
Ex.: Contador

• Sistema computacional: um hardware genérico tem sua função definida por um software.
Ex.: Microcontrolador

Arquitetura Harvard :

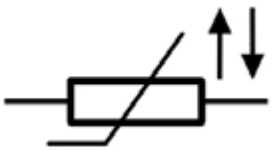
• Ciclos de leitura (fetch) e ciclos de execução simultâneos: pipeline;
• Cada instrução ocupa (tipicamente) um endereço de memória (1 instrução = 1 endereço);
• Mesmo tempo de execução para (quase) todas instruções;
• Memória de programa é mais larga que barramento de dados (operando + Operador).

3. Componentes Eletrônicos

. Microcontrolador: é um circuito integrado programável, capaz de executar as ordens gravadas em sua memória. Possui em seu interior três unidades funcionais principais: unidade central de processamento, memória e periféricos de entrada e saída. Os microcontroladores se diferenciam dos processadores pois, além dos componentes lógicos e aritméticos usuais de um microprocessador, o microcontrolador integra elementos adicionais em sua estrutura interna, como memória de leitura e escrita para armazenamento de dados, memória somente de leitura para armazenamento de programas, EEPROM para armazenamento permanente de dados, dispositivos periféricos como conversores analógico/digitais (ADC), conversores digitais/analógicos (DAC) em alguns casos; e, interfaces de entrada e saída de dados.São geralmente utilizados em automação e controle de produtos. Por reduzir o tamanho, custo e consumo de energia, e se comparados à forma de utilização de microprocessadores convencionais, aliados a facilidade de desenho de aplicações, juntamente com o seu baixo custo, os microcontroladores são uma alternativa eficiente para controlar muitos processos e aplicações.

. Termistor (NTC ou PTC) : O termistor NTC (do inglês Negative Temperature Coefficient) é um componente eletrônico semicondutor sensível à temperatura, utilizado para controle, medição ou polarização de circuitos eletrônicos. Possui um coeficiente de variação de resistência que varia negativamente conforme a temperatura aumenta, ou seja, a sua resistência elétrica diminui com o aumento da temperatura.

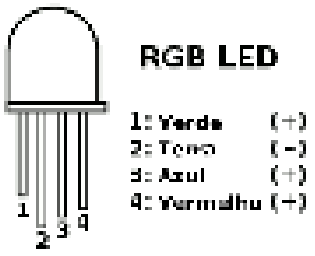
Símbolo :



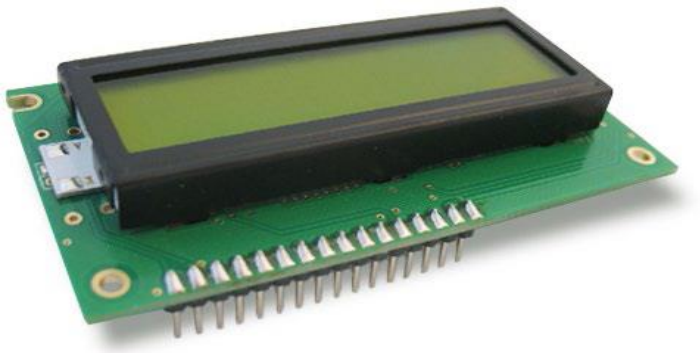
Componente :

. Led RGB : é um LED que incorpora em um mesmo encapsulamento três LEDs, um vermelho (Red), um verde (Green) e outro azul (Blue). Desta forma é possível formar milhares de cores ajustando de maneira individual cada cor. Os três LED´s estão unidos por um negativo ou cátodo.

Componente :



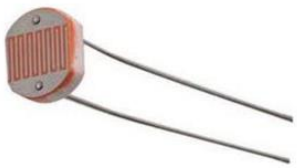
. Display de LCD : Um display de cristal líquido, ou LCD (liquid crystal display), é um painel fino usado para exibir informações por via eletrônica, como texto, imagens e vídeos. Um LCD consiste de um líquido polarizador da luz, eletricamente controlado, que se encontra comprimido dentro de celas entre duas lâminas transparentes polarizadoras. Suas principais características são leveza e portabilidade. Seu baixo consumo de energia elétrica lhe permite ser utilizado em equipamentos portáteis, alimentados por bateria eletrônica. Um display de LCD pode variar o número de linhas e caracteres por linha, a cor dos caracteres e a cor do fundo, assim como ter ou não luz de fundo.



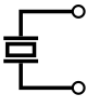
. Botão : Um botão em um dispositivo eletrônico funciona geralmente como um interruptor elétrico. No seu interior há dois contatos, e se é um dispositivo normalmente fechado ou normalmente aberto, ao pulsar o botão, se ativará a função inversa à que se está realizando no momento.

. LDR (Fotocélula) : O LDR (Light Dependant Resistor) é uma resistência cujo valor em ohms varia de acordo com a luz incidente. Uma fotocélula apresenta um baixo valor de resistência na presença de luz e um alto valor na sua ausência.
Símbolo :

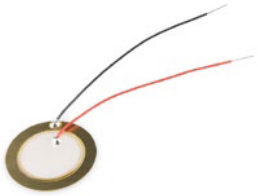
Componente :



Transdutor Piezoelétrico :Um transdutor piezoelétrico é muito prático para detectar vibrações ou golpes. Pode ser usado como sensor através da leitura da voltagem de saída. Este transdutor eletroacústico também pode ser usado como um pequeno buzzer para produzir um som ou zumbido contínuo ou intermitente.
Símbolo :



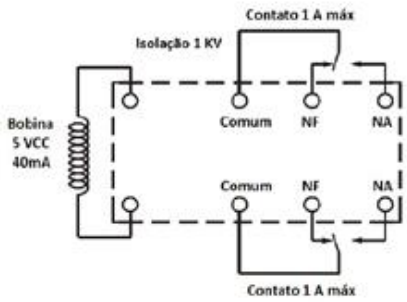
Componente :



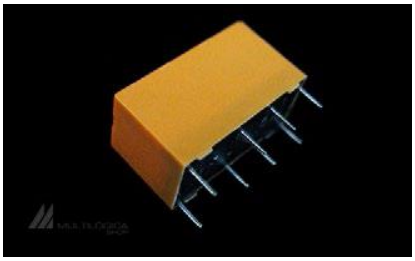
. Motor CC : O motor de corrente contínua (CC) é uma máquina que converte a energia elétrica em mecânica provocando um movimento rotatório. Esta máquina de corrente contínua é uma das mais versáteis. Seu fácil controle de velocidade, parada e sentido de rotação a tornam uma boa opção em aplicações de controle e automação de processos.
Por exemplo, pode-se encontrar na tração de carros de brinquedo a pilhas ou nas rodas de um robô.
Símbolo :

Componente :

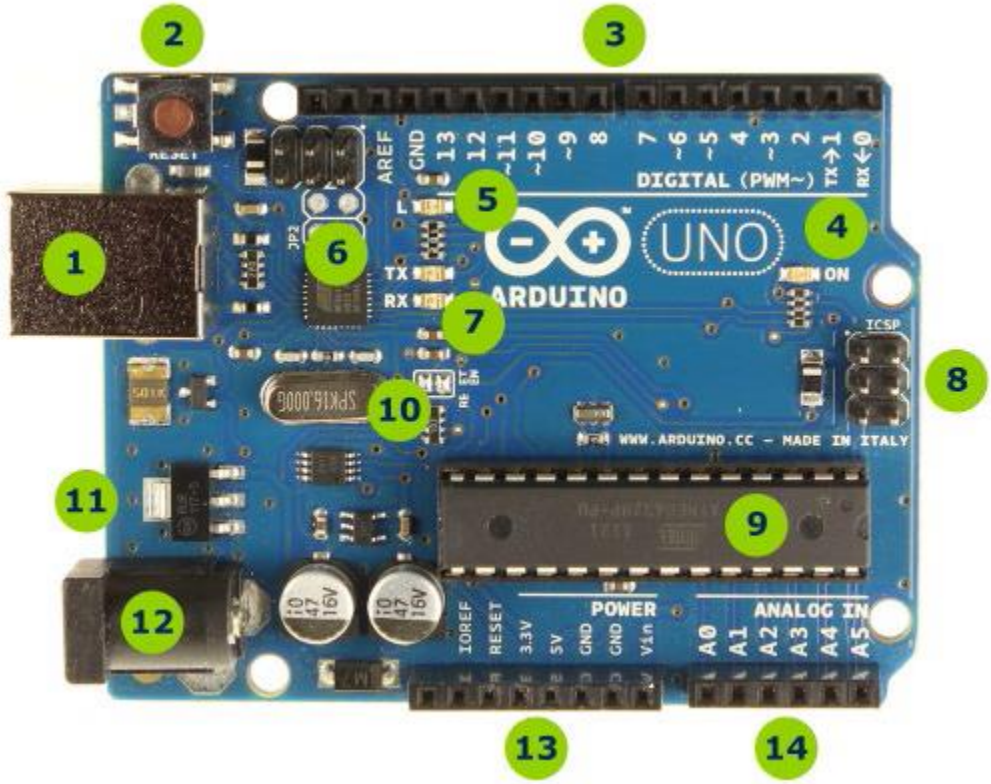
. Rele : É um interruptor eletromecânico usado para ligar ou desligar dispositivos. Quando uma corrente circula pela bobina interna, esta cria um campo magnético que atrai um ou uma série de contatos fechando ou abrindo circuitos. Ao retirar a corrente da bobina o campo magnético cessa, fazendo com que os contatos voltem para a posição original.
Símbolo :



Componente :



Arduino :



1 - Conector USB para o cabo tipo AB
2 - Botão de reset
3 - Pinos de entrada e saída digital e PWM
4 - LED verde de placa ligada
5 - LED laranja conectado ao pin13
6 - ATmega encarregado da comunicação com o computador
7 - LED TX (transmissor) e RX (receptor) da comunicação serial
8 - Porta ICSP para programação serial
9 - Microcontrolador ATmega 328, cérebro do Arduino
10 - Cristal de quartzo 16Mhz
11 - Regulador de voltagem
12 - Conector fêmea 2,1mm com centro positivo
13 - Pinos de voltagem e terra

Família Arduino:

Leonardo

Yun

Ethernet

Mega

Arduino Micro

Robot

Alguns Shields :

Motor

USB - Serial

WIFI

4. Programação Arduino

Softwares escritos usando Arduino são chamados de Sketches. Estes Sketches são escritos no editor de texto da IDE do Arduino e são salvos com a extensão de arquivo .ino. Este editor tem características de cortar/colar e para buscar/substituir texto. A área de mensagem dá feedback ao salvar e exportar arquivos e também exibe informações de erros ao compilar Sketches. O canto direito inferior da janela exibe a **placa atual e a porta serial**. Os botões da barra de ferramentas permitem que você verifique, carregue, crie, abra e salve Sketches ou abra o monitor serial.

Monitor Serial :Exibe dados seriais sendo enviados da placa Arduino para o computador. Para enviar dados para a placa, digite o texto e clique no botão "enviar" ou pressione enter.A comunicação entre a placa Arduino e seu computador pode acontecer em várias velocidades padrão pré-definidas. Para que isso ocorra é importante que seja definida a mesma velocidade tanto na Sketch quanto no Monitor Serial. Na Sketch esta escolha é feita através da função Serial.begin.

A comunicação serial com a placa Arduino também pode ser feita através de outras linguagens de programação como Processing, Flash, Delphi e muitas outras.

Biblioteca Arduino : O ambiente Arduino pode ser estendido através da utilização de bibliotecas, assim como a maioria das plataformas de programação. Bibliotecas fornecem funcionalidades extras para uso em sketches. Por exemplo, para trabalhar com hardware ou manipulação de dados.Algumas bibliotecas já vêm instaladas com a IDE Arduino, mas você também pode fazer download ou criar a sua própria.Para usar uma biblioteca em um sketch, selecione em sua IDE Arduino: Sketch> Importar Biblioteca. Dentro da programação você inclui as funcionalidades de uma biblioteca já existente a partir do comando: **#include** <LiquidCrystal.h>.

Arduino se programa em uma linguagem de alto nível semelhante a C/C++ e geralmente tem os seguintes componentes para elaborar o algoritmo:

- Estruturas
- Variáveis
- Operadores booleanos, de comparação e aritméticos
- Estrutura de controle
- Funções digitais e analógicas

Para mais detalhes visite a Referência da linguagem de programação Arduino, em português. Veja a referência estendida para características mais avançadas da linguagem Arduino e a página das bibliotecas para interação com tipos específicos de hardware, no site oficial do Arduino.

4.1 Estruturas:

São duas funções principais que deve ter todo programa em Arduino.

A função setup() é chamada quando um programa começa a rodar. Use esta função para inicializar as suas variáveis, os modos dos pinos, declarar o uso de bibliotecas, etc. Esta função será executada apenas uma vez após a placa Arduino ser ligada ou ressetada.

```
setup()
{
  //  configurações
}
```

Após criar uma função setup() que declara os valores iniciais, a função loop() faz exatamente o que seu nome sugere, entra em looping (executa sempre o mesmo bloco de código),permitindo ao seu programa fazer mudanças e responder. Use esta função para controlar ativamente a placa Arduino.

```
loop()
{
   // comandos
}
```

4.2 Variáveis :

São expressões que você pode usar em programas para armazenar valores como a leitura de um sensor em um pino analógico, elas reservam espaço de memória de acordo com o tipo especificado. Aqui destacamos alguns tipos:

- Variáveis Booleanas - Variáveis boolenas, assim chamadas em homenagem a George Boole, podem ter apenas dois valores: verdadeiro (true) e falso (false);
- Int : Inteiro é o principal tipo de dado para armazenamento numérico capaz de guardar números de 2 bytes. Isto abrange a faixa de -32.768 a 32.767;
- Char : Um tipo de dado que ocupa 1 byte de memória e armazena o valor de um caractere ASCII.Caracteres literais são escritos entre aspas.

Exemplos :

```
char letra = 'A';
int valor = 47;
boolean teste ;
```

4.3 Operadores lógicos booleanos :Estes operadores podem ser usados dentro da condição em uma sentença if.

- && ("e" lógico) :

Verdadeiro apenas se os dois operandos forem verdadeiros, ou seja, a primeira condição e a segunda forem verdadeiras.

Exemplo:

```
if (digitalRead(2) == 1 && digitalRead(3) == 1)
{
      // executa se os dois pinos estiverem em 5V
}
```

- || ("ou" lógico) :

Verdadeiro se algum dos operandos for verdadeiro, ou seja, se a primeira ou a segunda condição for verdadeira.

Exemplo:

```
if (x > 0 || y > 0)
{
    // executa se x ou y for maior que zero
}
```

- ! (negação) :
Verdadeiro apenas se o operando for falso.
Exemplo:

```
 if (!x)
{
    // executa se a variavel booleana x for falsa ( x = 0 )
 }
```

Operadores de comparação :

```
x == y (x é igual a y)
x != y (x é não igual a y)
x < y (x é menor que y)
x > y (x é maior que y)
x <= y (x é menor ou igual a y)
x >= y (x é maior ou igual a y)
```

Operadores aritméticos :

```
= (atribuição)
+ (adição)
- (subtração)
* (multiplicação)
/ (divisão)
% (resto da divisão)
```

4.4 Estruturas de programação :
Estruturas de controle -São instruções que permitem decidir e realizar diversas repetições de acordo com alguns parâmetros. Entre os mais importantes podemos destacar:

- IF : o fluxo de execução do programa depende da condição especificada na estrutura. Sintaxe :

```
IF ( condição )
{
   // executa se a condição for verdadeira
}
Else IF
{
   // executa se a condição for falsa
}
```

OBS.: o else é opcional ; se existir uma única linha ligada ao IF podemos omitir as chaves

- Switch/case: do mesmo modo que as sentenças if, as switch/case controlam o fluxo dos programas.Switch/case permite ao programador construir uma lista de “casos” dentro de um bloco delimitado por chaves. O programa checa cada caso com a variável de teste e executa o código se encontrar um valor idêntico.
Sintaxe :

```
switch (var)
{
 case 1:
```

```
        //faça alguma coisa quando var == 1
  case 2:
       //faça alguma coisa quando var == 2
 default:
      // se nenhum valor for idêntico, faça o default
     // default é opcional
  }
```

- While : fará com que o bloco de código entre chaves se repita contínua e indefinidamente até que a expressão entre parentesis () se torne falsa. Algo tem que provocar uma mudança no valor da variável que está sendo verificada ou o código vai sempre ficar dando voltas dentro do while. Isto poderia ser o incremento de uma variável ou uma condição externa, como o teste de um sensor. Sintaxe :

```
valor = <Valor Inicial>;
while(valor < Valor Final)
{
      // algum código que se repete do valor inicial até o valor final
      // no passo indicado
      <Passo>;  // por exemplo valor++ , incrementa valor em uma unidade
}
```

- For :A sentença for é utilizada para repetir um bloco de código delimitado por chaves. Um contador com incremento normalmente é usado para controlar e finalizar o loop. A sentença for é útil para qualquer operação repetitiva, e é frequentemente usada com arrays (matrizes) para operar em conjuntos de dados ou de pinos. Sintaxe :

```
for (int variavel=<valor inicial>; <condição final>; <passo>)
{
      // repete do valor inicial ao final no passo descrito
}
```

- Do While : Verifica a condição somente no final do laço e repete as estruturas entre chaves até a condição ser falsa. Sintaxe :

```
do
{
      Comandos;
}
While ( condição );
```

4.5 Funções digitais :

Orientadas a revisar o estado e a configuração das entradas e saídas digitais.

- pinMode(): Configura o pino especificado para que se comporte ou como uma entrada (INPUT) ou uma saída (OUTPUT).

Sintaxe:

```
pinMode(pino, tipo)
pinMode (9, OUTPUT); // determina o pino digital 9 como uma saída.
```

- digitalRead():Lê o valor de um pino digital especificado, HIGH ou LOW.
Sintaxe:

```
digitalRead(pino);
botao = digitalRead(9); // Leitura do estado de um botão no pino 9.
```

- digitalWrite(): Escreve um valor HIGH ou um LOW em um pino digital.
Sintaxe:

```
digitalWrite(pino, valor);
digitalWrite(9, HIGH); // Coloca o pino 9 em estado HIGH.
```

OBS.: HIGH = 1 = 5V ; LOW = 0 = 0V.

4.6 Funções analógicas:
Ideais para a leitura ou escrita de valores analógicos.

- analogRead(): Lê o valor de um pino analógico especificado. A placa Arduino contém um conversor analógico-digital de 10 bits com 6 canais. Com isto ele pode mapear voltagens de entrada entre 0 e 5 volts para valores inteiros entre 0 e 1023. Isto permite uma resolução entre leituras de 5 volts / 1024 unidades ou 0,0049 volts (4.9 mV) por unidade.
Sintaxe:

```
analogRead(pino);
int a = analogRead (A0); // Lê o valor do pino analógico A0 e armazena
                         //este valor na variável "a".
```

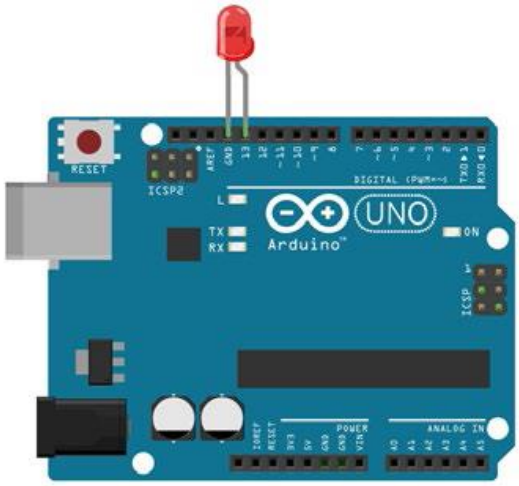
- analogWrite(): Escreve um valor analógico (onda PWM) em um pino. Pode ser usado para acender um LED variando o brilho ou girar um motor a velocidade variável.
Sintaxe:

```
analogWrite(pino, valor);
analogWrite (9,134); // Envia o valor analógico 134 na escala do PWM
```

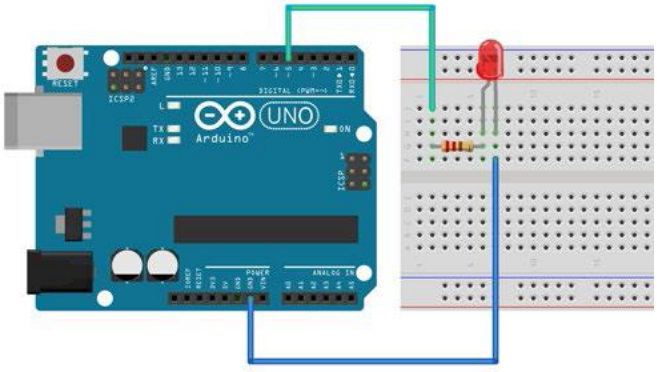
5.Programas de aplicação

5.1 Programa pisca-pisca

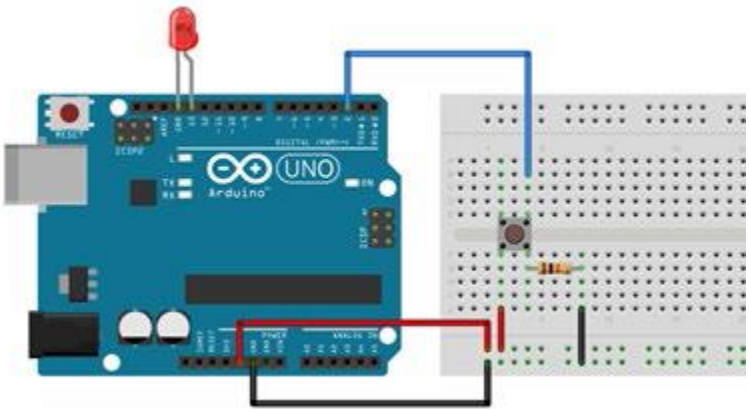
```
/* Pisca-Pisca
 * ---------------------------------------------------
 *ligar um led e um resistor de 220 ohms entre o pino 13 e o gnd
 */

int ledPin = 13;                // nome para pino 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // define pino 13 como saída
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // seta o pino 13
  delay(2500);                  // aguarda 2,5 seg
  digitalWrite(ledPin, LOW);    // zera pino 13
  delay(4000);                  // aguarda 4 seg
}
```



Exercício : Fazer um programa para piscar um led no pino 5 de 1 em 1 segundo.



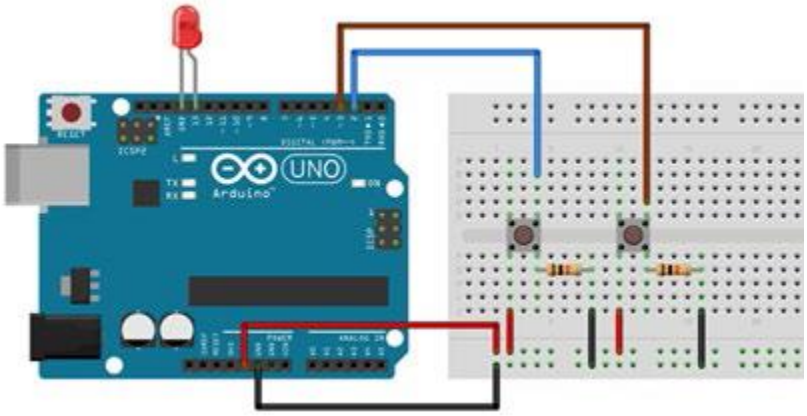
5.2 Acionar um led no pino 13 quando pressiona o botão ligado ao pino 2




```
/*
Alexandre Lima de Carvalho
Botao
Liga e desliga um LED conectado ao pino digital 13 quando pressionado um
botao conectado ao pino 2.
O Circuito:
* LED conectado ao pino 13 e ao terra
* botao conectado ao pino 2 desde 5V
* resistor de 10K conectado ao pino 2 desde o terra
*/
// Sao usadas aqui para definir os numeros dos pinos: ( constantes )
const int botao = 2; // o numero do pino do botão
const int led = 13; // o numero do pino do led
// declaração de variaveis
int estadobotao = 0; // variavel para ler o estado do botao

void setup()
{
// inicializa o pino do LED como saida:
   pinMode(led, OUTPUT);
// inicializa o pino do botao como entrada:
   pinMode(botao, INPUT);
}

void loop()
{
// faz a leitura do valor do botao:
     estadobotao = digitalRead(botao);
     // verifica se o botao esta pressionado.
      if (estadobotao == HIGH)
     {
       // liga o LED:
       digitalWrite(led, HIGH);
     }
     else
     {
        // desliga o LED:
```

```
        digitalWrite(led, LOW);
    }
}
```

Exercício : Alterar o programa acima para que somente se acionar dois botões (nos pinos 2 e 3) o led acenda .

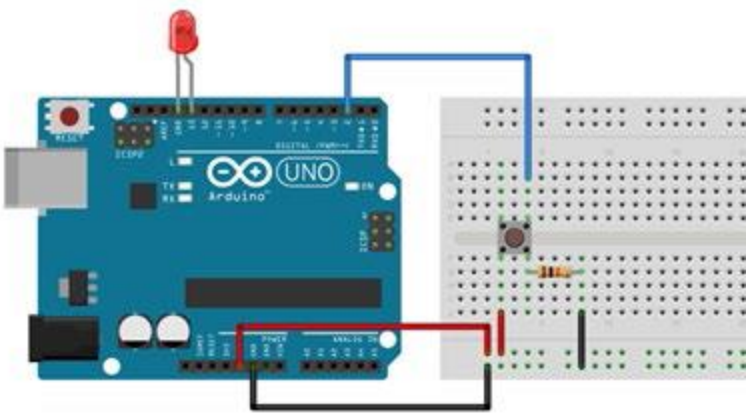


5.3 a) Leitura serial de uma entrada digital

OBS.: . configurar velocidade no monitor serial de 9600bps
. instrução print : envia um dado para o monitor do PC
. instrução println : envia o dado e vai para a linha de baixo

```
/*
Alexandre Lima de Carvalho
DigitalReadSerial
Le a entrada digital no pino 2 e imprime o resultado no monitor serial.
Este exemplo e de dominio publico.
*/
int botao = 2; // o pino 2 tem um botao ligado nele.
int led = 13; // entrada do LED no pino 13.

void setup()
{
   // Inicializa a comunicacao serial a 9600 bits por segundo:
   Serial.begin(9600);
   pinMode(botao, INPUT); // define o botao como uma entrada.
   pinMode(led, OUTPUT); //define o LED como uma saída.
}
void loop()
{
  // faz a leitura do pino de entrada:
  int estado = digitalRead(botao);  // define a variavel inteira estado
  if (estado == 1)
  {
    digitalWrite(led, HIGH);
  }
  else
```

```
  {
    digitalWrite(led, LOW);
  }
// imprime o estado do botao:
  Serial.println(estado);
  delay(1000); // delay entre leituras (em milissegundos)
}
```



b) Contador de pulsos no monitor serial

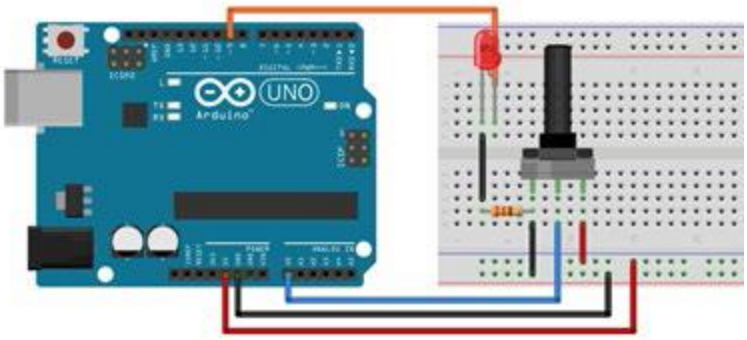
```
/*
Alexandre Lima de Carvalho
Le a entrada digital no pino 2 , conta quantas vezes a entrada variou em borda
de subida. A cada 5 variações acende um led no pino 13
Este exemplo e de dominio publico.
*/


// constantes nao sao alteradas:
const int botao = 2; // o numero do pino do botao
const int led = 13; // o numero do pino do LED

// variaveis que devem mudar:
int contador = 0; // contador para o numero de impressoes do botao
int estado = 0; // atual estado do botao
int anterior = 0; // anterior estado do botao

void setup()
{
  pinMode(botao, INPUT); // inicializa o pino do botao como entrada
  pinMode(led, OUTPUT); // inicializa o pino digital como saida
  Serial.begin(9600); // inicializa a comunicacao serial
}
void loop()
{
  // faz a leitura do valor do botao:
  estado = digitalRead(botao);
```

```
  // compara o estado atual do botao com seu estado anterior
  if (estado != anterior)
  {
      // se o estado do botao foi alterado em borda de subida
      if (digitalRead (botao) == HIGH)
      {
         contador++; // incrementa contador de pulsos contador++= contador +1
         Serial.print("numero de pulsos: ");  // escreve a mensagem
         Serial.println(contador); // escreve a quantidade de pulsos e pula linha
      }
  }

 // salva o estado atual do botao como ultimo estado para iniciar o próximo loop
  anterior = estado;

  // Liga o LED cada 5 pulsacoes
  if (contador % 5 == 0)
  {
     digitalWrite(led, HIGH);
  }
  else
  {
    digitalWrite(led, LOW);
  }
}
```

5.4 Leitura na porta serial de uma entrada analógica

```
/*
Alexandre Lima de Carvalho

Entrada Analogica, Saida Analogica, Saida serial
Le o pino de entrada analogica, mapeia o resultado para um intervalo entre 0
e 255 e usa o resultado para estabelecer o pulso PWM do pino de saida.
Tambem e possivel acompanhar o resultado atraves do Monitor Serial.
O circuito:
- O pino central do Potenciometro conectado ao pino analogico 0. Os pinos
lateriais do potenciometro conectados no terra e 5V.
- LED conectado no pino digital 9 e no terra.

*/
// constantes nao sao alteradas:
const int entradaAnal = A0; // Entrada analogica do potenciometro
const int saidaAnal = 9; // Saida analogica onde o LED esta conectado
int sensor = 0; // leitura do potenciometro
int valor = 0; // leitura da saida PWM (analogica) inicializa com Zero

void setup()
{
   // inicializa a comunicacao serial:
```

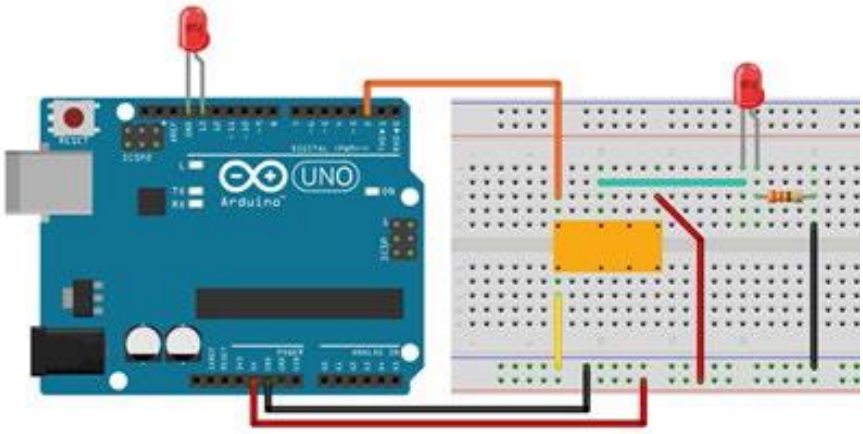
```
    Serial.begin(9600);
}

void loop()
{
     // faz a leitura da entrada analogica:
     sensor = analogRead(entradaAnal);

    // mapeia o resultado da entrada analogica dentro do intervalo de 0 a 255 -
   //muda para a escala de PWM de saída( 0 - 255 ), escala de leitura (0 - 1023)
     valor = map(sensor, 0, 1023, 0, 255);

     // muda o valor da saida analogica:
     analogWrite(saidaAnal, valor);

     // imprime o resultado no monitor serial:
     Serial.print("sensor = " );
     Serial.print(sensor);
     Serial.print("\t saída = ");
     Serial.println(valor);
    // Aguarda 100 milissegundos antes do proximo loop:
    delay(100);
}
```



5.5 Comando de comunicação serial

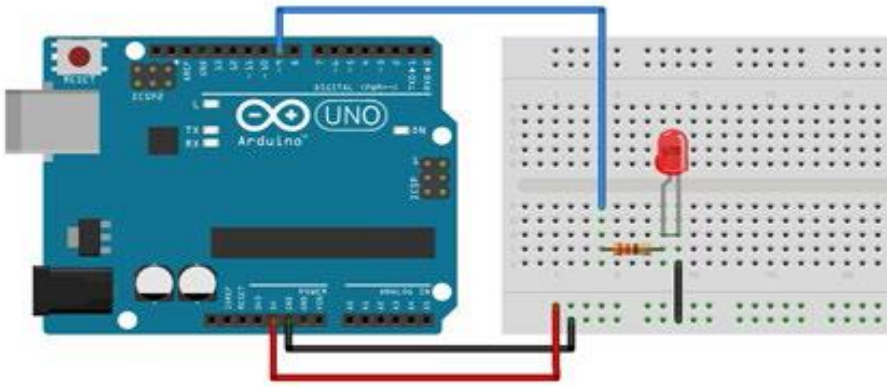
```
//*************************************************************************
//* Codigo para teste de Arduino acionando rele do kit Multilogica,
//* ligado na saida digital 2 e GND, monitorado pelo Led 13
//* Alexandre Lima de Carvalho
//*************************************************************************
//inicializa uma variavel do tipo char que utiliza 1 byte para armazenar 1
//caracter

char le= 0;
int rele=2;   // saída para o rele
```

```
int led=13;   // saída para led - pode utilizar o próprio led interno
boolean y=true; //inicializa uma variavel do tipo booleano para inverter posição

void setup()
{
     pinMode(rele,OUTPUT);
     pinMode(led,OUTPUT);
     Serial.begin(9600);  // inicializa comunicação e mensagem de comunicação
     Serial.println();
     Serial.print("**Codigo para acionar rele conectado ao pino 2 do Arduino ");
     Serial.println("atraves do monitor serial**");
     Serial.println("");
     Serial.println("Pressione 1 e depois ENTER para inverter o estado do rele
novamente");
     Serial.println("Aguardando comando :");
}

void loop()
{
   if (Serial.available() > 0)   // verifica se pode receber informação serial
   {
       le= Serial.read();  // le a porta serial
       if (le =='1')     // se digitou o número 1
       {
          Serial.print("O rele agora esta ");
          if(y)
          {
              digitalWrite(rele, HIGH);
              digitalWrite(led, HIGH);
              Serial.println("ligado");
          }
          else
          {
             digitalWrite(rele, LOW);
             digitalWrite(led, LOW);
             Serial.println("desligado");
          }
          y=!y; // inverte o valor da variavel booleana para inverter o rele no
                //próximo comando
       }
       else
       {
          Serial.println("Comando invalido");//se não digitou 1 aparece essa
                                              // mensagem
       }
   }
}
```



5.6 Variação do brilho (Fade)

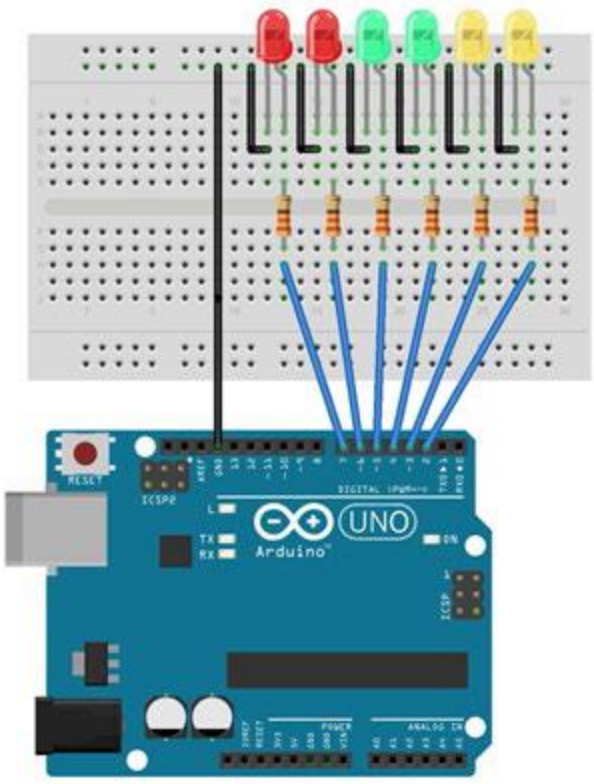
```
/*
Alexandre Lima de Carvalho
Fade ( variação gradual )
Este exemplo mostra como executar um fade em um LED no pino 9 usando a
funcao analogWrite()- escrita com valor analógico PWM ( 8 bits - 0 a 255 ).

*/
int led = 9; // pino do LED
int brilho = 0; // intensidade do brilho do LED
int passo = 5; // em quantos pontos aplicar o fade no LED

void setup()
{
   // define o pino 9 como saida:
   pinMode(led, OUTPUT);
}

// o loop roda em sequencia continuamente:
void loop()
{
    // define o brilho do pino 9:
    analogWrite(led, brilho);
   // muda o brilho para o proximo loop:
    brilho = brilho + passo;
// inverte a direcao do fade ao final do range analógico
    if (brilho == 0 || brilho == 255)
    {
         passo = -passo ;
    }
// aguarda 50 milissegundos para ver o efeito dimer:
delay(50);
}
```



5.7 Testando as estruturas repetitivas

```
/*
Alexandre Lima de Carvalho
equivale a montagem da super máquina do primeiro ano
Demonstra o uso da funcao for() loop.
Acende varios LEDs em sequencia e depois apaga

* LEDs entre os pinos 2 ao 7 e ao terra
*/

int tempo = 80; // Quanto maior o valor, mais lenta a sequencia de Leds.
void setup()
{
    // Use for loop para inicializar cada pino como saida:
    for (int pino = 2; pino < 8; pino++)  // declara a variavel pino com inteira e já
                                          //atribui um valor inicial dentro do laço for
    {
        pinMode(pino, OUTPUT);
    }
}

void loop()
{
    // loop desde o pino mais baixo ate o mais alto:
     for (int pino = 2; pino < 8; pino++) // declara a variavel pino novamente já
                                          // que ela é local
     {
         // liga este pino:
         digitalWrite(pino, HIGH);
         delay(tempo);
         // desliga este pino:
         digitalWrite(pino, LOW);
     }
    // loop desde o pino mais alto ate o mais baixo:
     for (int pino = 7; pino >= 2; pino--)
     {
```

```
        // liga este pino:
        digitalWrite(pino, HIGH);
        delay(tempo);
        // desliga este pino:
        digitalWrite(pino, LOW);
    }
}
```

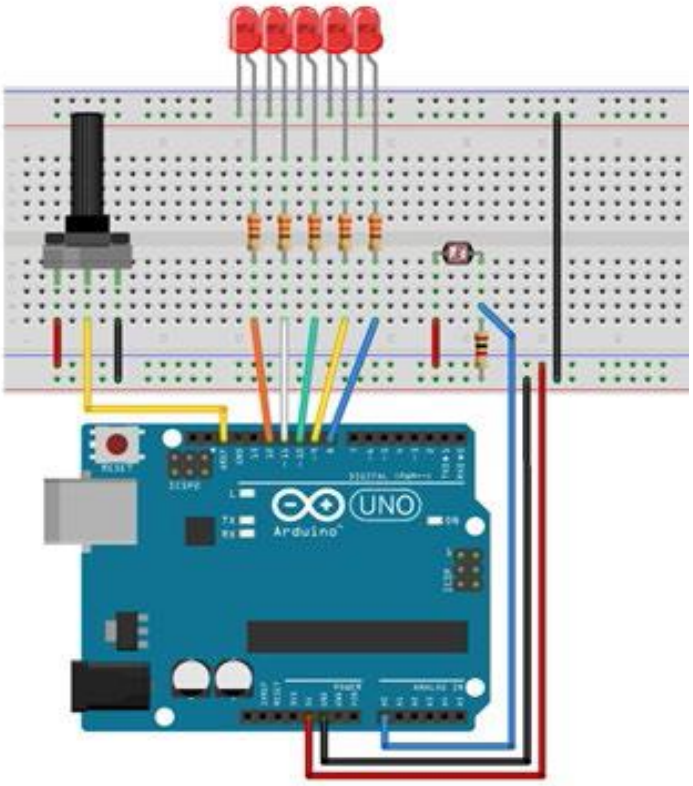


5.8 Sensor de luz – LDR

```
/*Alexandre Lima de Carvalho
Sensor LDR
Conectar um LDR a uma entrada analogica (A0) para controlar cinco saidas em
funcao da luz ambiente ( saidas 8 a 13 ).

*/
//Armazenar os dados recolhidos pelo sensor LDR:
int valorLDR = 0;
//Definir os pinos de entrada dos LEDs:
int led1 = 12;
int led2 = 11;
int led3 = 10;
int led4 = 9;
int led5 = 8;
//Definir pino de entrada do sensor LDR
```

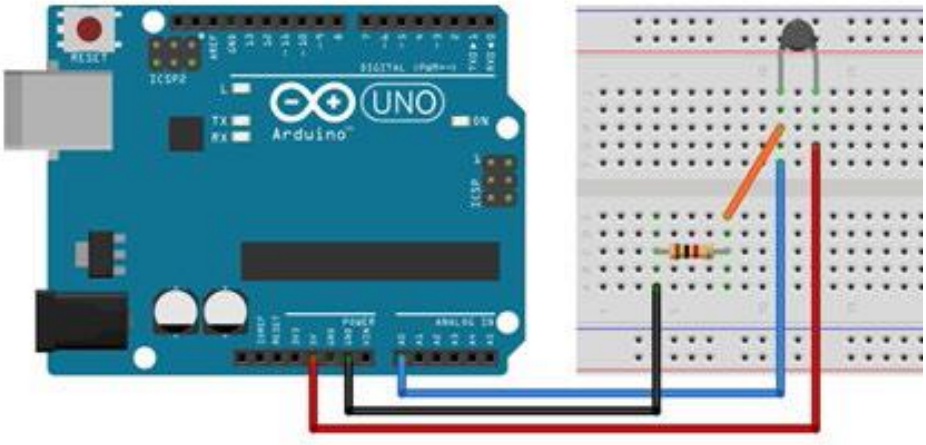
```
int pinLDR = 0;

void setup()
{
      Serial.begin(9600);
      //Definir os pinos de saida dos LEDs:
      pinMode(led1, OUTPUT);
      pinMode(led2, OUTPUT);
      pinMode(led3, OUTPUT);
      pinMode(led4, OUTPUT);
      pinMode(led5, OUTPUT);
      //  Definimos o uso de uma referencia externa , com um potenciometro na
entrada Aref :  pinMode(EXTERNAL);
}

void loop()
{
    //Guardar o valor da leitura de uma variavel:
    valorLDR = analogRead(pinLDR);  // leitura de valor de tensão segundo a
                                    // referencia ( potenciomentro na entrada external )
    Serial.println(valorLDR);

   //Definicao do padrao de controle dos LEDs:
    if(valorLDR >= 1023)  // todos apagados
    {
       digitalWrite(led1, LOW);
       digitalWrite(led2, LOW);
       digitalWrite(led3, LOW);
       digitalWrite(led4, LOW);
       digitalWrite(led5, LOW);
    }
    else if((valorLDR >= 823) & (valorLDR < 1023))   // acende primeiro led
    {
        digitalWrite(led1, HIGH);
        digitalWrite(led2, LOW);
        digitalWrite(led3, LOW);
        digitalWrite(led4, LOW);
        digitalWrite(led5, LOW);
    }
    else if((valorLDR >= 623) & (valorLDR < 823)) // acende primeiro e segundo
led
    {
        digitalWrite(led1, HIGH);
        digitalWrite(led2, HIGH);
        digitalWrite(led3, LOW);
        digitalWrite(led4, LOW);
        digitalWrite(led5, LOW);
    }
    else if((valorLDR >= 423) & (valorLDR < 623))  // acende led1, 2 e 3 e assim
                                                   //sucessivamente cada if
```

```
    {
        digitalWrite(led1, HIGH);
        digitalWrite(led2, HIGH);
        digitalWrite(led3, HIGH);
        digitalWrite(led4, LOW);
        digitalWrite(led5, LOW);
    }
    else if((valorLDR >= 223) & (valorLDR < 423))
    {
        digitalWrite(led1, HIGH);
        digitalWrite(led2, HIGH);
        digitalWrite(led3, HIGH);
        digitalWrite(led4, HIGH);
        digitalWrite(led5, LOW);
    }
    else
    {
        digitalWrite(led1, HIGH);
        digitalWrite(led2, HIGH);
        digitalWrite(led3, HIGH);
        digitalWrite(led4, HIGH);
        digitalWrite(led5, HIGH);
    }
}
```



5.9 Controle de temperatura – NTC

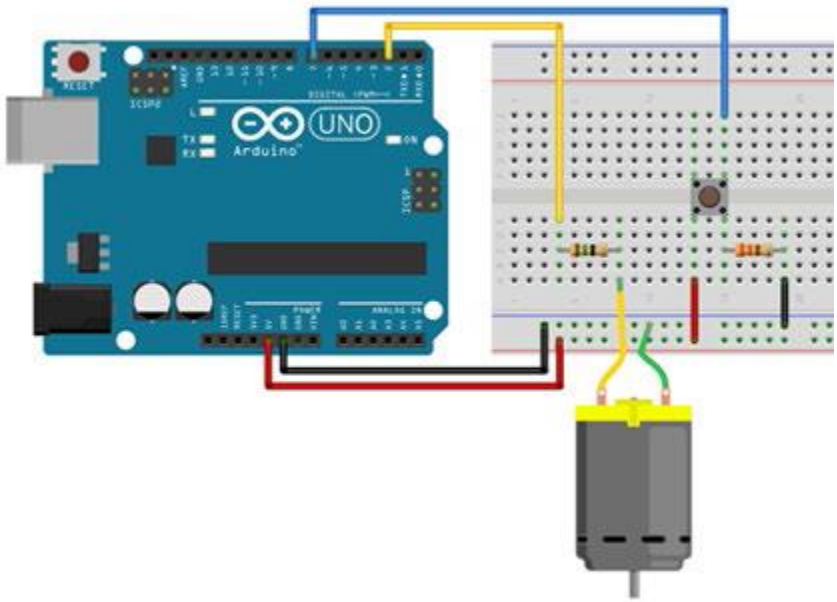
```
/* Codigo para leitura aproximada de temperatura
utilizando termistor de 1K do kit Multilogica
Note que nao e um termometro preciso, apenas um exemplo
aproximado baseado em dados empiricos.
Ligar resistor 1k de A0 para terra e termistor de +5V para A0 */

#define pino_termistor A0   // usando definição em vez de variavel ou constante

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

void loop(void)
{
  float leitura;
  float leitura1;
  leitura = analogRead(pino_termistor);  // leitura real da porta
  Serial.print("Leitura pino A0 = ");
  Serial.println(leitura);
  leitura1 = (leitura*0.2027)-72;      // leitura em escala
  Serial.print("Temperatura aprox. Celsius = ");
  Serial.println(leitura1);
  Serial.println("");
  delay(3000);
}
```



5.10 Motor cc

```
// Alexandre Lima de Carvalho
// Ligar motor no pino 2 em serie com um resistor de 15 ohms
// para limitar a corrente em 40mA para nao sobrecarregar o Arduino

const int motor = 2;
```

```
const int botao = 7;
int estado = 0;

void setup()
{
   pinMode(botao, INPUT);
   pinMode(motor, OUTPUT);
}

void loop()
{
   estado = digitalRead(botao);
   if (estado == HIGH)
   {
      digitalWrite(motor, HIGH);
   }
   else
   {
      digitalWrite(motor, LOW);
   }
}
```

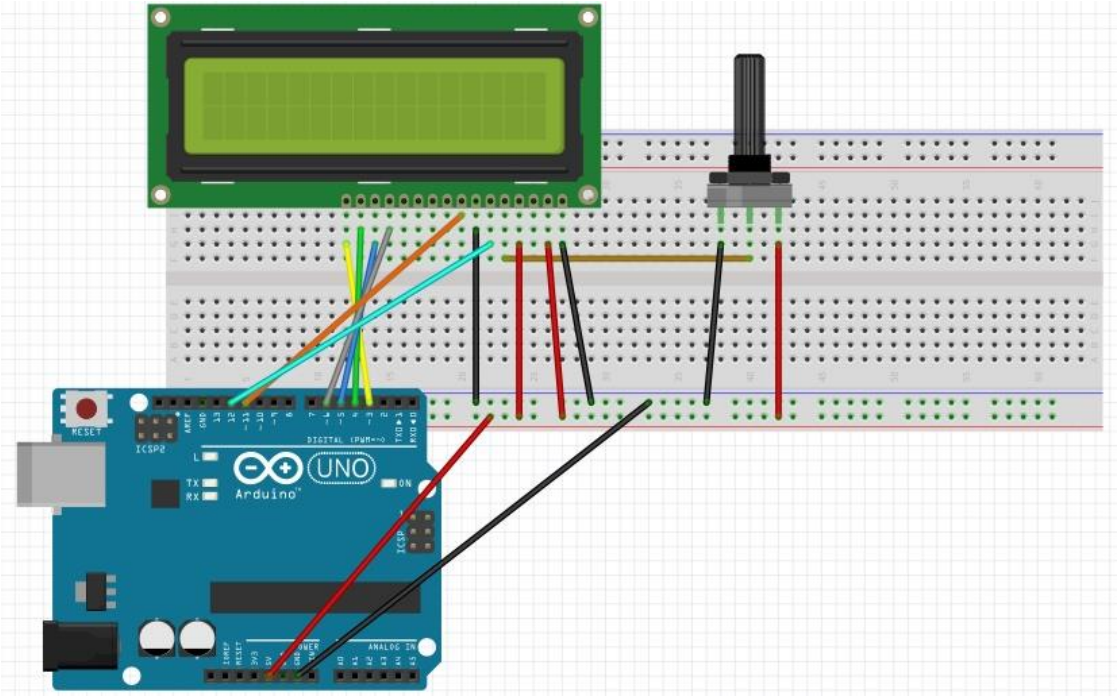


5.11 Trabalhando com LCD
A)

```
/*
Alexandre Lima de Carvalho
Biblioteca LiquidCrystal
Demonstra o uso do display de 16x2 caracteres
Esta biblioteca funciona com todos displays compativeis com o
driver Hitachi HD44780.
```

Circuito :

```
* LCD pino RS no pino digital 12
* LCD pino Enable no pino digital 11
* LCD pino D4 pin no pino digital 5
* LCD pino D5 pin no pino digital 4
* LCD pino D6 pin no pino digital 3
* LCD pino D7 pin no pino digital 2
* LCD pino R/W no terra
* Trimpot de 10K :
* +5V no +5V
* Terra no terra
* wiper to LCD VO pin (pin 3)
Codigo de dominio publico baseado no tutorial original :
http://www.arduino.cc/en/Tutorial/LiquidCrystal
*/

// Inclui o codigo da biblioteca:
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
   // define o numero de colunas e linhas do Display :
   lcd.begin(16, 2);
   // Envia a mensagem para o display.
   lcd.print("Alexandre Carv");
   lcd.setCursor(0, 1); //Posiciona o cursor na primeira coluna(0) e na   segunda
linha(1) do Display
   lcd.print(" ETECSP - 2014 ");
}

void loop()
{
}
```



B)

```
/*
Alexandre Lima de Carvalho
Biblioteca LiquidCrystal

// Inclui o codigo da biblioteca:
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()      // inicializa display
{
    // define o numero de colunas e linhas do Display :
    lcd.begin(16, 2);
    // Envia a mensagem para o display.
    lcd.print("Alexandre Carv");
    lcd.setCursor(0, 1); //Posiciona o cursor na primeira coluna(0) e na   segunda
linha(1) do Display
    lcd.print(" ETECSP - 2014 ");
}
void loop()
{
   delay (10000);
   lcd.setCursor (0,0);
   lcd.print ( " segundo ano ELO");
   lcd.setCursor (10,1);
   lcd.print ( "SM I ");   // notar que ele não apaga os caracteres de baixo
   delay ( 8000 );
}
```

C)

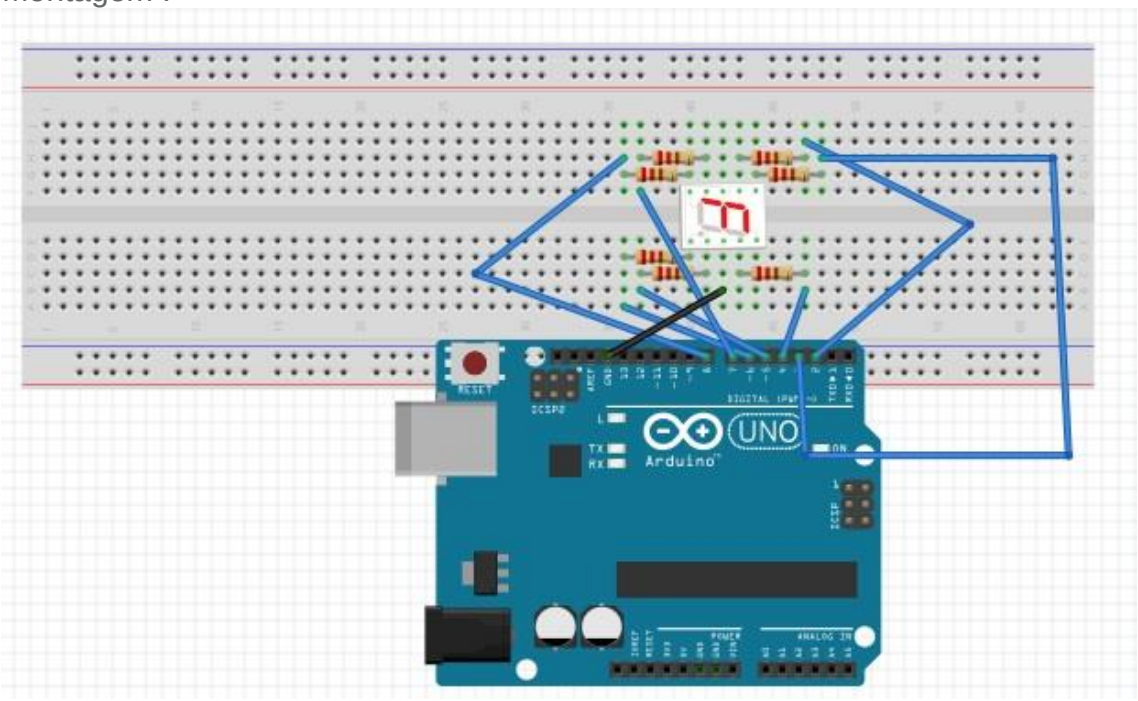
```
/*
Alexandre Lima de Carvalho
Biblioteca LiquidCrystal
Demonstra o uso do display de 16x2 caracteres
*/
// Inclui o codigo da biblioteca:
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()      // inicializa display
{
    // define o numero de colunas e linhas do Display :
    lcd.begin(16, 2);
    // Envia a mensagem para o display.
    lcd.print("Alexandre Carv");
    lcd.setCursor(0, 1); //Posiciona o cursor na primeira coluna(0) e na   segunda
linha(1) do Display
    lcd.print(" ETECSP - 2014 ");
}
void loop()
{
   delay (10000);
   lcd.setCursor (0,0);
   lcd.print ( " segundo ano ELO");
   lcd.setCursor (10,1);
   lcd.print ( "SM I ");   // notar que ele não apaga os caracteres de baixo
   delay ( 8000 );

   for (int posicao = 0; posicao < 16; posicao++)
   {
      // caminha uma posicao para a esquerda:
      lcd.scrollDisplayLeft();
      // Aguarda um instante:
      delay(400);
    }
     // caminha 32 posicoes para o texto sair do display a direita:
    for (int posicao = 0; posicao < 32; posicao++)
    {
      // caminha uma posicao para a direita:
      lcd.scrollDisplayRight();
      // Aguarda um instante:
      delay(300);
    }
    // caminha 16 posicoes para a esquerda para mover de novo ao centro:
    for (int posicao = 0; posicao < 16; posicao++)
    {
      // caminha uma posicao para a esquerda:
      lcd.scrollDisplayLeft();
      // Aguarda um instante:
      delay(250);
```

```
    }
    // delay no final do  loop:
    delay(2000);
}
```

5.12 Trabalhando com display de 7 segmentos

Utilizaremos o conceito de matriz e de funções auxiliares para implementar este exemplo.

Montagem :



Programa :

```
// Alexandre Lima de Carvalho
// Programa display de 7 segmentos - catodo comum
// ligar pinos : 2 no a , 3 no b , ... , 8 no g do display

// definindo matrizes dos números
int zero[7] = {1,1,1,1,1,1,0};
int um[7] = {0,1,1,0,0,0,0};
int dois[7] = {1,1,0,1,1,0,1};
int tres[7] = {1,1,1,1,0,0,1};
int quatro[7] = {0,1,1,0,0,1,1};
int cinco[7] = {1,0,1,1,0,1,1};
int seis[7] = {1,0,1,1,1,1,1};
int sete[7] = {1,1,1,0,0,0,0};
int oito[7] = {1,1,1,1,1,1,1};
int nove[7] = {1,1,1,1,0,1,1};
int envia[7] = {0,0,0,0,0,0,0};
int cont =0;

void setup()
{
```

```
  // configurando do pino 2 ao 8 como saída
  for (cont=2;cont<9;cont++)
  {
     pinMode(cont, OUTPUT);
  }
}


void loop()
{

// não foi vantajoso criar a função pois NÂO podemos copiar uma matriz
//  diretamente para outra

   for (cont=0;cont<7;cont++)
      envia[cont] = zero[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = um[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = dois[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = tres[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = quatro[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = cinco[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = seis[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = sete[cont];
   numero();

   for (cont=0;cont<7;cont++)
      envia[cont] = oito[cont];
   numero();
```

```
   for (cont=0;cont<7;cont++)
       envia[cont] = nove[cont];
   numero();
}

// função extra para exemplificar a utilização de funções que se repetem no
 // programa

void numero()
{
      for (cont=0;cont<7;cont++)
  {
        digitalWrite(cont+2, envia[cont]);   // turn the LED on (HIGH is the voltage
level)
  }
  delay(1000);
}
```

6. Alimentação Externa Arduino

.Terminal Bateria: Centro Positivo.

Podemos alimentar a placa do Arduino com uma bateria de 9 V , adquirindo os adaptadores do desenho abaixo; lembrando que o terminal central do adaptador deve ser conectado ao fio vermelho (positivo da bateria).