UNO BÁSICO COMPLETO

Por Eng. Roberto Bairros dos Santos

email: professor.bairros@gmail.com
www.bairrospd.com
Data:10/08/2017

Sumário

Prefácio:

Este tutorial foi criado para mostrar exemplos de programas usando o Arduino UNO e simulando no Proteus ISIS com o componente SIMULINO.

Para este tutorial você deve ter conhecimentos básicos de linguagem “C”, hardware do Arduino UNO e da interface de programação do Arduino, o tutorial UNO BASICO 01 abrange estes tópicos!

Para a simulação foi usado o Proteus, você pode baixar da internet no link abaixo.
http://wle.ir/hadi/4-Program/Proteus/P85SP0/P85SP0.rar

A primeira vez observe a data de validação, o melhor é alterar para uma data mais longa!

Para alterar a data e validar a licença primeiro feche o Proteus, esta alteração tem que ser feita com o Proteus fechado.

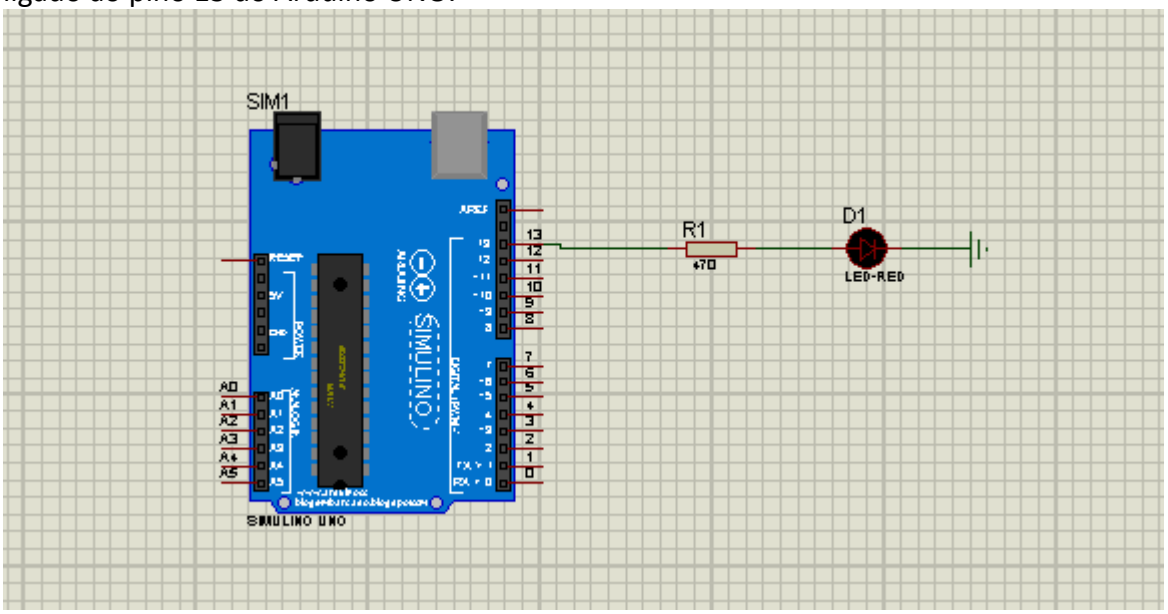
Para alterar a data abra o arquivo LicenseKey.lxk usando um editor de texto e altere a configuração “EXPIRY=31/02/2050”.
Para achar a configuração “EXPIRE” no editor de texto abra o menu “Editar” opção “Localizar” e digite no campo “EXPIRY”!

Para atualizar a chave rode a “LicenseKey.lxk” com o Proteus fechado.

Rode o Proteus e confirma a data de validação!

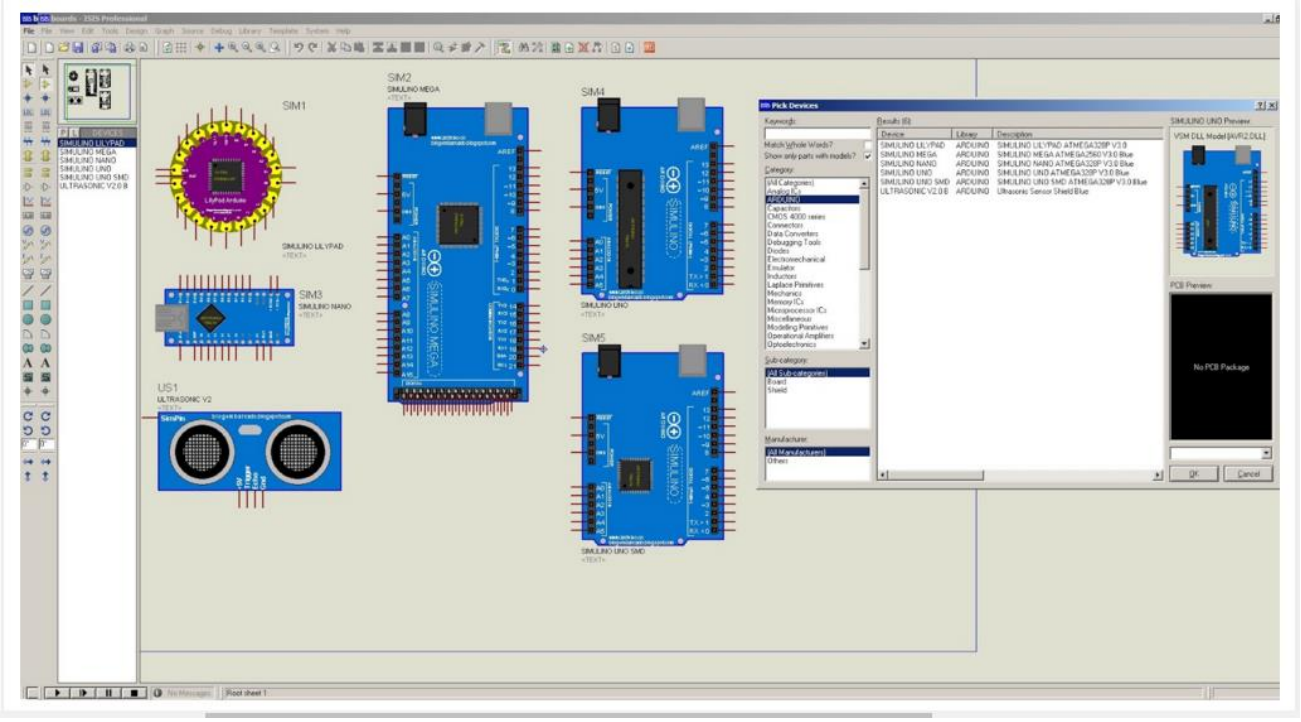
Meu primeiro programa pisca-pisca (blink):

Neste tutorial você vai aprender a usar e testar os exemplos do Arduino.
Os exemplos são úteis para auxiliar você a aprender a trabalhar com o Arduino e a orientar em novos projetos.
O primeiro passo para criar um projeto é criar uma pasta para colocar todos os arquivos do seu projeto como o circuito montado no ISIS, os arquivos compilados para serem carregados no Arduino do simulador.
O circuito deverá ser montado no programa Proteus Isis conforme o desenho abaixo com um led ligado ao pino 13 do Arduino UNO.

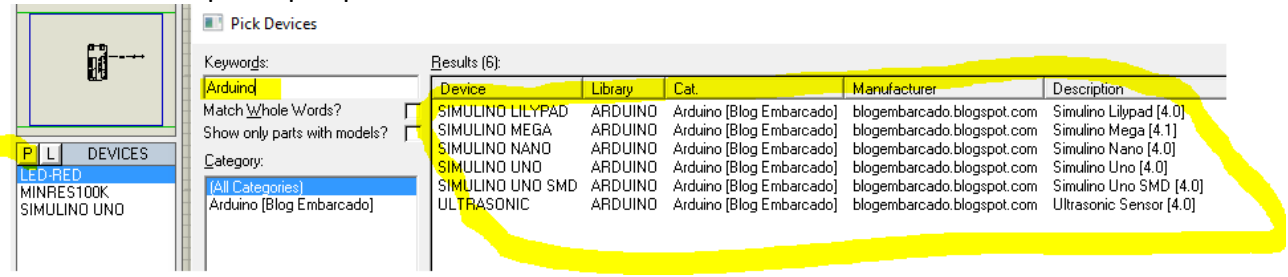


O componente do Arduino:

O componente do Arduino é da biblioteca do site no link abaixo.
http://blogembarcado.blogspot.com.br/2013/06/simulino-v20-biblioteca-para-proteus.html
O site dá informações de como baixar e instalar a biblioteca no Proteus.

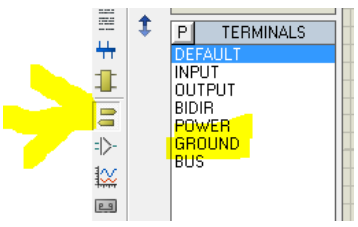


Para inserir o componente no Proteus Isis clique no botão “P” para pegar o compoente de digite Arduino no campo de pesquisa.



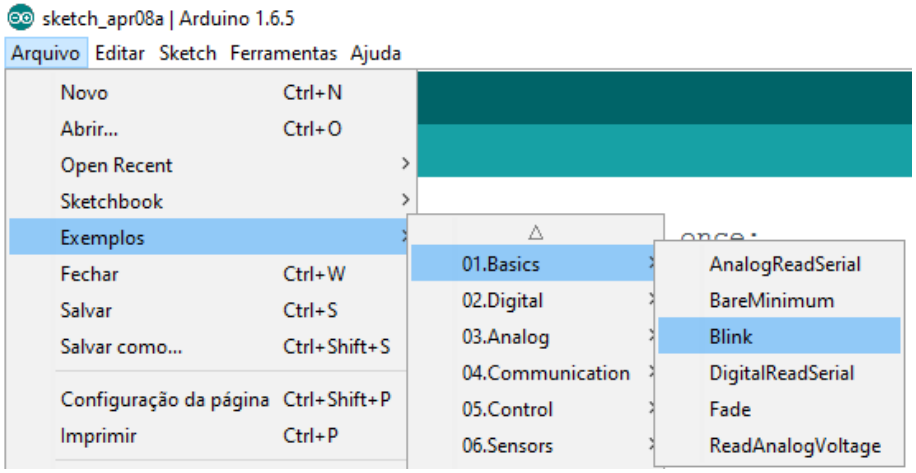
Você deverá proceder da mesma forma para pegar o LED e o resistor, altere o valor do resistor para 470 Ohm!

Relembrando que o pino de terra (groud), alimentação (power) e conexão simples (default) você encontra na aba da figura abaixo no Proteus Isis.



Usando um programa Exemplo do Arduino:

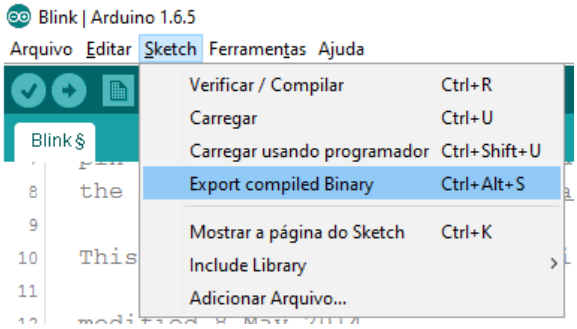
Depois de montar o diagrama e salvar na pasta do seu projeto, depois o seu trabalho será montar o software na interface do Arduino.
Para abrir um exemplo do pisca pisca (blink em inglês) você deve seguir o caminho mostrado abaixo.



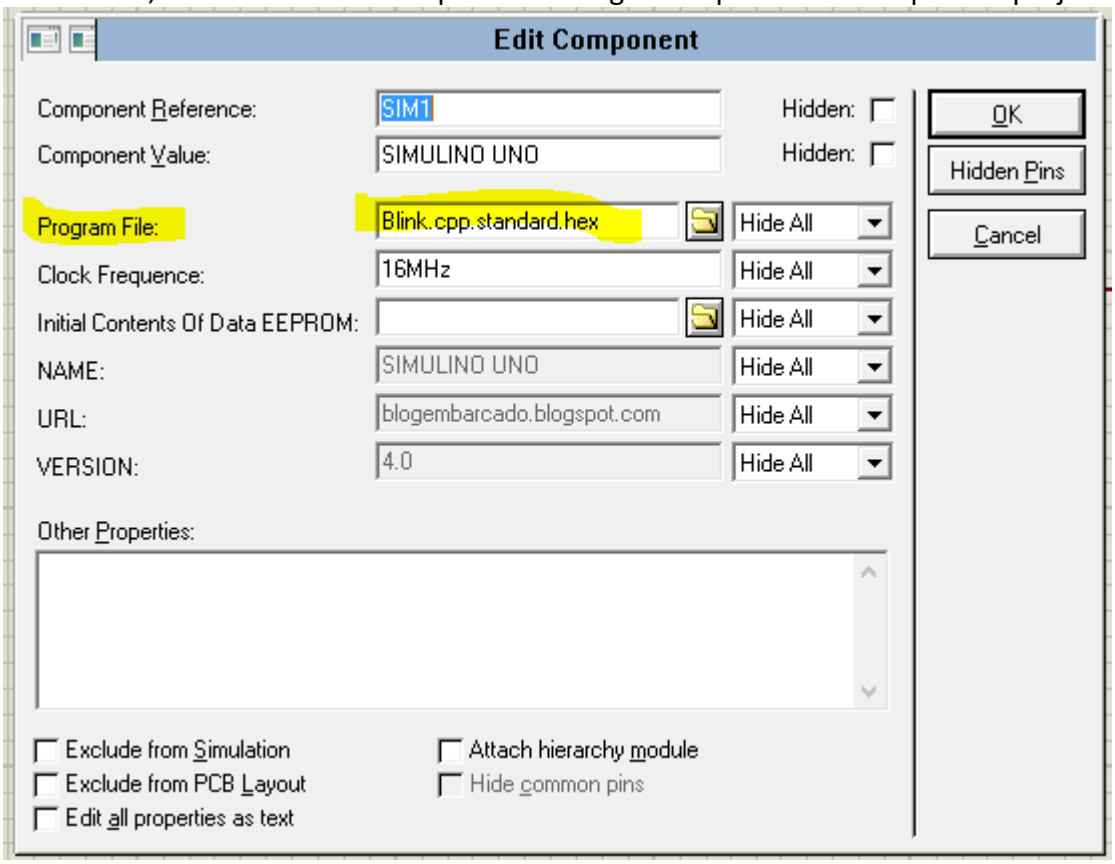
Após abrir o projeto a tela abaixo é mostrada.

Como gerar o arquivo .hex para usar no simulador:

Este é um exemplo completo onde o led está configurado na saída 13.
Uma vez pronto o software você deverá gerar o arquivo .hex que será carregado no Arduino do circuito desenhado anteriormente.
Para gerar o arquivo ponto hexa (.hex) siga o caminho da figura abaixo que exporta o arquivo compilado para a pasta do seu projeto.

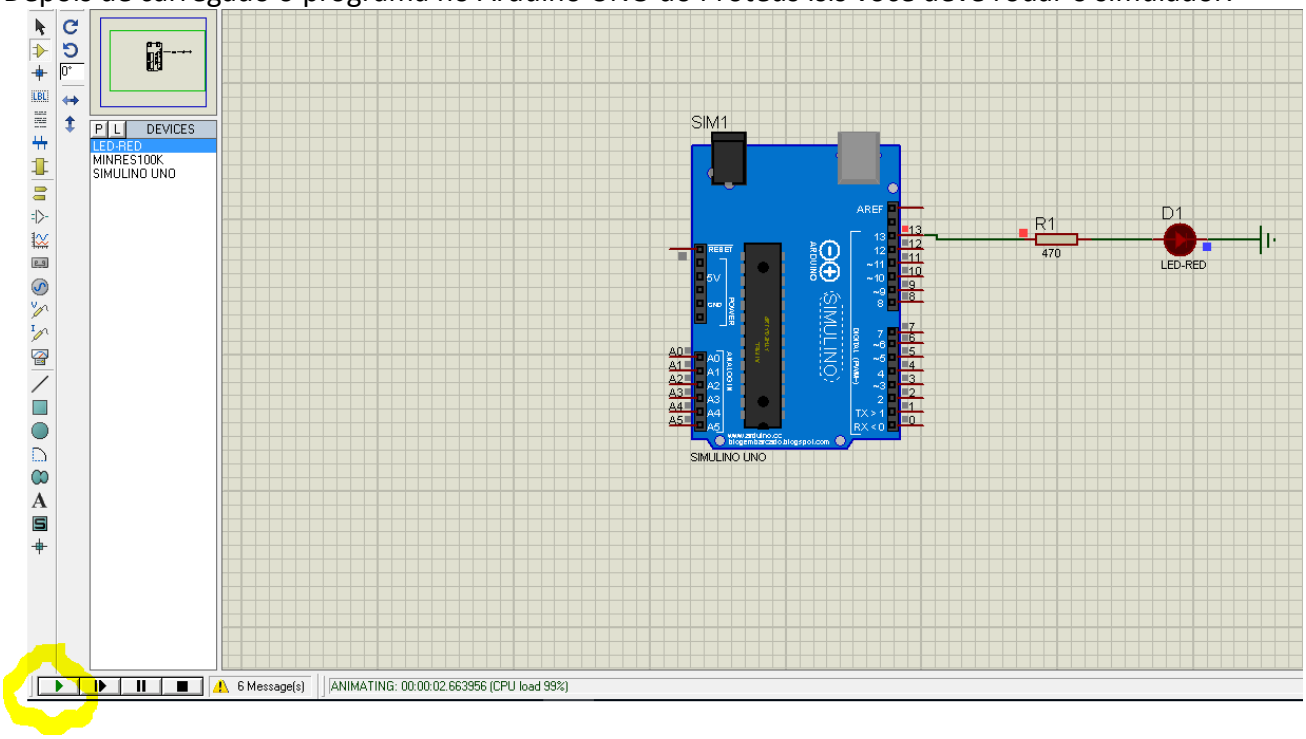


Uma vez gerado o arquivo .hex você deverá carregar este arquivo no Arduino, para isto, clique no Arduino dentro do Proteus Isis e selecione a opção “Preference”, a janela abaixo será aberta.
Use o browse para procurar o arquivo .hex que deverá estar em uma pasta com o nome do arquivo do Arduino, neste caso é blink! A pasta Blink é gerada quando você exporta o projeto do Arduino.



Como carregar o arquivo ponto hex no simulador:

Depois de carregado o programa no Arduino UNO do Proteus Isis você deve rodar o simulador.



Revisando o ambiente de programação e as instruções do Arduino:

Se você não conhece o ambiente de programação do Arduino sugiro que veja o tutorial Arduino 01, mas agora vamos fazer uma revisão!
Bem no início do programa existe um comentário que descreve o programa, é interessante você sempre comentar os seus programas. Os comentários seguem o padrão da linguagem “C”.
Para mais de uma linha inicie o comentário com barra invertida e asterisco “/*” e feche com asterisco e barra invertida “*/”, para comentário de uma linha coloque duas barras invertidas ” //”!

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */

// the setup function runs once when you press reset or power the board
```

O ambiente de programação do Arduino possui duas funções o setup() e o loop().

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

A função setup():

Na função setup() você irá configurar o microcontrolador descrevendo o hardware como: pinos e funções das portas d entrada e saída, configuração do LCD, configuração da Serial etc.
O setup roda somente ao ligar o microcontrolador ou o reset ser ativado!

```
15
16 // the setup function runs once when you press reset or power the board
17 void setup() {
18   // initialize digital pin 13 as an output.
19   pinMode(13, OUTPUT);
20 }
```

Instrução para configuração da porta digital.

A função abaixo mostra como configurar uma porta do Arduino.
O primeiro parâmetro define o endereço da porta, este endereço vem descrito na placa do Arduino.
O segundo parâmetro define se a porta será usada como entrada (INPUT) ou saída (OUTPUT).
Não esqueça que a linguagem "C" é sensível ao caso e que a instrução deve terminar sempre com ponto e vírgula!

```
// initialize digital pin 13 as an output.
pinMode(13, OUTPUT);
```

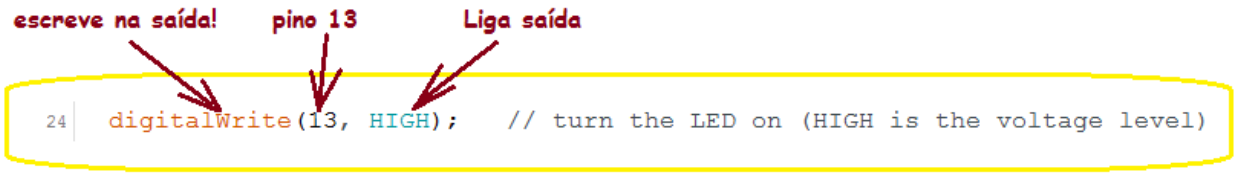
A função loop():

A função loop é o laço infinito, ficará rodando enquanto o microcontrolador tiver ligado.
A na função loop() que você irá escrever o programa em linguagem “C”!

```
21
22 // the loop function runs over and over again forever
23 void loop() {
24   digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
25   delay(1000);              // wait for a second
26   digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
27   delay(1000);              // wait for a second
28 }
```

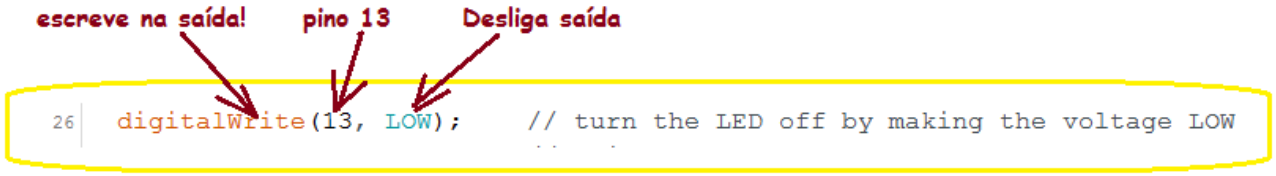
Instrução para ligar uma porta de saída digital:

Pra ligar uma saída use a função digitalWrite().
O primeiro parâmetro é o endereço da porta.
O segundo parâmetro é a ação; Ligar (HIGH), desligar (LOW)!
HIGH significa nível lógico alto (5V), e LOW significa nível baixo baixo (0V)!



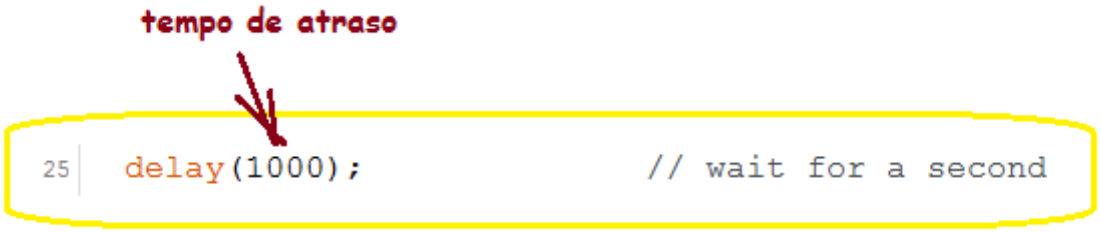
Instrução para desligar uma saída digital!

Pra desligar uma saída use a função digitalWrite() definindo o segundo parâmetro como LOW!
Observe que uma vez ligada uma saída específica ela permanece ligada até que uma instrução de desligar esta saída seja encontrada.



Instrução de tempo de atraso!

Na instrução de tempo de atraso prende o programa por um espaço de tempo de forma a retardar a sequência de processamento dando um atraso de tempo entre a instrução anterior e a seguinte.
Atraso em inglês é delay.
A função delay() (atraso) é mostrada abaixo e o parâmetro é definido em milissegundos!



Descrição da rotina do pisca-pisca:

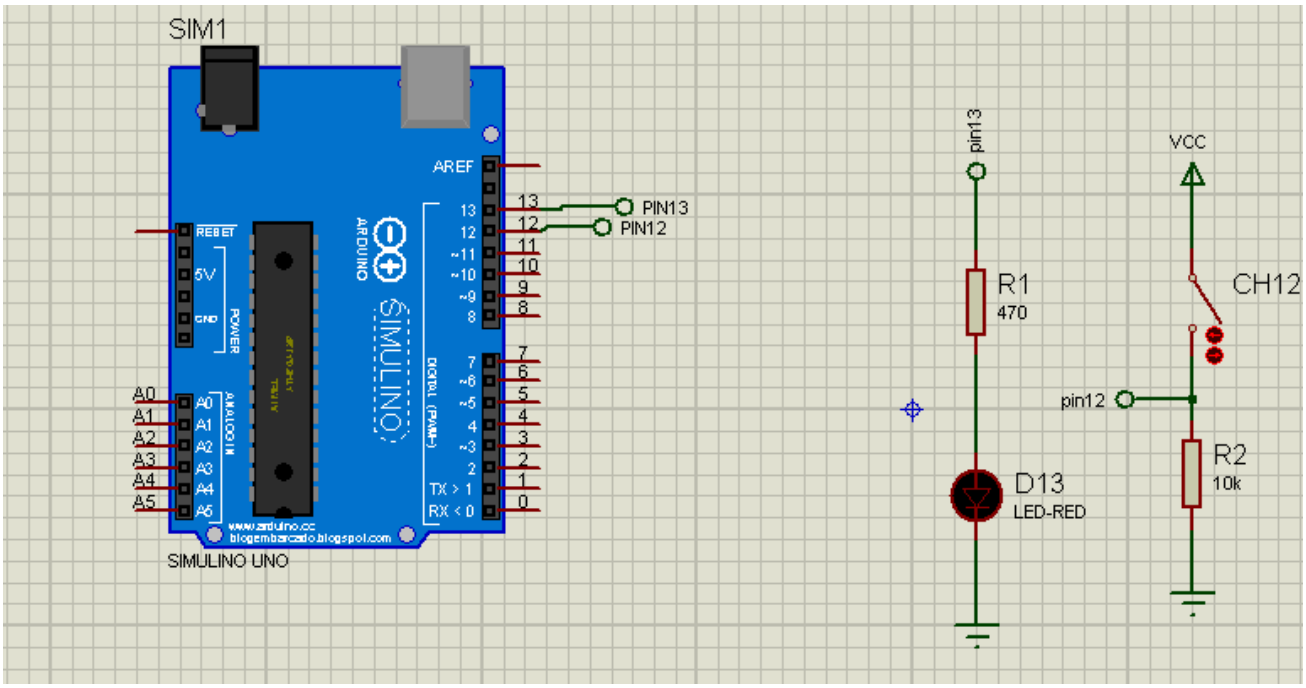
A rotina do pisca-pisca está toda dentro da função loop() e é composta dos seguintes passos:
Linha 24, liga o LED.
Linha 25, espera 1 segundo (1000 ms) com o LED ligado.
Linha 26, desliga o LED.
Linha 27, pera 1 segundo (1000 ms) com o LED ligado.
Linha 28, o colchete termina o loop e a sequência volta para a linha inicial 24!

```
21
22 // the loop function runs over and over again forever
23 void loop() {
24   digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
25   delay(1000);              // wait for a second
26   digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
27   delay(1000);              // wait for a second
28 }
```

Programa chave LED:

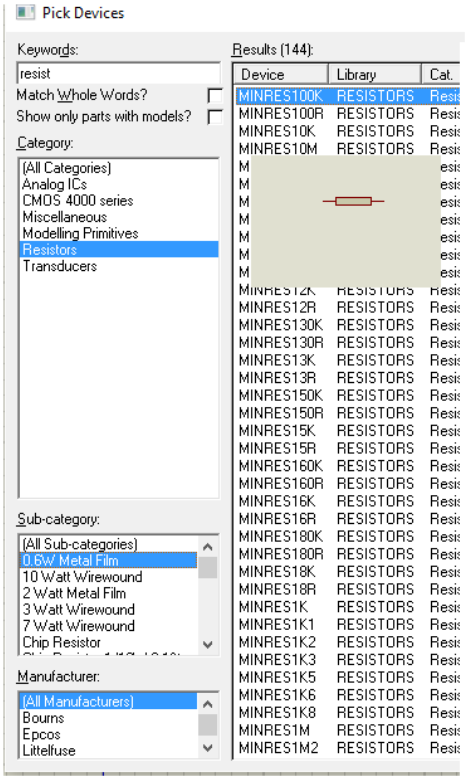
Neste tutorial você irá adicionar uma chave no projeto do pisca-pisca para ligar e desligar o pisca-pisca!
Note que no diagrama foram usados conectores, isto deixa o diagrama mais limpo e colocando a chave do lado do LED fica mais simples de visualizar o resultado!

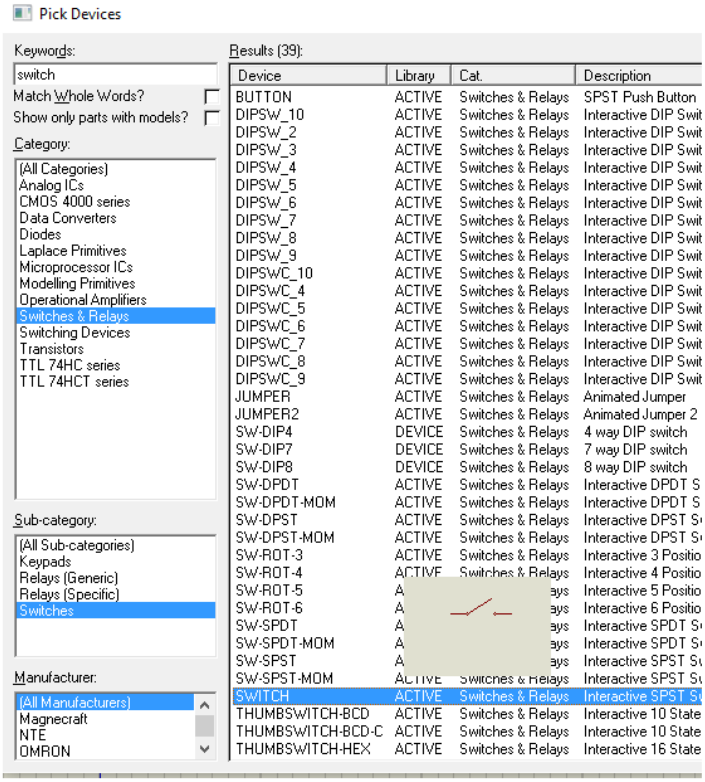
Para que o circuito tenha uma relação com o comando é de boa prática numerar os componentes que vão ligados ao microcontrolador com o número da porta onde estão conectados, assim fica mais fácil entender o circuito. A chave CH12 está ligada na porta 12 do Arduino!



Alterando o diagrama no Proteus Isis:

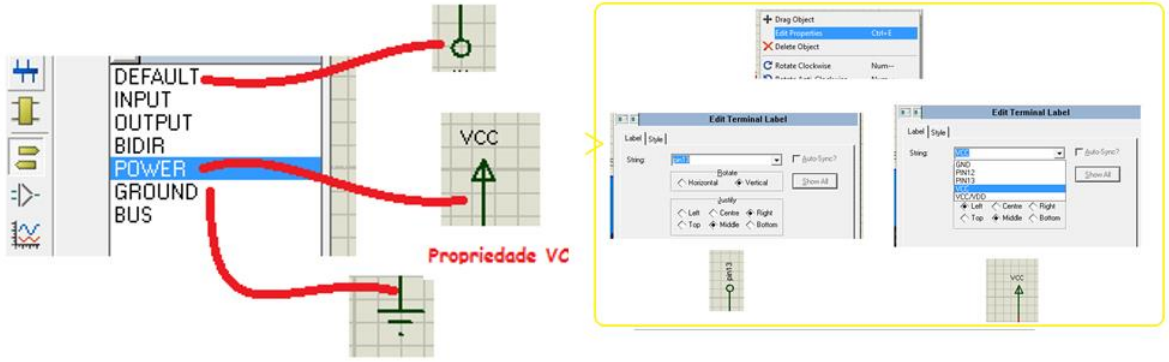
Primeiro você altera o diagrama, desta forma você saberá os pinos das entradas e saídas usadas, pois você precisa desta informação para alterar o software.
Pegue na biblioteca os componentes para completar o circuito, a chave descreva como “switch” na pesquisa, refina a pesquisa selecionando “Swithes & Relays” no campo “Category” e depois switch no campo “sub-category”! Escolha SWITCH ACTIVE!
Para resistor digite “resist” selecione qualquer um de 0,6W depois você irá alterar o valor!





Revisando os conectores:

Para definir o nome do conector use a caixa de propriedades e escreva o nome se não tiver ainda, se já tiver o nome selecione do botão de seleção.
Para o conector de potência selecione no botão de opções o “VCC”!



Alterando o software do Arduino:

O programa completo é mostrado abaixo, e a seguir é feita uma descrição de cada uma das linhas de instrução, altere o programa “Blink” conforme descrito abaixo, compile e exporte e rode no ISIS!

```
int led13=13;
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(led13, OUTPUT);
  pinMode(12, INPUT);
}
// the loop function runs over and over again forever
void loop() {
  int chave12=digitalRead(12);//lê o valor da entrada digital
  if (chave12==HIGH){
    //se a chave estiver ligada
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);              // wait for a second
  }
   else{
    //senão chave deslighada
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  }
 }
```

Sempre que você colocar um componente de entrada ou saída no seu circuito, você deverá configurar este componente na função setup(), antes de usar na função loop()!

Para que o software fica mais legível é bom dar nomes aos endereços dos pinos, você pode fazer isto de várias formas:
Definindo o nome da porta com o endereço usando a diretiva “#define” ou declarando uma variável com valor igual ao número da porta, ou ainda declarando uma constante estes métodos são mostrados abaixo.
No uso da diretiva define o nome usado não poderá ser usado como de nome de outra variável.

Vamos usar neste trabalho a recomendação do fabricante do Arduino que dá preferência a definição usando variáveis.

```
#define 13
int led=13;
const int led=13;
```

A definição da variável pode ser feita em qualquer parte do programa, normalmente é feita antes da função setup().

No caso da chave12 a definição é feita ao pegar o dado da chave dentro do loop(), esta é uma prática bastante comum no Arduino.

```
24   int chave12=digitalRead(12);//lê o valor da entrada digital
```

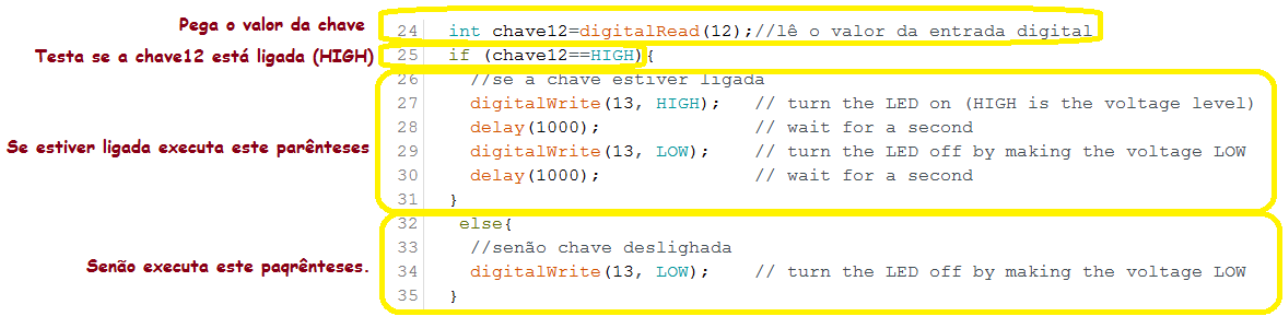
Depois de ler a chave você vai testar o valor da chave, lembrando que a chave é um dispositivo digital só pode assumir dois valores 1 (HIGH) ou 0 (LOW).

A instrução que testa o valor é a instrução if() que significa “SE” em inglês, esta é uma instrução muito importante em linguagem de computador.

A instrução if() testa o valor dentro do parênteses, neste caso se a chave estiver ligada.

```
25  if (chave12==HIGH){
26    //se a chave estiver ligada
```

Se a resposta for afirmativa, o programa executa a instrução dentro do primeiro parênteses, senão (resposta é negativa) o programa executa as instruções dentro dos parênteses depois do else que significa “SENÃO” em inglês!



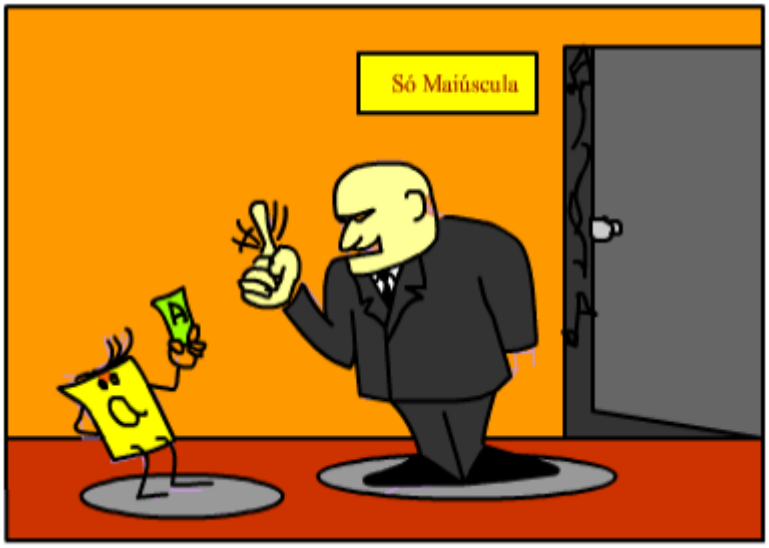
Desafio 1:

DESAFIO 1:
Insira um segundo led que deverá piscar ao inverso do led já instalado, quando a chave estiver desligada os dois leds deverão ficar apagados.

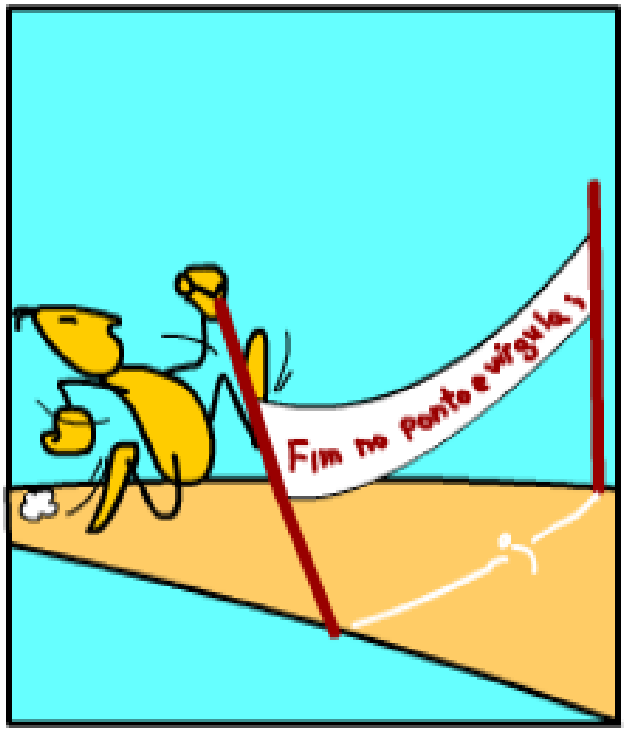
Regra básicas da linguagem “C”

As características básicas da linguagem "C" e "C++" são listadas abaixo.

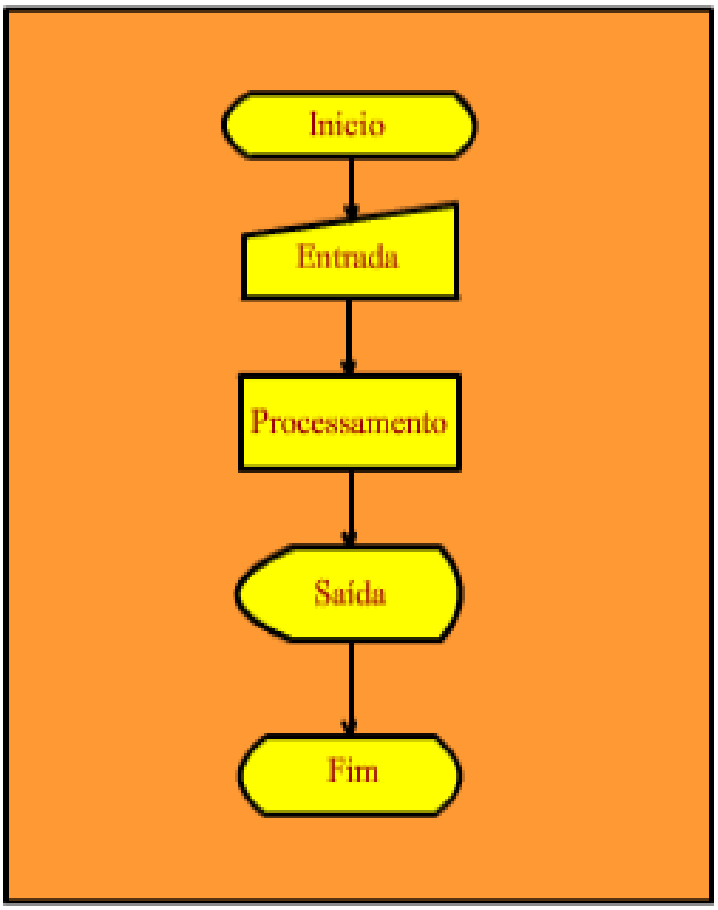
A linguagem “C” é sensível ao caso, isto é, diferencia letras minúsculas de maiúsculas.



Uma instrução sempre termina em ponto e vírgula.



Um programa é feito por uma sequência de instruções, o microcontrolador executa uma a uma cada instrução!



Um comentário não é uma instrução, ele simplesmente descreve a instrução servindo para que você lembre o que estava fazendo ou outro programador entenda o seu programa.

Você escreve um comentário depois de duas barras invertidas!

```
/*
Este é um comentário
de mais de uma linha
você pode colocar quantas linhas quiser!
*/

// este é um comentário de uma linha
//tem que iniciar com as duas barras
//e não precisa terminar com duas barras
//tudo que está escrito depois das duas barras é comentário
```

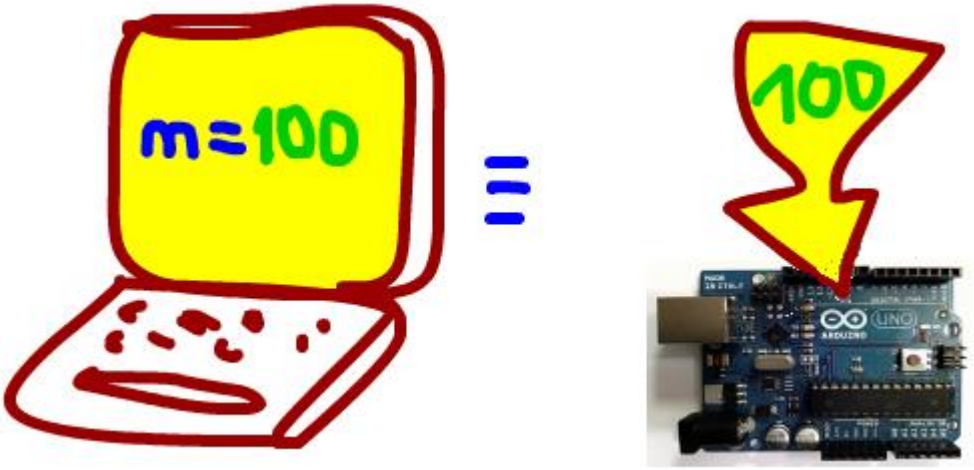
Comente sempre seus programas!

As variáveis precisam ser declaradas antes de serem usadas.

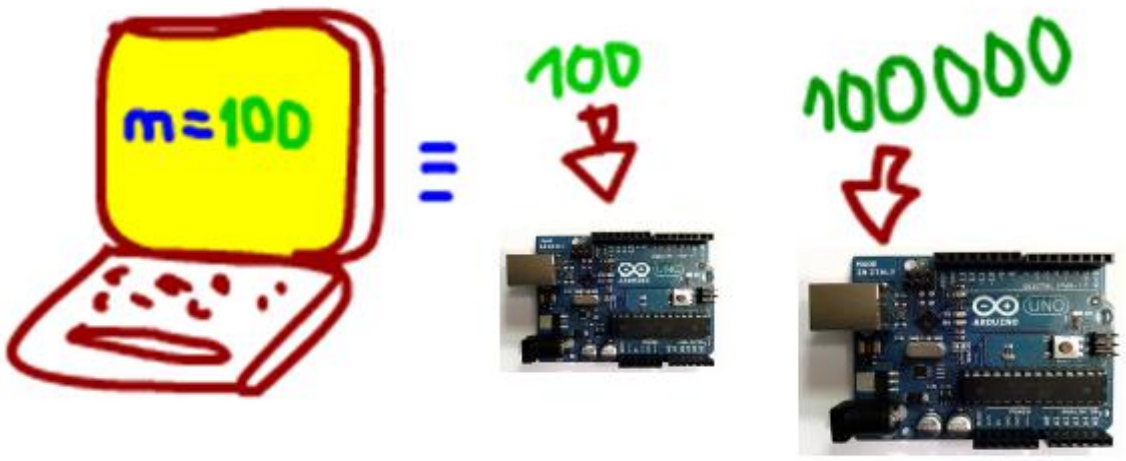
Como definir uma variável em linguagem “C”

Uma variável é um valor numérico usado no seu programa e que deve ser armazenado na memória interna do microcontrolador.

Uma memória é guardada em um endereço específico da memória, em linguagem “c” este endereço é descrito como um nome.



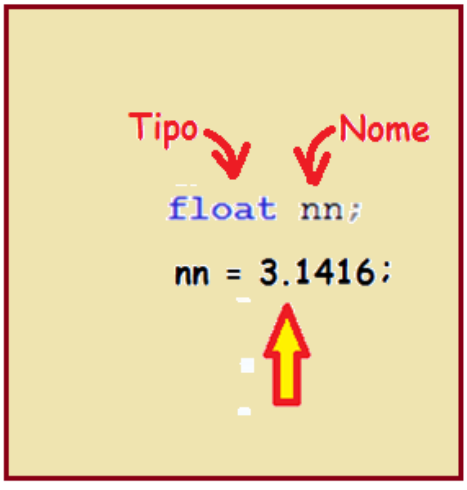
Você tem que dizer ao programa qual o tamanho da variável para que o programa reserve uma área apropriada para guardar este dado.

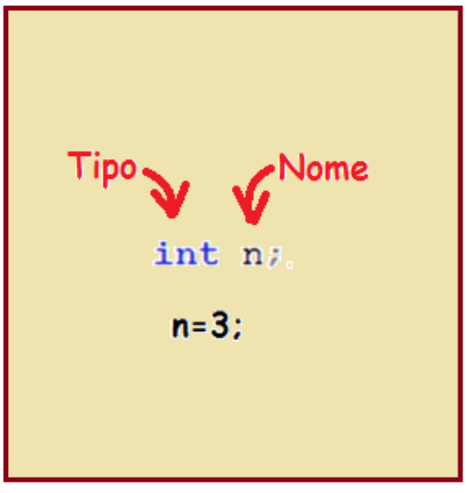


Dizer o nome e o tamanho da variável é chamado de declarar uma variável e você deverá sempre declarar uma variável antes de usar, é como batizar uma variável.

Ao declarar uma variável você deve dizer o tipo e dar um nome a esta variável.

Observe que uma variável do tipo float (com vírgula decimal) é escrita com ponto, este é um programa americano onde a parte decimal é separada por um ponto.





O tipo da variável descreve o tipo de dado numérico a ser armazenado estes tipos são listados abaixo com os respectivos ranges de valores.

Uma vez declarado o tipo da variável você não pode usar números que não se enquadrem dentro do formato e range declarado, pois ao declarar uma variável o computador reserva a área de memória necessária para armazenar este dado, se o valor for maior do que a área reservada o compilador irá indicar uma falha.

Detalhes dos tipos e usos.

- Na maioria das vezes você vai usar o tipo int(inteiro), use para números com sinal, mas sem virgula (valor decimal). Cuidado que o valor máximo a ser escrito com tipo inteiro é +32767 e o menor é -32768!
- Para cálculos com números com vírgula use o tipo float.
- O char pode ser usado para variáveis binárias que guarda somente números 1 e 0, pois o tipo char é o menor de todos.

Tipo	Tamanho (em bits)	Intervalo
char	8	-128 a 127
int	16	-32768 a 32767
float	32	3,4E-38 a 3,4E+38
double	64	1,7E-308 a 1,7E+308
void	0	sem valor

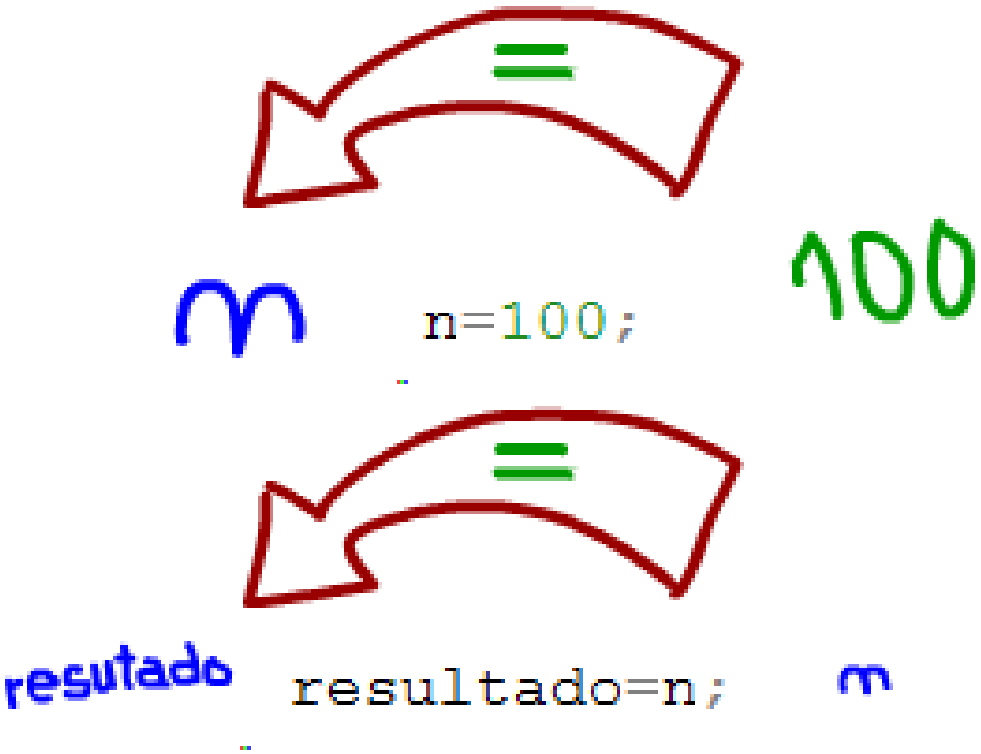
Como alterar o valor da variável.

Você altera o valor guardado na variável usando o operador de atribuição de igual “=”.

O nome da variável que vai receber o valor (destino) deverá ficar sempre a esquerda, será sempre a primeira a ser escrita na linha.

Você pode guardar um número direto.

Você pode transferir o valor de uma variável para outra variável.



Operações com variáveis:

Você pode fazer operações matemáticas com as variáveis, em cada operação somente o valor da variável a esquerda é alterado.
O microprocessador começa a execução da operação pela expressão escrita a direita do sinal de igual seguindo as regras recursivas da matemática.
As quatro operações básicas são permitidas, outras operações somente com inclusão de biblioteca especial.
As operações básicas são descritas abaixo:

Operador	Ação
+	soma
-	subtração
*	multiplicação
/	divisão
%	módulo (resto da divisão)

Exemplo:
Se da linha da instrução abaixo a variável n estava com o valor zero, ao final da instrução abaixo o valor da variável n será igual a "1"!

```
n= n+1;
```

Na sequência abaixo se n1=1 e n2=2, apenas final da sequência somente o n1 estará alterado para o valor 3!

```
n1=1;
n2=2;
n1=n1+n2;
```

Como escrever o nome de uma variável em linguagem “C”

O nome da variável é definido por seu identifica e deverá conter letras, números e o caractere sublinhado e não poderá conter caracteres especiais, as regras para definir uma variável é descrita abaixo.

O primeiro caractere do nome da variável deverá ser uma letra ou o caractere sublinhado.

No nome da variável não é permitido caractere em branco ou caractere especiais como @,#,$,%,&, ç e letras acentuadas, como por exemplo “é” etc.!

Não é permitido usar as palavras reservadas nos identificadores, ou seja, palavras que pertençam a linguagem de programação!

Exemplo identificadores usados para declarar o nome de uma variável:

A, A2, Nota, nota , nota_semestre, notaSemestre, _dia Idade!

Dicas:

Inicie nome de variáveis sempre com letra minúscula, exemplo: int nota;!

Se a variável é composta por dois nomes inicie o segundo com letra maiúscula, exemplo: int notaSemestre;!


```
int n;//variável inteiro de uso geral
```

Não esqueça o ponto e vírgula no final!

Toda a instrução em linguagem “C” deve terminar com o ponto e vírgula

```
int n;//variável inteiro de uso geral
```

O tipo void.

O toipo “void” é um tipo especial!

Você usa o “Tipo void” quando não souber o tipo, o computador reserva uma área especial para este tipo, e ajusta a área conforme a necessidade, este tipo de variável consome muita memória do computador, deve ser usada somente em último caso.

Então se você não souber o tipo a ser usado, não esquente a cabeça, use o tipo void, ela funciona como um coringa!

Como escrever letras.

Uma variável que guarda letras é chamada de string em linguagem “C”!

As strings são armazenadas no tipo char (caractere).

Uma frase é armazenada como uma sequência de caracteres por isto ao definir a variável para guardar uma frase você deverá definir o tamanho da frase.

O último caractere de uma frase é reservado pela linguagem para guardar o caractere de finalização, assim configure o tamanho com um caractere a mais!

Exemplo: char[17]; //string de 16 caracteres mais o caractere de finalização

Uma string deve ser sempre escrita entre os sinais de duas aspas!

A linguagem “C” não consegue trabalhar com letras como da mesma forma que trabalha com números, por exemplo, não é possível transferir uma frase usando o sinal de atribuição igual, não é possível compara frases dentro da função if!

Mas não se preocupe, quando for preciso usar letras você vai baixar funções prontas que codificam letras como números usando uma tabela de conversão chamada ASCII que é reconhecida por equipamento que trabalha com letras, mas terá que incluir estes arquivos no seu projeto!

Como funciona a função “if”:

A instrução “if” é a chave para o programa tomar decisões, esta instrução faz parecer que o programa pensa.

A instrução “if” faz uma pergunta “Se” para o valor contido no parênteses, se este valor for diferente de zero o programa executa a sequência de instrução entre as chaves depois da pergunta, senão (o valor do parênteses for igual a zero) o programa executa a sequência dentro das chaves depois do palavra “”else”!

```
int n=digitalRead(chave12);//lê o valor da entrada digital

if (n)
```


```
digitalWrite( led13, HIGH);// turn the LED on (HIGH is the voltage level)
delay(1000);               // wait for a second
digitalWrite( led13, HIGH);// turn the LED off by making the voltage LOW
delay(1000);               // wait for a second

}
else{
```


```
digitalWrite( led13 , LOW);// turn the LED off by making the voltage LOW

}
```

Como escrever um comentário:

Um comentário não é executado pelo seu programa, ele não é transformado em linguagem de máquina, serve para organizar o seu projeto informado a outras pessoas o porquê do seu código e serve para você mesmo relembrar mais tarde o que você fez!

A forma de escrever o comentário é livre, você pode abreviar ou escrever como num "chat" o importante é a informação!

Comentar um projeto é de fundamental importância, você deve se acostumar a inserir o máximo de comentários possíveis.

As linhas em amarelo abaixo são comentários!

```
// the loop function runs over and over again forever
void loop() {
  int n=digitalRead(chave12);//lê o valor da entrada digital
  if (n){
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);              // wait for a second
  }
   else{
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  }
}
```

Um comentário pode ser de uma linha ou várias linhas.

O comentário de uma linha é todo o texto incluído depois de duas barras invertidas, como no exemplo abaixo, exemplo "//comentário! “

Note que não precisa terminar o comentário de uma linha com mais duas barras invertidas, todo o texto depois das barras é considerado comentário.

Esta é a forma mais comum de comentar um programa simples!

O comentário de mais de uma linha é todo o texto incluído entre os símbolos barra invertida asterisco (/*) e asterisco barra invertida (*/).

```
/*
Este é um comentário
de mais de uma linha
você pode colocar quantas linhas quiser!
*/

// este é um comentário de uma linha
//tem que iniciar com as duas barras
//e não precisa terminar com duas barras
//tudo que está escrito depois das duas barras é comentário
```

Diga fundamental:

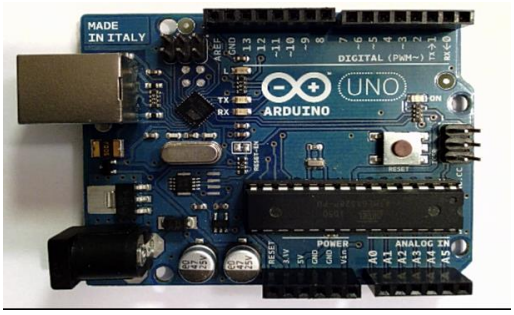
Sempre comente os seus programas!

Hardware do UNO:

- Este capítulo você vai abordar os seguintes tópicos:
- Conhecer o hardware da placa Arduino UNO.
- Conhecer e identificar os conectores de sinais digitais, sinais analógicos, sinais do tipo PWM e potência!

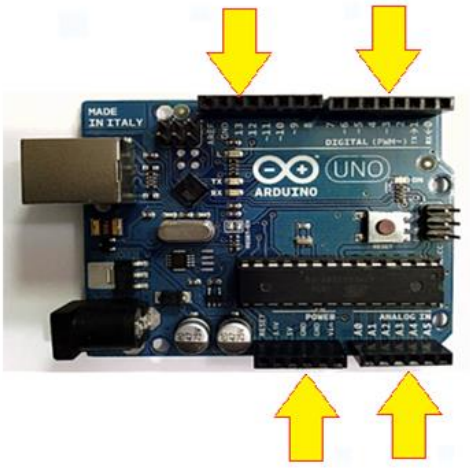
Hardware da placa UNO.

Esta é a placa do Arduino UNO

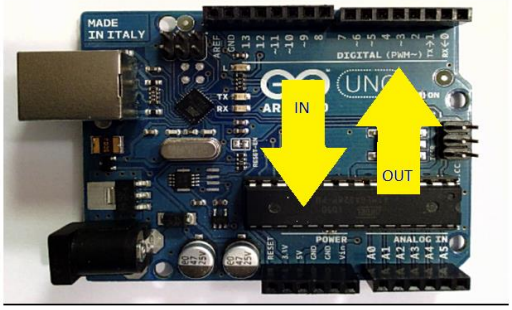


Os conectores são usados para ligar os sinais digitais ou analógicos do mundo externo ao microcontrolador!

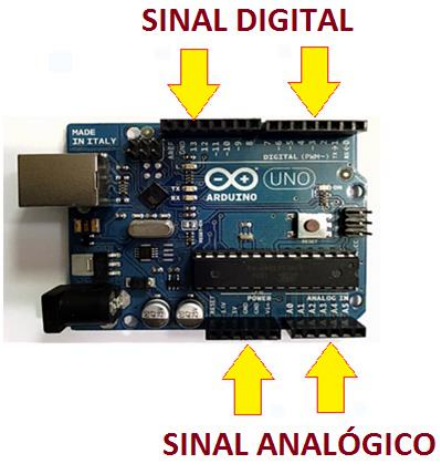
Os pinos para os sinais de entrada e saídas são chamados de portas.



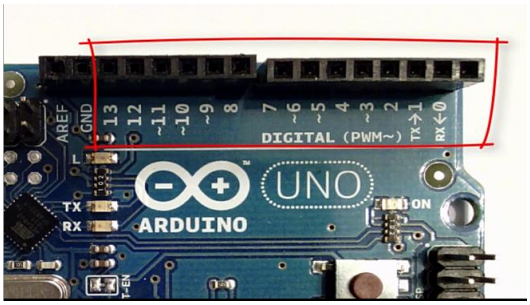
Através dos conectores você tem acesso aos pinos do microcontrolador!



As portas poderão ser do tipo digitais ou analógicos!



Os pinos dos conectores de sinais digitais são numerados de “0” a “13”.



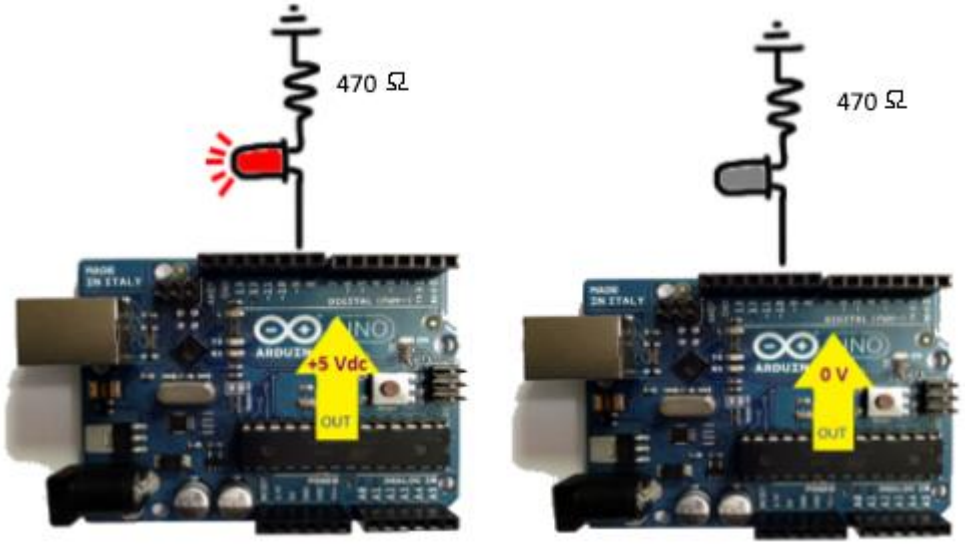
As portas podem ser de entradas ou saídas!

Você configura no programa os pinos que serão usados como portas de entradas ou saídas!

Portas digitais.

Quando uma saída digital é ligada a placa do Arduino coloca 5 Vdc no respectivo pino com uma corrente máxima de 10mA!

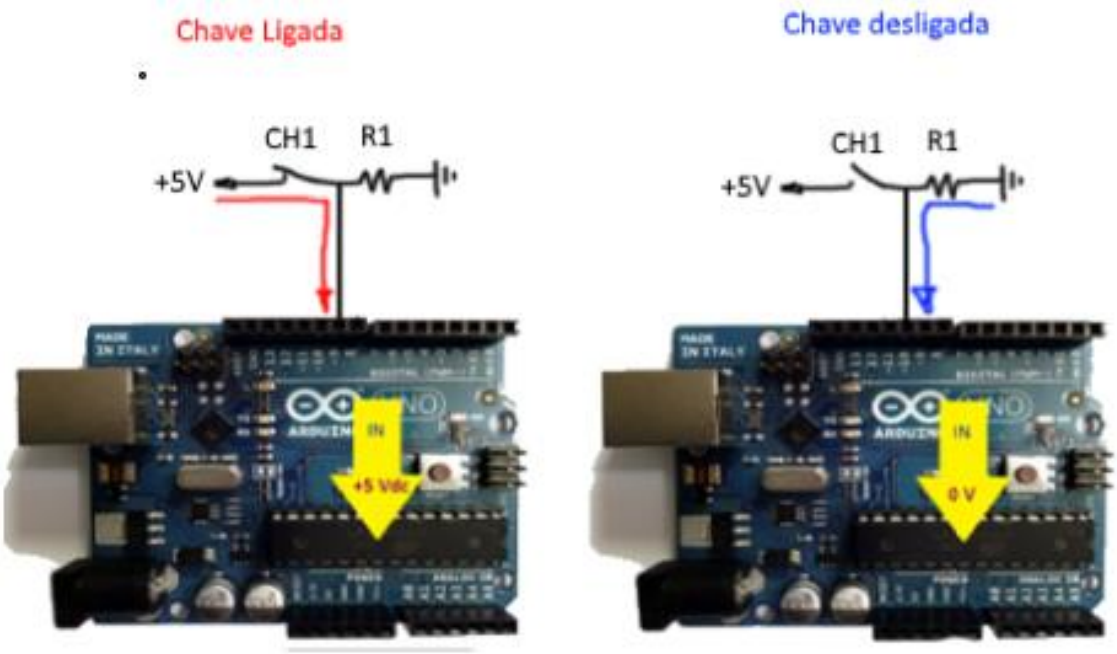
Quando uma saída digital é desligada a placa do Arduino coloca 0 V no respectivo pino!



Para ligar uma entrada digital o circuito externo deve ligar 5 Vdc na placa do Arduino!

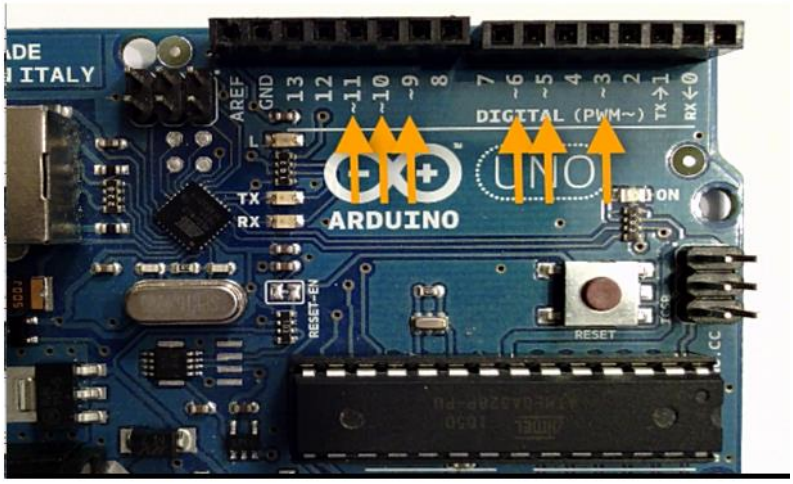
Para desligar uma entrada digital o circuito externo deve ligar 0 V na placa do Arduino!!

O circuito para uma entrada digital acionada por chave deve ter um resistor de pull-down se a chave é ligada ao positivo!



Portas PWM.

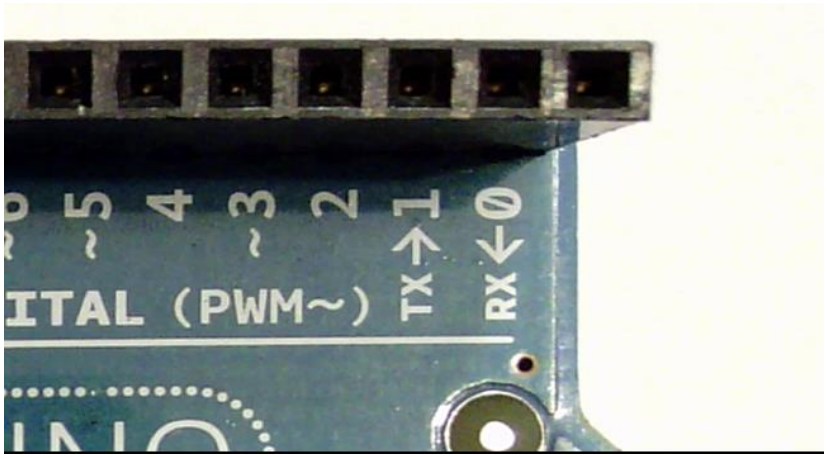
A placa Arduino UNO possui saídas especiais, como por exemplo, os pinos de saída do tipo PWM para controle de velocidade de motor, circuitos de potência etc.!



Portas de comunicação!

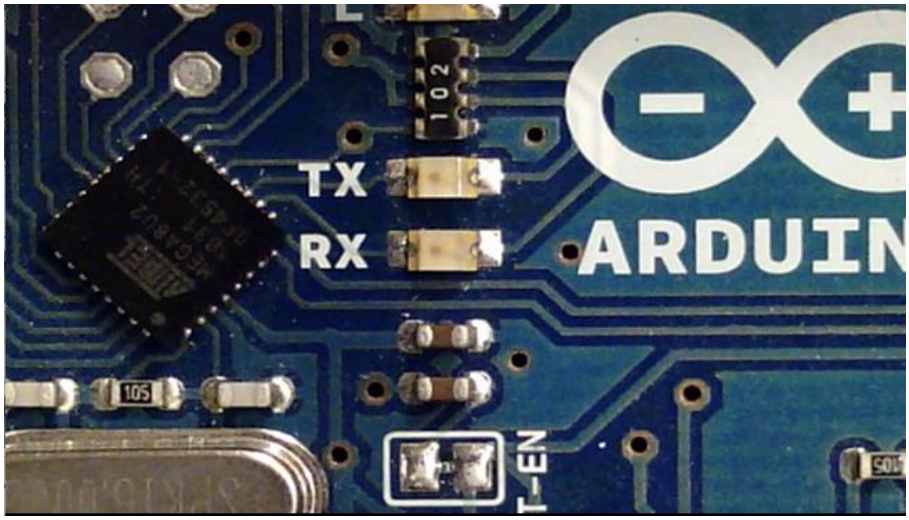
Estes pinos são usados para comunicação com outras placas Arduino, PC e outros equipamentos que usem o padrão de comunicação serial!

- TX significa transmitindo dados!
- RX significa recebendo dados!



A placa possui LEDs de status da comunicação junto ao CI de comunicação!

Durante uma comunicação os LEDs piscam!



Portas analógicas:

A placa possui um conector para ligar sinais analógicos!

Existem 6 entradas analógicas numeradas de A0 a A5.

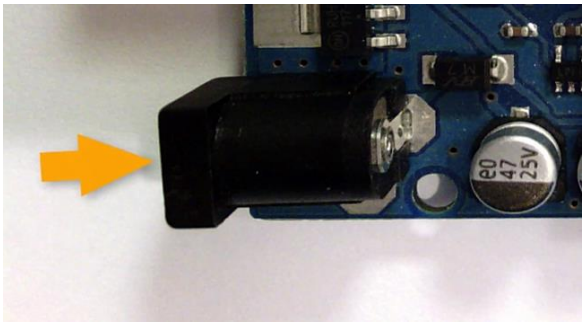
O nível do sinal analógico é de 0 a 5Vdc.

O sinal analógico é convertido em digital através de ADC interno!

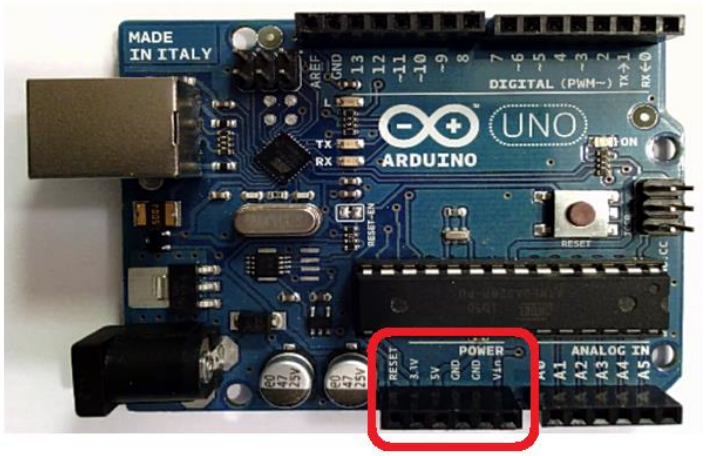
Pinos de alimentação.

A placa pode ser alimentada diretamente pela USB do PC, esta é a forma usada durante a programação e testes.

A placa pode ser alimentada por um conector auxiliar com tensões que podem variar de 9VDC a 15VDC fornecida por uma fonte externa, normalmente uma fonte de celular!

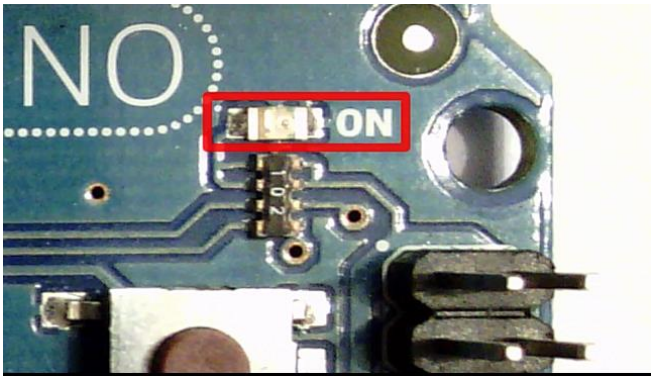


A placa também possui um conector de força que dá acesso ao circuito de alimentação interno da placa, podendo ser usado para alimentar os circuitos de entrada e saída externo



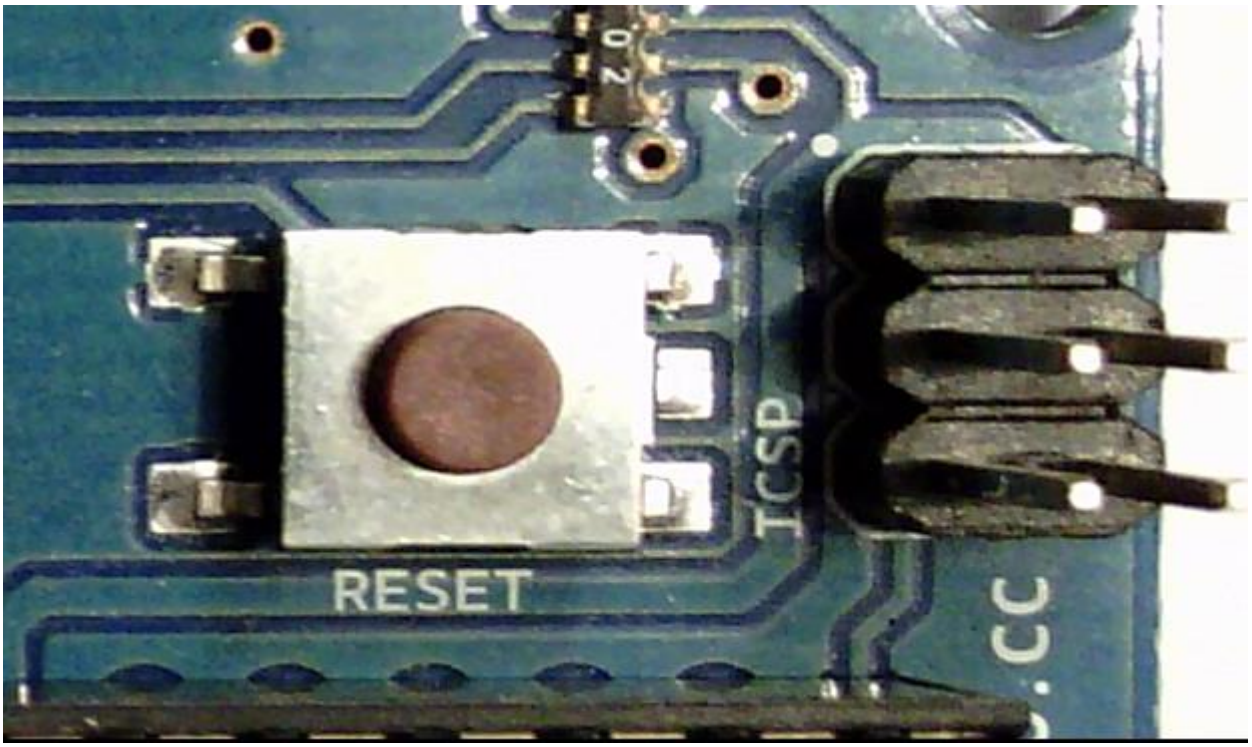
!

A placa possui um LED de indicação da placa energizada!



Botão de RESET.

A placa possui um Botão de RESET que força o programa a reiniciar do zero!



Desafio 2:

DESAFIO 2

Nome:

1)Coloque um “C” ao lado da linha que declara corretamente uma variável?

int salario do mês 10;

float salarioMes10;

int 10salarioMes;

float salario_mes10

int salario_mes10;

2)Na sequência de instruções abaixo qual o valor da variável nota1, nota2 e media ao final da sequência?

```
nota1=60;
nota2=80;
media=(nota1+nota2)/2;
nota1=nota1+nota2;
```

3)Na instrução if abaixo se a chave12 estiver ligada o LED estará aceso ou apagado?

```
void loop() {
  int n=digitalRead(chave12);//lê o valor da entrada digital
  if (n){
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  }
   else{
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  }
 }
```

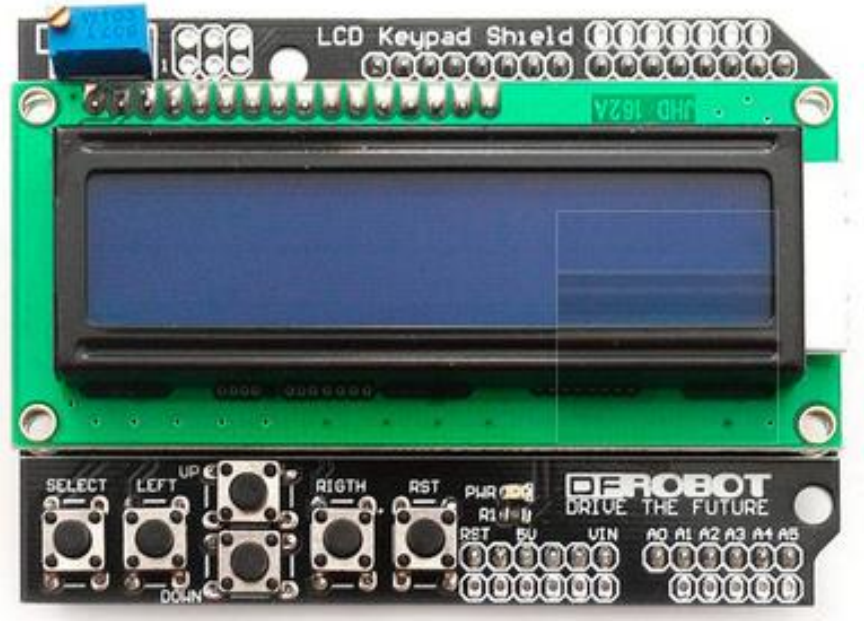
4)Qual a tensão máxima que você pode aplicar a entrada digital?

5)Qual a máxima corrente na saída digital?

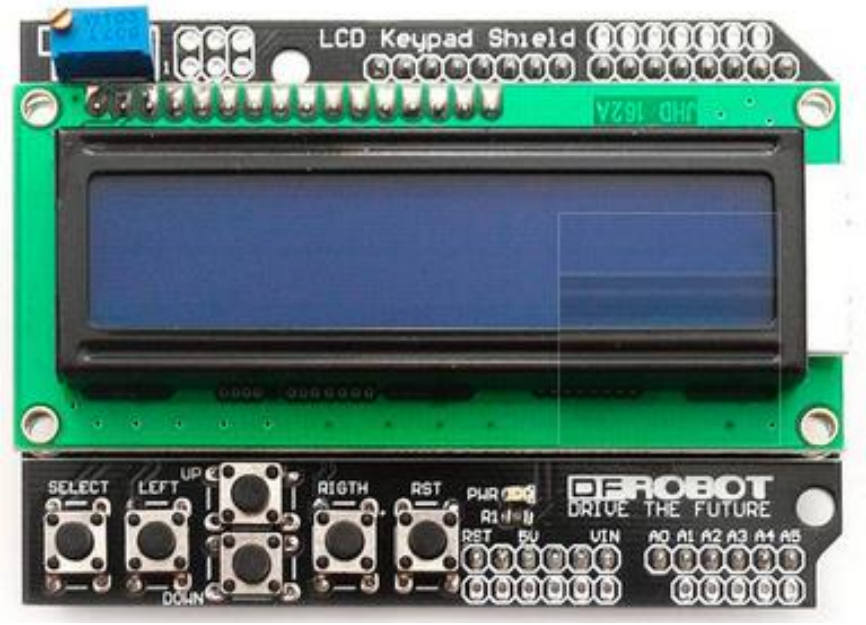
6)A conexão USB com PC serve também para alimentar a placa UNO e o circuito externo?

Display LCD:

O Arduino pode ser montado com vários circuitos prontos chamados de shield, neste tutorial você verá como criar um programa para acionar o LCD do shield LCD.



Este shield conecta diretamente um LCD ao Arduino de forma fácil e rápida, não sendo necessário solda ou protoboard, deixando o seu projeto mais compacto e ergonômico.
O Shield é compatível com a biblioteca LCD4Bit que pode ser encontrada no site oficial da Arduino juntamente com exemplos passo-a-passo. Não é possível vender este Shield sem o LCD incluso pois ambos vem lacrados.



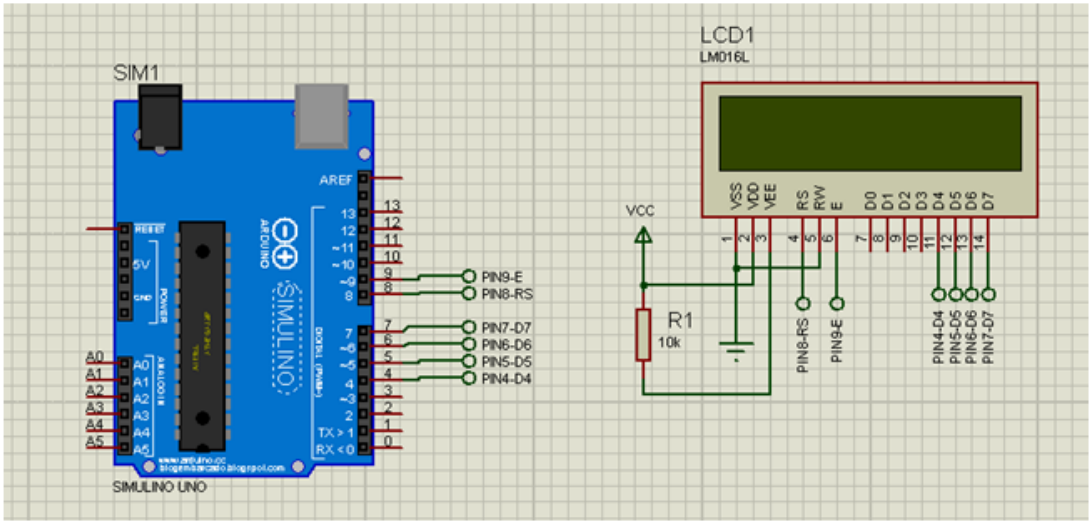
Conexão do shield LCD:

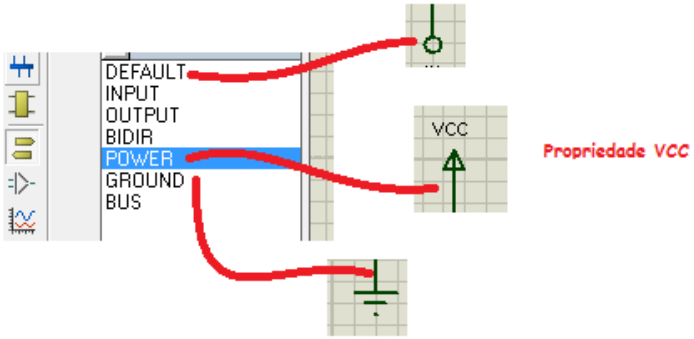
O diagrama mostra a interconexão do LCD com os pinos do Arduino, no Arduino original esta conexão é feita automaticamente ao inserir o shield no Proteus ISIS você deverá montar.

Pin	Function
Analog 0	Button (select, up, right, down and left)
Digital 4	DB4
Digital 5	DB5
Digital 6	DB6
Digital 7	DB7
Digital 8	RS (Data or Signal Display Selection)
Digital 9	Enable
Digital 10	Backlit Control

Montando o circuito no Proteus ISIS:

O circuito no Proteus Isis é mostrado abaixo, note que o LDC é de 16 linhas 2 colunas (16x2) e os pinos são ligados conforme descrito na conexão do shield.
Para pegar o LDC digite lcd no campo de pesquisa, categoria Optoelectronic e subcategoria Alphanumeric LCDs na lista selecione o LM016.
Use conectores para simplificar o diagrama.





Quando ao software do Arduino:

O Shield é compatível com a biblioteca LCD4Bit que pode ser encontrada no site oficial da Arduino juntamente com exemplos passo-a-passo. Todas as funções já estão prontinhas nesta biblioteca.
Todas as funções para o acionamento do display podem ser visualizadas no link abaixo:
https://www.arduino.cc/en/Reference/LiquidCrystal
Note que neste shield o LDC é ligado usando somente 4 pinos de dados e o sinal RW (Ler escrever) está aterrado sendo usado somente para escrever dados no LCD, com isto o circuito economiza pinos do Arduino.
Para que estas funções funcione você deve acrescentar a diretiva abaixo no início do programa.

```
#include <LiquidCrystal.h>
```

Descrição das principais funções:

As principais funções são:

```
LiquidCrystal(rs, enable, d4, d5, d6, d7) :
lcd.begin(cols, rows);
lcd.setCursor(col, row);
lcd.print(data);
lcd.print(data, BASE);
```

Função LiquidCrystal():

LiquidCrystal(rs, enable, d4, d5, d6, d7) : Inicializa o LCD e você deverá colocar nos campos do s parâmetros o endereço dos pinos usados com as funções descritas no caso do uno a programação correta é descrita abaixo:
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

Função lcd.begin():

lcd.begin(cols, rows)
Inicializa o LCD.
No campo cols você deve colocar o número de colunas do LCD e no campo rows você deve colocar o número de linhas.
No caso do shield o número de colunas é 16 e o número de linhas é 2!
lcd.begin(16, 2);

Função lcd.setCursor():

lcd.setCursor(col, row):
Posiciona o cursor na linha e coluna apropriado.
Observar que a primeira coluna e primeira linha possuem o índice zero.
Exemplo:
lcd.setCursor(0, 1); //coloca o cursor na coluna 0 linha 1
lcd.setCursor(0, 0); //coloca o cursor na coluna 0 linha 0 Home!

Função lcd.clear() :

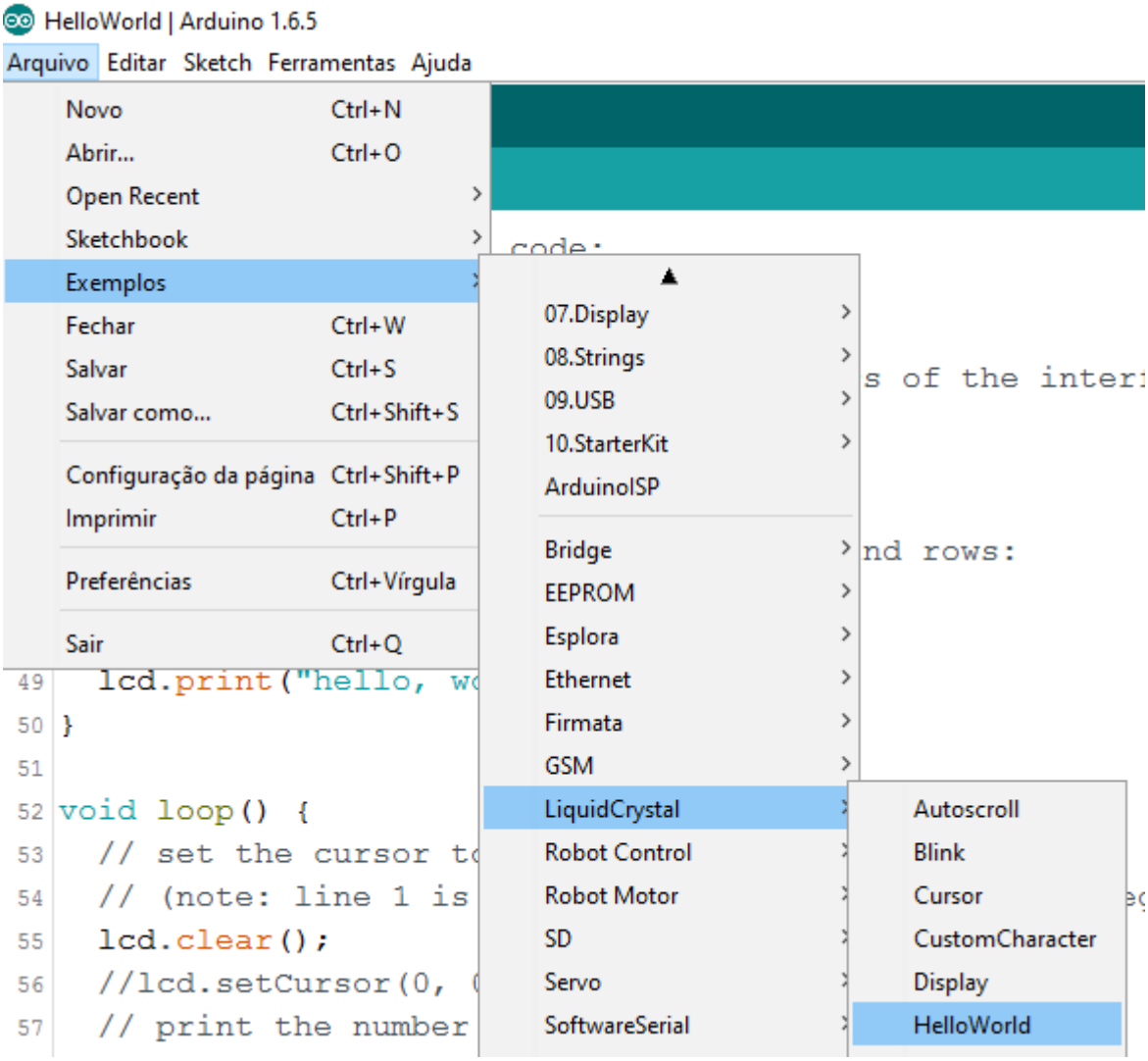
Limpa o display e leva o cursor para a posição inicial coluna zero e linha zero.

Função lcd.print():

lcd.print(data) ou lcd.print(data, BASE):
Esta função escreve uma linha no LCD, para escrever duas linhas você deve antes posicionar o cursor.
Você pode escrever somente uma dado (data) de cada vez, este dado poderá ser string (frases) que deverá ser escrita entre aspas, veja o exemplo abaixo.
lcd.print("hello, world!");
Para escrever números escreva a variável ou o número sem as aspas, como no exemplo abaixo.
lcd.print(millis() / 1000);
Para escrever números e letras você terá que escrever duas vezes:
lcd.print("Numero= ");
lcd.print(100);
O printf do Arduino é muito simples e você deverá usar com cuidado, é aconselhável usar a função lcd.clear() para limpara o display e posicionar o cursor no inicio (home), neste caso pode ocorrer um piscar no brilho do display, para tirar esta falha inclua um delay(100) depois da escrtira no display!

O programa exemplo “Hello World!”do Arduino:

O programa exemplo que escreve no display você pode baixar dos exemplos conforme é descrito abaixo.



Este programa é mostrado abaixo sem o cabeçalho:

HelloWorld

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Descrição do programa:

A linha abaixo inclui as bibliotecas do LCD.

```
// include the library code:
#include <LiquidCrystal.h>
```

A linha abaixo inicializa o LCD configurando os pinos, no exemplo da Arduino os pinos estão ERRADOS, corrija para a configuração abaixo para os pinos usados no shield do LCD Arduino com chaves!

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

Na linha abaixo escrita no setup é declarado um LCD de 16 linhas e 2 colunas e ainda é escrito a frase “Hello World” na primeira linha, esta é a forma de escrever letras no display.

```
void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}
```

No loop são escritas as linhas abaixo onde o cursor é ajustado para a segunda linha e em seguida o programa escreve no display uma função interna do Arduino chamada millis() que volta o tempo em milissegundos que o programa está ligado, esta é uma forma de escrever um número do display.

```
void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Exporte o código binário compilado (Sketch/ Export compiled Binary), carregue no Arduino do simulador e teste!

Desafio:

Altere a frase “Hello World” colocando o seu nome?

Montando um temporizador com o Arduino:

O programa consiste em gerar um contador de segundos de zero a 59, quando a contagem de segundos chegar a 60 deverá ser zerada!

Neste exemplo você verá como mostrar mais de uma informação no display.

Crie uma pasta par ao seu projeto, copie o diagrama do ISIS do projeto “Hello Wold” e cole na pasta do seu projeto ou monte um novo diagrama com o LCD conforme descrito no exemplo “Hell World”!
Abra o programa do Arduino, adicione o exemplo “Hello World!” e salve como na pasta que você criou para o seu projeto alterando o nome para temporizador!
Altere o programa “Hello world” conforme descrito abaixo.

Altere os valores da função LiquidCristal e insira uma variável do tipo inteiro com o nome segundos, conforme descrito abaixo..

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
int segundos;
```

Apague a mensagem inicial do setup que deverá ficar como descrito abaixo.

```
46 void setup() {
47   // set up the LCD's number of columns and rows:
48   lcd.begin(16, 2);
49   }
```

Apague tudo que estiver dentro do loop(), escreva as instruções abaixo dentro do loop()!

```
50 void loop() {
51     delay(1000);//gera o tempo de 1 segundo
52     segundos++;//increment ao segundo de 1
53     if (segundos==60){
54       segundos=0;//zera o segundo
55     }
56     //mostra
57     //Limpa o display
58     lcd.clear();
59     //Escreve Temporizador na coluna 0
60     lcd.print("Temporizador");
61     // (note: (coluna, Linha) primeira coluna=0, primeira linha=0:
62     //posiciona o cursor na linha 1 coluna 0.
63     lcd.setCursor(0, 1);
64     lcd.print(segundos);
65 }
```

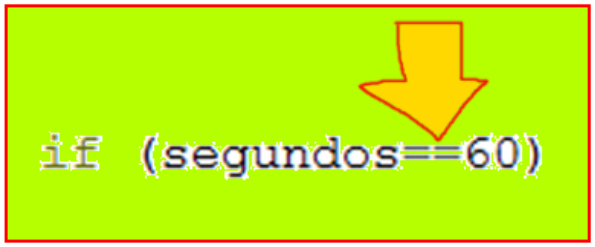
A primeira linha gera um tempo de atraso de um segundo que será a base de tempo do temporizador.

```
delay(1000);//gera o tempo de 1 segundo
```

A linha abaixo incremente o valor do segundo de um equivale a segundo=segundo+1!

```
52      segundos++;//increment ao segundo de 1
```

A linha abaixo pergunta se segundo é igual a 60, na linguagem “C” a comparação igual é descrita com dois sinais de iguais para diferenciar do igual que atribui um valor na variável.



Se a resposta for “sim” zera o valor do segundo, note que esta pergunta não tem o else e como este ajuste é feito antes de mostrar o número 60 nunca aparece!

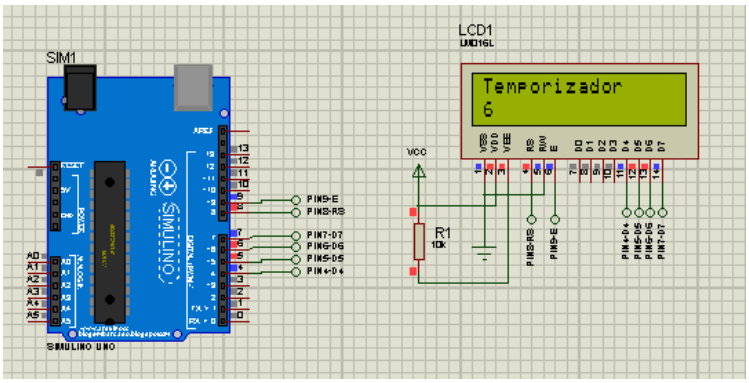
```
53      if (segundos==60){
54        segundos=0;//zera o segundo
55      }
```

As linhas abaixo mostra o valor; primeiro limpando a tela do display e posiciona o curso na coluna zero e linha zero com a instrução lcd.clear(), isto se faz necessário porque a instrução lcd.print não limpa o display e traços da mensagem anterior podem continuar sendo mostrados; segundo escreve o título “Temporizador”; terceiro posiciona o cursor na próxima linha coluna zero; e por último escreve o segundo!

Observe que você tem que escrever textos e números (variáveis numéricas) em linha diferentes, a função lcd.print do Arduino é bem simples e ajustar a aparência do display pode dar algum trabalho.

```
56      //mostra
57      //Limpa o display
58      lcd.clear();
59      //Escreve Temporizador na coluna 0
60      lcd.print("Temporizador");
61      // (note: (coluna, Linha) primeira coluna=0, primeira linha=0:
62      //posiciona o cursor na linha 1 coluna 0.
63      lcd.setCursor(0, 1);
64      lcd.print(segundos);
65  }
```

Compile, gere o código binário, note que se você salvou na pasta do seu projeto o programa não pede o local para salvar, ele salvará na pasta do seu projeto, carregue no Arduino do simulador e teste, o resultado é mostrado abaixo!



Como escrever um número float no LCD:

A função lcd.print() ajusta o número conforme o tipo de variável, e números float a função mostra com duas casas depois da virgula.
A matemática do programa do Arduino trabalha com números inteiros para operar com números do tipo float (com vírgula) você deverá escrever (float) no início da operação, no programa exemplo abaixo, você irá criar uma variável "float n" e irá executar o cálculo n=10/3, e então, irá mostrar o n. Observe o ajuste da variável float no cálculo, se não ajustar não vai aparecer a parte decimal da divisão.

```
40 // include the library code:
41 #include <LiquidCrystal.h>
42
43 // initialize the library with the numbers of the interface pins
44 LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
45 float n;
46 void setup() {
47   // set up the LCD's number of columns and rows:
48   lcd.begin(16, 2);
49   // Print a message to the LCD.
50   lcd.print("hello, world!");
51 }
52 void loop() {
53   n=(float)10/3;
54   // set the cursor to column 0, line 1
55   // (note: line 1 is the second row, since counting begins with 0):
56   lcd.setCursor(0, 1);
57   // print the number of seconds since reset:
58     lcd.print(n);
59 }
```

Desafio 3:

DESAFIO 3:

Crie um programa em que escreva na primeira linha “Tempo” e na segunda linha fique contando os segundos de forma contínua até 200, quando a contagem chegar a 200 deverá ser zera iniciando a contagem novamente.

Como usar a comunicação serial no Arduino:

O Arduino possui dois pinos para a comunicação serial: TX (Transmite dado) e RX (Recebe dado).

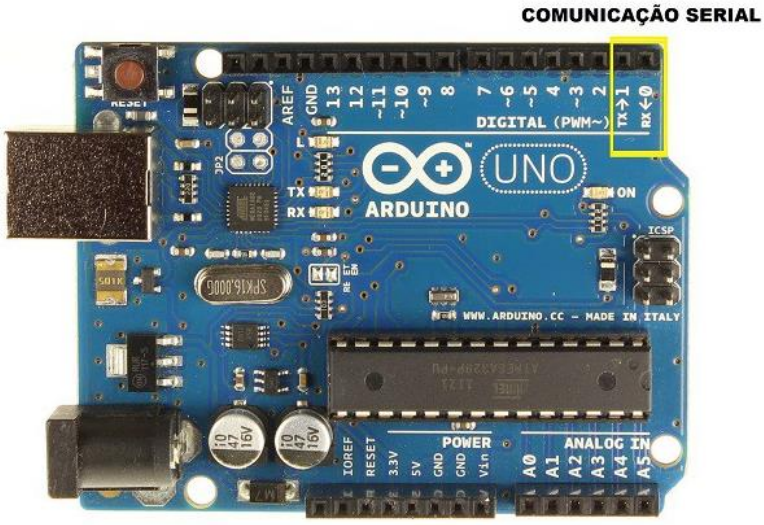


Figura 1 - Pinos para comunicação serial

Você deve ficar atento a dispositivos conectados a esses pinos pois podem interferir no upload do seu programa. Em alguns casos é recomendável desconectar os dispositivos ou shields ligados a esses pinos antes de fazer o upload.

COMUNICAÇÃO SERIAL

Figura 1 - Pinos para comunicação serial

O sinal de comunicação na placa Arduino UNO, é um sinal TTL de 5V. Para comunicação com um computador ou outro dispositivo que não tenha o mesmo nível de tensão é necessário um conversor de nível. Existem várias opções de conversores, como por exemplo TTL/ RS232, TTL/RS485, TTL/USB, entre outros. A seguir são exibidos alguns modelos de conversores.

Figura 2 - Shields e módulos para comunicação serial

Terminal Serial:

Além do recurso de upload através da comunicação serial, a IDE trás um terminal serial que auxilia no recebimento e envio de dados para a placa sem a necessidade de recorrer a uma ferramenta externa. Para acessar essa ferramenta basta clicar no ícone Serial Monitor ou acessar o menu Tools> Serial Monitor. É aberta a janela a seguir.
Este recurso só funciona com um módulo real, na simulação usaremos um simulador de terminal para esta função.

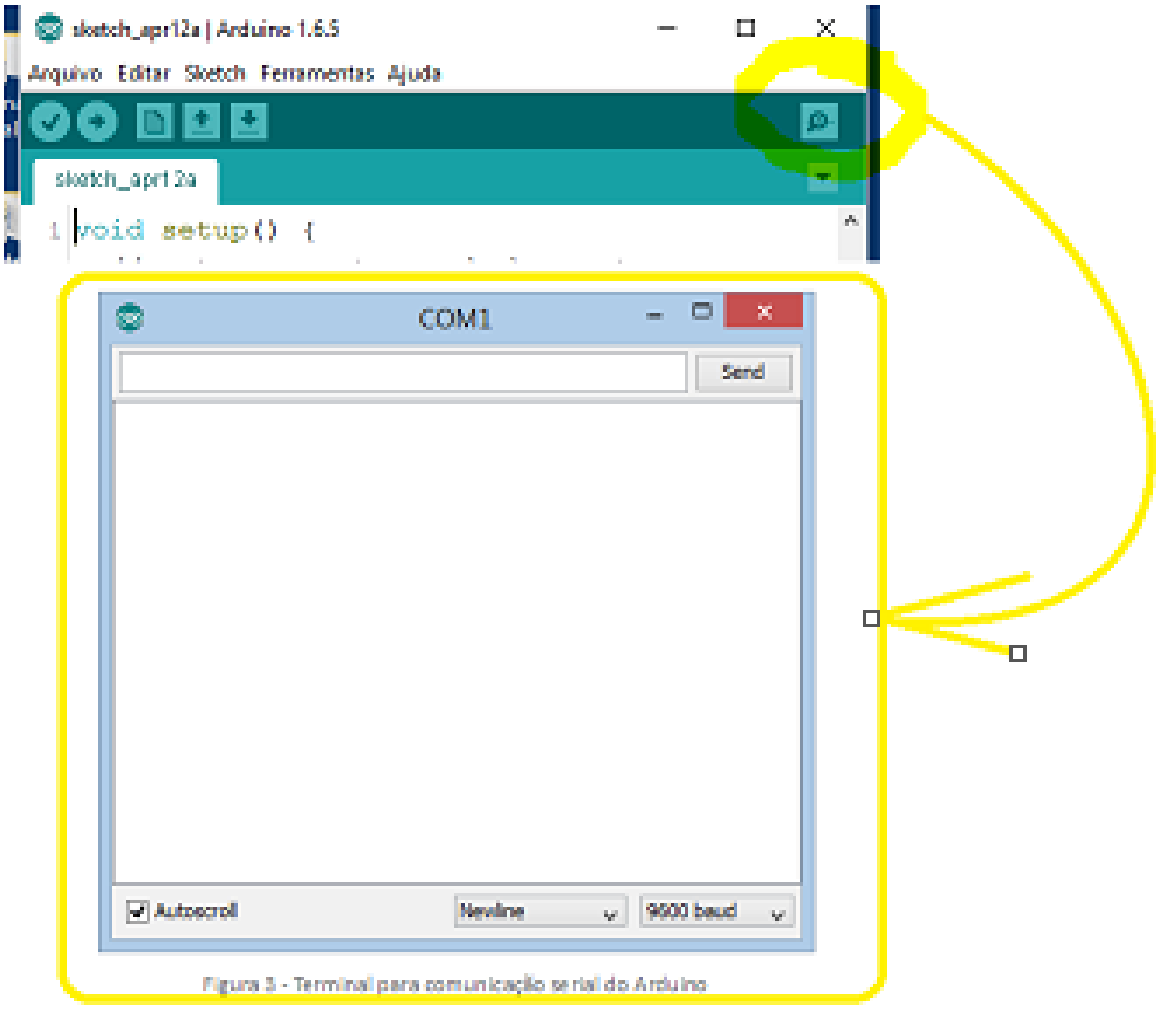
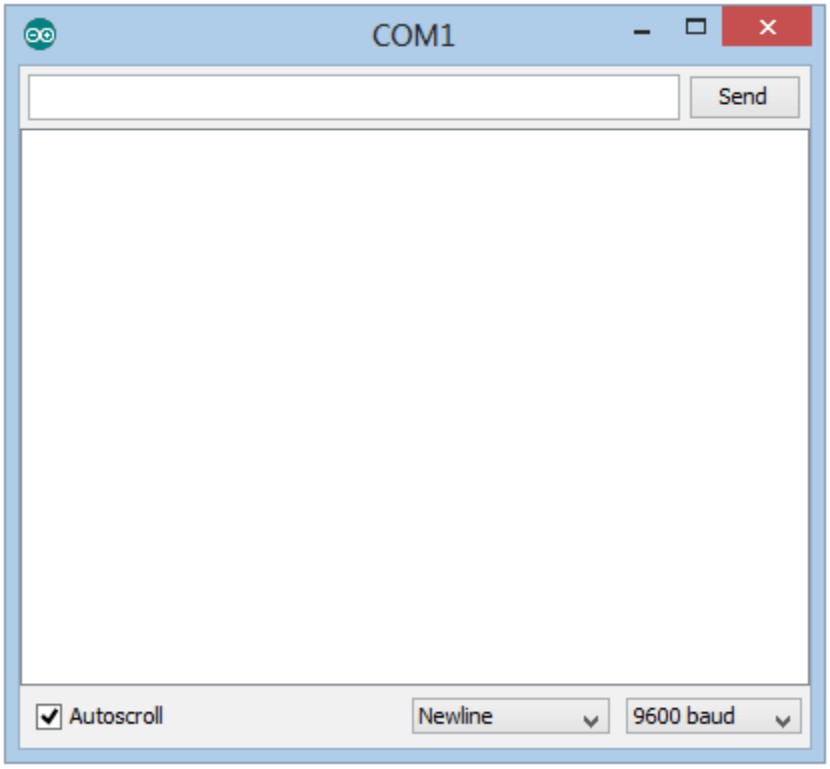


Figura 3 - Terminal para comunicação serial do Arduino

A ferramenta é bem simples, contendo apenas alguns parâmetros de configuração, onde se pode definir a taxa de envio (baud rate). Possui dois campos, um onde pode ser inserido a mensagem a ser enviada e outro maior onde é exibido os caracteres enviados pela placa para o computador.



Funções usadas na comunicação serial:

```
Serial.begin();
Serial.available();
Serial.read();
Serial.readBytes(buffer, length)
Serial.print ();
Serial.write();
```

Serial.begin()

É a primeira função a ser utilizada quando vai trabalhar com a comunicação serial. Ela configura a taxa de comunicação em bits por segundo (baud rate). Possui um segundo parâmetro opcional para a definição da quantidade de bits, paridade e stop bits. Se for omitido esse parâmetro o padrão será 8 bits, sem paridade e 1 stop bit.

Sintaxe:

Serial.begin(speed)
Serial.begin(speed, config)

Parâmetros:

speed: velocidade em bit por segundo (baud rate) - long
config: configura a quantidade de bits, paridade e stop bits. Os valores válidos são :
SERIAL_5N1
SERIAL_6N1
SERIAL_7N1
SERIAL_8N1 (padrão)
SERIAL_5N2
SERIAL_6N2
SERIAL_7N2
SERIAL_8N2
SERIAL_5E1
SERIAL_6E1
SERIAL_7E1
SERIAL_8E1
SERIAL_5E2
SERIAL_6E2
SERIAL_7E2
SERIAL_8E2
SERIAL_5O1
SERIAL_6O1
SERIAL_7O1
SERIAL_8O1
SERIAL_5O2
SERIAL_6O2
SERIAL_7O2
SERIAL_8O2

Serial.available()

Retorna a quantidades de bytes disponíveis para leitura no buffer de leitura. Essa função auxilia em loops onde a leitura dos dados só e realizada quando há dados disponível. A quantidade máxima de bytes no buffer é 64.

Sintaxe:
Serial.available();

Serial.read()

Lê o byte mais recente apontado no buffer de entrada da serial.

Sintaxe:
Serial.read();

Serial.readBytes(buffer, length)

Esta função lê um determinado número de caracteres configurado no parâmetro “length”!
A recepção termina quando o número de caracteres for alcançado ou quando o tempo máximo de comunicação for atingido, este tempo pode ser configurado na função Serial.setTimeout()!
Esta função retorna o número de caracteres recebidos ou zero se nenhum dado foi encontrado.

Syntax:

Serial.readBytes(buffer, length)

Parâmetros:

buffer: O buffer onde será armazenado os bytes recebido pode ser do tipo char[] ou byte[].
length: Número de bytes a ser recebido.

Serial.readBytesUntil()

Esta função lê uma sequência de caracteres até que o caractere de terminação seja encontrado ou o comprimento máximo tenha sido alcançado ou o tempo de comunicação tenha se esgotado (Serial.setTimeout())!

Esta função retorna o número de caracteres recebidos ou zero se nenhum dado foi encontrado.

Esta é uma função prática para receber dados na forma de texto.

Os terminadores mais usados são:
CR: Carr Return = 13 (D em hexa)
LF: Line Feed = 10 (A em hexa)

Algumas vezes são usados os dois LF CR, mas a maioria das vezes é usado somente o CR, no terminal do ISIS quando você pressiona o ENTER é gerado o CR (13)!

Você não deve esquecer de declarar a variável buffer do tipo char no início do programa como um array como mostra o exemplo abaixo onde o nome do buffer foi declarado com o nome sbuffer e tamanho 30! O tamanho maior do que o tamanho do texto a ser recebido.
Char sbuffer[30]

Sintaxe:

Serial.readBytesUntil(character, buffer, length)

Parâmetros:

character: caractere de terminação.

buffer: Buffer onde os caracteres serão armazenados char[] ou byte[].

Lenght: Número máximo de bytes a ser lido, comprimento máximo da frase a ser lida!

Serial.print()

Sintaxe:
Serial.print(val)
Serial.print(val, format)

Escreve na serial texto em formato ASCII descrito no campo val, se este campo for um número o programa converte para texto antes de enviar!
Essa função tem muitas possibilidades. Números inteiros são escritos usando um caractere ASCII para cada dígito. O mesmo ocorre para números flutuante e, por padrão, são escritos com duas casas decimais. Bytes são enviados como caracteres únicos e strings e caracteres são enviados como texto.

Vejamos alguns exemplos:

```
Serial.print ( “123” ); // Envia "123"
Serial.print ( 1.234567 ); // Envia "1.23"
Serial.print ( 'N' ); // Envia "N".
Serial.print ( "Hello world" ); // Envia "Hello world".
```

Obs.: caracteres são enviados com aspas simples e strings com aspas duplas.

Um segundo parâmetro (format) é opcional define a base numérica para formatar o valor enviado. São aceitos os seguintes parâmetros:

- BIN - binário, base 2
- OCT - octal, base 8
- HEX - hexadecimal, base 16
- DEC - decimal, base 10

Para números em ponto flutuante esse parâmetro define a quantidade de casas decimais a serem enviadas após o ponto. Exemplos:

- Serial.print(78, BIN) envia em binário "1001110"
- Serial.print(78, OCT) envia emr octal "116"
- Serial.print(78, DEC) envia em decimal "78"
- Serial.print(78, HEX) envia em hexadecimal "4E"
- Serial.println(1.23456, 0) envia apenas "1", sem casas decimais
- Serial.println(1.23456, 2)envia "1.23", ou seja, duas casas decimais
- Serial.println(1.23456, 4) envia "1.2346", ou seja, 4 casas decimais

Serial.println()

Funciona praticamente igual a função Serial.print(), a única diferença é que esta função acrescenta ao fim da mensagem o caractere de retorno de carro (ASCII 13 ou '\r') e o caractere de nova linha(ASCII 10 ou '\n'). A sintaxe, os parâmetros e o retorno são os mesmos da função Serial.print().

Você vai usar esta função se o equipamento que está recebendo a mensagem exigir o uso dos dois caracteres.

Serial.write()

Escreve um byte na porta serial.

Sintaxe:
Serial.write(val)
Serial.write(str)
Serial.write(buf, len)

Parâmetros:
val: um valor para ser enviado como um único byte.
str: uma string (texto para ser enviada como uma sequência de bytes.
buf: um array para ser enviado como uma serie de bytes.
len: o tamanho do buffer a ser enviado.

Linhas de configuração da porta:

Você deverá sempre escrever as linhas de configuração da porta serial descritas abaixo.
A linha descrita abaixo descreve o Baud Rate da porta que normalmente é 9600 e espera a porta ser conectada. Estas linhas devem ser escritas dentro do setup()!

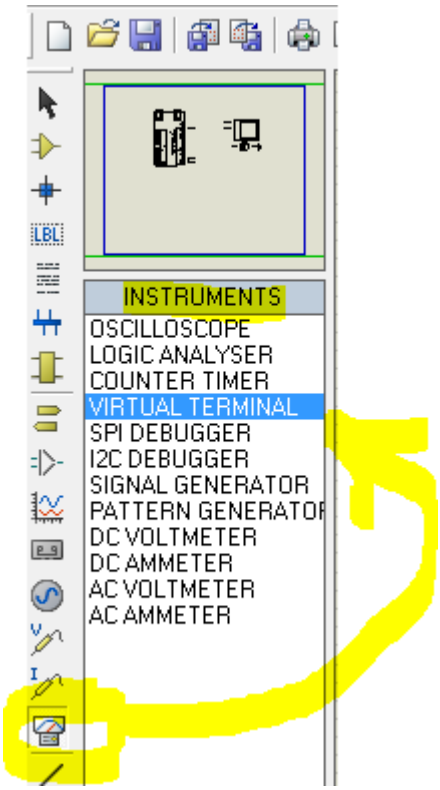
```
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
```

Programa exemplo “Echo”:

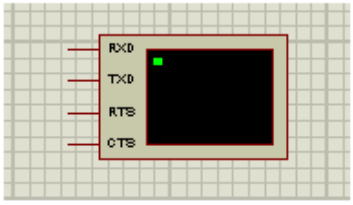
Crie uma pasta para o seu projeto ECHO!

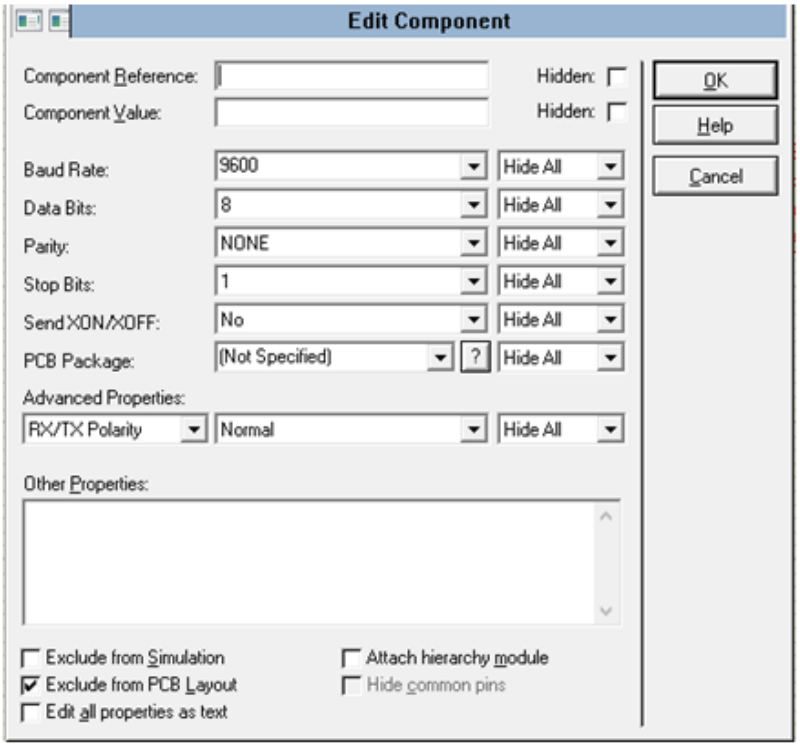
Este é um programa padrão para teste de uma comunicação serial, consiste em um terminal conectado aos pinos da comunicação serial do Arduino, este terminal envia uma mensagem para o Arduino que lê esta mensagem e devolve (echo), se o terminal receber a mensagem de volta a comunicação com o Arduino está perfeita.

Para este exemplo você deverá montar o circuito com o Arduino UNO e um VIRTULA TERMINAL no Proteus Isis. O componente VIRTUAL TERMINAL você pega do menu “INSTRUMENTS” como mostra a figura abaixo.



O terminal virtual deve ser configurado com os mesmos parâmetros da comunicação serial do Arduino, a figura abaixo mostra a configuração padrão!

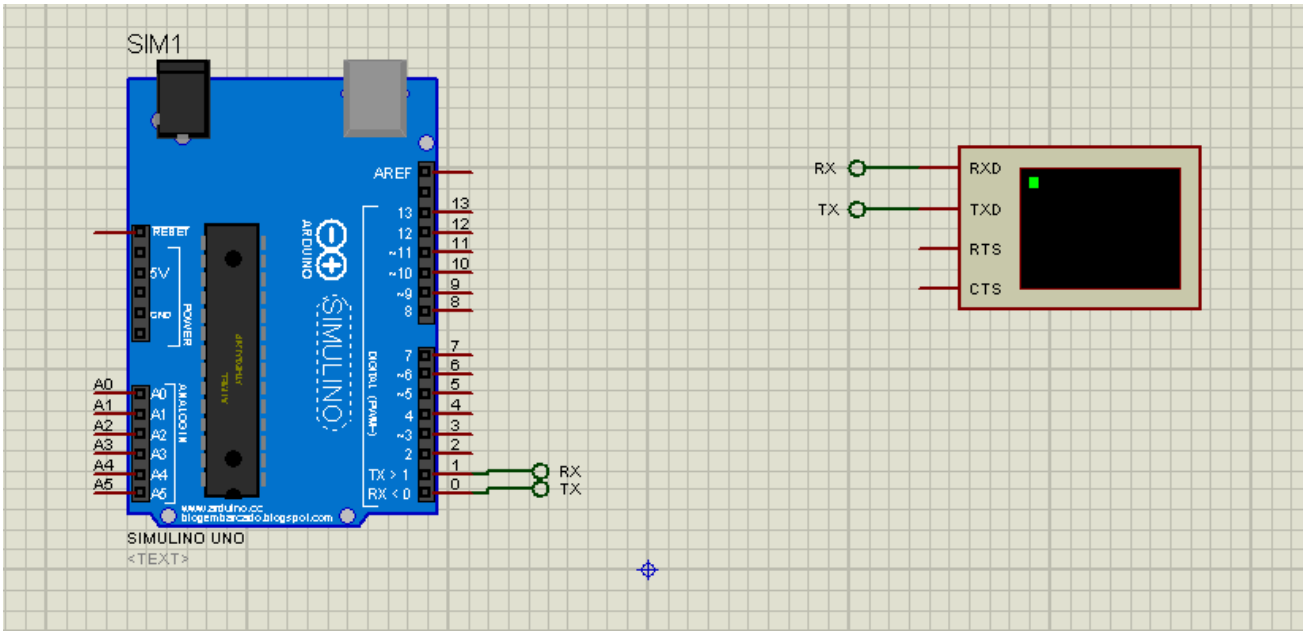




O circuito completo a ser montado no Proteus Isis é mostrado abaixo:

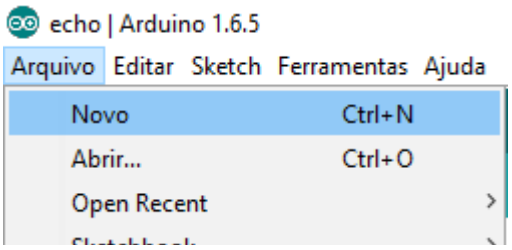
Observe que o pino TX do terminal é ligado ao pino RX do Arduino UNO, eo pino RX do terminal é ligado ao pino TX do Arduino. Na comunicação serial normalmente os cabos são invertidos o TX vai ligado ao RX e o RX vai ligado ao TX, pois o circuito que recebe o sinal deve receber o sinal do transmissor do outro dispositivo!

Salve o seu desenho como o nome ECHO na pasta que você criou para o seu projeto.



Criando o programa exemplo:

Você irá criar um programa a partir do zero.
Abra um projeto novo clicando no menu Arquivo e Novo.



Altere o programa conforme descrito abaixo.
Este é o programa de teste da serial!
Depois clique Arquivo Salvar como e selecione a pasta do seu projeto, salve como “echo”!

```
 byte byteRead;
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available())//se recebeu alguma coisa via serial
  {
    byteRead=Serial.read();//le o dado da entrada serial
    Serial.print("\rEcho=");//\r faz trocar de linha
    Serial.write(byteRead);//volta a string
    Serial.print("\rDECIMAL=");// \r faz trocar de linha
    Serial.print(byteRead);//mostra numero decimal
    Serial.print("\rHEXA=");//\r faz trocar de linha
    Serial.println(byteRead, HEX);//mostra numero HEXADECIMAL e troca linha
    delay(1);        // delay in ms between reads for stability
  }
}
```

As linhas que fazem a comunicação são mostradas abaixo.

A linha abaixo declara uma variável de comunicação que irá guardar cada byte recebido pela serial, uma palavra é composta por uma sequência de byte.

```
byte byteRead;
```

A linha abaixo colocada dentro do setup() cria uma serial com baud rate de 9600 bits por segundo.

```
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
```

A linha abaixa verifica se chegou algum byte via serial.

```
if(Serial.available())//se recebeu alguma coisa via serial
```

A linha abaixo lê um byte da serial e guarda na variável de comunicação.

```
byteRead=Serial.read();//le o dado da entrada serial
```

A linha abaixo a função print() transmite o texto “ECHO=” pela serial informando que o Arduino está na rede, o caractere “\r (Car return)” faz com que o texto mude de linha antes de escrever “Echo=”.

```
Serial.print("\rEcho=");//escreve um texto na serial \r faz trocar de linha
```

O ato de escrever de volta o byte recebido é chamado de echo!
Note que você deve escrever em linhas diferentes textos e números,

As linhas abaixo escrevem na serial o byte recebido no formato de letras “string”, para isto é usado a função Serial.Write()!

```
Serial.print("\rEcho=");//\r faz trocar de linha
Serial.write(byteRead);//volta a string
```

As linhas abaixo mostram o byte recebido no formato de número decimal, para isto é usado a função SerialPrint() ou SerialPrintln() .

```
Serial.print("\rDECIMAL=");// \r faz trocar de linha
Serial.print(byteRead);//mostra numero decimal
```

As linhas abaixo mostram o byte no formato Hexadecimal.
O parâmetro HEX indica que o byte de retorno será mostrado na base Hexadecimal do código ASCII!

```
Serial.print("\rHEXA=");//\r faz trocar de linha
Serial.println(byteRead, HEX);//mostra numero HEXADECIMAL e troca linha
```

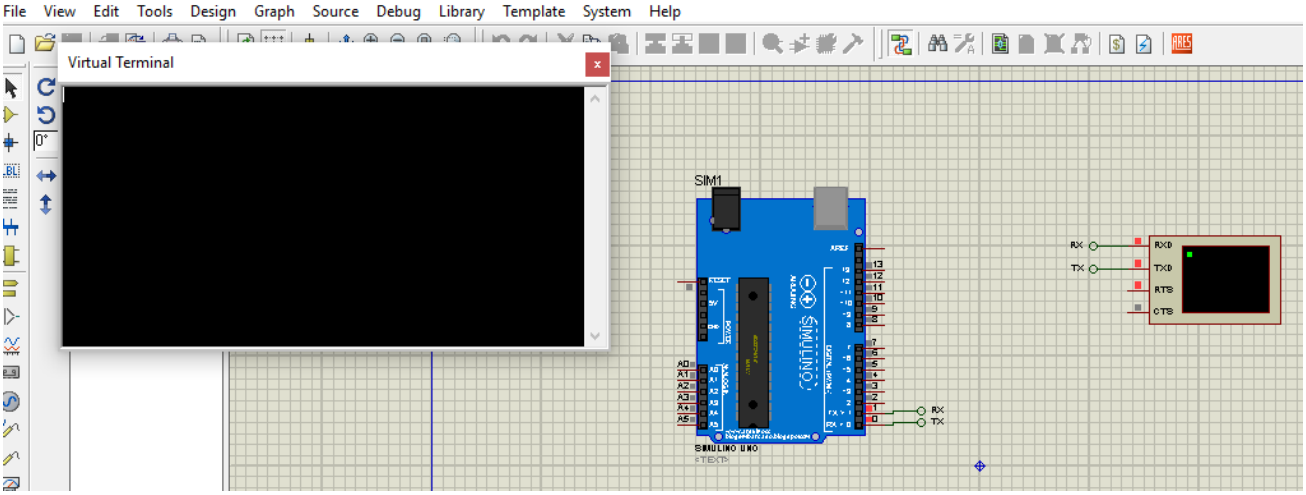
Resumo.
Você pode usar três instruçoes para escrever na serial.

```
Serial.write(byteRead);//volta a string
Serial.print("\rDECIMAL=");// \r faz trocar de linha
Serial.println(byteRead, HEX);//mostra numero HEXADECIMAL e troca linha
```

A função Serial.Println() no final da escrita troca de linha automaticamente, por isto as letras “ln” no final que significa “nova linha”!

Você pode dizer para trocar de linha na mensagem escrita “string” escrevendo o código “\r” dentro da frase!

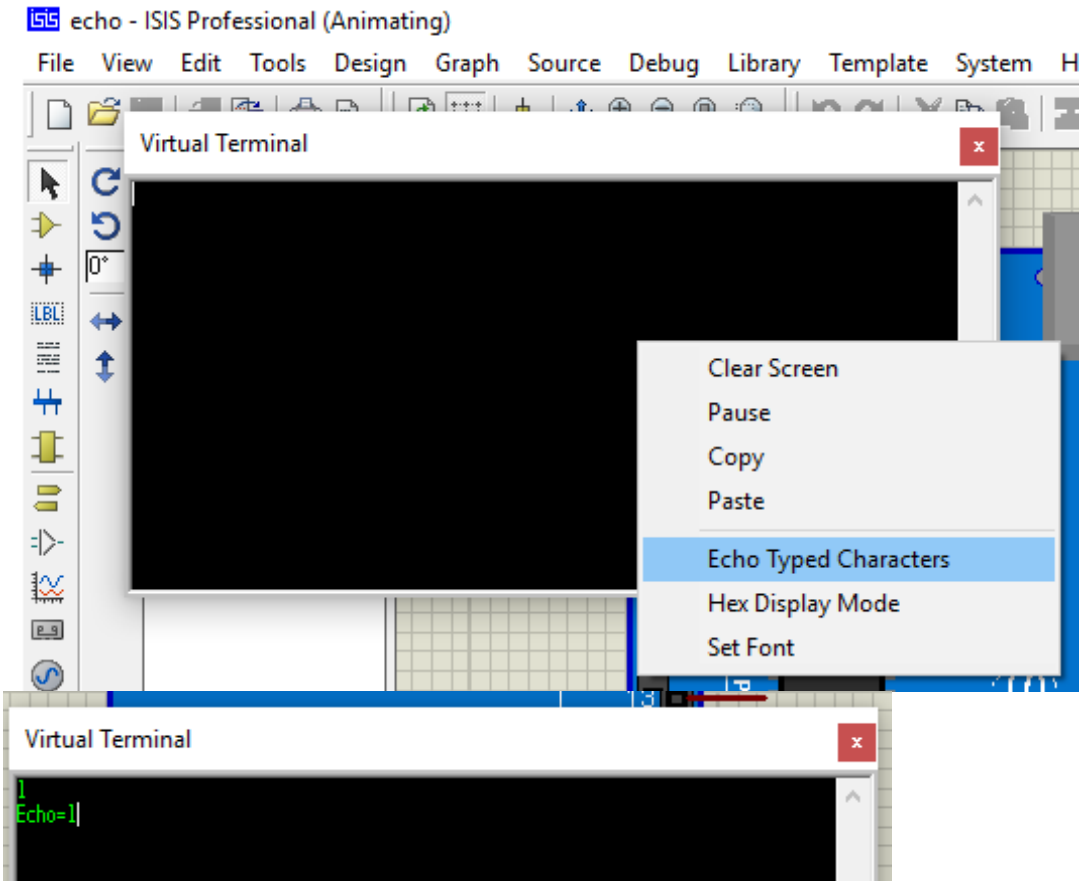
Compile e exporte o seu programa para a pasta criada para o projeto ECHO, note que se você já salvou como na pasta ECHO ao compilar e exportar o programa não pede para você escolher o endereço!
Abra o diagrama, carregue o programa no Arduino UNO e ligue o circuito.
Ao ligar o circuito aparecerá uma janela do Virtual Terminal, digita uma letra dentro da janela ao digitar a letra o terminal transmite para o Arduino a letra e o Arduino devolve com a palavra ECHO= na frente!



O resultado é mostrado abaixo.
Digite uma letra no terminal virtual para ver o efeito!

A função echo do terminal.

Você pode ativar a opção echo usando o botão direito do mouse sobre o terminal com isto a palavra digitada é escrita primeiro no terminal e depois enviada e ao ser recebida aparece novamente, assim para cada letra que você digita aparece duas, uma que você digitou outra depois do echo!



A Tabela de conversão ASCII.

Toda a comunicação via rede seja RS232, ou internet os textos e números são enviados seguindo a tabela de codificação chamada ASCII, esta tabela é mostrada abaixo.
A tabela abaixo está descrita em Hexadecimal, exemplo: A letra "A" tem o código char(41)!
Tabela ASCII Padrão de 1 BYTE (de 0 a 255)!

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Caractere alfanumérico são aqueles códigos que mostram letras e números, estes caracteres aparecem no monitor.
Já o caractere ENTER não aparece no monitor mas faz o cursor mudar de linha, estes não são caracteres alfanuméricos, mas são importantes para a comunicação!

Dos caracteres que não são alfanuméricos os mais importantes são mostrados abaixo e estão bem no início da tabela ASCII.

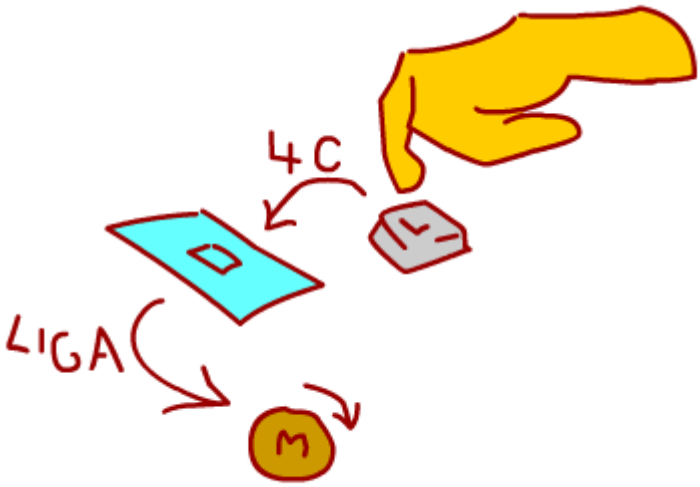
	ESPAÇO	HEX 20	Teclado espaço.
BEL	BUZINA	HEX 7	Liga a buzina do computador
CR	Retorno do carro	HEX D	Usado para mudar o cursor de linha
LF	NOVA LINHA	HEX A	Usado para mudar o cursor de linha

O CR\LF são usados em conjunto tudo depende do tipo de equipamento e são usados para indicar que uma mensagem completa terminou!

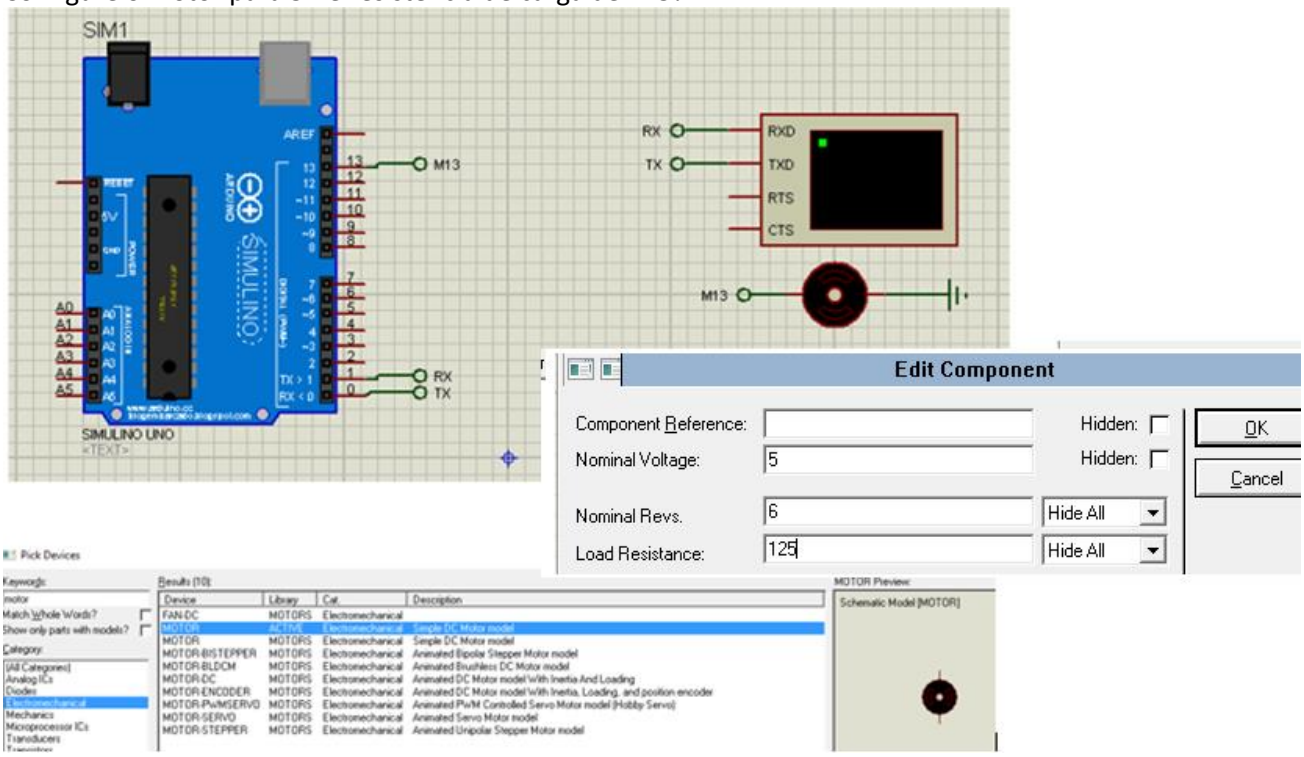
*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/

Usando um comando remoto:

Você pode enviar um comando via rede serial e então o Arduino irá tomar uma ação, o comando é simplesmente um caractere codificado em ASCII que é recebido via serial e armazenado na variável byteRead que poderá ser testada mais tarde!



Como exemplo você irá montar um motor no projeto ECHO anterior e programar o Arduino para ligar este motor quando a letra “L” for digitada.
O circuito é mostrado abaixo, o motor usado é do tipo MOTOR ACTIVE.
Configure o motor para 5V e resistência de carga de 125!



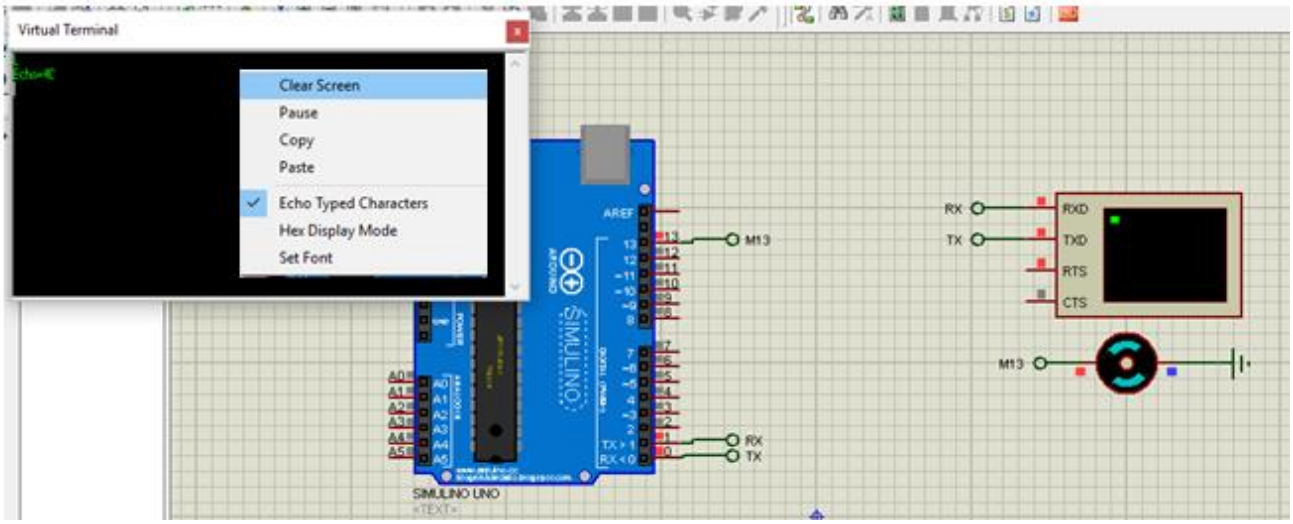
O programa é mostrado abaixo.
Primeiro declare o motor com o nome M13 e configure para saída!

```
 int M13=13;
 byte byteRead;
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(M13,OUTPUT);
}
```

Depois você deverá incluir a linha de teste do caractere contido na variável byteRead.
Observe que a pergunta dentro do if compara a variável byteRead com o caractere 'L', um caractere alfa numérico deve ser escrito entre aspas simples, diferente dos textos que deve ser escrito com aspas(k) duplas "texto").
Outro detalhe é que o caractere de letra maiúscula e minúscula possuem códigos diferentes!

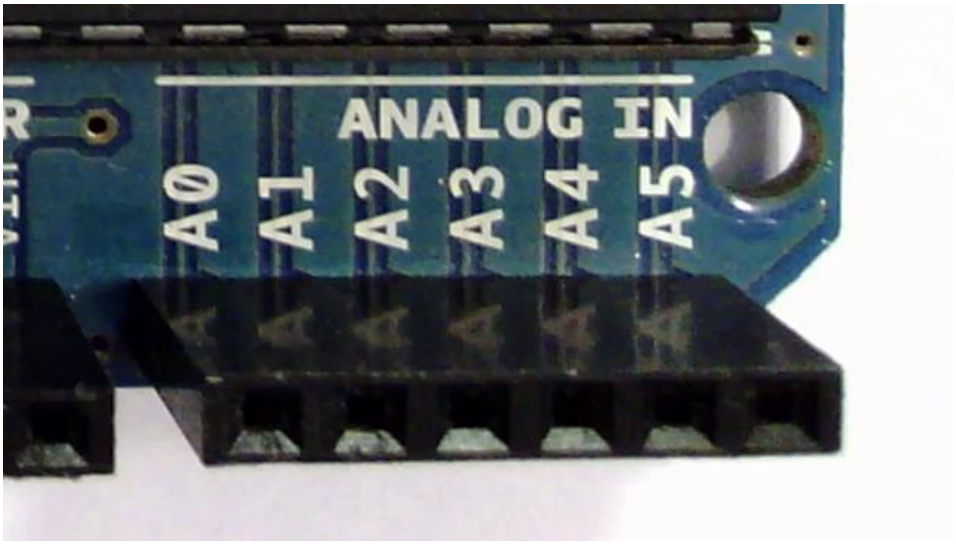
```
void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available())//se recebeu alguma coisa via serial
  {
    byteRead=Serial.read();//le o dado da entrada serial
    Serial.print("\rEcho=");//escreve um texto na serial \r faz trocar de linha
    Serial.print(byteRead, HEX); //escreve um byte na serial
    Serial.println();
    delay(1);        // delay in ms between reads for stability
    }
    if (byteRead=='L'){//se receber a letra L
      digitalWrite(M13,HIGH);//liga o motor
    }
    else//se não receber a letra L
    {
      digitalWrite(M13,LOW);//desliga o motor
    }
  }
```

Compile e gere o arquivo .hex.
Teste o projeto digitando a letra “L” para ligar o motor e de
Ligue digitando outro caractere!
Para aparecer a letra antes do Echo não esqueça de ativar o Echo Typed Mode!

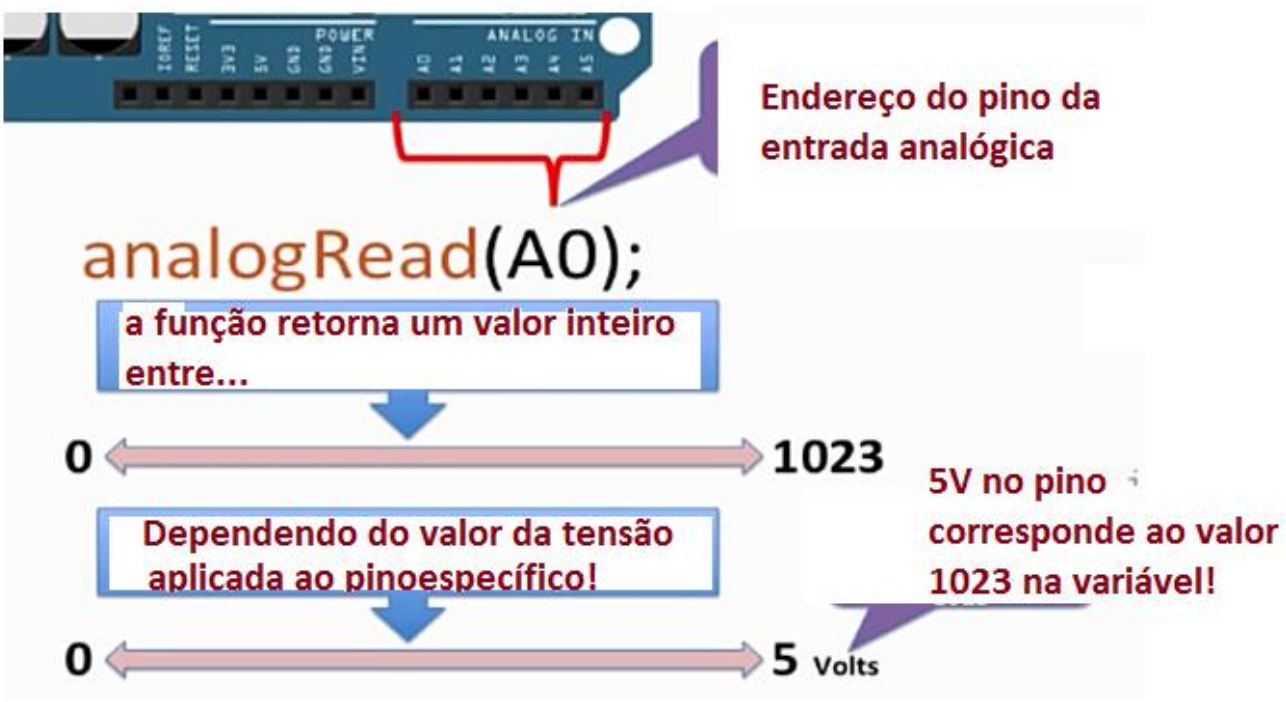


Usando as portas analógica no Arduino UNO.

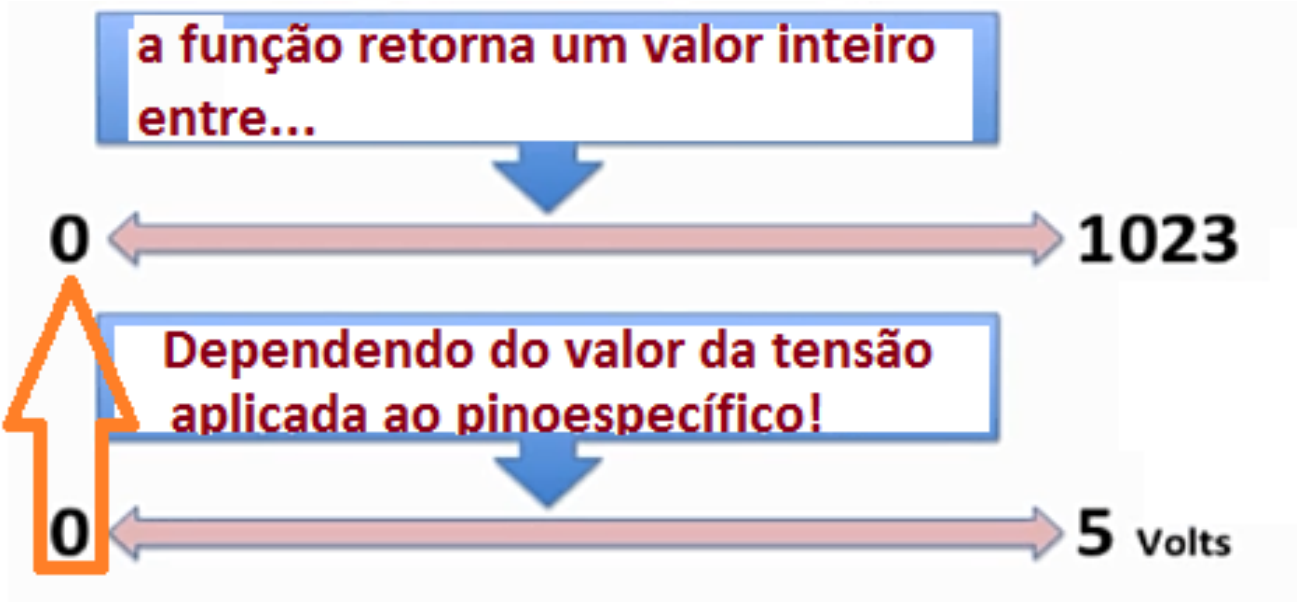
O Arduino UNO possui seis portas analógicas de A0 a A5.
A entrada analógica lê valores de 0 a 5Vdc e converte em um valor digital de 10 bits que poderá ser usado no seu programa.



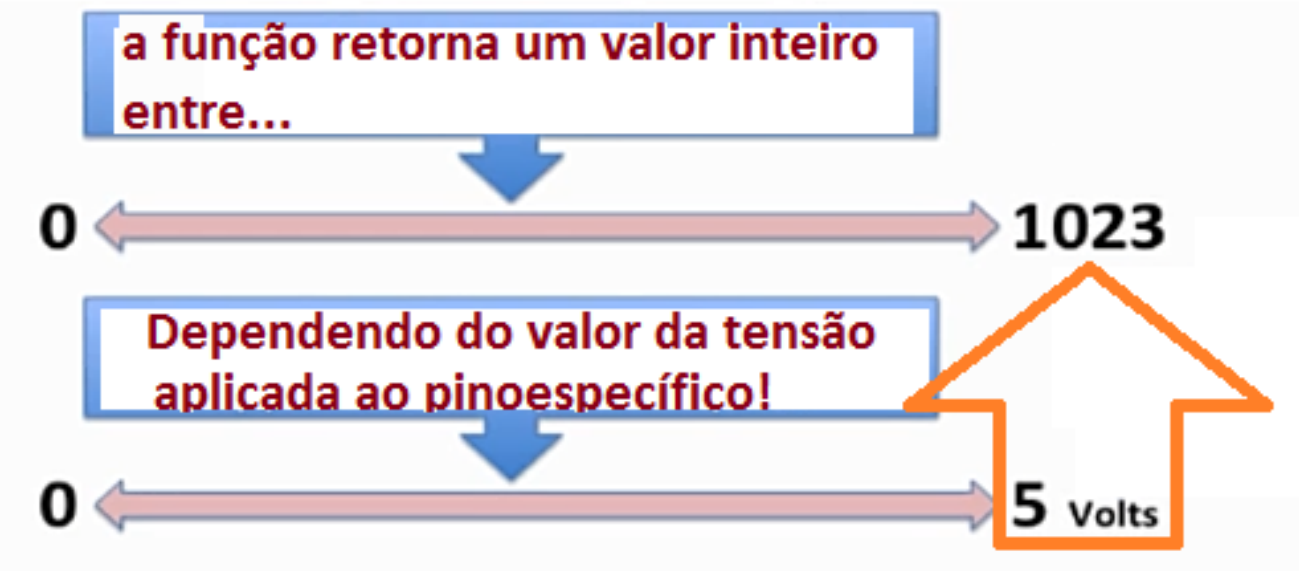
O conversor do Arduino é de 10 bits podendo gerar um valor máximo de 1023 quando a tensão de entrada é de 5V, o restante dos valores é proporcional.



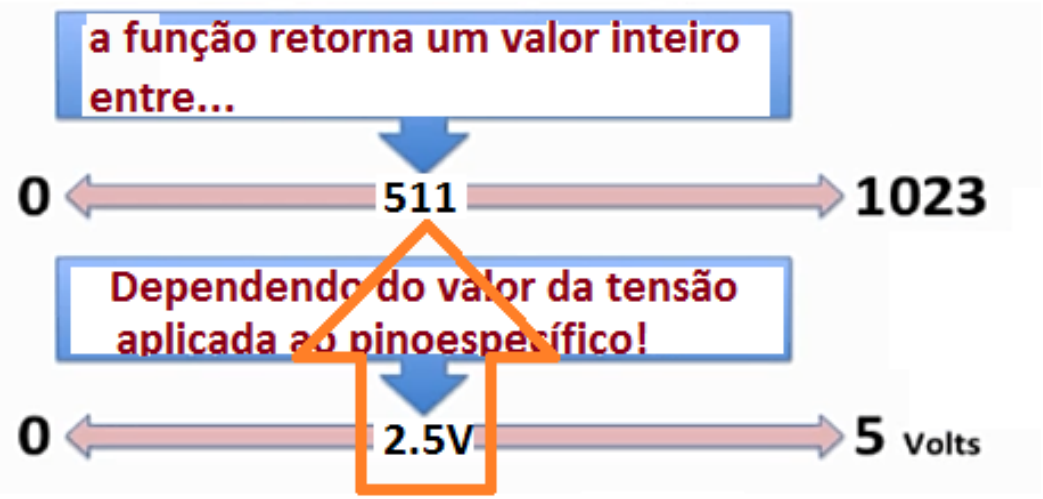
Se aplicar zero volt ao pino A0, isto é o potenciômetro estiver posicionada para a terra, o valor na variável analógica será 0!



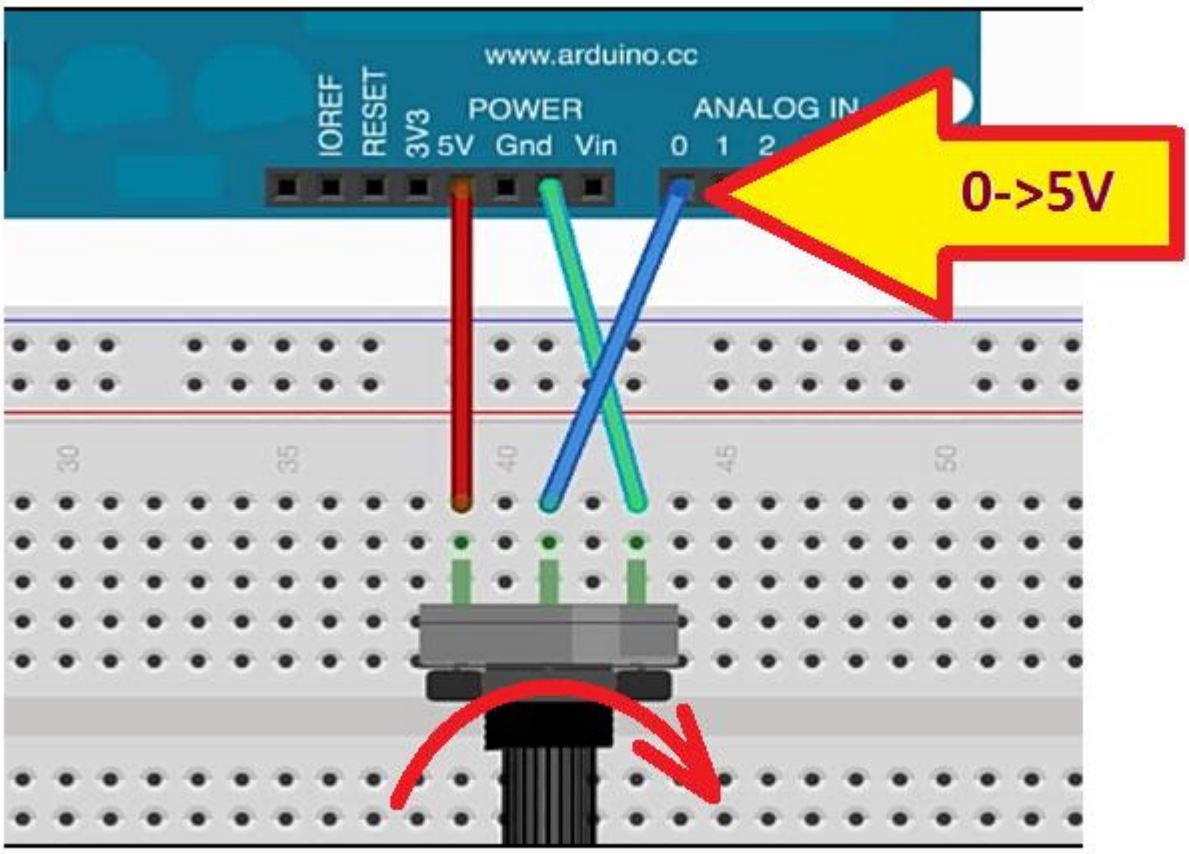
Se você aplicar 5 Volts ao pino A0,isto é gira o potenciômetro para o lado da tensão VCC, então a variável analógica vai assumir o valor 1023!



Para posições intermediárias o valor irá variar na mesma proporção, assim para a metade do curso a tensão no pino será de 2.5 Volts e o valor da variável será de 511!



Quando você girar o potenciômetro a tensão na entrada analógica vai variar de 0-5 Volts e o valor da variável analógica vai variar de 0 a 1023!



Funções para trabalhar com as portas analógicas:

O Arduino já possui as funções de leitura e escrita de valores analógicos nas portas analógicas já prontas em sua biblioteca, são três as funções disponíveis:

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

analogRead():

Lê o valor analógico na porta específica, esta é a função mais importante, pois ler valores analógicos é a principal função das portas analógicas!

O valor pode variar de 0 a 1023 (1024 unidades) ou 4,9mV por unidade, este é o menor valor de tensão que a entrada analógica pode ler.
O valor de retorno é um número do tipo inteiro!
O exemplo abaixo mostra uma leitura na entrada analógica A0 e o valor é armazenado na variável do tipo inteiro de nome sensorValue, esta é a forma padrão de ler uma entrada analógica lembrando que a variável de leitura pode ser declarada no início do programa!

Syntax

analogRead(pin)

```
// read the input on analog pin 0:
int sensorValue = analogRead(A0);
```

analogReference():

Configura a tensão de referência a ser usada para uma leitura analógica, o valor normal é 5V (tensão VCC). As opções são:

- DEFAULT: o valor padrão para a tensão de referência é de 5 volts nas placas de 5V ou 3,3 nas placas de 3,3V!
- INTERNAL: Referência interna igual a 1,1V quando o microprocessador for do tipo ATmega168 ouATmega328 e 2.56 volts para o ATmega8(*não disponível no Arduino Mega*)
- INTERNAL1V1: Referência interna igual a 1,1V (somente par ao *Arduino Mega*)
- Referência externa uma tensão de (0 a 5V) deve ser ligada ao pino AREF para ser usada como referência.

Normalmente esta função não é usada e a referência é a DEFAULT de 5volts!

Syntax

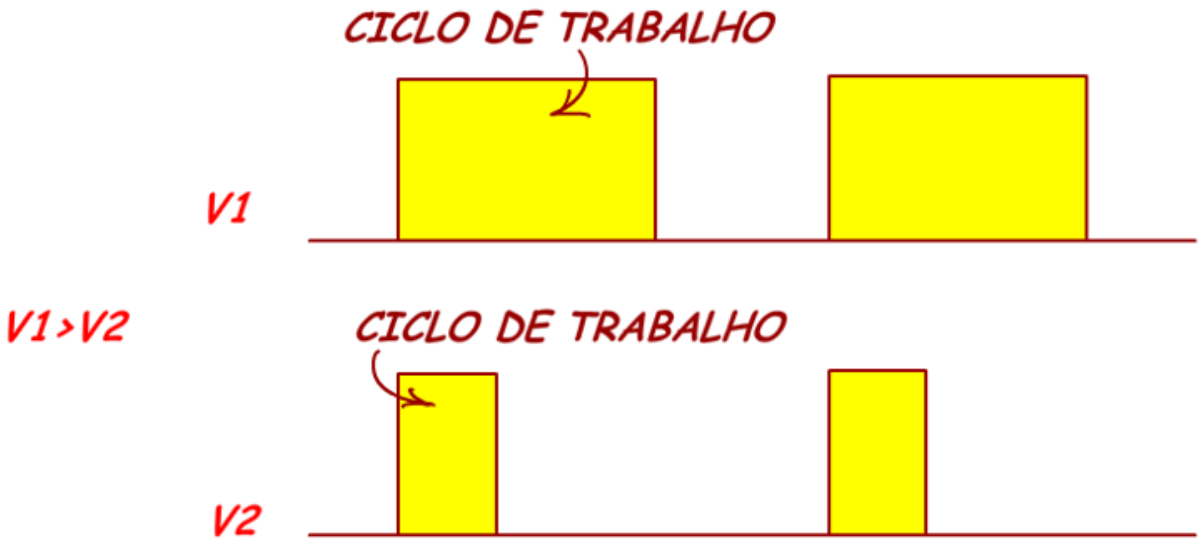
```
analogReference(type)
```

analogWrite()

Este não é uma função que gera 0 a 5V na porta analógica e sim uma função que gera uma onda quadrada do tipo PWM onde a tensão média de saída é função do ciclo de trabalho!
Nesta função o valor programado pode variar de 0 a255, valor bem diferente da entrada analógica, gerando uma tensão média de 0 a 5V .

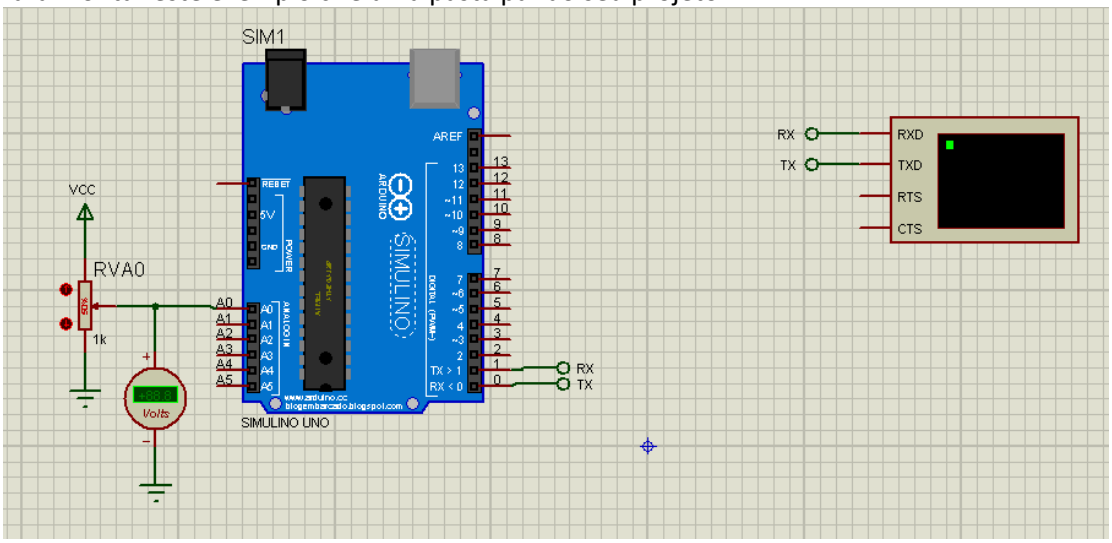
Syntax

analogWrite(pin, value)



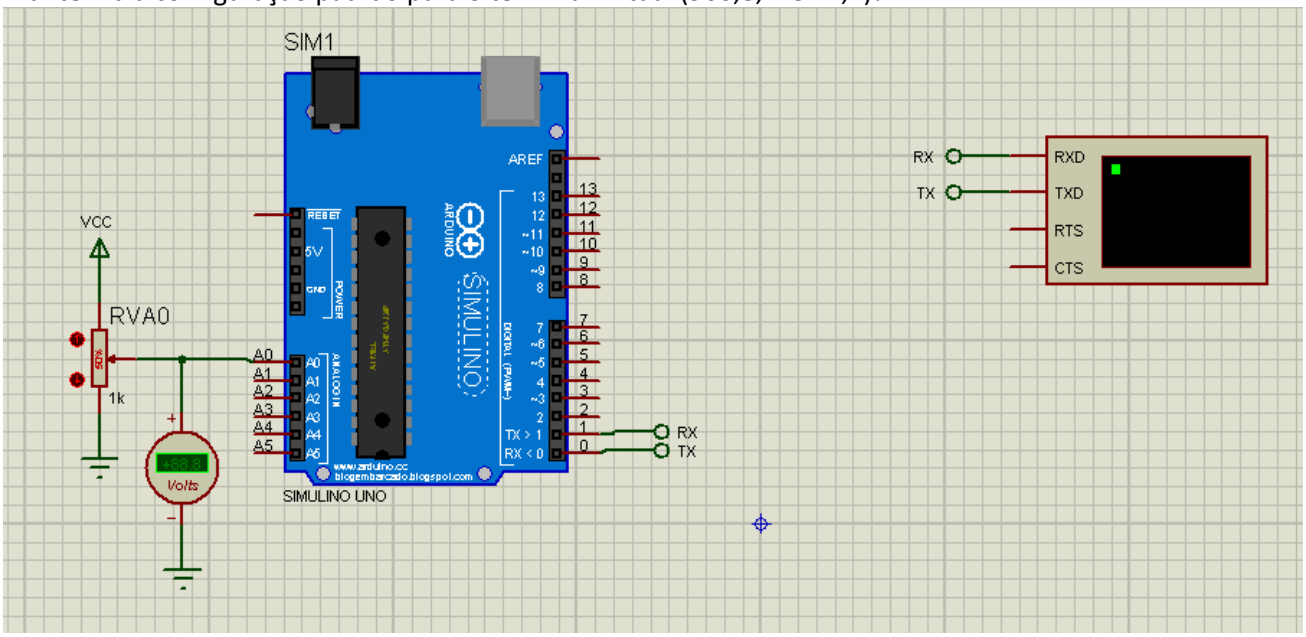
Exemplo de programa para leitura de uma entrada analógica.

Neste exemplo você será orientado a montar um projeto para ler o valor da entrada analógica ajustado por um potenciômetro ligado a entrada A0, e este valor será mostrada no terminal virtual, se você tiver uma placa real o valor será mostrado no terminal seria da interface.
Para montar este exemplo crie uma pasta par ao seu projeto.

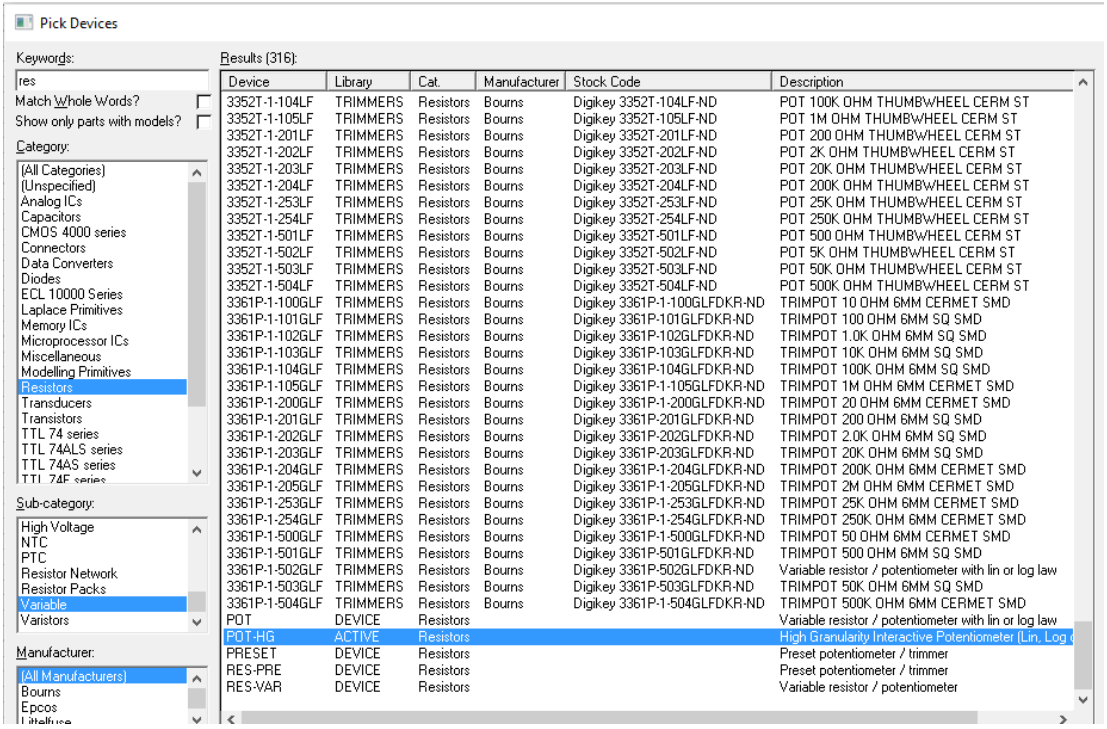


Montando o circuito:

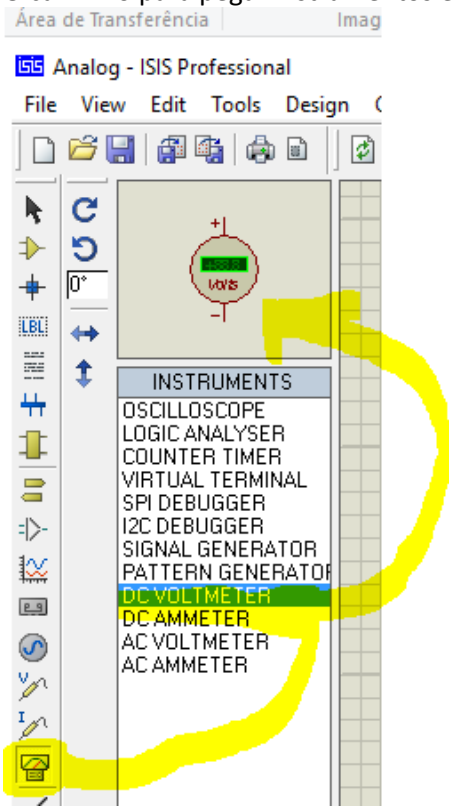
Monte o circuito abaixo no Proteus Isis.
O potenciômetro você pega da biblioteca digitando Res na área de procura selecione Resistors no cmapo de Category e variable na sub-Category e então pegue o POT-HG bem no fim da lista.
O instrumento você pega do menu “Virtual Instruments Mode”
Mantenha a configuração padrão para o terminal virtual (960,8, NONE,1)!



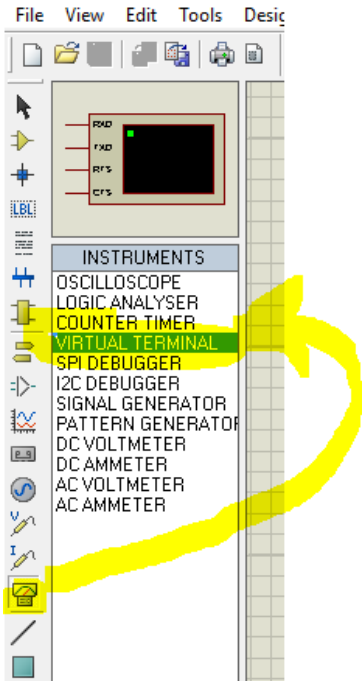
Como pegar o potenciômetro na biblioteca.
Este é um potenciômetro interativo!



Onde pegar o voltímetro no menu “INSTRUMENTS”:
O caminho para pegar instrumentos é mostrado abaixo

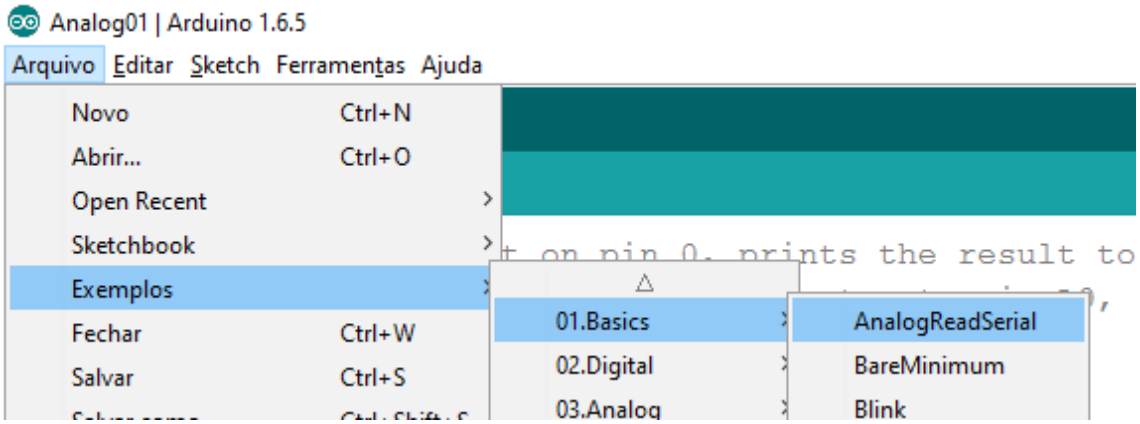


Como pegar o terminal virtual no menu “INSTRUMENTS”!



Montando o programa:

Abra a interface de programação do Arduino, salve na sua pasta de projetos com o nome Analog01.
Vá no menu Arquivo Exemplo/01.Basic;/AnalogReadSerial e abra este exemplo!



O programa exemplo é mostrado abaixo:
No setup inicia serial:

```
Serial.begin(9600);
```

No loop lê a entrada analógica A0 e coloca o valor na variável sensorValue:

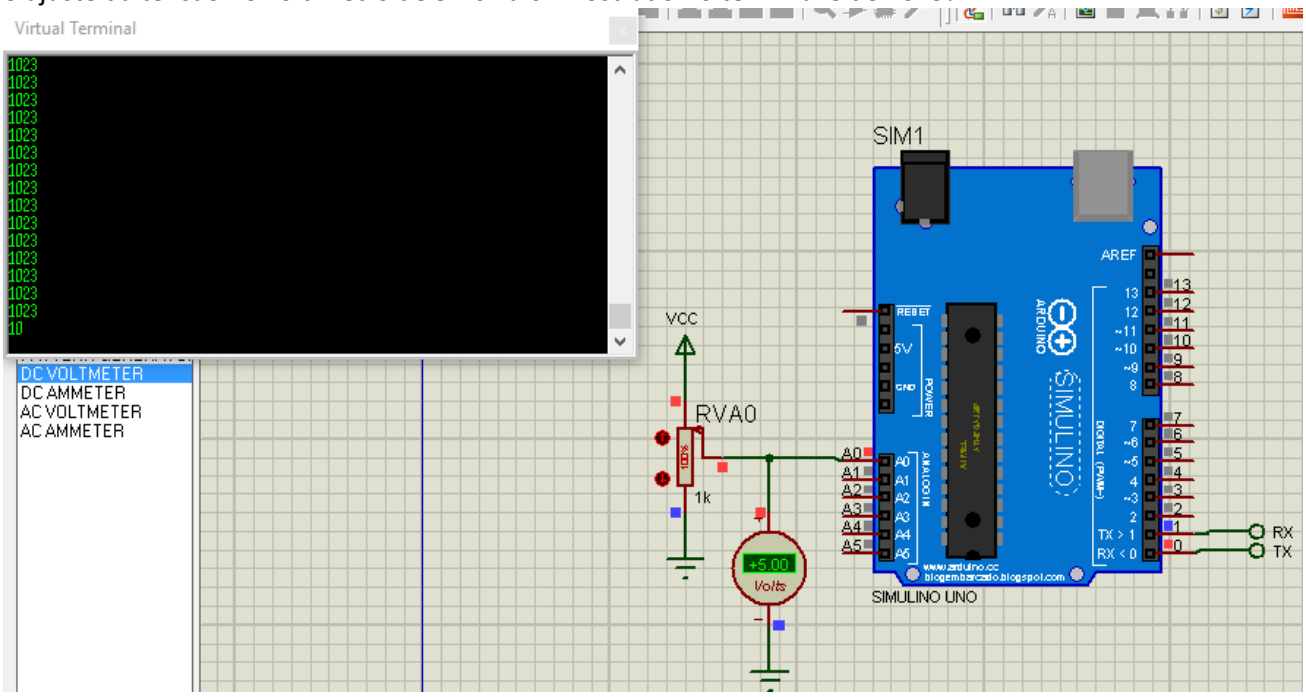
```
int sensorValue = analogRead(A0);
```

Escreve o valor na serial:

```
Serial.println(sensorValue);
```

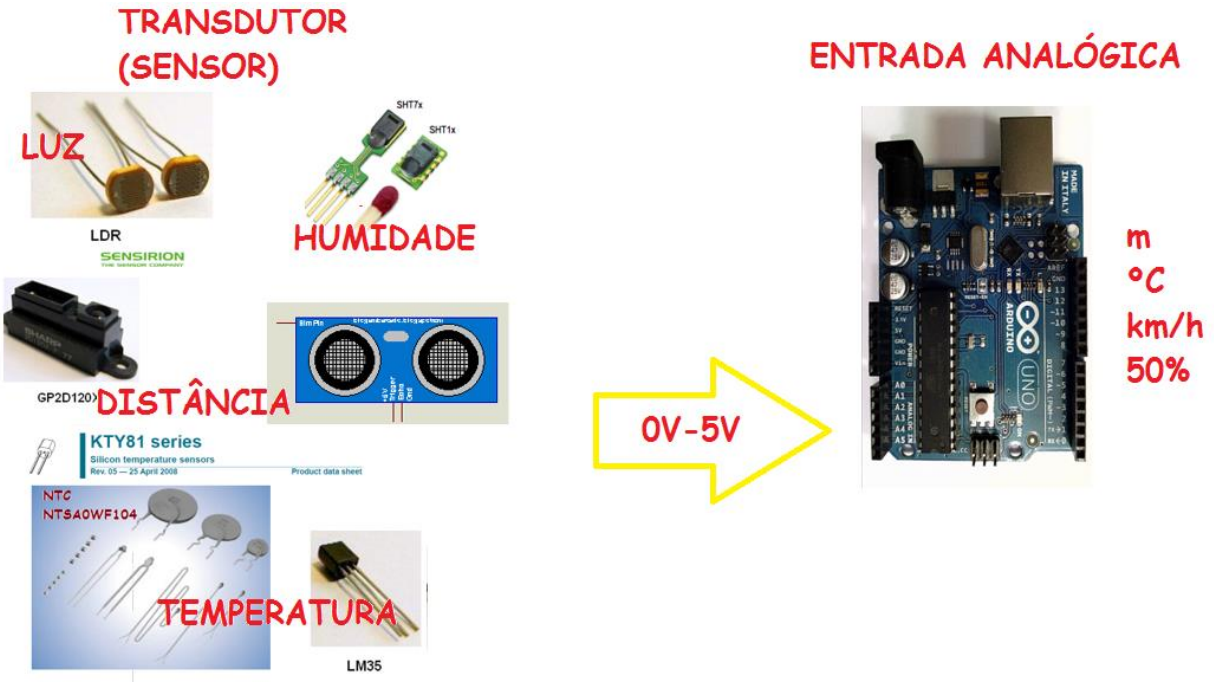
```
9  // the setup routine runs once when you press reset:
10 void setup() {
11   // initialize serial communication at 9600 bits per second:
12   Serial.begin(9600);
13 }
14
15 // the loop routine runs over and over again forever:
16 void loop() {
17   // read the input on analog pin 0:
18   int sensorValue = analogRead(A0);
19   // print out the value you read:
20
21   Serial.println(sensorValue);
22   delay(1);        // delay in between reads for stability
23 }
```

Compile e carrega o programa no Arduino do simulador e rode o programa.
Gire o potenciômetro para variar o valor observe o valor de tensão no voltímetro e o valor no terminal, com o ajuste da tensão no voltímetro de 5V o valor mostrado no terminal é de 1023!

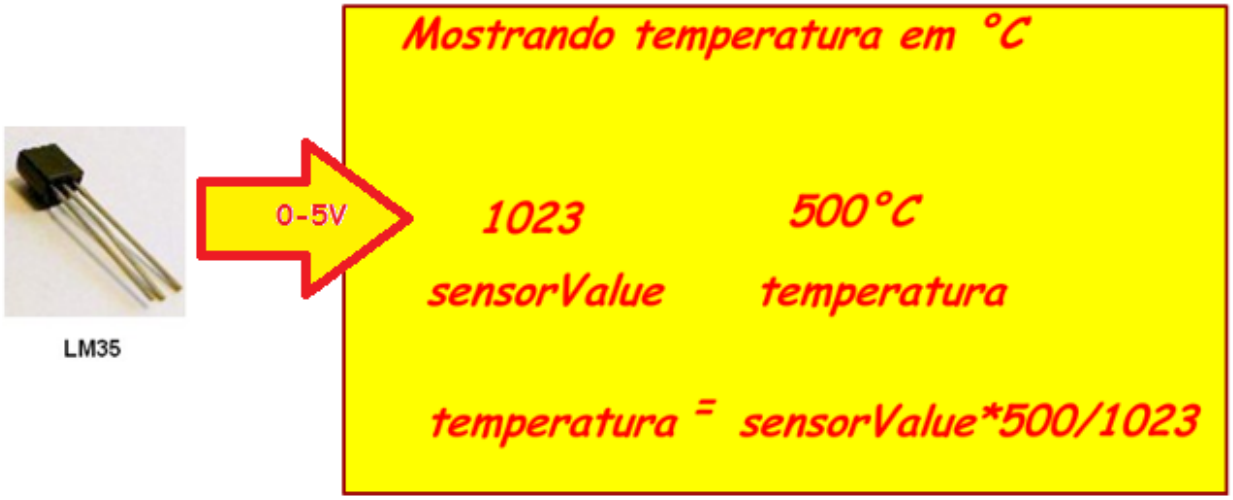


Como ajustar o valor analógico mostrado para a grandeza desejada.

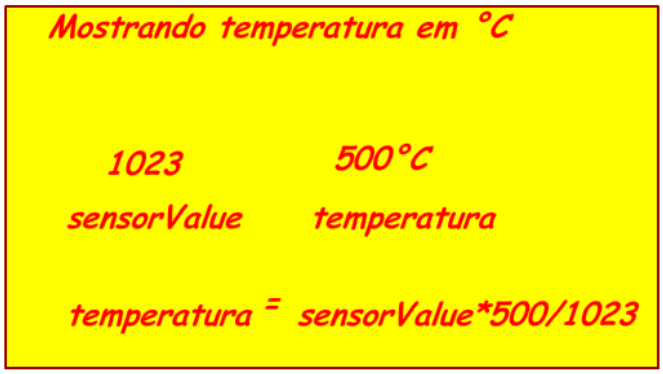
O valor mostrado no terminal é o valor da variável gerada pelo conversor ADC, no entanto você vai querer mostrar as grandezas que estão sendo medidas na sua unidade, por exemplo, temperatura em graus Celsius, velocidade em quilômetro por hora, tensão em Volt, corrente em Ampére, humidade etc.



Para fazer isto você terá que montar uma regra de três no seu programa, você só tem que saber a relação entre a tensão de saída do seu sensor e o valor da unidade da grandeza a ser medida, por exemplo, o sensor de temperatura LM35 gera 1V com 100°C. Para leitura de temperatura com o LM35 5V na entrada analógica significa 500°C que tem o valor analógico de 1023.



No caso do nosso exemplo, o Arduino poderia funcionar como voltímetro mostrando o mesmo valor do instrumento, neste caso 5V na entrada tem o valor analógico de 1023, a regra de três para ajuste do valor do sensor LM35 e do valor lido pelo potenciômetro são mostrados abaixo.



Altere o programa do exemplo para funcionar como um voltímetro como é mostrado abaixo!

O programa completo é mostrado abaixo.

```
float tensao;//declara uma variável para mostrar a tensão
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  tensao=(float)sensorValue* 5 /1023;//ajusta valor para mostrar tensão no display
  Serial.println(tensao);//mostra o valor da  tensaõ via serial
  delay(1);        // delay in between reads for stability
}
```

Os principais detalhes do programa são descritos a seguir.
A linha abaixo declara uma variável tensão que mostra valores com virgula por isto o tipo é float:

```
8 float tensao;//declara uma variável para mostrar a tensão
```

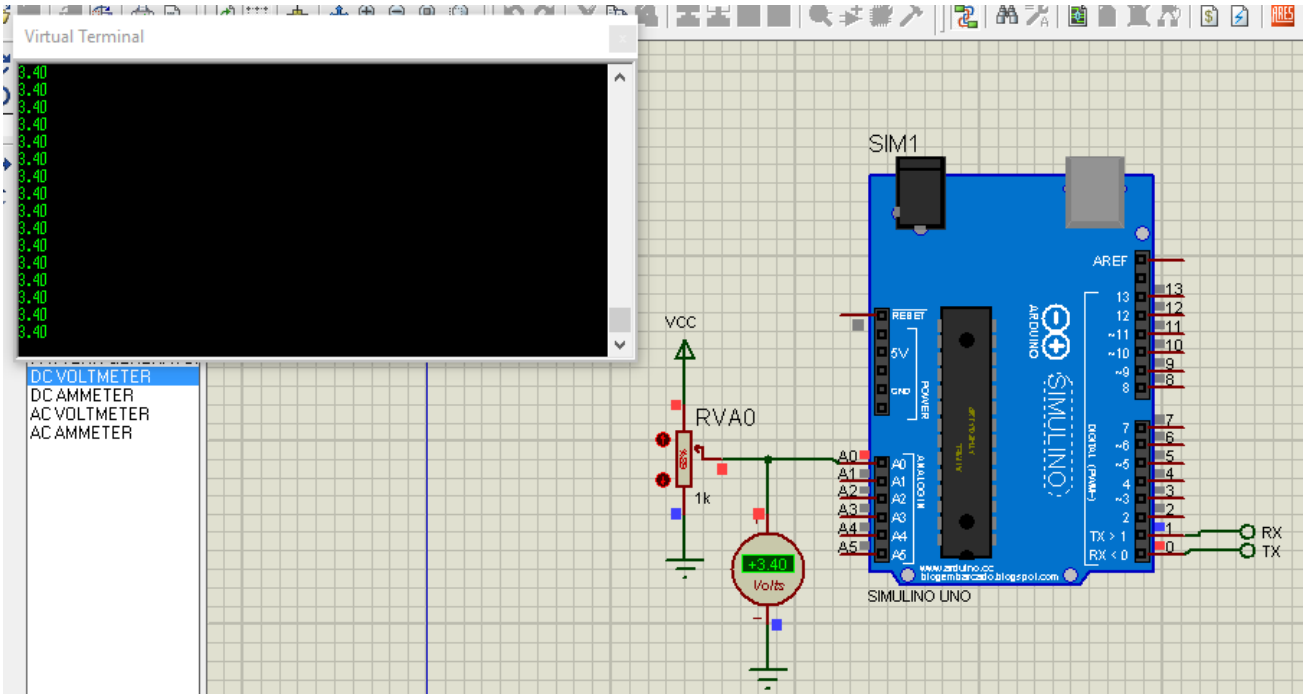
A linha abaixo ajusta o valor usando uma regra de três simples.

```
20   tensao=(float)sensorValue* 5 /1023;//ajusta valor para mostrar tensão no display
```

A linha abaixo transmite o valor da tensão via serial!

```
Serial.println(tensao);//mostra o valor da  tensaõ via serial
```

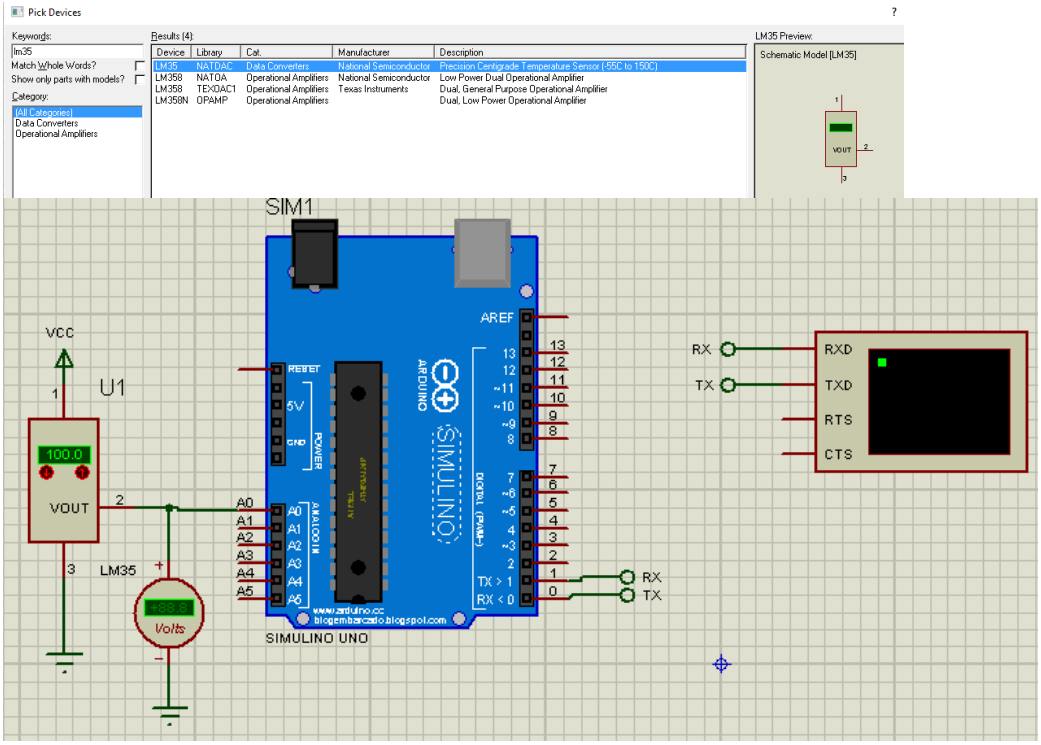
Compile e carregue o programa no Arduino do simulador e observe o resultado.
Note que os valores do voltímetro e o mostrado no terminal nem sempre coincidem, isto é, devido ao erro da conversão e a aproximação do cálculo que no Arduino gera valor com duas casas decimais!



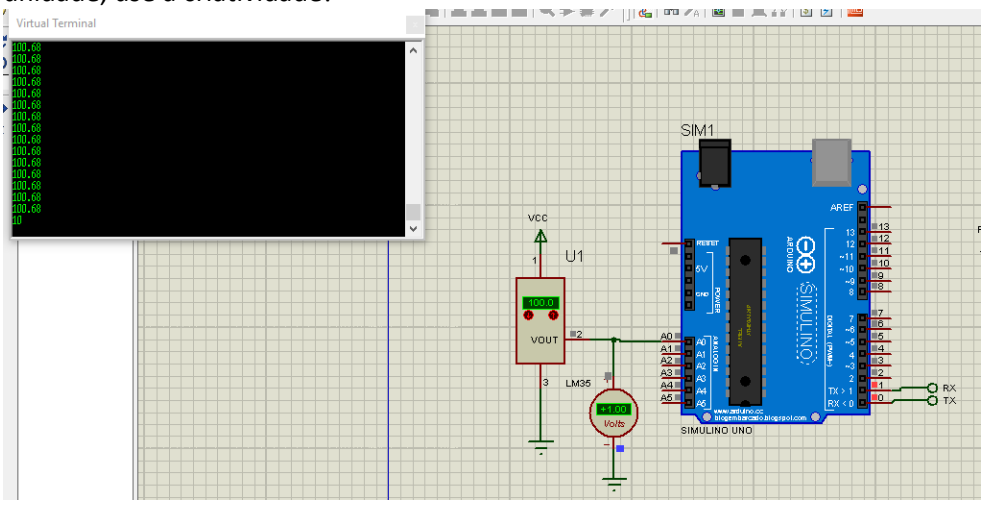
Desafio 4:

DESAFIO 4:

Monte o circuito abaixo com o sensor de temperatura LM35 e ajuste o programa para mostrar a leitura em graus Celsius!



O resultado é mostrado abaixo, ajuste para que o resultado apareça de forma mais interativa mostrando a unidade, use a criatividade!



Usando uma saída analógica.

Portas das saídas analógicas:

Antes de usar esta função você deve especificar a porta a ser usada na função (pino) como saída, esta porta deverá do tipo PWM estas portas possui o desenho de um “~ til” antes do número.

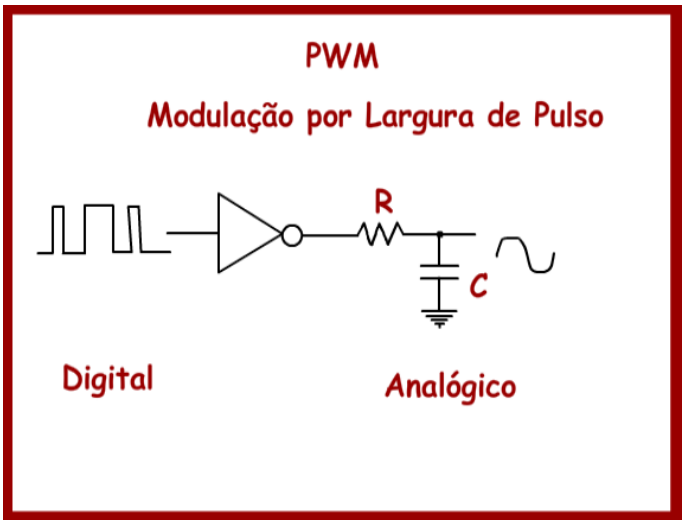
Saídas analógica PWM.

A função analogWrite() fornece uma forma simples de simular uma saída analógica, na verdade o Arduino gera um sinal do tipo PWM onde a largura do pulso é proporcional a tensão.

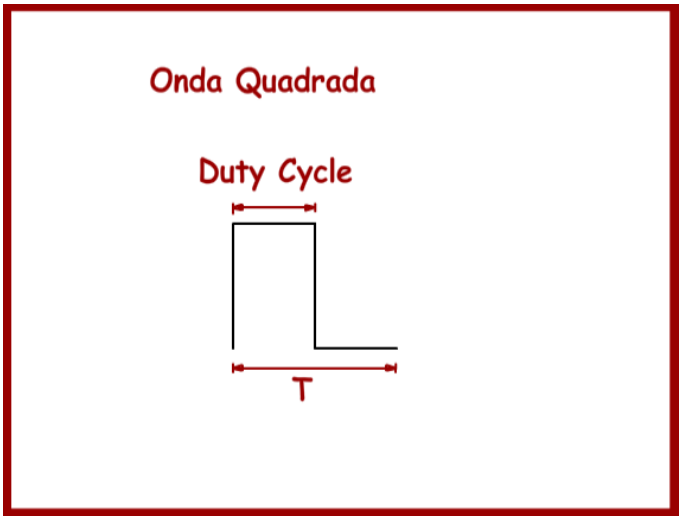
O que é PWM?

PWM é acrônimo do inglês PULSE WIDTH MODULATION, modulação por largura de pulso.

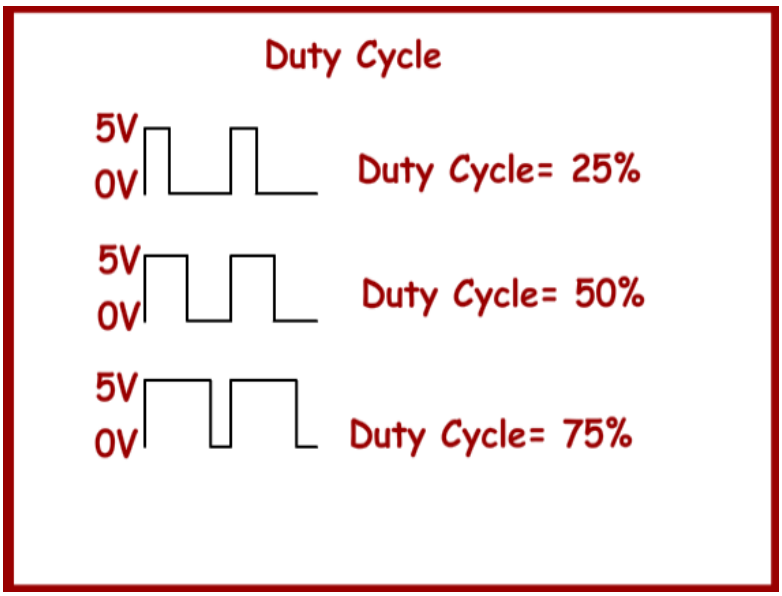
O PWM é a forma mais simples de gerar um sinal analógico através de um circuito digital.



O sinal do tipo PWM consiste em uma onda quadrada gerado por um circuito digital onde o tempo de ligado e desligado são diferentes. O tempo de ligado da onda quadrada é chamado de Duty Cycle (tempo útil). A frequência "f" da onda quadrada é fixa no PWM, o que é a mesma coisa que dizer que o período "T" da onda quadrada é constante!



O Duty Cycle é especificado em percentagem do tempo de ligado em relação ao desligado. Zero por cento significa totalmente desligado, e cem por cento significa totalmente ligado.

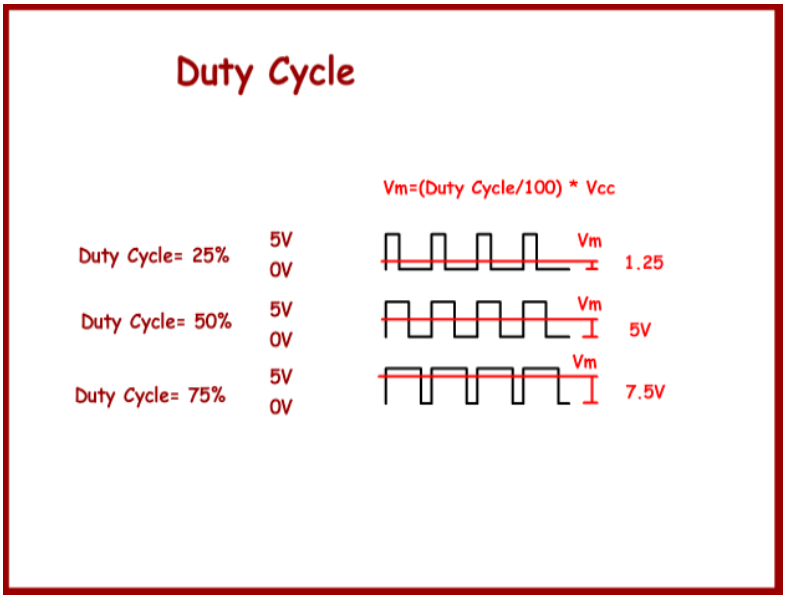


A tensão analógica será a média da onda quadrada (Vm), quando mais tempo ligado maior a tensão analógica. A tensão média é dada fórmula mostrada na figura abaixo onde:

Vm é a tensão média.

Duty Cycle em percentagem.

Vcc tensão máxima da saída digital, no Arduino é 5Vcc.

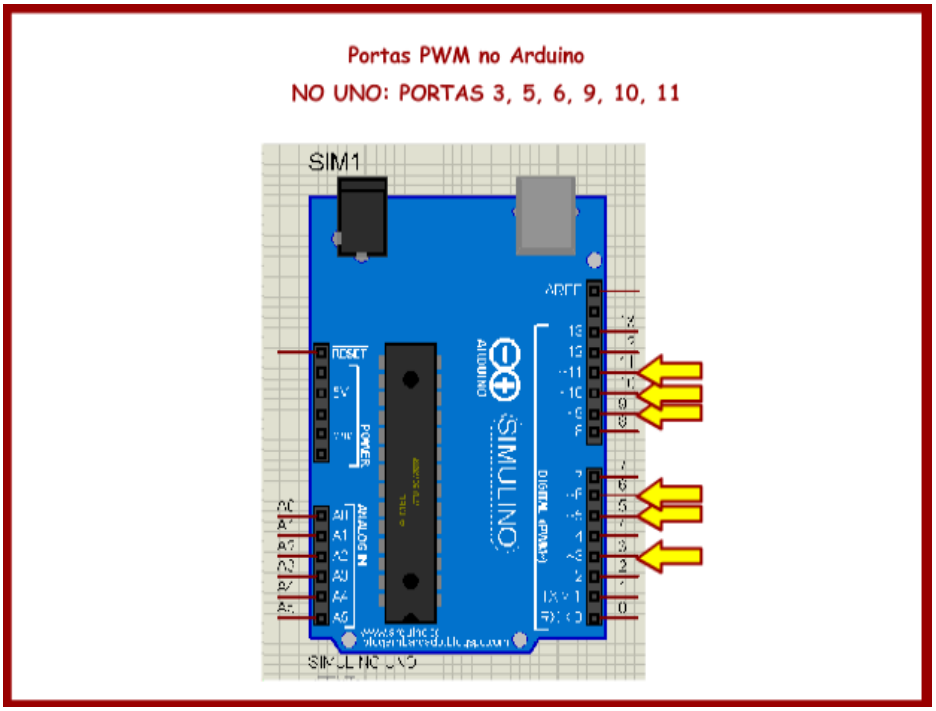


O PWM é usado para diversas aplicações, como por exemplo:

- Controle velocidade de motores CC.
- Variação luminosidade de LEDs e lâmpadas incandescentes.
- Geração de sinais analógicos com senoides.
- Geração de sinais de áudio.

O PWM no Arduino.

O Arduino possui algumas portas digitais que podem ser usadas como saídas PWM, estas portas estão marcadas com o sinal " ~ " na frente do número, como mostra a figura abaixo.



Para gerar o sinal PWM o Arduino possui a função analogWrite().

Para usar uma porta com esta função, a porta deverá ser configurada como saída, e somente as portas do tipo PWM podem ser usadas.

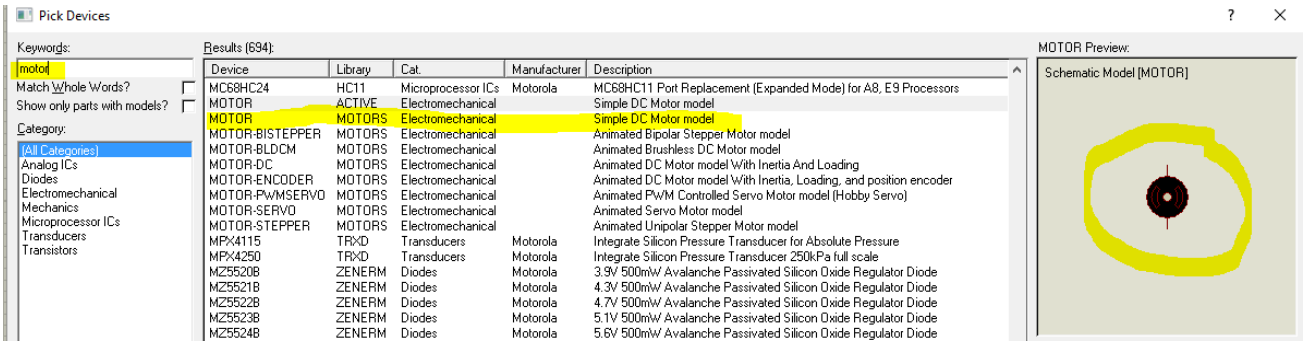
A sintaxe da função é mostrada na figura abaixo.

A frequência do PWM no Arduino é fixa, na maioria dos pinos é 490Hz no pino 5 e 6 do Arduino UNO é 980Hz.

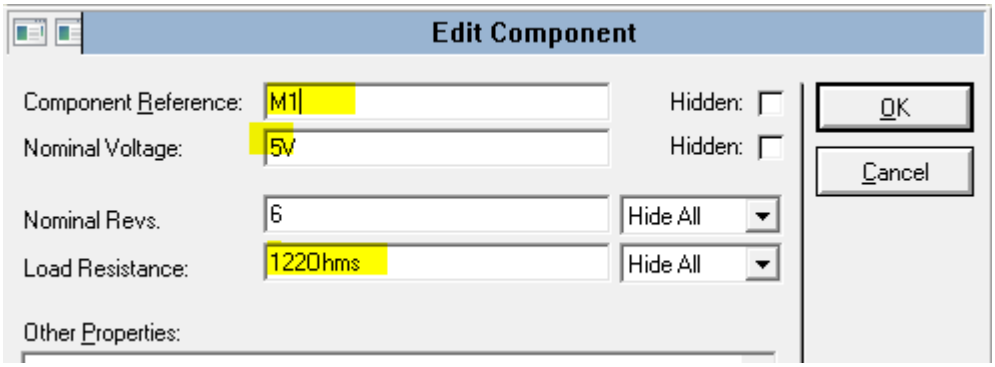
Se você quiser gerar uma função do tipo senoidal a frequência do PWM deverá ser no mínimo igual ao dobro, quanto maior a frequência mais precisa sai a forma de onda, assim no Arduino você poderá gerar senoides menores do que 248Hz ou 490 Hz se usar os pinos 5 ou 6!

Exemplo, acionamento de um motor CC.

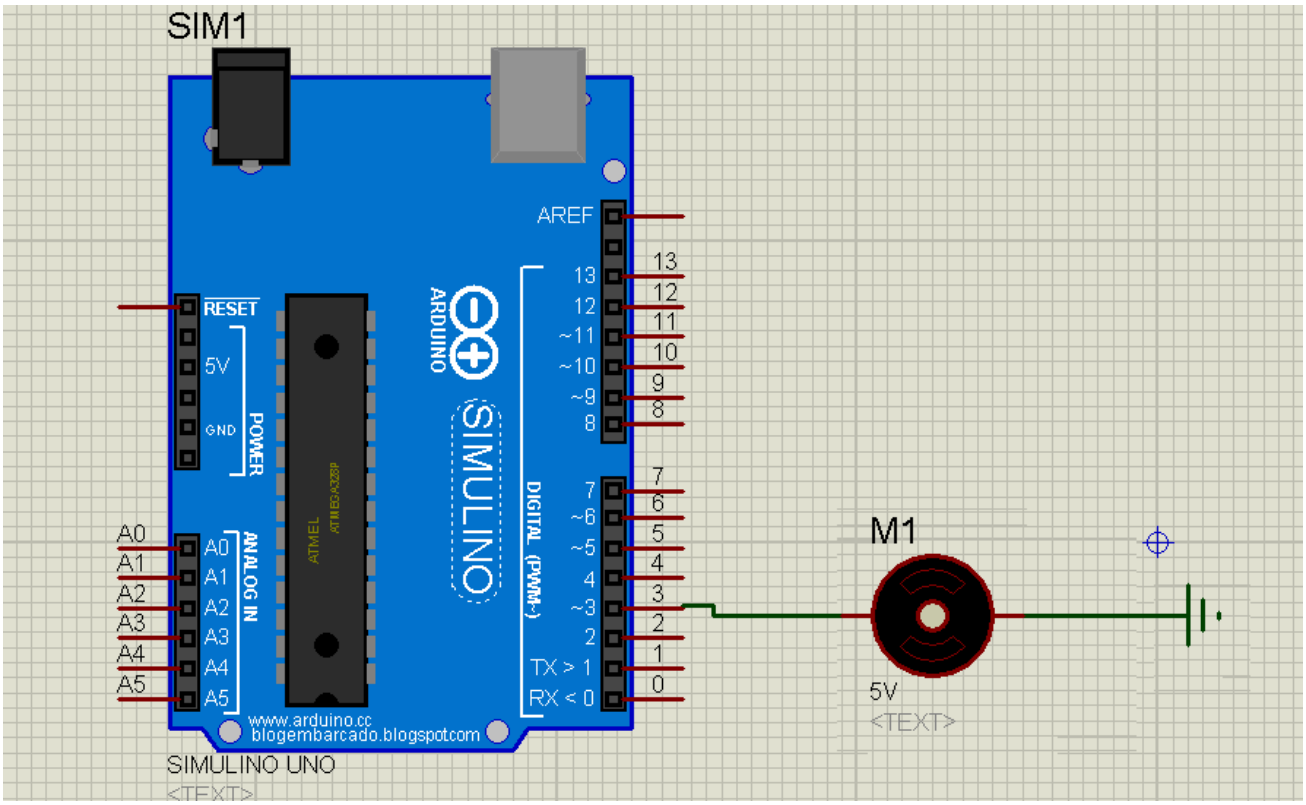
Monte o circuito abaixo onde você pega o motor da livraria como mostra a figura abaixo.



Altere as propriedades do motor, troque o nome (referência) para M1, 5V para a tensão nominal, e 122Ohm par a resistência interna.



O circuito.

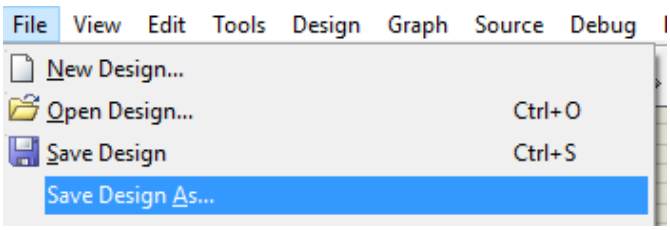


Abra a interface do Arduino, crie um programa novo, salve em uma pasta na área de trabalho com o nome PWM, ou outro se você preferir.

Altere o programa como mostra abaixo.

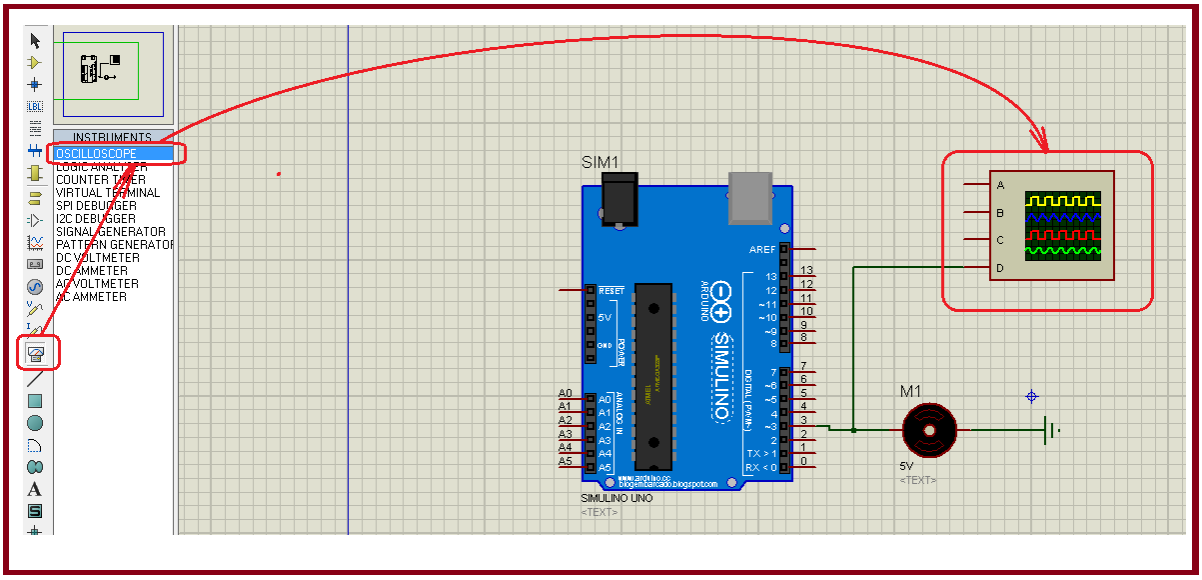
```
float velocidade;//velocidade do motor
void setup() {
  // put your setup code here, to run once:
  pinMode(3,OUTPUT);//saída 3 usada como PWM
}

void loop() {
  // put your main code here, to run repeatedly:
  velocidade=(50.0/100.0)*255.0;//Velocidade=50% do total
  analogWrite(3, velocidade);//liga o motor com 50
}
```

Salve o diagrama do ISIS usando a opção ”File/ Save Design as..”. Salve na mesma pasta do programa Arduino e altere o nome do diagrama para “PWM”.

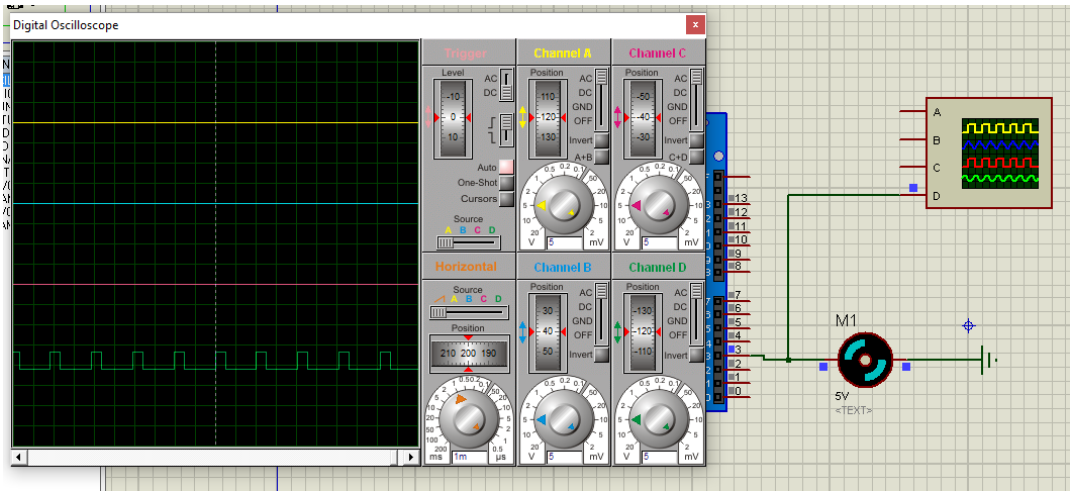


Compile, Exportar Binário compilado e teste e veja o motor girando.

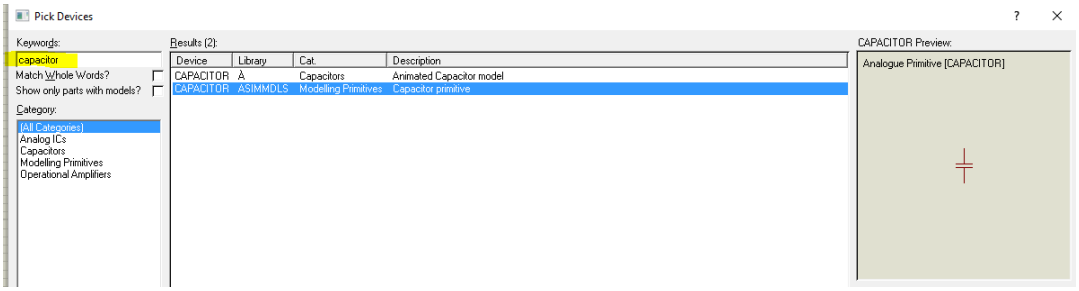
Insira um osciloscópio no diagrama para ver a forma de onda, para isto selecione o osciloscópio na aba do instrumento e conecte em paralelo com o motor e veja a forma de onda.



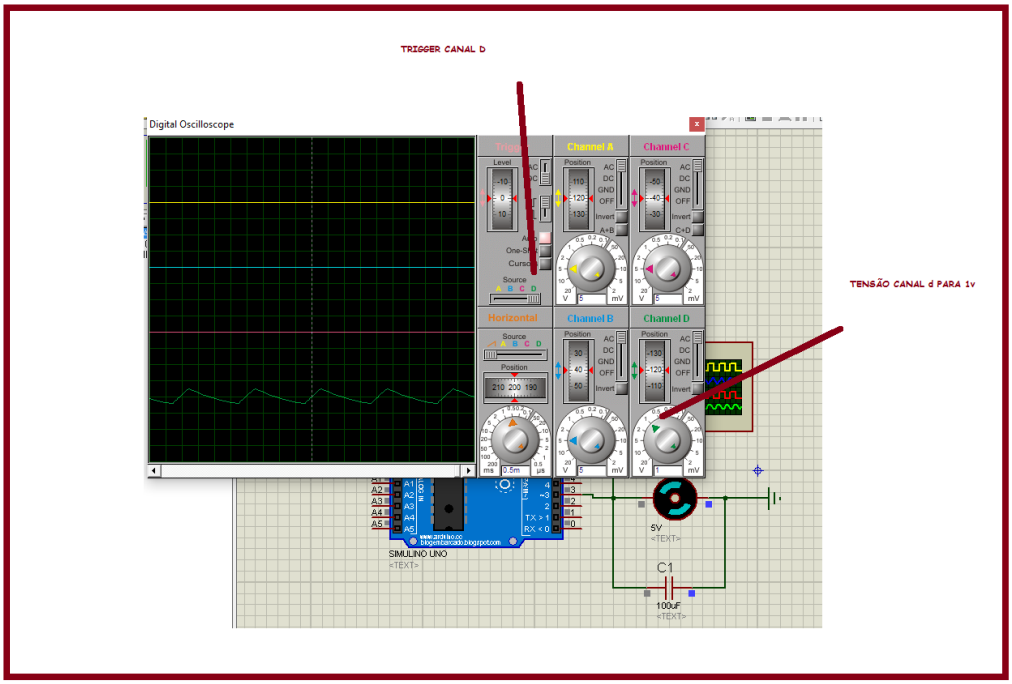
Observe a forma de onda verde no osciloscópio.



Monte um capacitor em paralelo com o motor, pegue o capacitor na livraria como descrito abaixo.



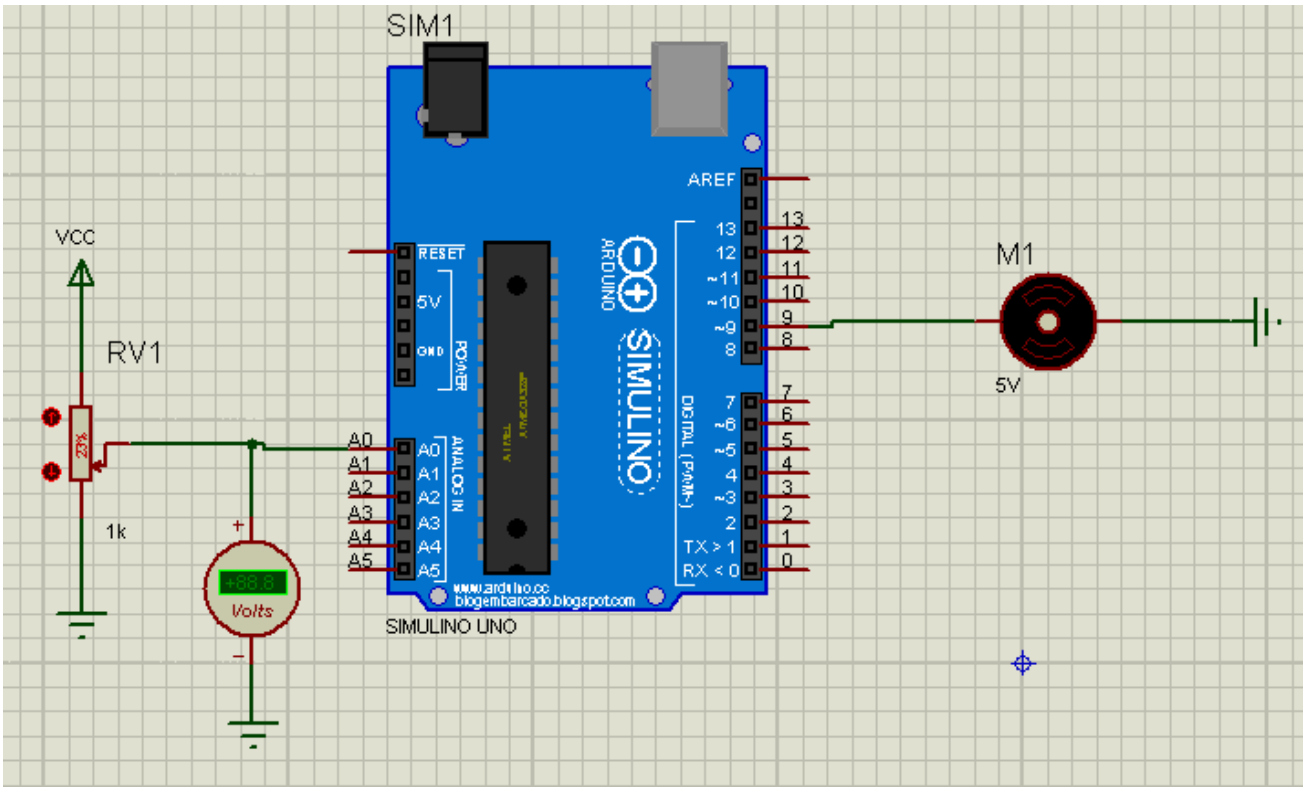
Altere o trigger e a tensão do canal como descrito na figura abaixo. Note a forma de onda mostrando a carga e descarga no capacitor.



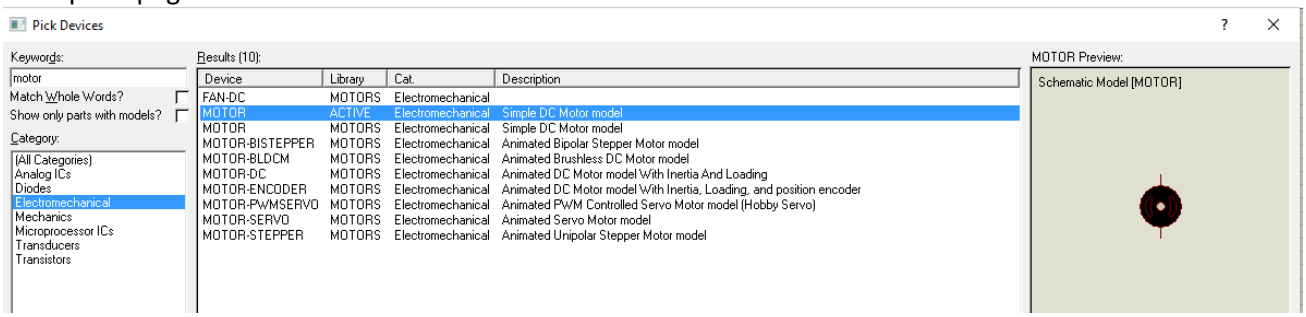
Exemplo de escrita e leitura analógica:

Neste exemplo a velocidade de um motor DC é controlada usando a saída PWM.
Crie uma pasta para o seu projeto.
Neste exemplo o pino usado para ligar o led é o pino 9 e o pino para pegar o valor analógico do potenciômetro é o pino 3 (A3), estes pinos são configurados no início do programa e no setup()!

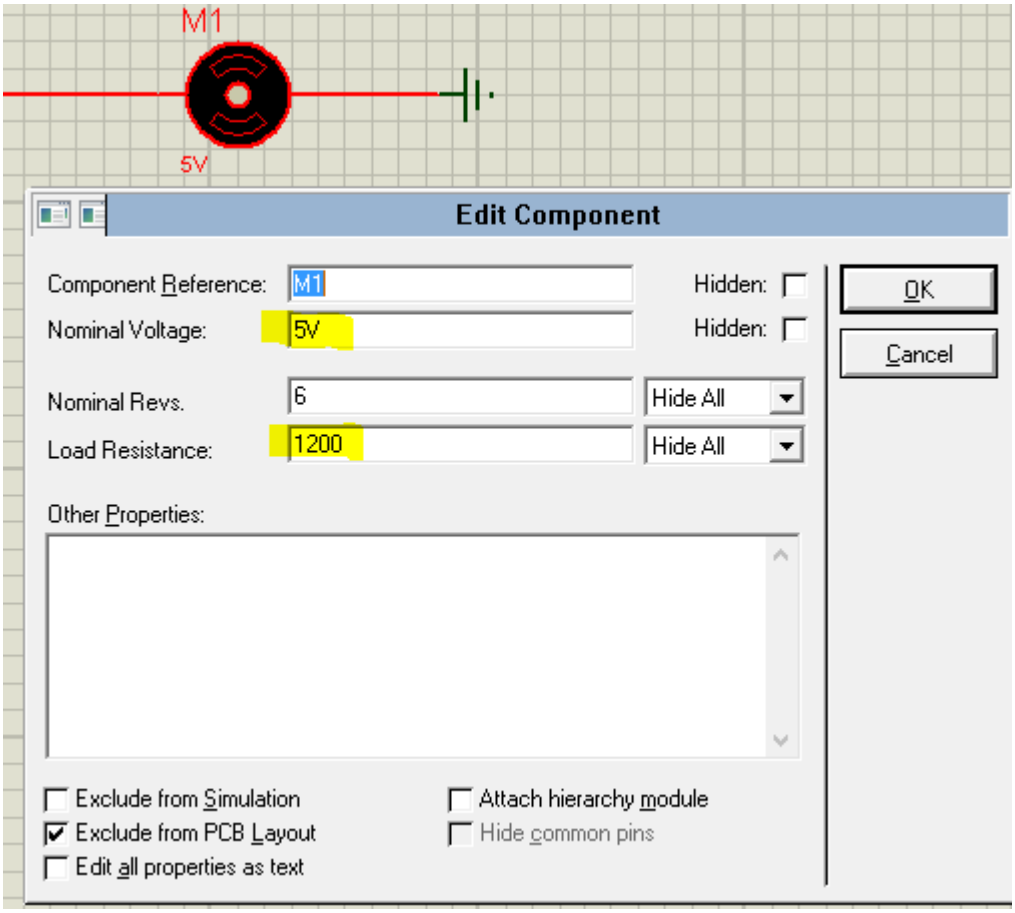
O circuito é mostrado abaixo:
Monte o circuito usando o programa ISIS do PROTEUS e salve na pasta do seu projeto, neste ciruciot será usado um ARDUINO UNO, um potenciômetro e um motor.



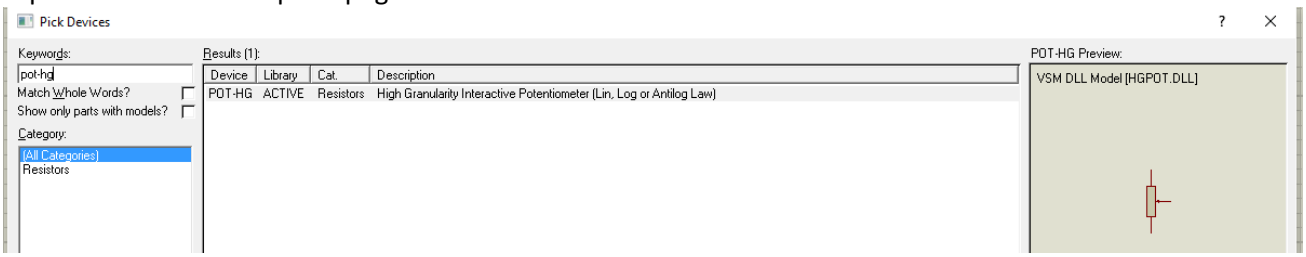
Você pode pegar o motor na biblioteca abaixo.



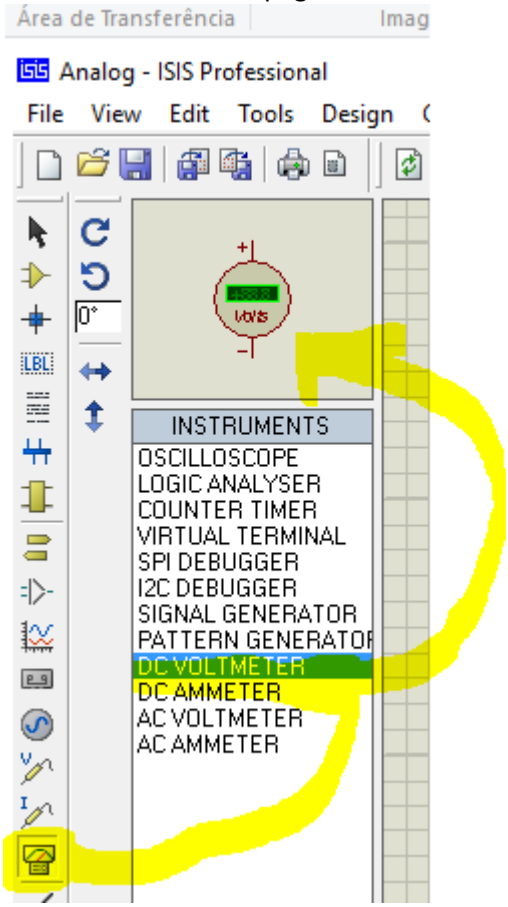
Ao inserir o motor no circuito não esqueça de alterar a tensão para 5V e a resistência de carga para 1200!



O potenciômetro você pode pegara na biblioteca “POT-HG” abaixo.

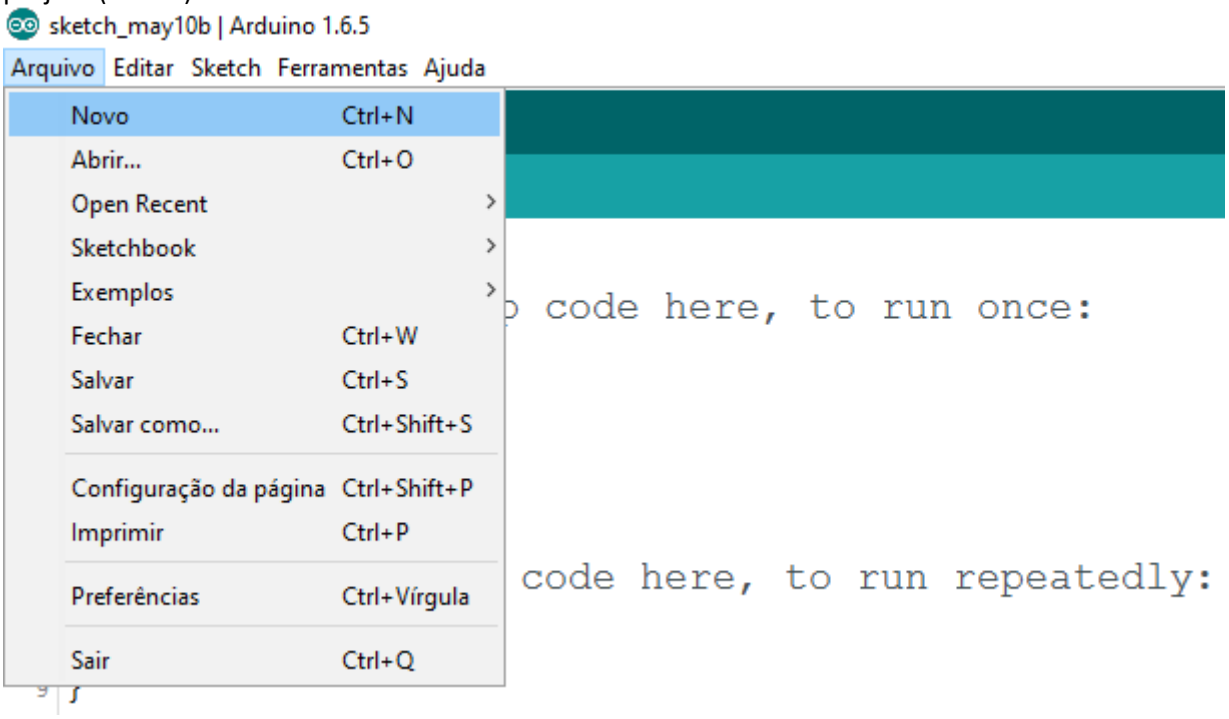


O voltímetro DC você pega no menu ”INSTRUMENTS”.

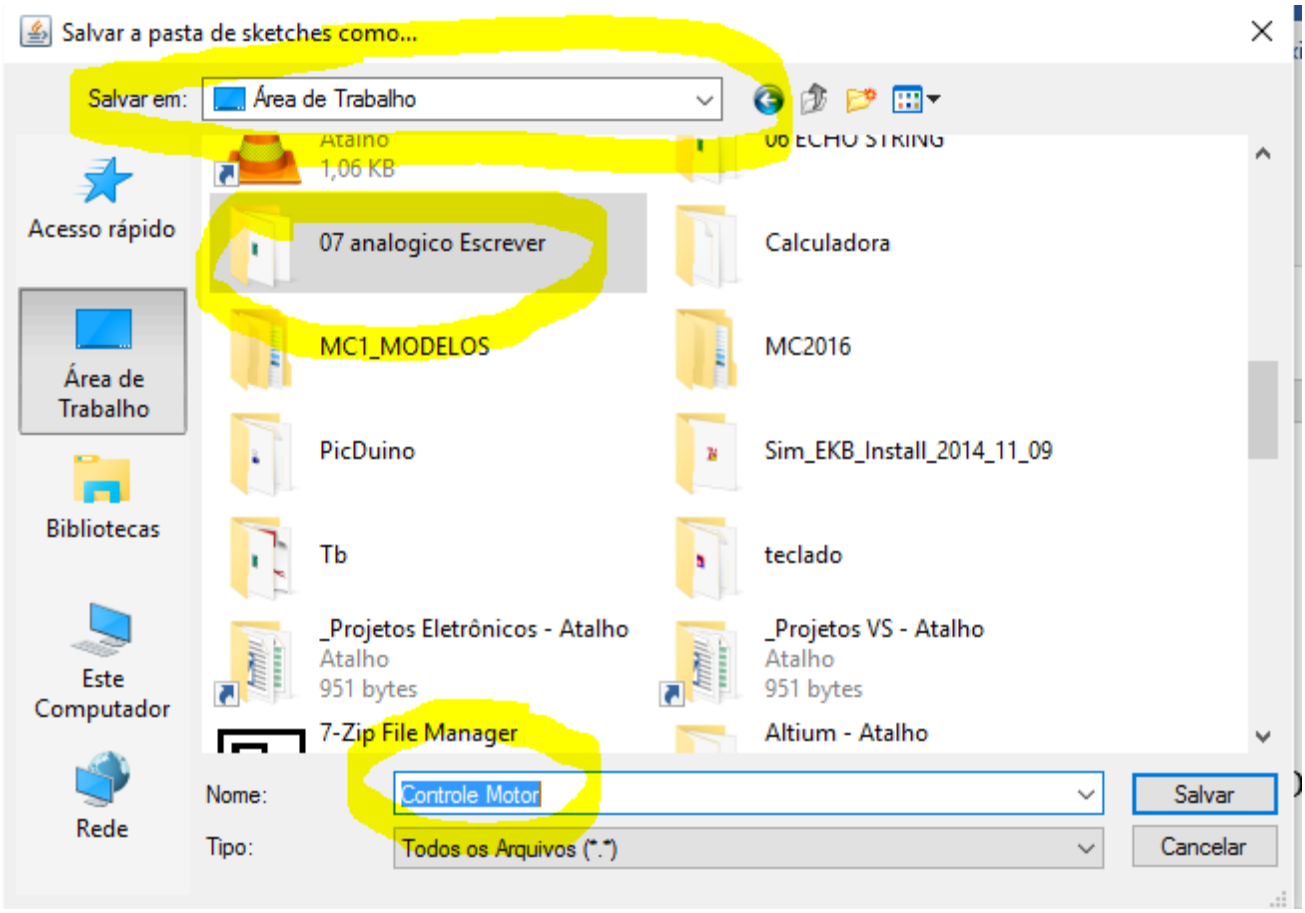


O programa é descrito abaixo.

Abra a interface do ARDUINO e verifique se foi aberto um novo projeto (sketch), se não foi crie um novo projeto (sketch) usando o menu abaixo.



Salve como na pasta criada par ao seu projeto alterando o nome, por exemplo, para “controleMotor”.



Escreva o programa abaixo na interface do ARDUINO.
Compile e Exporte como binário.

Para ajustar o valor da velocidade PWM ao valor da entrada você deve usar uma regra de três simples, como mostra a figura abaixo.

velocidade 255

analogA0 1023

velocidade= analogA0 * 255 /1023

Note que os valores da divisão podem dar números com casas decimais, por isto as variáveis velocidade e analogA0 devem ser declaradas como float.

Já as funções de entrada analógica e saída trabalham com números inteiros, por isto você deverá converter inteiros em float e float em inteiros.

Para fazer a conversão de inteiro para float basta escrever float () e dentro dos parênteses colocar a variável a ser convertida, isto é feito para pegar o valor analógico que é inteiro e deve ser convertido para float.

```
analogA0=float(analogRead(A0));
```

Já as operações matemáticas devem ser feitas usando variáveis do mesmo tipo, neste caso as variáveis são todas do tipo float.

```
velocidade=analogA0*(255.0/1023.0);
```

Para escrever na saída os parâmetros devem ser inteiros, se você colocar um parâmetro do tipo float o programa faz a conversão automática, mas para ficar bem claro é conveniente você colocar a conversão na própria instrução.

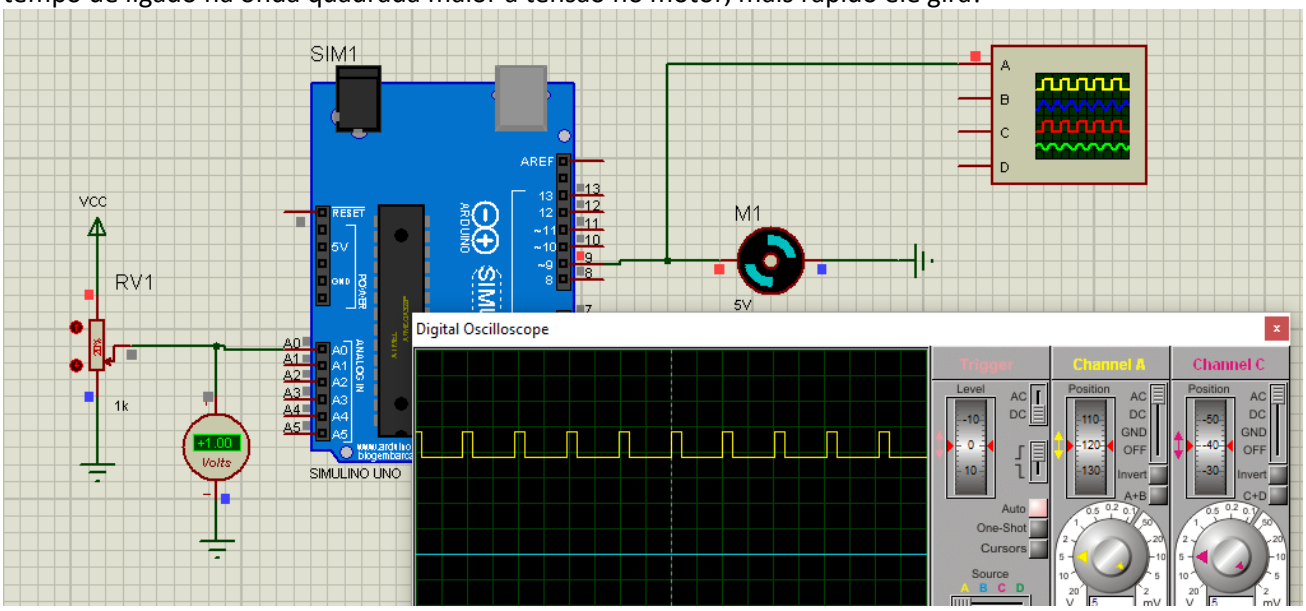
O programa completo é mostrado abaixo.

Controle_Motor §

```
//controle velocidade motor DC
int motor=9;//define o pino onde será ligado o motor
float velocidade=0;//variavel com a velocidade do motor.
float analogA0;//valor entrada analógica.
void setup() {
  // put your setup code here, to run once:
  pinMode(motor,OUTPUT);//define o pino PWM do motor como saída.

}

void loop() {
  // put your main code here, to run repeatedly:
  analogA0=float(analogRead(A0));//lê a entrada  onde está ligado o potenciômetro
  //regra de três velocidade=255 quando A0=1023
  velocidade=analogA0*(255.0/1023.0);// ajusta o valor pois o máximo da saída é 255 e da entrada é 1023
  analogWrite(motor,int(velocidade));//ajusta a velocidade do motor.

}
```

Carregue o programa no ARDUINO do simulador e rode.

Se você ligar um osciloscópio em paralelo com o motor poderá observar as ondas quadradas geradas pelo PWM, a relação ligado e desligado da onda quadrada irá gerar uma tensão média no motor, quanto mais tempo de ligado na onda quadrada maior a tensão no motor, mais rápido ele gira!



A instrução map()

A instrução map() monta a regra de três para ajustar o valor de uma entrada analógica a uma nova saída analógica proporcional a entrada.

Esta instrução só existe na biblioteca do Arduino.

A lógica usado a regra de três pode ser usada com qualquer microcontrolador!

Você pode ver abaixo como usar a instrução.

Int Val=map(value, fromLow, fromHigh, toLow, toHigh);

Onde Val é a variável analógica de saída do tipo inteiro, mas pode ser float também.

Onde value é a variável analógica de entrada.

Onde fromLow é o menor valor de entrada.

Onde fromHigh é o miaor valor de entrada.

Onde toLow é o menor valor de saída.

Onde toHigh é o maior valor de saída.

Exemplo.

Exemplo do uso da função map.

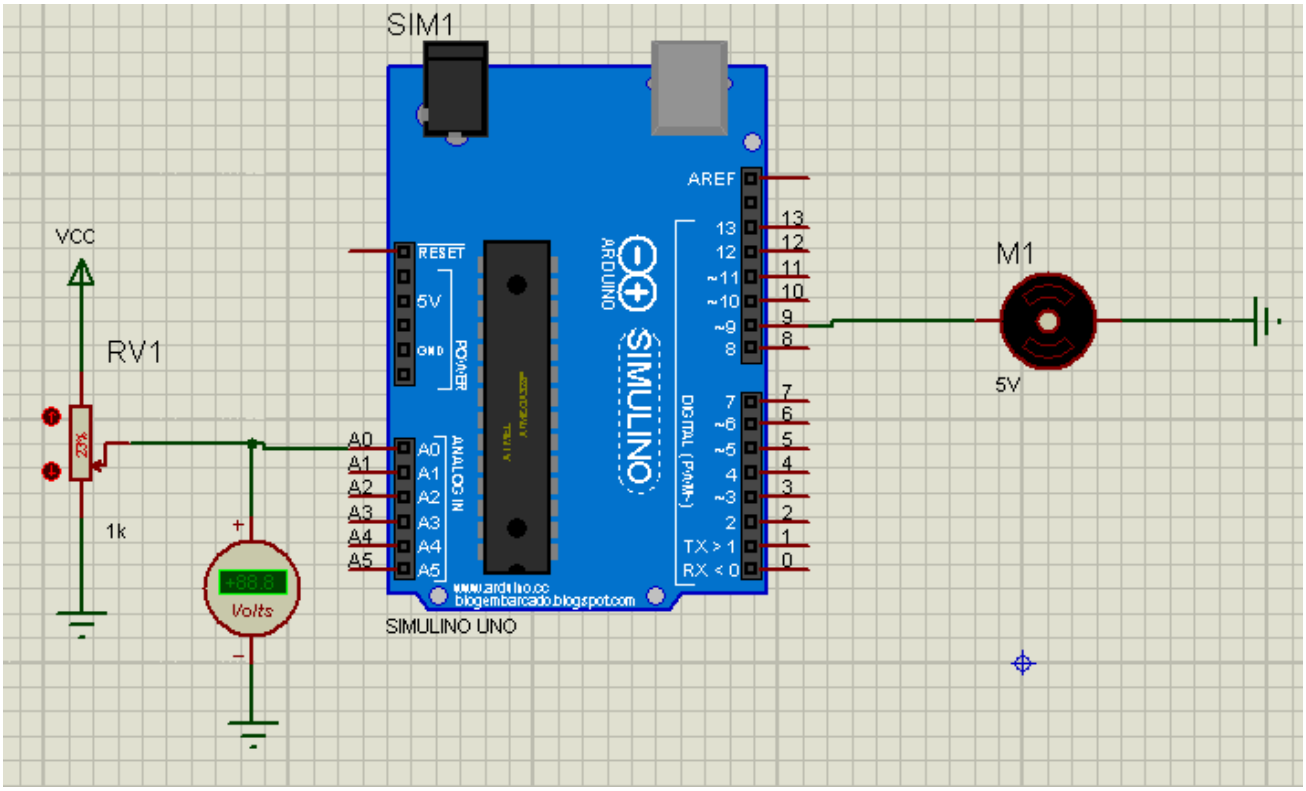
```
int velocidade=map (analogA0, 0,1023,0,255);
```

Note o sinal de igual, a variável de saída (velocidade) recebe o valor ajustado da variável de entrada analogA0!

Escreva o exemplo anterior usando a função map() e veja como ficou mais simples.

Monte o circuito abaixo que é o mesmo do capítulo anterior.

Não esqueça de alterar a propriedade do “Load Resistance” do motor para 122Ohm!



O programa completo é mostrado abaixo observando a instrução map() para gerar a regra de três simples!.

Controle_Motor_map §

```
1  //controle velocidade motor DC
2  int motor=9;//define o pino onde será ligado o motor
3  void setup() {
4    // put your setup code here, to run once:
5    pinMode(motor,OUTPUT);//define o pino PWM do motor como saída.
6
7  }
8
9  void loop() {
10   // put your main code here, to run repeatedly:
11   int analogA0=analogRead(A0);//lê a entrada  onde está ligado o potenciômetro
12   //regra de três velocidade=255 quando A0=1023
13   // ajusta o valor pois o máximo da saída é 255 e da entrada é 1023
14   int velocidade=map (analogA0, 0,1023,0,255);
15   analogWrite(motor,velocidade);//ajusta a velocidade do motor.
16 }
```

Reforçador de saída.

A saída do microcontrolador é muito baixa potência, não deve ser usada para acionar motores, para isto você deve usar um reforçador de sinal.

O transistor é o mais usado como reforçador, você pode usar tanto o transistor bipolar ou o transistor MOSFET, o tipo depende da corrente do motor.

Neste tipo de circuito o transistor deve ser polarizado como chave.

O transistor TIP122 é o mais prático, pode ser usado nas aplicações até 3A, mas a corrente máxima é de 5A. Você deve ter o cuidado de colocar um dissipador de calor para correntes maiores do que 1A. O transistor TIP122 é prático porque é do tipo Darlington com ganho da ordem 1000, podendo ser acionado com baixas correntes.

Como exemplo modifique o exemplo anterior como descrito abaixo.

O resistor R1 serve para limitar a corrente da base, seu valor é função da corrente no coletor, no exemplo é a corrente no motor.

Uma forma prática de determinar o valor da resistência R1 para que o transistor opere como chave é considerar o ganho do transistor igual a 10% do ganho real, assim para o transistor TIP122 operar como chave considere o ganho igual a 100.

A corrente de base é igual a corrente máxima do microcontrolador que na prática é 10mA.

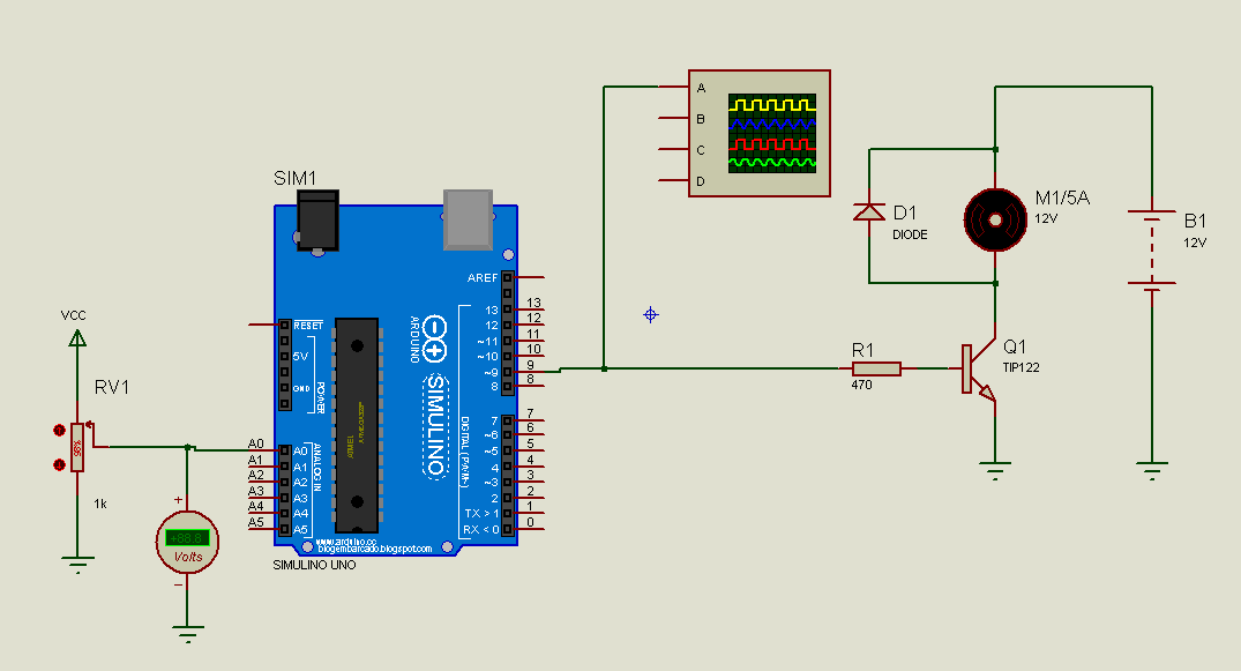
Sabendo a corrente de base e a tensão na saída no microcontrolador é possível calcular o valor do resistor R1. Neste cálculo a tensão VBE foi considerada de 0,7V, para o TIP122 esta tensão é um pouco maior, mas para simplificar vamos manter 0,7V.

$$R_1 = \frac{V_{CC} - V_{BE}}{I_B} = \frac{5V - 0{,}7V}{10mA} = 430\Omega$$

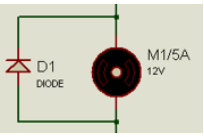
O valor comercial mais próximo é 470 Ohm!

Com este valor de resistor você pode usar transistores do tipo Darlington de ganho 100 até 5A e transistores de ganho 100 como o BC547 até 50mA!

O circuito é mostrado na figura a seguir.



Observe o diodo colocado em paralelo com o motor, este diodo é montado invertido com o catodo para o positivo e serve como supressor de ruído evitando que apareçam tensões elevadas quando o motor é desligado, se você esquecer este diodo o transistor e até mesmo o microcontrolador pode queimar.

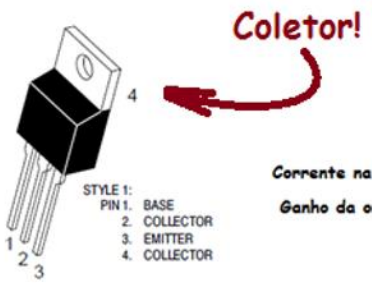


Depois de montar o circuito, use o mesmo programa do Arduino do exemplo anterior, só coloque a rodar, já que só foi alterado o circuito!

Os detalhes do transistor TIP122 é mostrado na figura abaixo.

Observar que o coletor é o pino do meio e está ligado ao metal do dissipador, por isto, não aterre o dissipador!

TIP 121/ 120/ 122
NPN



Corrente na carga entre 300mA e 3A.
Ganho da ordem de 1000

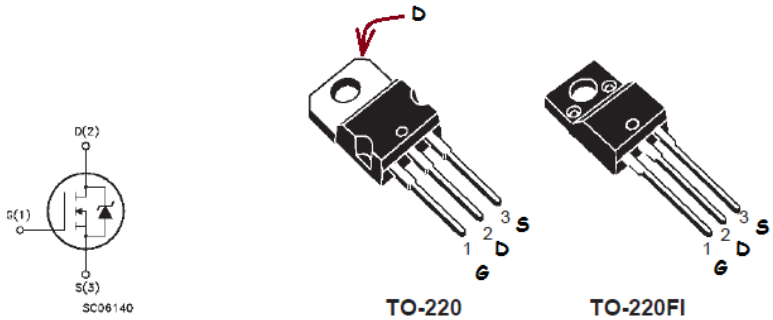
*MAXIMUM RATINGS

Rating	Symbol	TIP120,	TIP121,	TIP122,	Unit
Collector–Emitter Voltage	V_{CEO}	60	80	100	Vdc

Você também pode usar um MOSFET, é mais moderno e não tem que calcular a resistência de base, já que o MOSFET opera com tensão, basta você colocar uma tensão igual ou maior do que 3V no gate que ele liga, por precaução é usado um resistor de alto valor no gate, qualquer resistor maior do que 1K pode ser usado, na prática 10K é o mais usado.

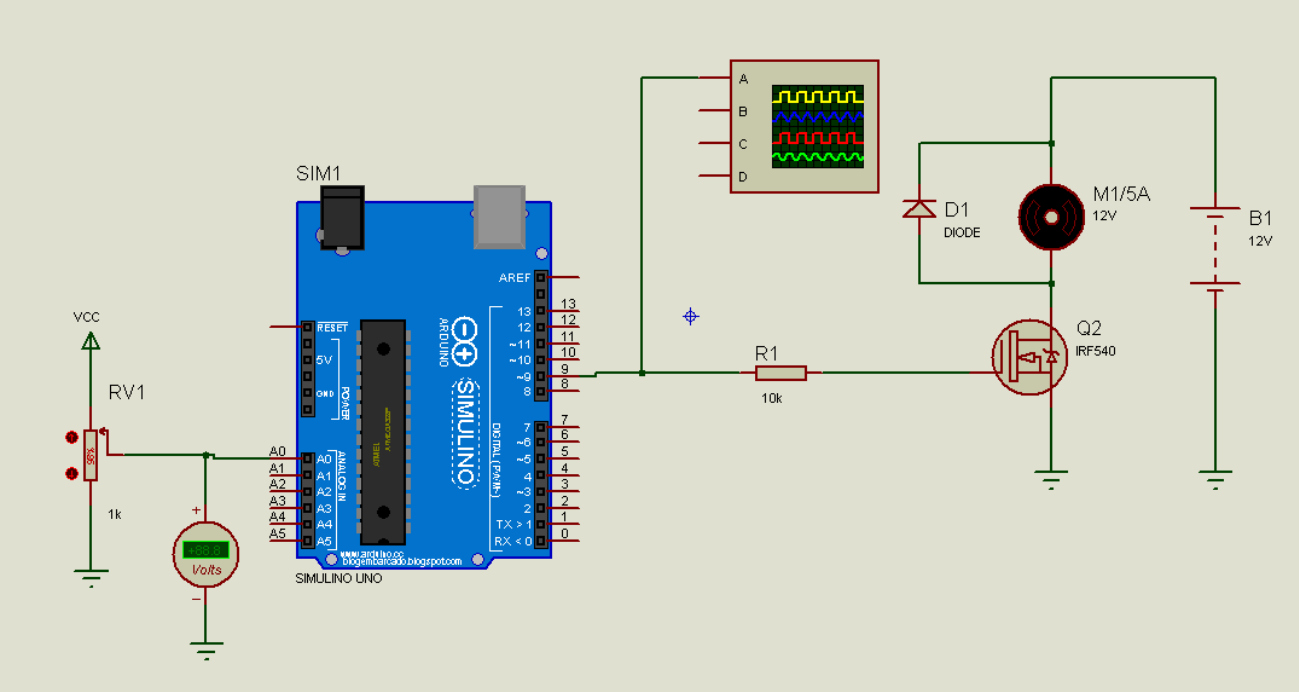
Você só tem que ter cuidado com o MOSFET que é muito sensível a tensões estáticas no gate, evite pegar o transistor pelos terminais, se não colocar o diodo supressor em paralelo com o motor pode ser fatal para o transistor!

O MOSFET mais usado é o IRF540 que pode trabalhar até 10ª o detalhe deste transistor é mostrado abaixo.



IRF 540

Como exemplo o circuito do exemplo anterior pode ser modificado como é mostrado na figura abaixo. Monte o circuito e teste, não precisa alterar o programa, note que você deve trocar o transistor e alterar o valor do resistor, não esqueça de manter o diodo D1, não precisa calcular nada, para qualquer corrente de saída o resistor R1 será sempre10K Ohm!



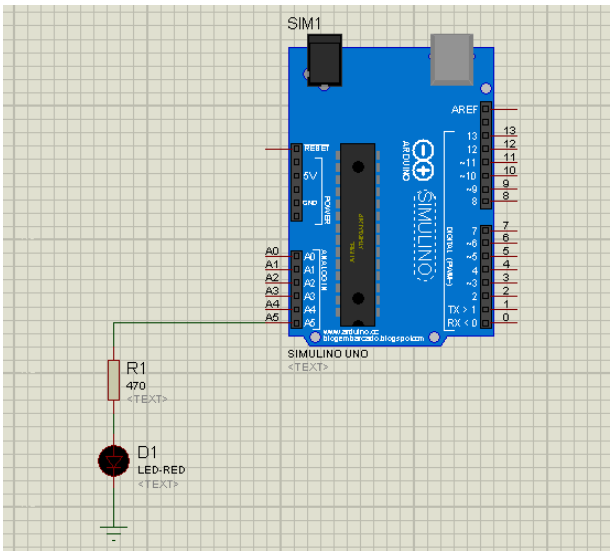
Usando uma porta analógica como digital.

Você pode usar uma porta analógica com entrada ou saída digital para ampliar o número de portas disponíveis, principalmente no UNO!

A aplicação é muito simples, basta definir no setup o endereço da porta como A0, A1, A2 etc. conforme o pino a ser usado, a para escrever na porta use a função da escrita digital.

Como exemplo será mostrado como alterar o programa BLINK (pisca-pisca) para usar a porta analógica A5 como saída digital, para isto crie uma pasta para o seu projeto, monte o circuito abaixo no ISIS e depois na interface do ARDUINO abra o programa Exemplo/01Basic/ Blink como foi feito no exemplo do pisca-pisca, depois altere o programa como mostrado abaixo.

O circuito é mostrado abaixo.




```
/*
  Exemplo do uso de uma porta analógica como digital
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(A5, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(A5, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(A5, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Desafio 5:

DESAFIO 5

Altere o programa de controle da velocidade do motor de forma a mostrar no monitor virtual a velocidade do motor!

Altere o programa com descrito abaixo:

Altere a função pinMode alterando o endereço para a porta analógica A5.

Alterena função sigitalWrite o endereço da porta a ser acionada para A5!

Ampliando o conceito da instrução if()!

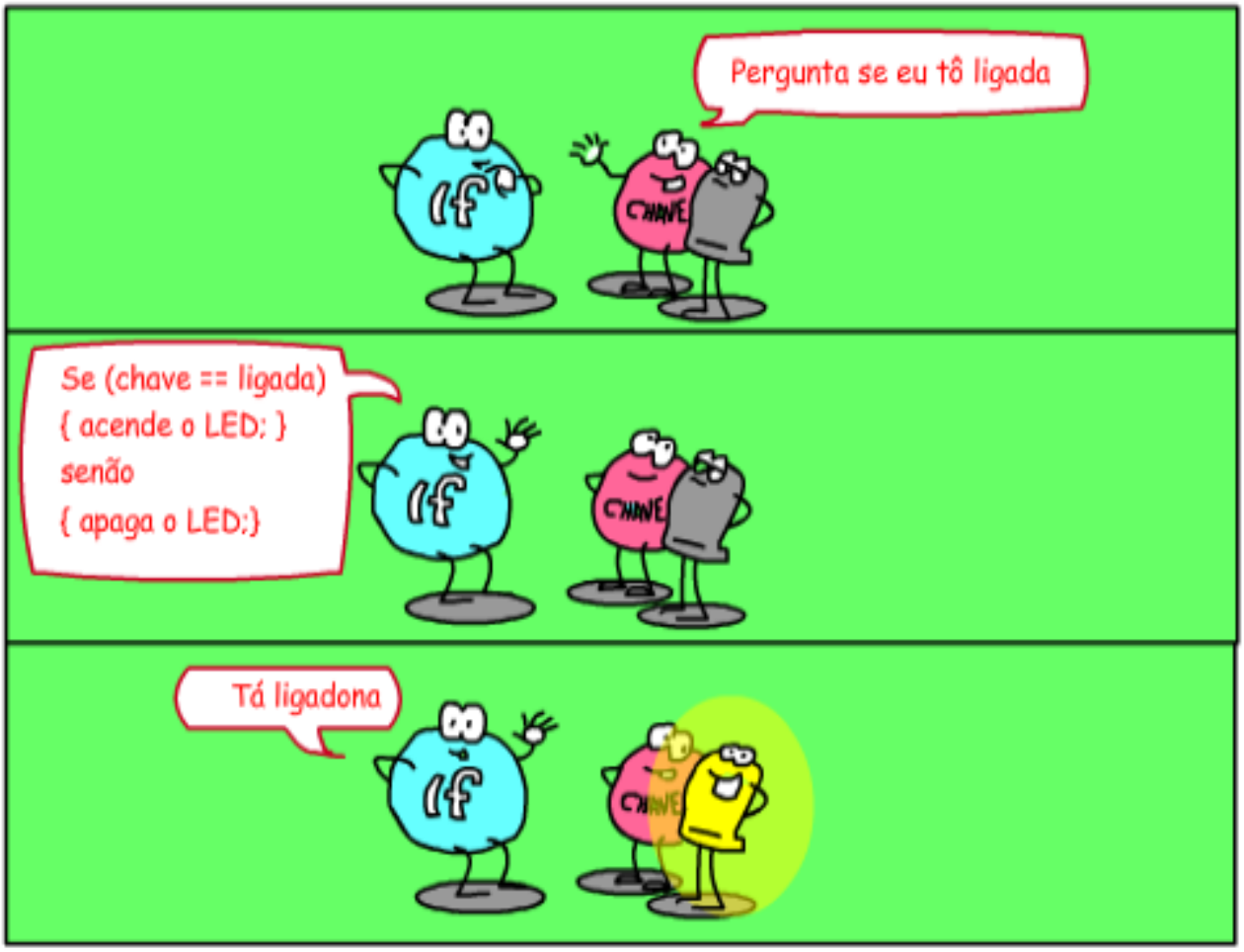
Você já conhece a instrução “if “e sabe da importância desta instrução neste capítulo você irá ampliar o seu conhecimento sobre esta função!

A função “if” responde uma pergunta lógica que só pode ter duas respostas: Verdade ou falso.

O compilador interpretado o verdadeiro como maior do que zero e o falso como igual a zero.

Para o número maior do que zero normalmente é usado o número 1.

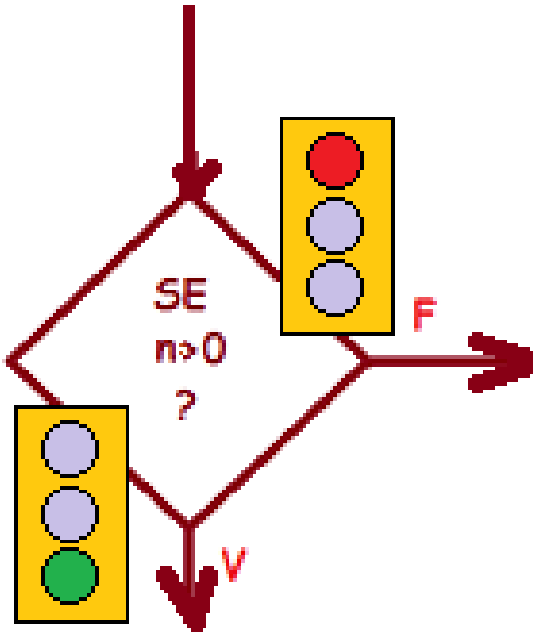
Em eletrônica uma entrada ligada manda par ao compilador o valor 1 e uma entrada desligada manda o valor 0!



A função do tipo if() é chamada de desvio condicional, isto é, o programa desvia o seu fluxo de processamento sequencial a partir do teste de uma condição.

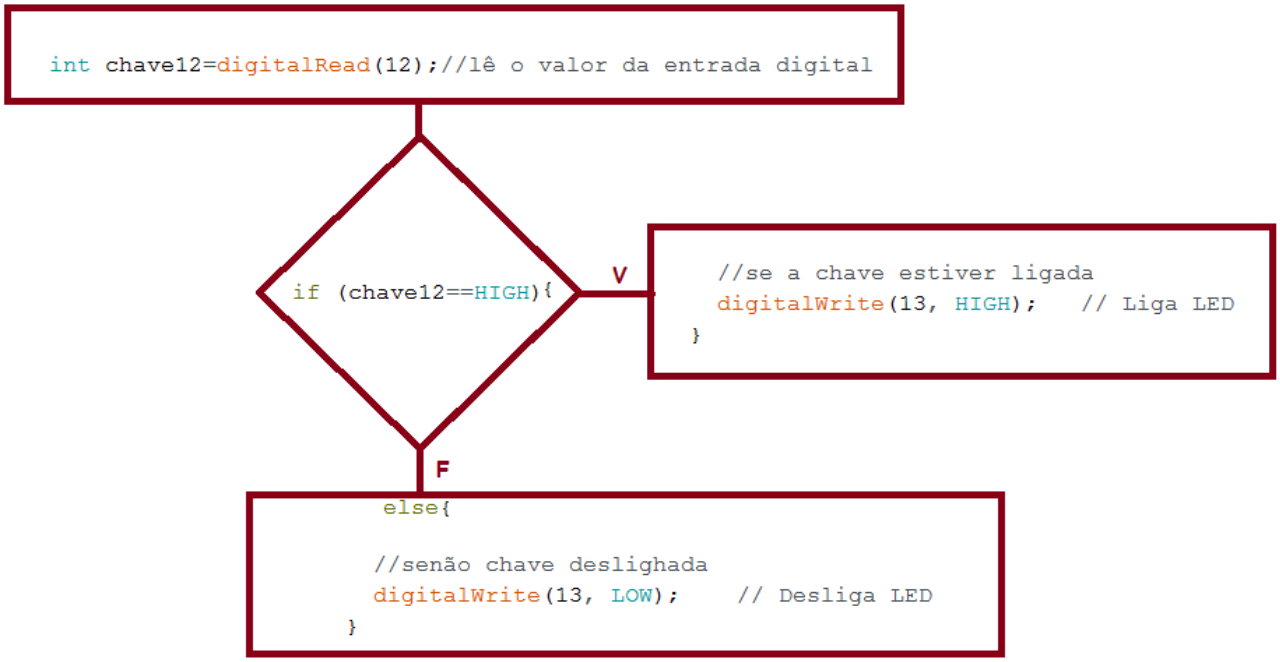
O retorno do teste é do tipo digital podendo voltar somente dois valores:

Zero (condição Falsa) ou maior do que zero (condição Verdadeira)!



A condição deve ser colocada dentro dos parênteses depois do if()!

Se a condição for verdadeira a sequência de instruções dentro da primeira chave depois do IF é executada, senão, a sequência dentro das chaves depois do ELSE será executada. Isto pode ser descrito através de um fluxograma como descrito abaixo.



Teste condicional?

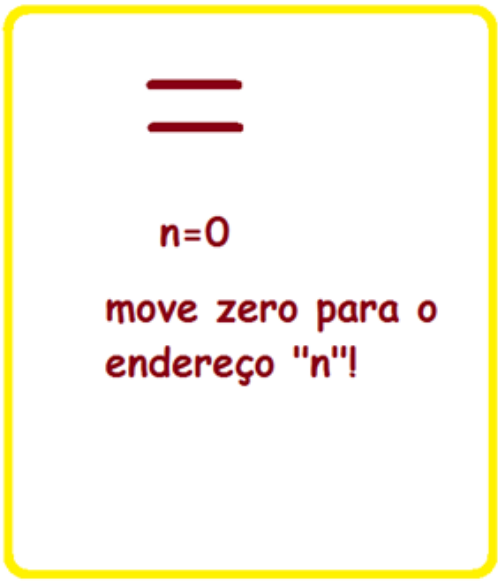
Existem duas formas de montar o teste condicional:

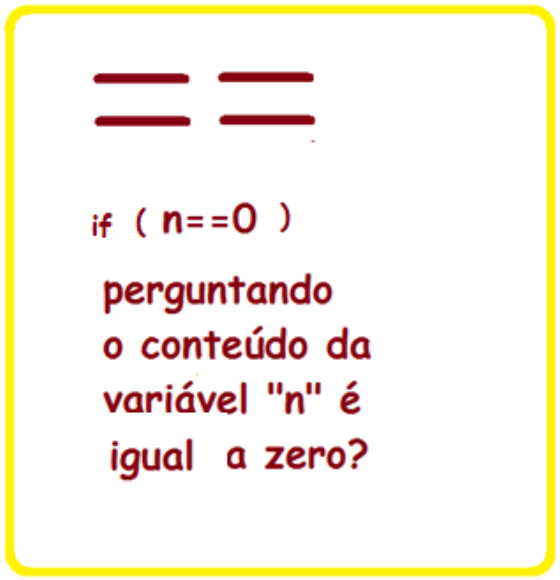
- Com operadores Relacionais.
- Com operadores Lógicos.

Os operadores Relacionais são operadores que relacionam números:

Operador	Ação
>	Maior
>=	maior ou igual
<	Menor
<=	menor ou igual
==	Igual
!=	não igual (diferente)

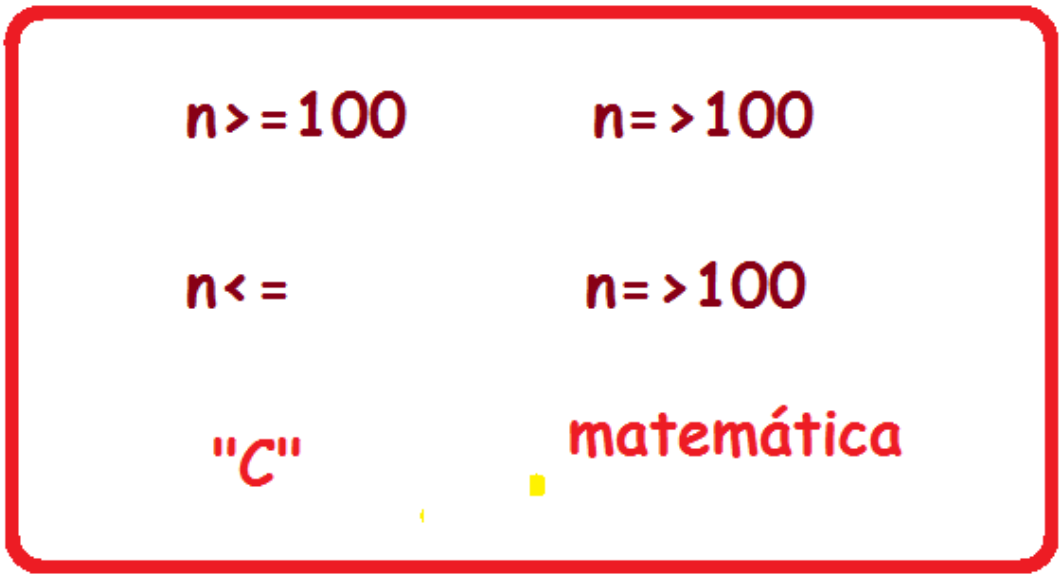
Observe que o operador relacional IGUAL (==) é diferente da atribuição igual (=) que move o valor para uma variável!



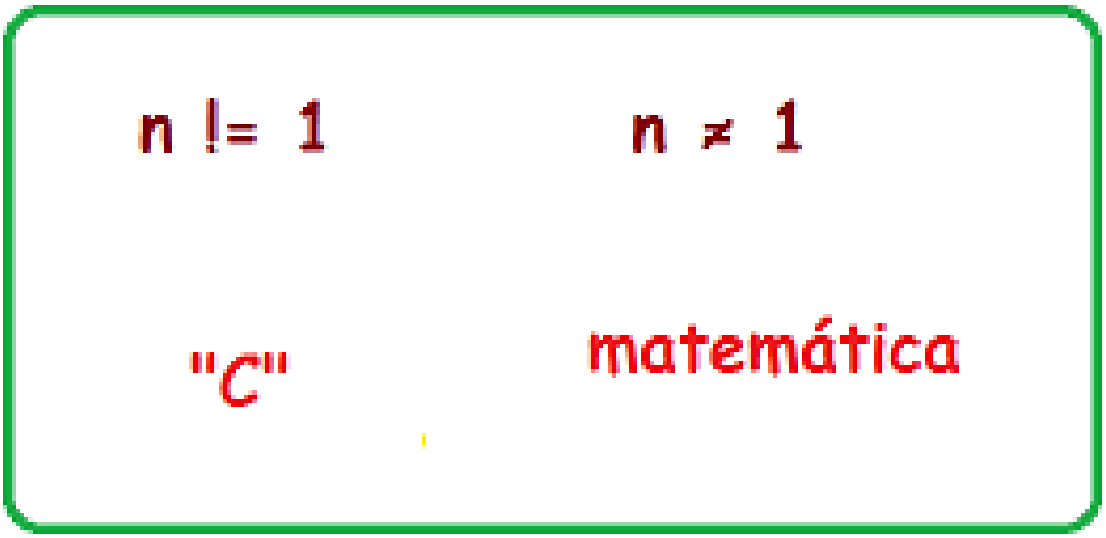


Observe que a forma de escrever o maior igual ou menor igual é diferente da forma de escrever normal na matemática.

Mas não se preocupe, se você inverter o compilador irá detectar o erro para você corrigir!



Observe que a forma de perguntar “se é diferente” é bem diferente da matemática!



E se não tiver operador dentro da condição?

Neste caso o valor numérico da variável ou expressão dentro da condição será usado na resposta, se este valor for maior do que zero será considerado verdadeiro, senão (se for igual a zero) será considerado falso!

Na instrução abaixo a pergunta sem operador equivale a “if(chave12==HIGH)”!

```
int chave12=digitalRead(12);//lê o valor da entrada digital
if (chave12){
  //se a chave estiver ligada
  digitalWrite(13, HIGH);   // Liga LED
}
 else{

  //senão chave deslighada
  digitalWrite(13, LOW);    // Desliga LED
}
```

Exemplo:

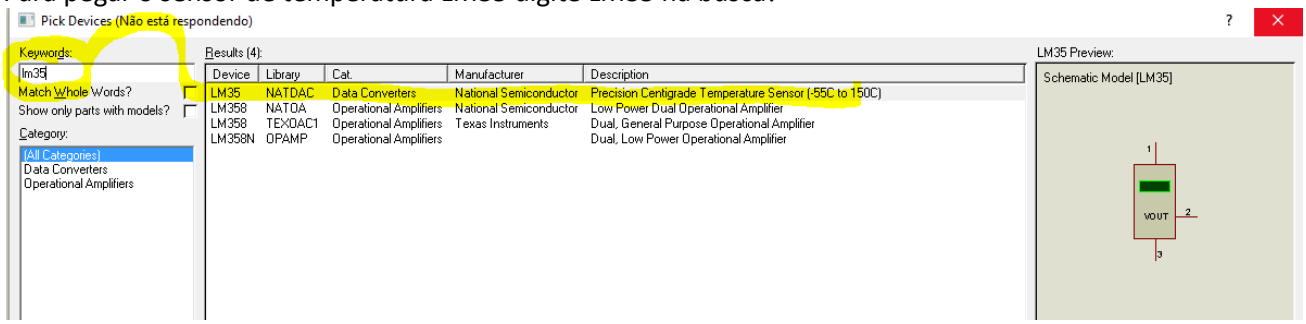
Você usa valores relacionais para comparar números.

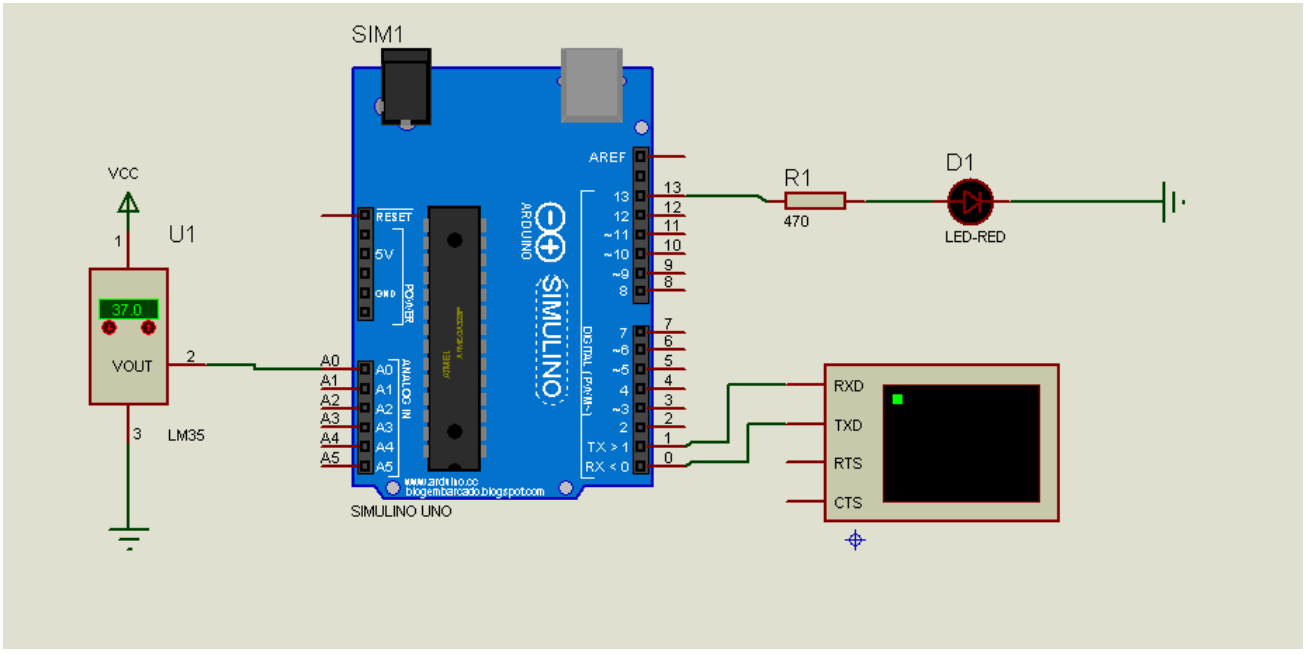
Para mostrar o uso dos comparadores trabalhe o exemplo.

No exemplo abaixo a entrada analógica é um sensor de temperatura LM35 onde 100°C gera 1V na entrada analógica.

O programa compara a entrada convertida em graus com o valor 38 graus, se o valor for maior ou igual o LED acende indicando que o paciente está com febre!

Montar o circuito abaixo.
Para pegar o sensor de temperatura LM35 digite LM35 na busca!





Para medir a temperatura de forma simples você pode usar o LM35 que converte a temperatura em nível de tensão de zero a 5V, a figura abaixo mostra este CI.

Note que quando a temperatura for 100 graus centígrados a saída do CI irá mostrar o valor de 1V este é o valor padrão.

Fazendo uma regra de três simples, o CI irá mostrar o valor máximo de 5V quando a temperatura for 500 graus.

Na prática este CI deve ser usado somente até 150 graus!!!

Typical Applications

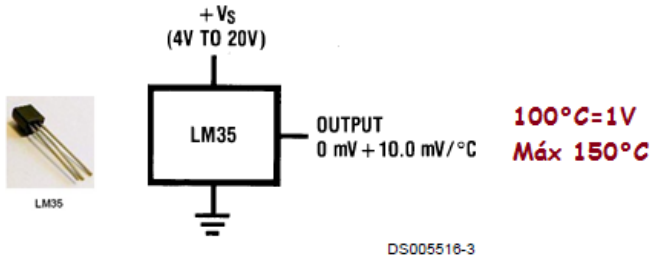
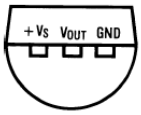


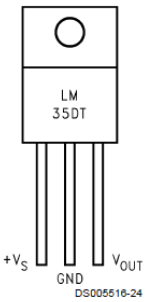
FIGURE 1. Basic Centigrade Temperature Sensor

TO-92
Plastic Package



Order Number LM35CZ,
LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

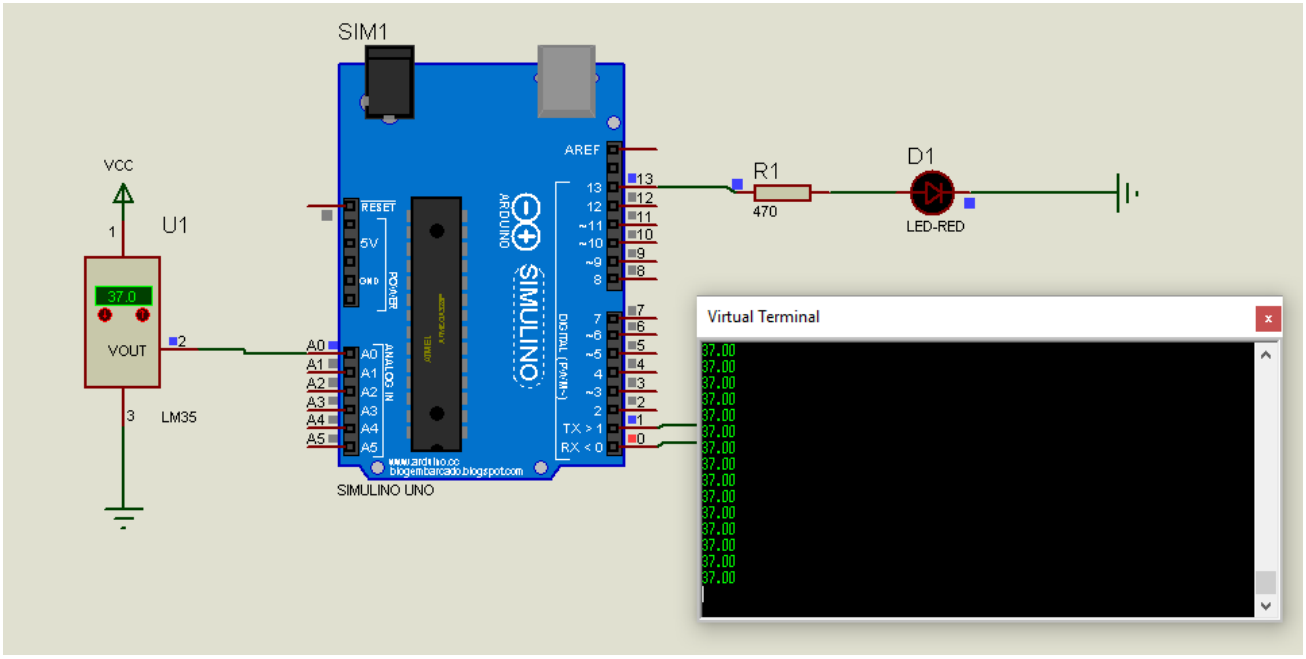
Order Number LM35DT
See NS Package Number TA03F

No Arduino monte o programa abaixo.

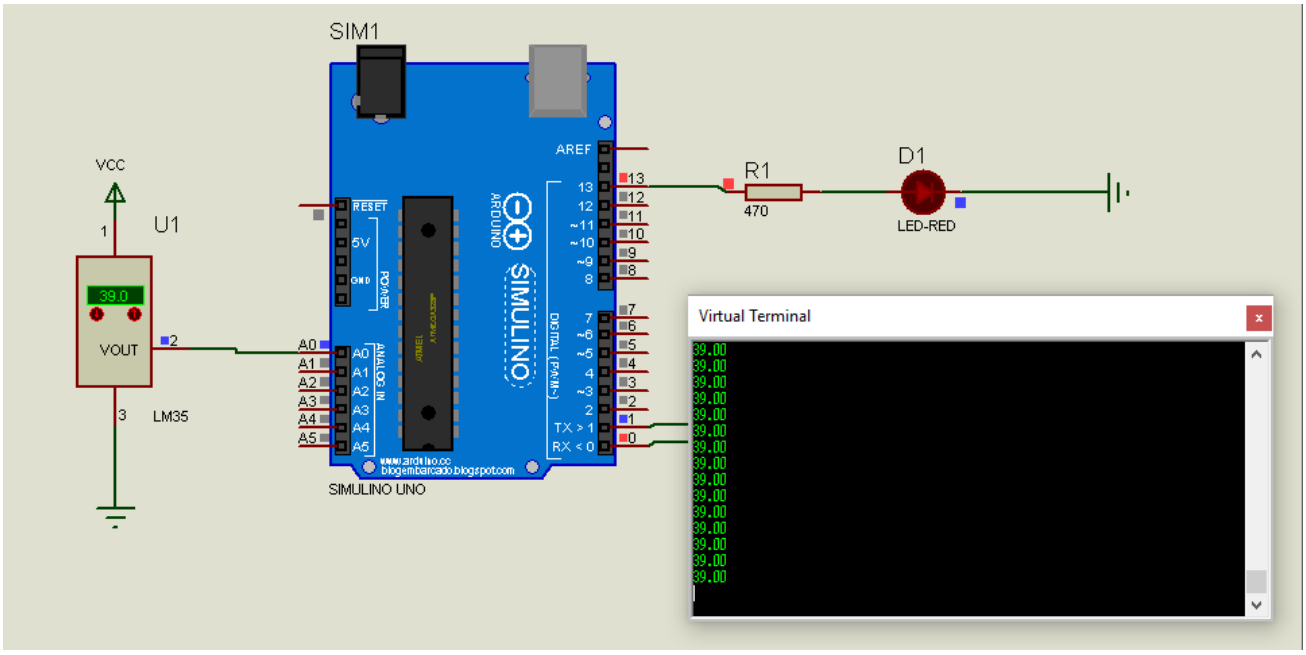
A inclusão do terminal serve para mostrar o valor da temperatura.
Note que no if a temperatura deve ser colocada com a parte decimal para que a comparação seja feita usando a variável do tipo float!
Observe que as variáveis temperatura foi declarada como float (número com parte decimal)!

```
//Usando o comparador relacional
int led=13;
// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT);//saida do LED
  Serial.begin(9600);
}
void loop() {
  int analogA0 = analogRead(A0);//Lê a entrada onde está ligado o potenciômetro
  // ajusta o valor para mostrar a temperatura
  float temperatura=map (analogA0, 0, 1023, 0, 500);
  //liga o led se temperatura maior do que 38 FEBRE
  if (temperatura>38.0){
    digitalWrite(led, HIGH);//liga LED
  }
  else{
    digitalWrite(led, LOW);//liga LED
  }
  Serial.println(temperatura);//mostra o valor da  tensaõ via serial
  delay(1);        // delay  para estabilizar
}
```

O Resultado é mostrado abaixo.
LED apagado.



LED aceso.



Comparação com operadores lógicos:

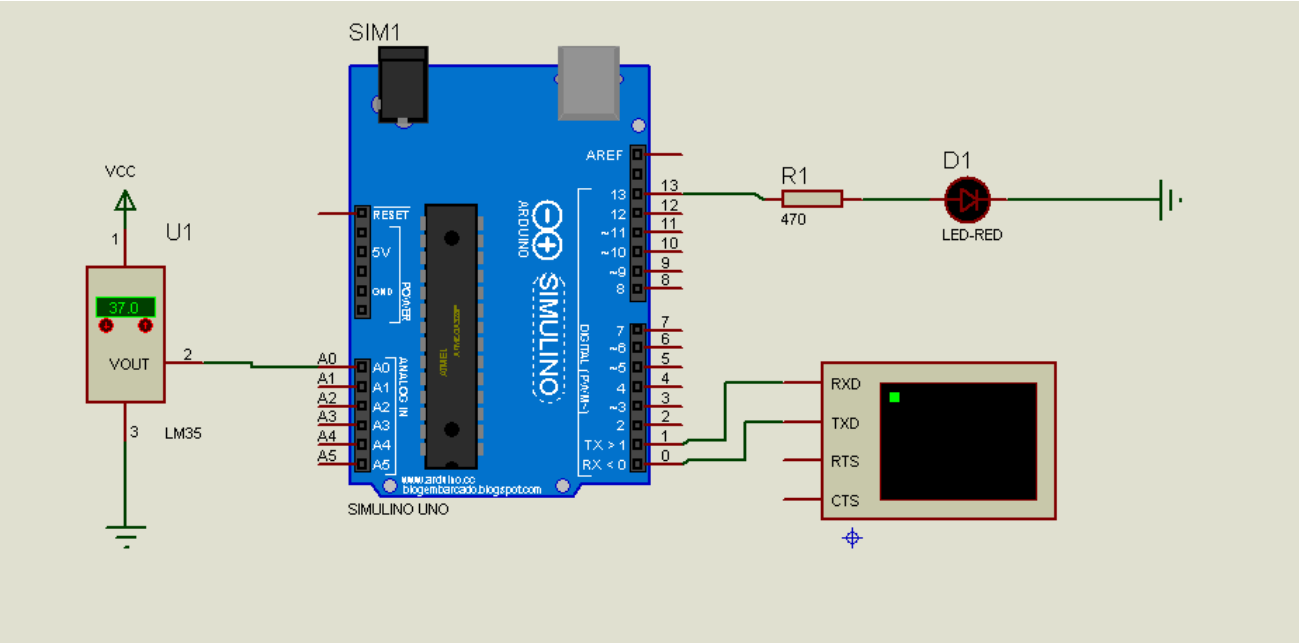
Você pode usar a comparação com variáveis que expressão uma função lógica cujo valor pode ser zero ou um, mais precisamente zero ou maior do que zero!

A tabela abaixo mostra como escrever estes operadores lógicos!

Operador	Ação
&&	operação AND
||	operação OR
!	operador de negação NOT (operador unário)

A minha sugestão é que você sempre use as palavras chaves “E”, “OU”, “NÃO”, “LIGAR”, “DESLIGAR”, “LIGADO” e “ DESLIGADO” ao explicar o funcionamento da máquina!

Como exemplo vamos aplicar este conceito ao circuito abaixo usado no exemplo anterior para medir a temperatura.



Agora altere o programa de forma a acender o LED quando a temperatura for maior do que 38 graus ou menor do que 20 graus.

Note que usando a palavra “OU” ficou bem claro o que eu quero.

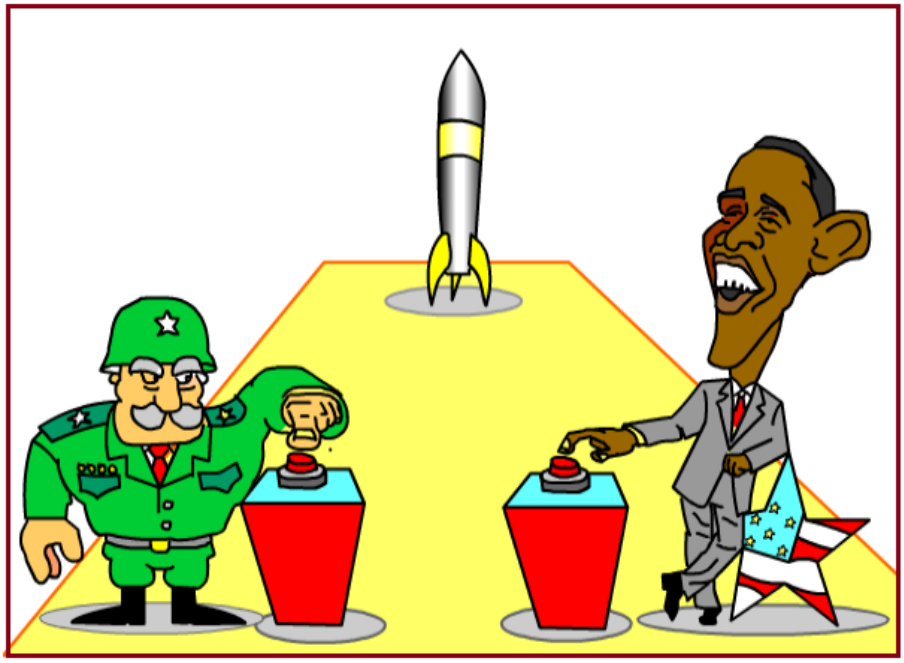
Uma possível solução é mostrada abaixo.

Observe em amarelo o uso da função lógica “OU”, note que você deve escrever o nome da variável e a condição duas vezes!

```
//Usando o comparador relacional
int led=13;
// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT);//saida do LED
  Serial.begin(9600);
}
void loop() {
  int analogA0 = analogRead(A0);//Lê a entrada onde está ligado o potenciômetro
  // ajusta o valor para mostrar a temperatura
  float temperatura=map (analogA0, 0, 1023, 0, 500);
  //liga o led se temperatura maior do que 38 FEBRE
  if (temperatura>38.0 || temperatura <20.0){
    digitalWrite(led, HIGH);//liga LED
  }
  else{
    digitalWrite(led, LOW);//liga LED
  }
  Serial.println(temperatura);//mostra o valor da  tensaõ via serial
  delay(1);        // delay  para estabilizar
}
```

Exemplo usando chaves!

Se alguém pede para você fazer uma máquina que dispare um míssil, mas para disparar existem dois botões, um com o presidente e outro com o general, somente quando os dois apertarem é que o míssil pode ser disparado!

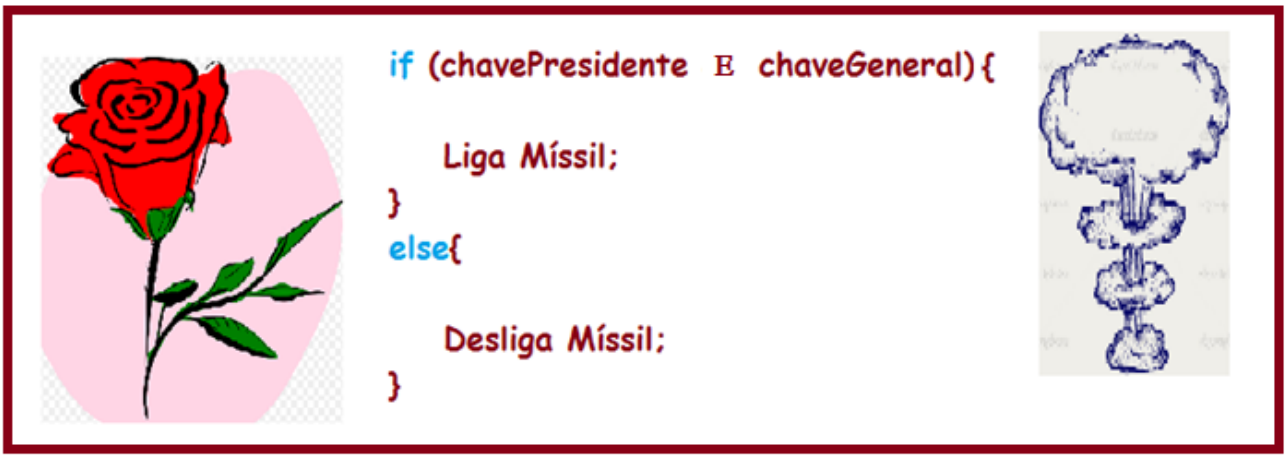


Parece claro para você, mas não está claro para o microprocessador!

Você deve descrever o programa de forma que possa ser traduzido para a linguagem do microcontrolador, linguagem usando lógica digital!

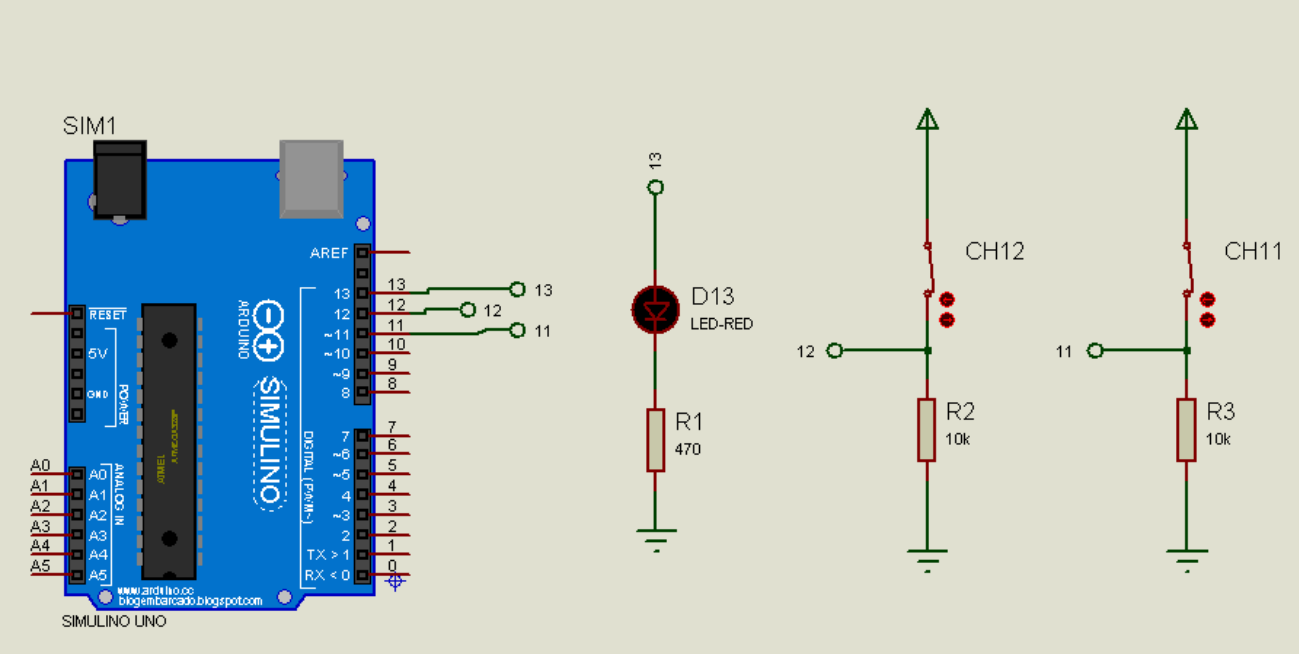
Para o microprocessador fica melhor falar assim:

Se a chave do presidente estiver ligada “E” a chave do general estiver ligada, então o míssil liga, senão o míssil desliga.



Para exemplificar monte o circuito abaixo para testar as condições lógicas.

Crie uma pasta para guardar o seu projeto, monte o diagrama abaixo e salve na pasta do seu projeto.



Neste primeiro exemplo você vai criar um programa que ligue o LED quando a chave CH13 “E” CH11 estiver ligadas, note que que ao descrever o seu projeto procure usar as palavras “E”, “OU”, ligada e desligada!

Para este exemplo monte o programa abaixo no Arduino e salve na pasta do seu projeto.

```
// exemplo de teste condicional
int led13=13;
void setup() {
  // put your setup code here, to run once:
  pinMode(12,INPUT);
  pinMode(11,INPUT);
  pinMode(led13,OUTPUT);
 }
void loop() {
  // put your main code here, to run repeatedly:
 int ch12=digitalRead(12);//le a chave 12
 int ch11=digitalRead(11);//le a chave 11
 if (ch11 && ch12){
  digitalWrite(led13,HIGH);
 }
 else{
  digitalWrite(led13,LOW);
 }
}
```

Compile e carregue o programa no microcontrolador e teste, o LED só liga quando as duas chaves estiverem ligadas!

Note que primeiro você pega os valores das entradas (chaves) e depois você testa!

Altere o programa de forma a ligar o LED quando CH12 “OU” CH11 estiverem ligadas.

A inversão!

A função lógica inversão em linguagem de computação deve ser usada sempre que a palavra “NÃO” entrar na descrição do problema, veja o exemplo abaixo.

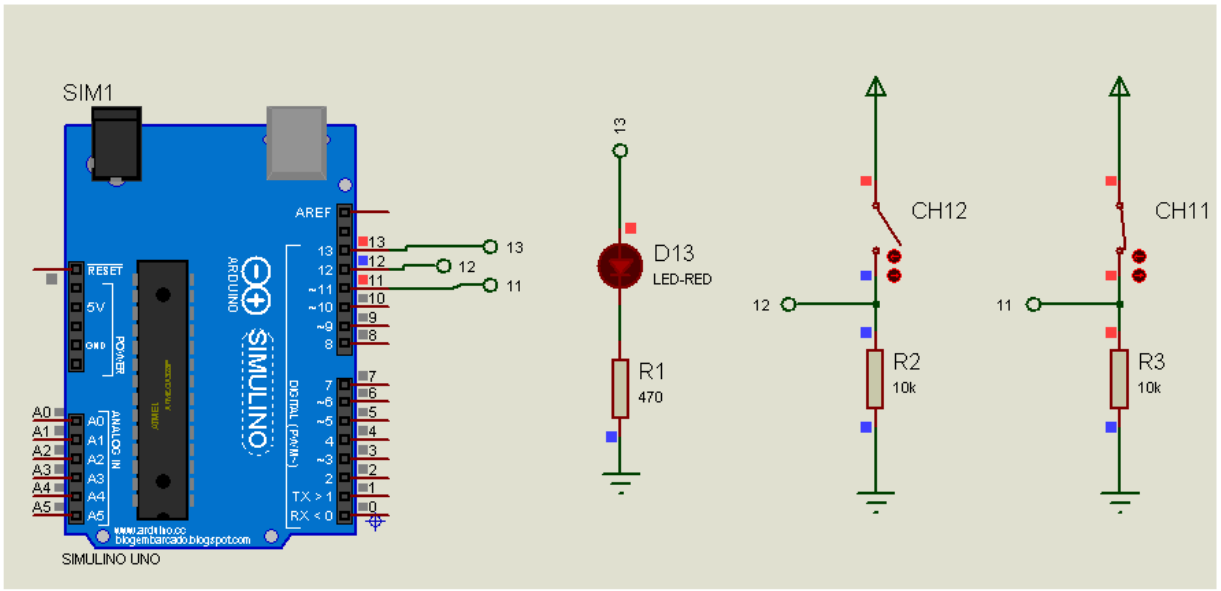
Monte um programa que ligue o LED quando a chave CH12 não estiver ligada e a chave CH11 estiver ligada.

A parte da frase não estiver ligada é o mesmo que dizer desligada, assim se na descrição a palavra desligada pode ser substituída por “Não Ligada”, esta inversão é mais aplicada as entradas!

Para escrever este “NÃO” dentro da condição do if() basta colocar o sinal de exclamação na frente da variável a ser invertida, observe o programa exemplo abaixo que soluciona o exemplo.

```
1  // exemplo de teste condicional
2  int led13=13;
3  void setup() {
4    // put your setup code here, to run once:
5    pinMode(12,INPUT);
6    pinMode(11,INPUT);
7    pinMode(led13,OUTPUT);
8   }
9  void loop() {
10   // put your main code here, to run repeatedly:
11  int ch12=digitalRead(12);//le a chave 12
12  int ch11=digitalRead(11);//le a chave 11
13  if (ch11 && !ch12){
14   digitalWrite(led13,HIGH);
15  }
16  else{
17   digitalWrite(led13,LOW);
18  }
19 }
```

O resultado é mostrado abaixo salientado a linha do programa onde a função NÃO é usada.



NÃO

```
if (ch11 && !ch12){
```

Agora altere o programa de forma a ligar o LED quando a chave CH1 não estiver ligada e a chave CH12 não estiver ligada!

Partida direta de motor com botão de emergência!

Um exemplo bastante comum é ligar um motor com partida direta tendo, um botão de liga “1”, um botão de desliga “0” e um botão de emergência para desligar o motor sempre que ela for pressionada.

O botão de emergência é do tipo “Normalmente Fechado”, isto é, quando você pressiona ele abre. Tem um tipo de botão de emergência que trava mecanicamente quando pressionado, o operador tem que destravar manualmente! O botão de emergência é do tipo “NF” porque se o cabo se romper por algum motivo tudo para, se fosse ao contrário você só iria saber ao pressionar e aí poderia ser tarde demais!

Símbolo

Caixa onde é montado o botão de emergência e fixado na máquina!

O botão de liga e o botão de desliga normalmente são montados em uma caixa. O botão de ‘liga” é do tipo sem retenção e contato “NA” e o botão de desliga é do sem retenção com contato tipo “NF”! Um botão sem retenção é do tipo campainha, só fica ligado enquanto você o mantém pressionado.

Caixa Partida direta de motor

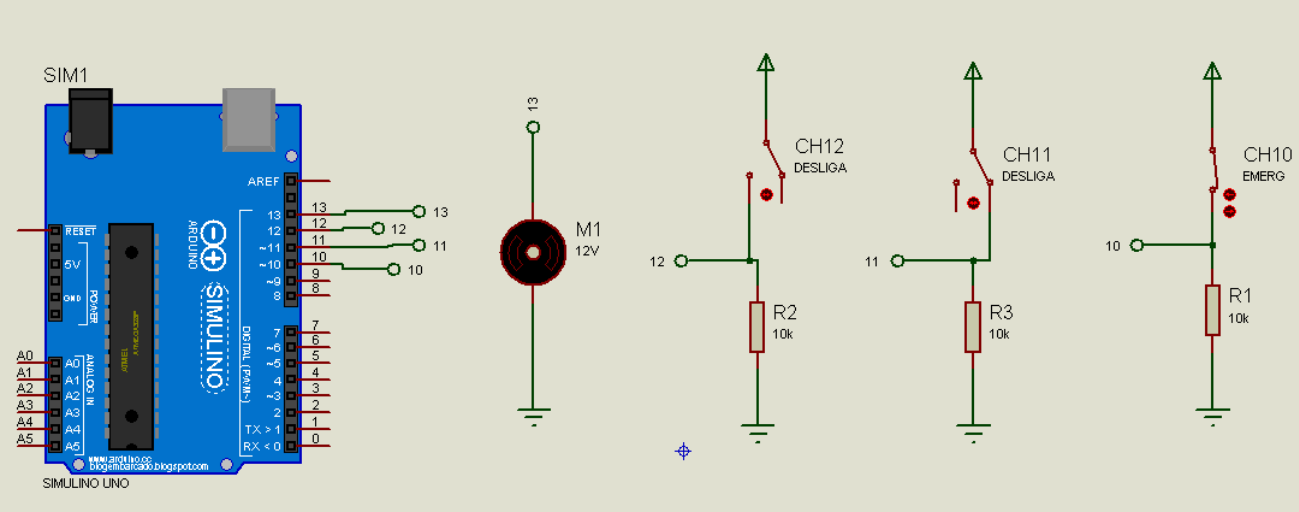
Símbolo

Para exercitar este exemplo crie uma pasta para o projeto partida direto do motor, monte o circuito abaixo.

O motor é do tipo “motor dc active”, altere a propriedade “Load REsistance” para 122Ohm!

Os botões sem retenção em inglês são chamados de “momentary” e no ISIS são chamados de “MOM” e você pega digitando “switch spdt mom! na busca.

Note que no botão liga você usa o contato “NA” e no botão desliga você usa o contato “NF”!



O programa é mostrado a seguir.

Você pode descrever a lógica como:

Se botão de emergência ligado e botão de liga ligado então liga motor!

Se botão de emergência ligado e botão desliga desligado (NÃO ligado) então desliga o motor!

Se botão de emergência desligado (NÃO ligado) então desliga o motor!

Quando o botão de emergência for desligado com o motor ligado este é desligado, e se o botão de emergência for ligado novamente o programa fica esperando liga ro botão “Liga” para ligar o motor novamente!

Se o botão de emergência for desligado como o motor ligado este irá desligar, e se o botão de emergência for mantido desligado não adiante ligar o botão liga, o motor não vai ligar até voê voltar a ligar o botão de emergência!

Monte o programa seguir, compile carregue no Arduino e teste observando todas as condições descritas antes!

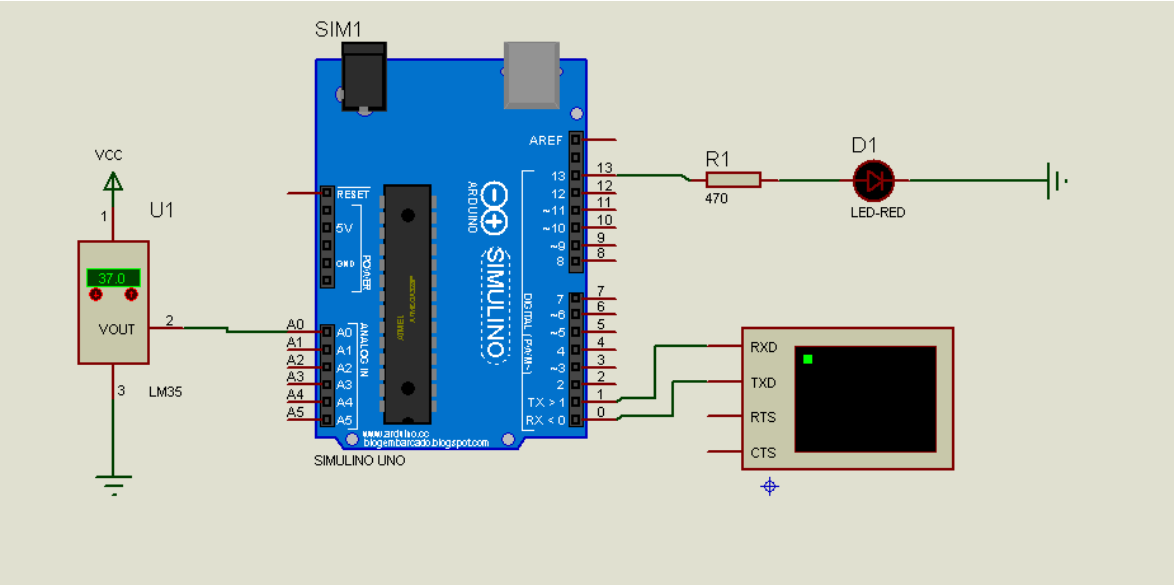
```
int motor=13;
void setup() {
  // put your setup code here, to run once:
  pinMode (motor,OUTPUT);
  pinMode(12, INPUT);
  pinMode(11, INPUT);
  pinMode(10, INPUT);
}

void loop() {
  //lê as chaves
  int chliga=digitalRead(12);
  int chdesliga=digitalRead(11);
  int emergencia=digitalRead(10);
  // se botão emergincia ligado E chave liga ligada
  if (emergencia && chliga){
    digitalWrite(motor, HIGH);
  }
  //se botão de emergênica ligado E chave desliga desligada (não ligada)
  if (emergencia && !chdesliga){
    digitalWrite(motor, LOW);
  }
  //se botão de emergência não ligado então desliga o motor
  if (!emergencia){
    digitalWrite(motor, LOW);
  }
}
```

Desafio 6:

DESAFIO 6

Coloque um segundo LED no exemplo do sensor de temperatura de forma que o LED ligue quando a temperatura for maior do que 30 graus e menor do que 50 graus?

O circuito.



O programa a ser alterado é mostrado a seguir.

```
//Usando o comparador relacional
int led=13;
// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT);//saida do LED
  Serial.begin(9600);
}
void loop() {
  int analogA0 = analogRead(A0);//Lê a entrada onde está ligado o potenciômetro
  // ajusta o valor para mostrar a temperatura
  float temperatura=map (analogA0, 0, 1023, 0, 500);
  //liga o led se temperatura maior do que 38 FEBRE
  if (temperatura>38.0){
    digitalWrite(led, HIGH);//liga LED
  }
  else{
    digitalWrite(led, LOW);//liga LED
  }
  Serial.println(temperatura);//mostra o valor da  tensaõ via serial
  delay(1);        // delay  para estabilizar
}
```

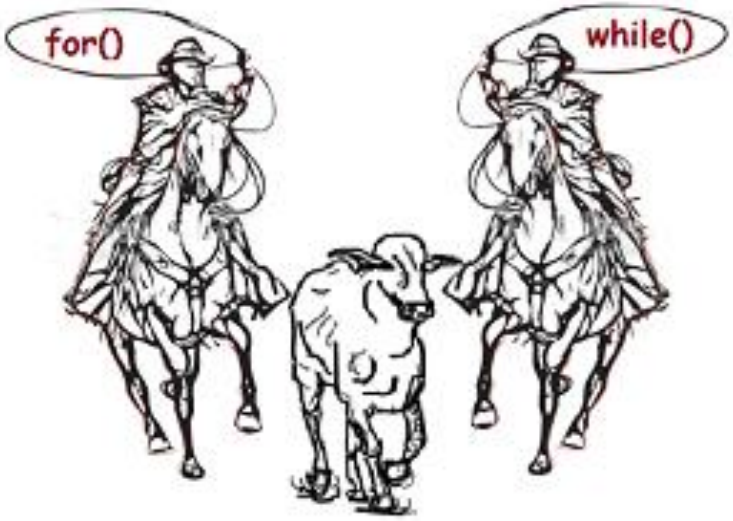
Instruções de laço (LOOP):

Uma instrução de laço prende a sequência de execução do programa em uma região definida entre chaves enquanto uma condição de teste for verdadeira.

As duas principais instruções de laço são:

A instrução while ().

A instrução for().



Começando pela instrução while ():

Uma instrução while () prende o programa entre os sinais de chaves enquanto a condição dentro dos parênteses for verdadeira.
Primeiro é testada a condição, se esta for verdadeira a sequência dentro das chaves é executada, se for falsa sequencia salta continuando depois das chaves!

Se for verdadeira?
Laço nela!

Se for falsa?
Salta fora!

Ao final da última chave de fechamento da instrução while() a sequência de execução volta para o início e testa a condição novamente!

A forma de montar a condição é a mesma usada na instrução if() usando operadores relacionais e operadores lógicos!

```
while (CONDIÇÃO)  {

    Sequencia dentro do laço;

}
```

Operador	Ação
>	Maior
>=	maior ou igual
<	Menor
<=	menor ou igual
==	Igual
!=	não igual (diferente)

Operador	Ação
&&	operação AND
||	operação OR
!	operador de negação NOT (operador unário)

Projeto do pisca-pisca.

Para testar a instrução while () você vai criar um programa que gere um pisca-pisca.

Neste exemplo você irá ligar o LED como um pisca-pisca quando a chave for pressionada!

A instrução de atraso usando o while() é mostrado abaixo.

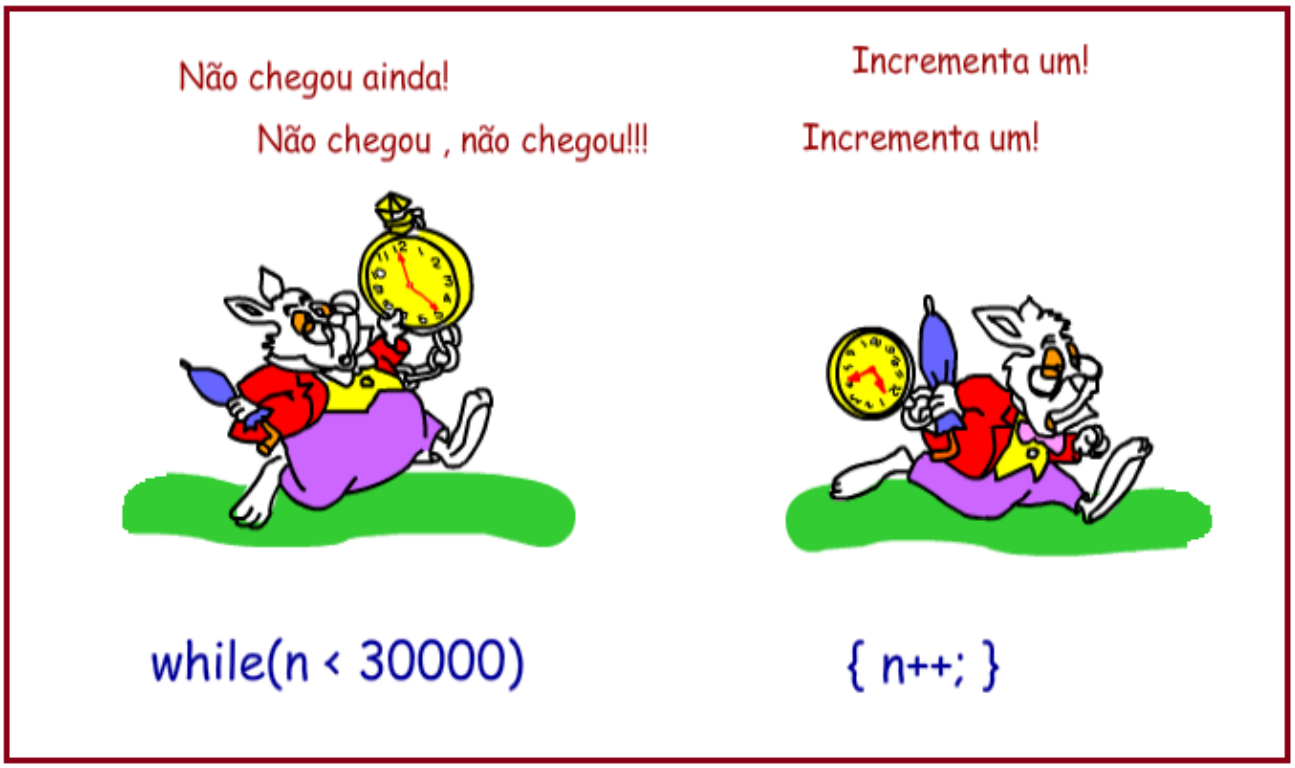
A instrução mostrada na figura abaixo é usada para prender o programa em laço do while(), este tipo de rotina é chamada de “DELAY” (atraso em inglês)!

O Arduino tem uma função delay() própria que você já usou no projeto “Blink” inicial, mas para um compilador normal da linguagem “C” que não possuam a função delay() você pode usar o while para implementar esta função!

A pergunta do while é: Enquanto a variável “n” for menor do que 30000 a sequência fica dentro do laço. A variável é contador de tempo. O tempo vai depender do clock de cada microprocessador, na prática você terá testar o programa e avaliar se o tempo desejado está bom.

Dentro do laço a variável “n” é incrementada de um cada vez que o processamento passa pela linha “n++”.

A instrução “n++” é uma instrução da linguagem “C” muito usada e que soma um ao conteúdo da variável, isto é chamado de incrementar a variável.

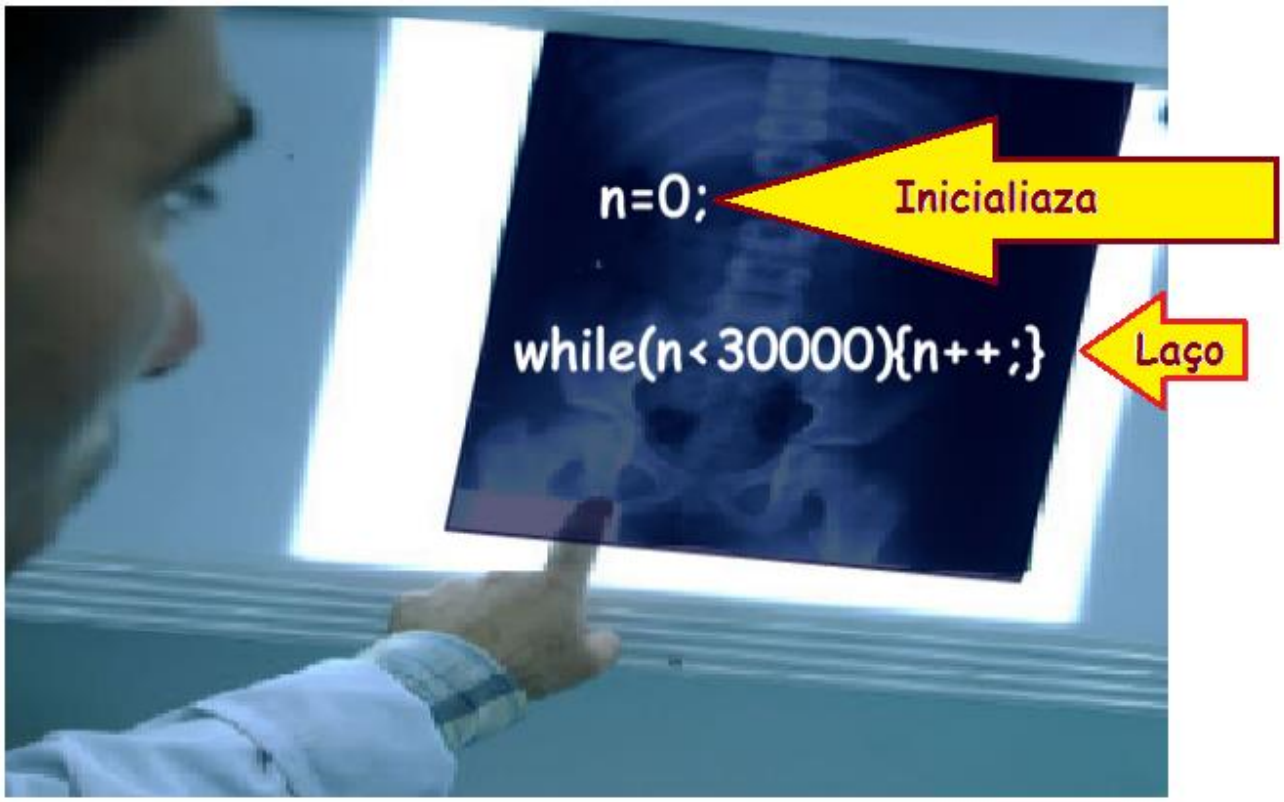


A rotina do atraso compreende duas linhas!

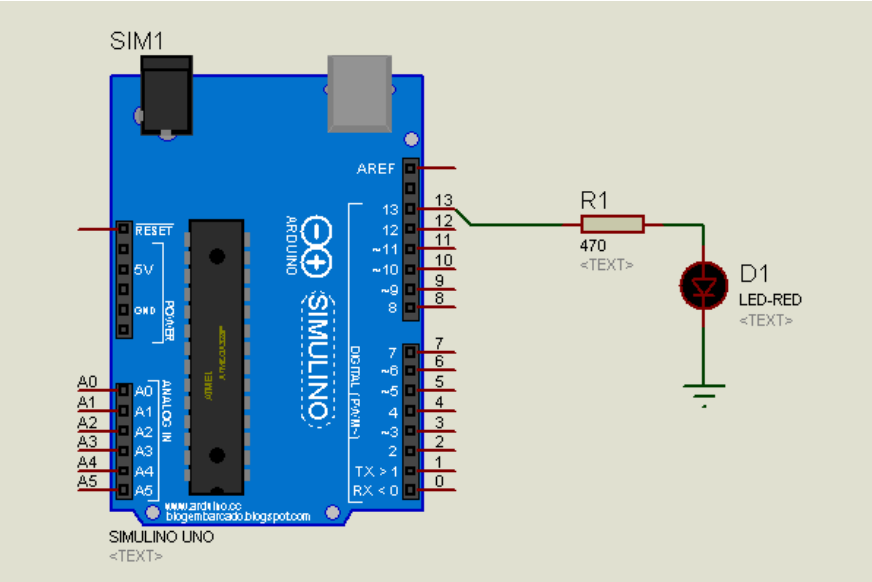
A primeira linha transfere o valor zero para a variável de uso geral “n” para garantir que o valor inicial seja zero!

A segunda linha é o laço de atraso propriamente dita!

Note que os colchetes foram colocados na mesma linha dos parênteses, insto simplifica a escrita porque tem somente uma instrução (n++;), se tiver mais de uma instrução é melhor colocar nas linhas abaixo como foi feito na função if()!!



Para testar o programa crie uma pasta na área de trabalho e nesta pasta monte o circuito abaixo com um LED ligado na saída 13.



Escreva o programa é mostrado abaixo e salve na pasta criada para o seu projeto!

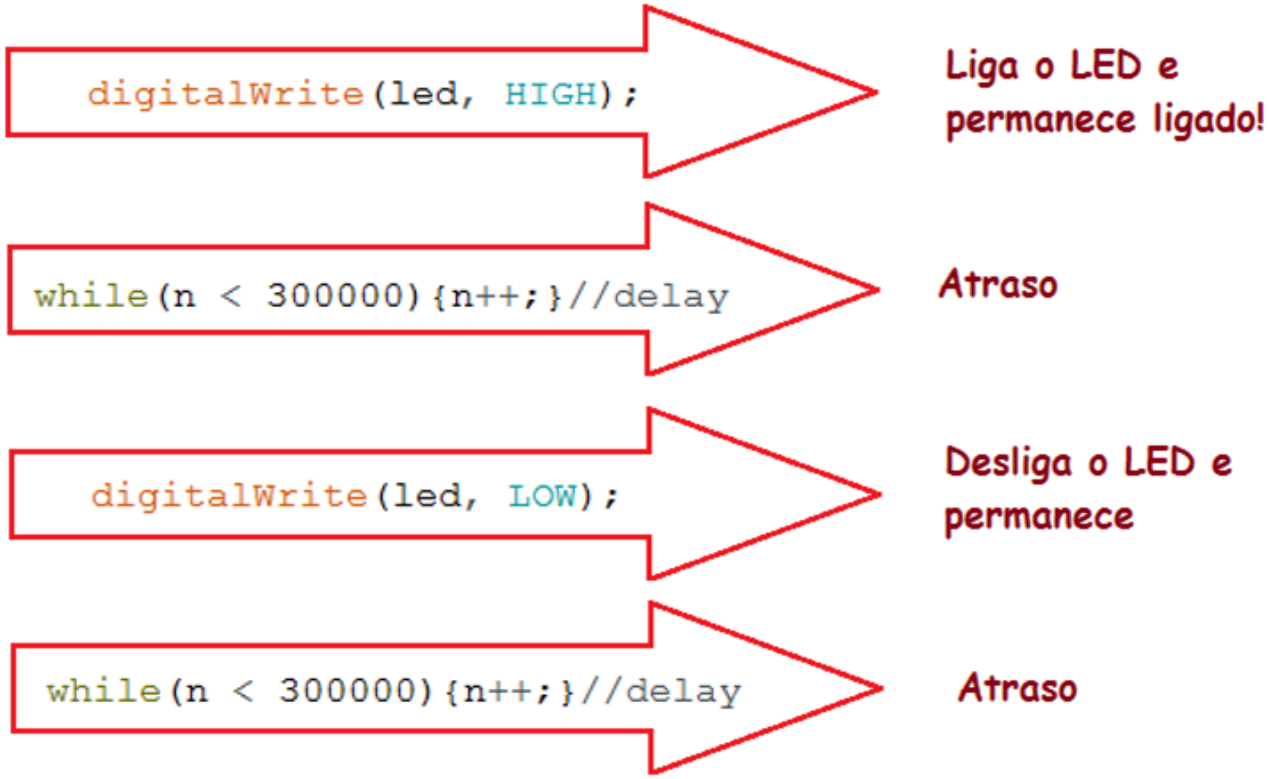
Note que a variável que conta o tempo “n” foi declarada do tipo “long”, este é um tipo de variável que pode alcançar valores muito alto deixando o ajuste de tempo bem sensível!

Note que a instrução “n=0” foi colocada antes de acionar o LED, pois no Arduino o acionamento do LED pode interferir na instrução seguinte!

```
long n;//variável que conta o tempo
int led=13;//led pino 13
void setup() {
  // put your setup code here, to run once:
  pinMode(led,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  n=0;//inicializa variável de contagem
  digitalWrite(led, HIGH);//liga led
  while(n < 300000){n++;}//delay
   n=0;//inicializa variável de contagem
  digitalWrite(led, LOW);//desliga led
  while(n < 300000){n++;}//delay
}
```

Compile carregue o programa no Arduino rode o programa, veja o led piscando!

Detalhando o exemplo do pisca-pisca:



Usando o tipo Byte, flag e a chave toogle.

A variável do tipo “byte” pode ser usada como flag em programação porque ocupa pouco espaço de memória e pode assumir dois estados HIGH e LOW!

Uma das aplicações deste tipo de variável é chamada de “flag” em alusão as bandeirolas usadas para sinalizar mensagens entre navios, nesta aplicação uma variável é usada para indicar ao restante do programa algo foi ligado ou desligado, neste caso usar uma variável do tipo byte fica mais claro.

Na linguagem “C” pura, como a usada no PIC não é possível ligar e desligar usando as palavras LOW e HIGH, mas você pode ligar e desligar colocando os valores “1” e “0”!

Flag é uma sinalização
que poderá ser usada adiante

Uma chave do tipo toogle (alternar) alterna o seu valor cada vez que é pressionada servindo para inverter o estado de uma saída, se estava ligada então desliga, se estava desligada então liga. Este é um caso típico do uso do flag que indica que a chave foi pressionada.

Uma chave toogle é usada com uma chave do tipo sem retenção (impulso), como aquelas usada em campainha!

Chave Impulso	
NA	
NF	

Veja o uso do flag e da chave toogle no exemplo a seguir!

No exemplo sempre que a chave for ligada e desligada o estado do LED inverte o estado do LED.

Neste exemplo será usado um flag chamado fchave que indica que a chave foi pressionada e um flag chamado fled que indica o estado do led! O “f” na frente do nome indica um flag e um flag chamado fled que indica o estado do LED!

O programa é dividido em três partes bem distintas que se comunicam via variáveis do tipo flag.

Na primeira parte é usado uma instrução de laço while() é usado para testar a chave toogle.

```
11  // verfica a chave toogle
12  while (int chave=digitalRead(12)){
13    fchave=HIGH;
14  }
```

Note o flag toogle é ligado dentro do while()!

Na segunda parte o flag toogle é testado e o flag do led é atualizado.

Um ponto importante é desligar o flag do toogle para que possa ser usado novamente a próxima vez que a chave for pressionada!

Note o uso do operador lógico inversão (NÃO) para inverter o estado do flag do led!

```
if (toogle){
  //desliga flag do toogle para que ele só funcione uma vez
  toogle=LOW;
  ligaLed=!ligaLed;//inverte o flag LED
}
```

Na terceira etapa o flag do led é usado para ligar e desligar o led.

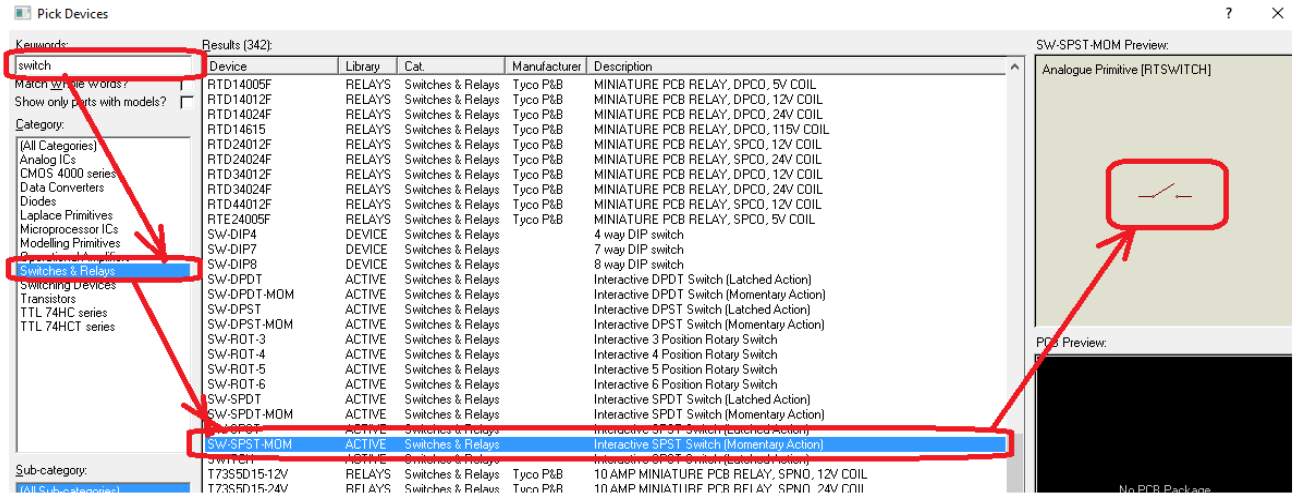
```
20  //atualiza o estado do LED
21  if (fled){
22    digitalWrite(led13, HIGH);
23  }
24  else{
25    digitalWrite(led13, LOW);
26  }
```

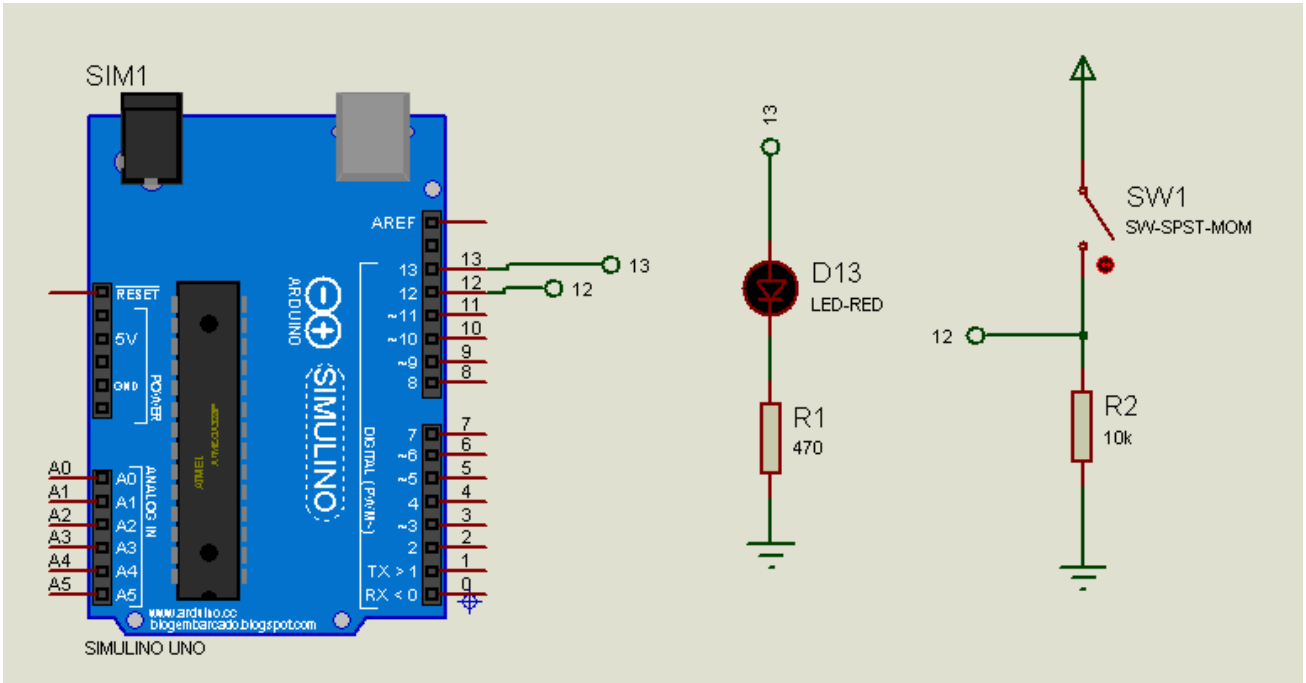
O programa complexo foi dividido em três pequenas rotinas simples, dividir um programa complexo em pequenas rotinas simples é uma boa política em programação!

Agora trabalhe no exemplo para isto crie uma pasta para o seu projeto, crie um novo projeto no ISIS monte o circuito desenhado abaixo e salve como (save design as...).

O circuito é mostrado abaixo.

A chave é do tipo contato momentâneo e você pega conforme descrito na figura abaixo digitando “switch mom spst” no campo de procura!



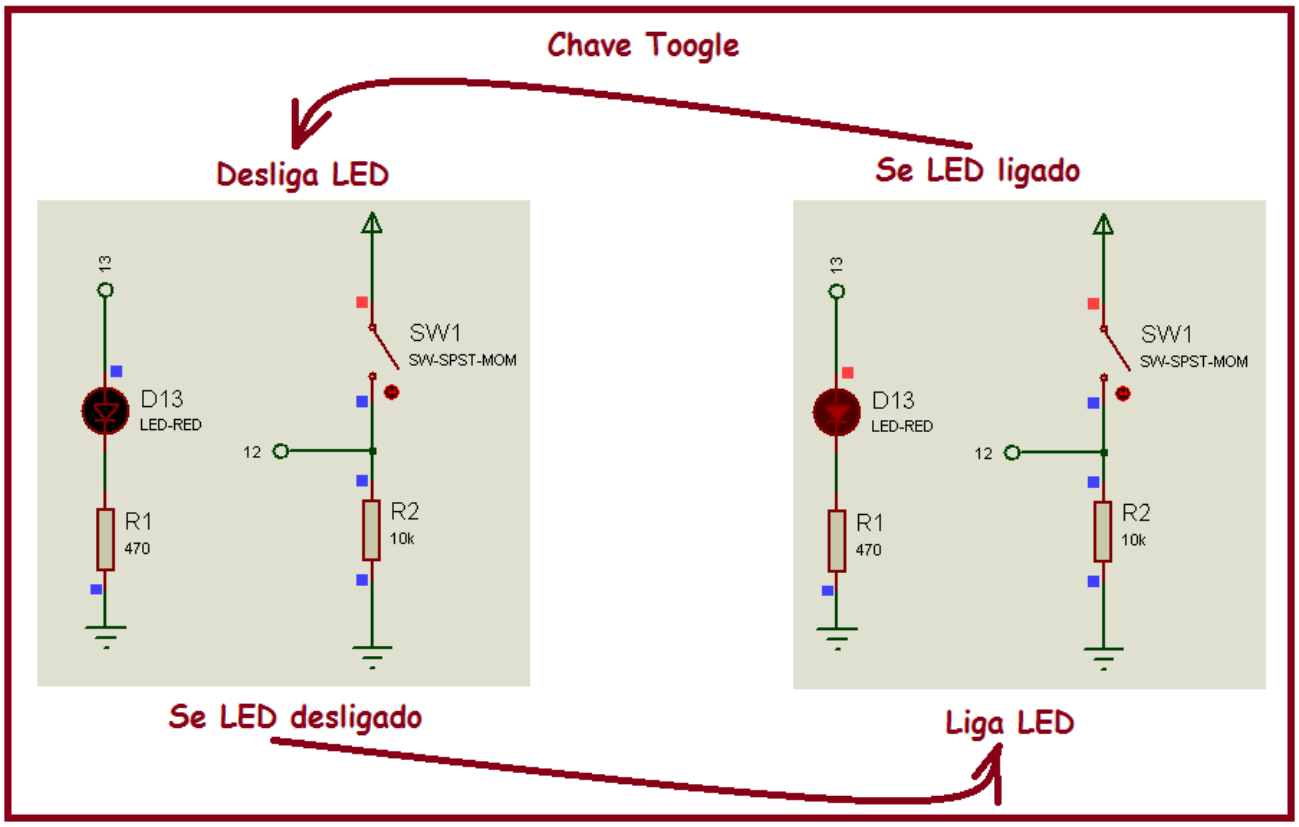


Na pasta criada para este projeto, crie um novo projeto no ARDUINO e salve “Salve como...” na pasta do seu projeto também, complete o projeto do ARDUINO como descrito abaixo depois compile e exporte binário, carregue no simulino e rode observando o funcionamento!

O programa completo é mostrado abaixo.

```
// exemplo de uso do flag e chave toogle
int led13=13;//pinos do LED
byte fchave;//flag chave
byte fled;//flag led
void setup() {
  // put your setup code here, to run once:
  pinMode(12,INPUT);//define pino 12 como entrada
  pinMode(led13,OUTPUT);//define pino 13 como saída
 }
void loop() {
  // verfica a chave toogle
  while (int chave=digitalRead(12)){
    fchave=HIGH;
  }
  //se a chave foi pressionada atualiza os flags
  if (fchave){
    fchave=LOW;//reset flag toogle
    fled=!fled;//inverte flag estado do led
  }
  //atualiza o estado do LED
  if (fled){
    digitalWrite(led13, HIGH);
  }
  else{
    digitalWrite(led13, LOW);
  }
}
```

Observe que ao pressionar e soltar a chave a condição do LED inverte, se estava desligado, liga! Se estava ligado, desliga!



Contador Chave pressionada.

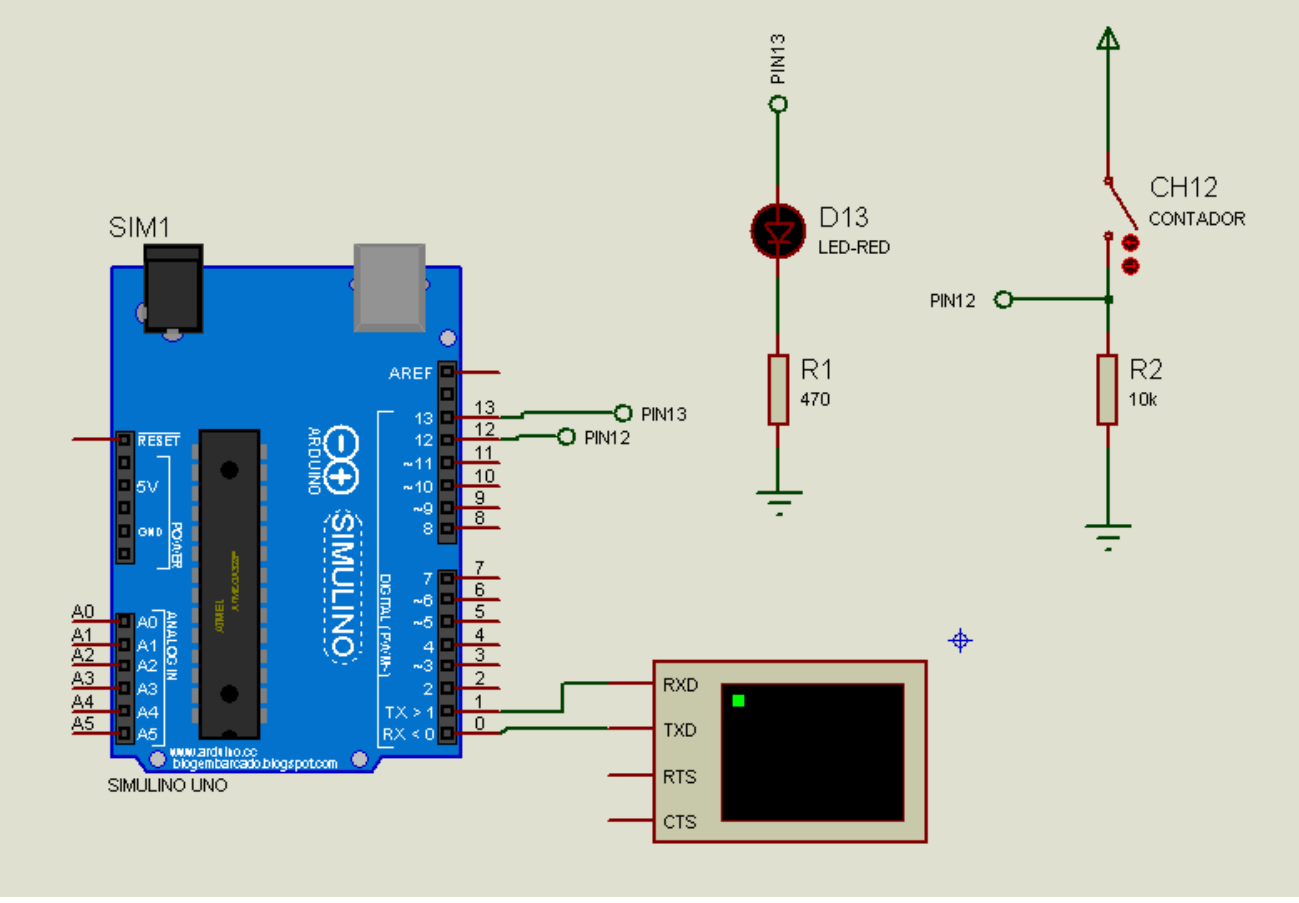
Uma aplicação importante do while() é detectar quando uma chave foi pressionada e depois liberada, é o que acontece quando você digita uma tecla no seu teclado!

Outro detalhe deste exemplo é mostrar a você como fazer um contador usando a linguagem “C”.

Um contador é simplesmente uma variável que é incrementado em sincronismo com um evento, pode ser uma chave, ou um sensor ou ainda clock interno do microprocessador. O contador pode ser usado para contar a produção de uma máquina ou o tempo tendo um pulso de tempo como base etc.

A instrução que detecta a chave é do tipo toogle.

Agora trabalhe no exemplo para isto crie uma pasta para o seu projeto, crie um novo projeto no ISIS monte o circuito desenhado abaixo e salve como (save design as...).



Na pasta criada para este projeto, crie um novo projeto no ARDUINO e salve “Salve como...” na pasta do seu projeto também, complete o projeto do ARDUINO como descrito abaixo depois compile e exporte binário, carregue no simulino e rode observando o funcionamento!

Note o uso da chave toogle.

A variável contador é declarada do tipo ”long” que pode alcançar valores maiores do que 2 bilhões!

A variável contador é atualizada dentro do teste da chave toogle para evitar contagem indesejáveis!

```
// programa exemplo de contador
int led=13;//pino do led
byte fchaveligada;//flag tooglechave pressionada
long contador;
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);//declara pino do led como saída
  pinMode(12, INPUT);//delara pino da chave como entrada
  Serial.begin(9600);//inicializa a serial
}
void loop() {
    //Teste da chave toogle
   while(digitalRead(12)){
    delay(100);//delay tira ruido
    fchaveligada=HIGH;//flag chave ligada
   }
   //testa o flag toogle
   if (fchaveligada){
    //Se a chave foi pressionada
    fchaveligada=LOW;//reset flag chave ligada
    contador++;//incrementa contador
     Serial.println(contador);
     //Se o contador for maior do que 5 liga LED
     if (contador>3){
      digitalWrite(led, HIGH);
      contador=0; //zera contador para iniciar nova contagem
     }
     else{
      digitalWrite(led, LOW);
     }
   }
}
```

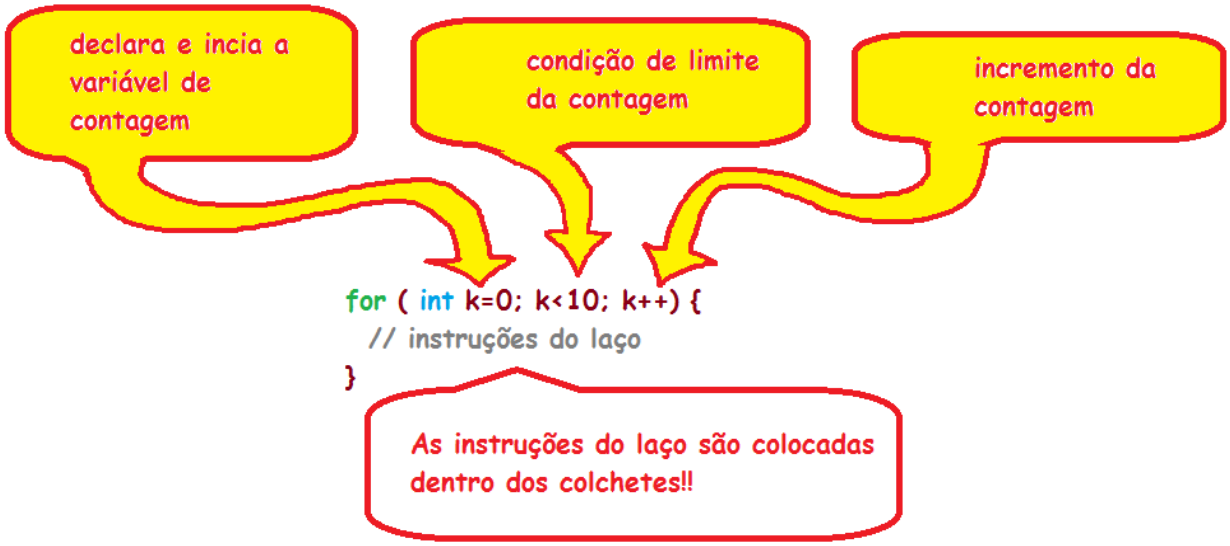
O laço loop().

A função loop() é similar ao while() que prende o programa dentro dos colchetes enquanto a condição dentro dos parênteses for verdadeira.

O que diferencia o loop() é que a variável de contagem e os limites de contagem são especificados como parâmetros da instrução ficando mais simples.

Se você precisar de uma sequência de repetição a preferência será o loop() em relação ao while()!

A forma de escrever o laço for() é mostrado abaixo.



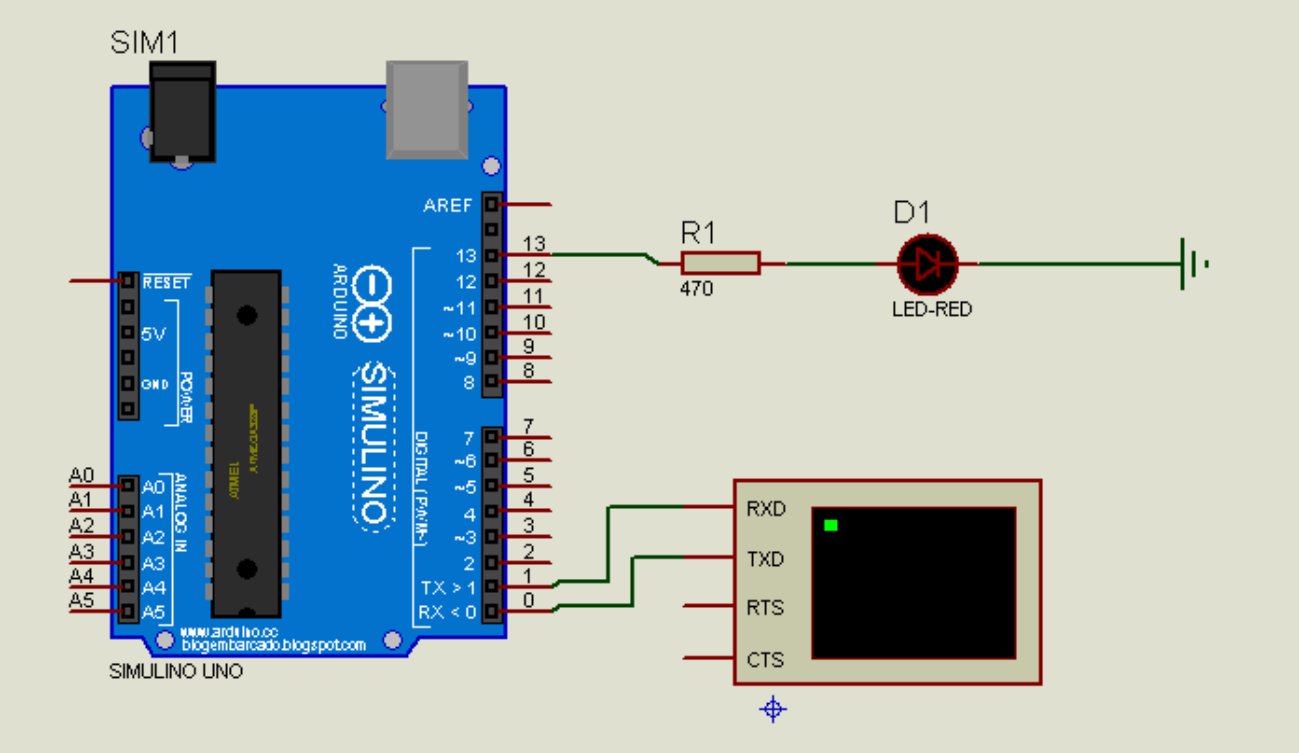
Note que você não vai mais precisar inicializar a variável de contagem fora do laço e também não vai precisar ter uma linha só para incrementar a variável de contagem, tudo isto é feito no parâmetro da função!

A variável de contagem é atualizada e testada se chegou no limite programado quando a sequência do programa chegar ao colchete final do loop(), e então, se não chegou no limite as instruções dentro dos colchetes voltam a ser executadas, caso contrário a sequência do programa salta para fora do loop().

Para testar faça o exercício a seguir.

Neste exemplo você irá montar um programa que mostra no terminal a variável de contagem “k” que é atualizada a cada 1 segundo. Nas instruções dentro do loop() existe um teste da variável de contagem, se esta for menor do que 10 então o LED é ligado, senão o led é desligado. Este programa é uma espécie de pisca-pisca com tempo de ligado diferente do tempo de desligado!

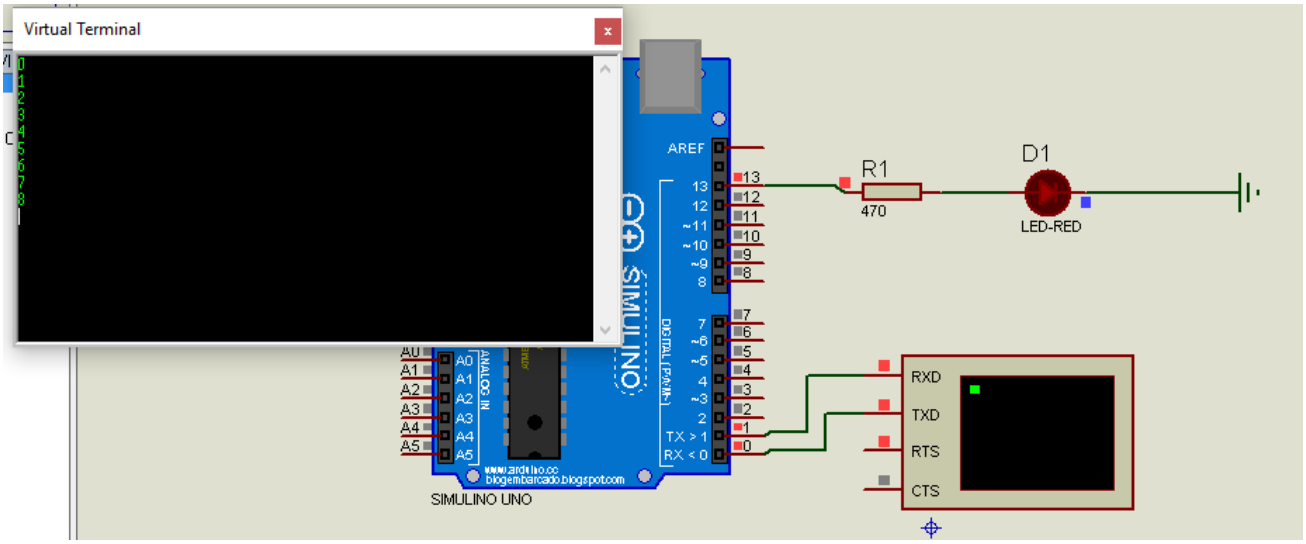
Agora trabalhe no exemplo para isto crie uma pasta para o seu projeto, crie um novo projeto no ISIS monte o circuito desenhado abaixo e salve como (save design as...).



Na pasta criada para este projeto, crie um novo projeto no ARDUINO e salve “Salve como...” na pasta do seu projeto também, complete o projeto do ARDUINO como descrito abaixo depois compile e exporte binário, carregue no simulino e rode observando o funcionamento!

```
// testando o loop()
int led=13;//pino do led
void setup() {
  // put your setup code here, to run once:
  pinMode(led, OUTPUT);//define led como saída
  Serial.begin(9600);//inicia a serial
}

void loop() {
  // put your main code here, to run repeatedly:
  for (int k=0;k<15;k++){
    //loop com contador k iniciando do zero até 15 com incremento de um
    Serial.println(k);//mostra o valor do contador k
    if (k<10){
      digitalWrite(led, HIGH);//liga led
    }
    else{
      digitalWrite(led, LOW);//desliga led
    }
    delay(1000); //delay de 1000ms=1s
  }
}
```

Resultado.



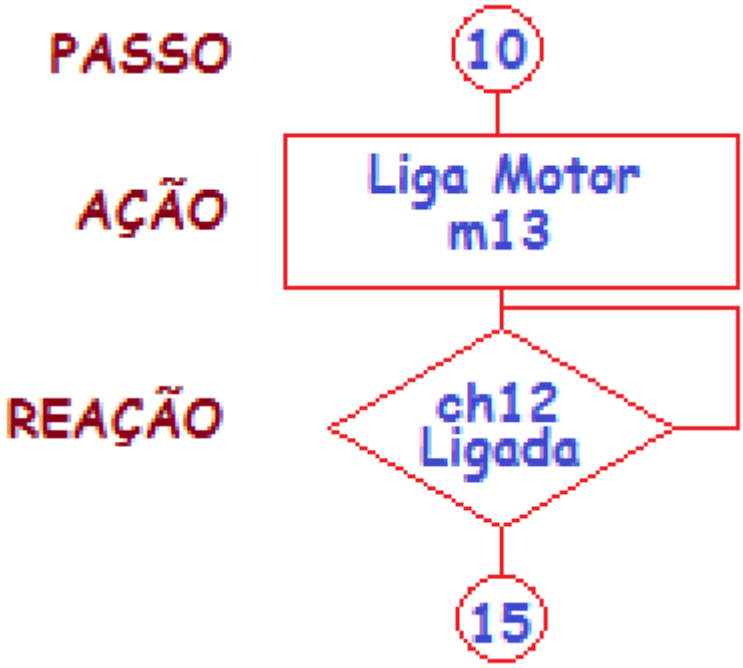
Sequencia temporizada:

Uma sequência é uma rotina que executa tarefas em sequência de passos onde somente um passo pode ser executado de cada vez.

A maioria das máquinas em automação industrial executam tarefas sequencias.

Uma sequência pode ser descrita na forma de um fluxograma, onde cada passo é descrito com um número (número do passo), ações (ligar ou desligar uma saída digital, ligar ou desligar um flag, ligar um temporizador etc..), e uma ou mais reações que são condições para mudar de passo.

A figura abaixo mostra o exemplo de um fluxograma do passo “10” de um sequencia onde a ação é ligar o motor M13 e a reação consiste em monitorar a chave12 (desliga motor), quando esta chave for ligada a sequência muda para o passo 15!

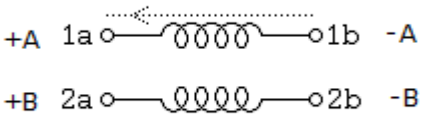


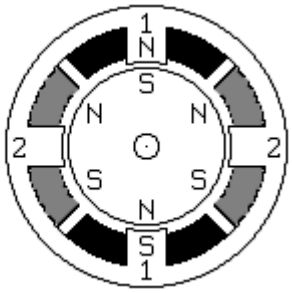
Uma forma simples de sequência é chamada de sequência temporizada onde a troca do passo depende somente de um temporizador, um exemplo típico deste tipo de sequência é o acionamento de um motor de passo, como será mostrado no exemplo abaixo, antes você verá como acionar um motor de passo.

Como é acionado um motor de passo:

O acionamento de um motor de passo pode ser feito por uma sequência temporizada já que a comutação das bobinas deve ser feita na sequência correta, a velocidade será função do tempo de atraso, quanto menor o intervalo de tempo mais rápido irá girar o motor.

Neste exemplo você vai ligar um motor de passo bipolar, existem outros tipos de motores de passo, o motor bipolar é um dos mais simples e mais usados, este tipo de motor possui somente duas bobinas, como mostra a figura abaixo.





As conexões das bobinas são chamadas de +A, -A, +B e -B na maioria dos motores!

Um motor de passo é muito usado na indústria pois apresenta boa precisão, a velocidade pode ser controlada com facilidade, e se as bobinas forem mantidas energizadas o motor fica freado.

Para que o motor de passo gire é preciso acionar as bobinas de forma sequencial correta, quando o motor possui duas bobinas para controlar o giro você deverá alternar a polaridade da bobina, isto é hora botar o positivo em um lado da bobina (lado 1a por exemplo) e hora aplicar o positivo no outro lado (1b por exemplo) invertendo a polaridade.

Um motor de passo é um componente que consome muita energia, a corrente típica é da ordem de 2A, por isto, há necessidade de colocar um circuito de potência entre o microcontrolador e o motor de passo.

Existe um circuito comercial bastante comum para este tipo de trabalho, é o CI L293D, este circuito além de ajustar a tensão de trabalho do motor para um valor maior do que 5V, também será responsável por fornecer a corrente necessária ao motor. O seu circuito interno é chamado circuito em ponte que possibilita a troca de polaridade da bobina com facilidade. Este circuito possui duas alimentações, uma alimentação de baixa tensão (5V) para o circuito de entrada e outra de tensão mais alta (12V ou 24V) para o circuito de saída. Os pinos identificados como "Em", são usados para habilitar o CI a funcionar, as letras Em significam Enable em inglês. Para habilitar o CI você deve colocar 5V nestes pinos, se você colocar 0V ele será desabilitado e o motor fica sem energia para de girar e todas as saídas do motor ficarão desligadas, o motor para, mas não fica freado, esta é uma forma de desligar o motor economizando energia. Um circuito usado para aumentar a potência de saída do sinal é chamado de "DRIVER" em eletrônica!

Um motor de passo anda aos passos, os passos são determinados pela construção interna, quanto maior o número de passos, mais preciso é o movimento e tanto mais caro será o motor, neste exemplo o motor irá trabalhar girando 90 graus a cada passo ou 45 graus dependendo da sequência de comutação das bobinas.

Quando um motor de passo está em um passo e as boinas estão energizadas ele fica parado e travado, esta é uma vantagem deste tipo de motor sobre todos os outros motores, pois os outros motores precisam de circuito ou dispositivo mantê-lo travado, se o motor de passo for desabilitado ele não ficará travado. Quando um motor está travado você não consegue girar o seu eixo com a mão, quando está destravado você consegue girar!

Para que o motor de passo gire você deverá ligar as bobinas na sequência correta, por isto, é um bom exemplo para a aplicação dos circuitos sequenciais!

Você pode ver nas tabelas abaixo os duas sequencias para o acionamento do motor de passo bipolar, para passos de 90 graus e 45 graus!

Tabela para passo de 90°:

-B	+B	-A	+A	ANG
0	0	0	1	0º
0	1	0	0	90º
0	0	1	0	180º
1	0	0	0	270º

Esta tabela move o motor na direção horária, isto é 0, 90, 180, 270,0!

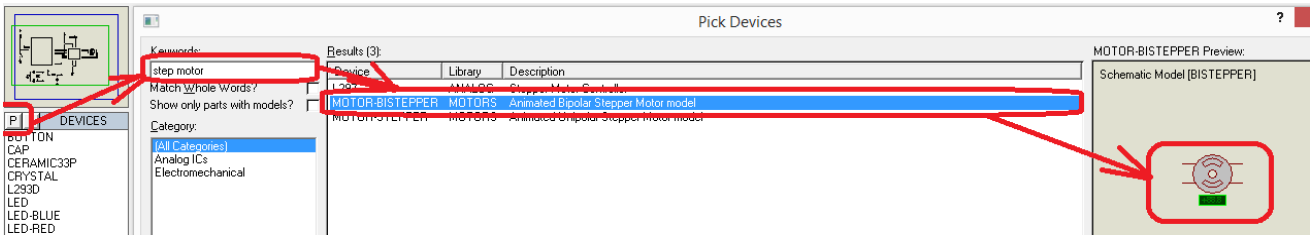
Para girar ao contrário, basta inverter a sequência, isto é, 0, 270, 180, 90, esta é outra vantagem do motor de passo, é fácil de inverter a rotação!

Tabela para passo de 45°

-B	+B	-A	+A	ANG
0	0	0	1	0º
0	1	0	1	45º
0	1	0	0	90º
0	1	1	0	135º
0	0	1	0	180º
1	0	1	0	225º
1	0	0	0	270º
1	0	0	1	315º

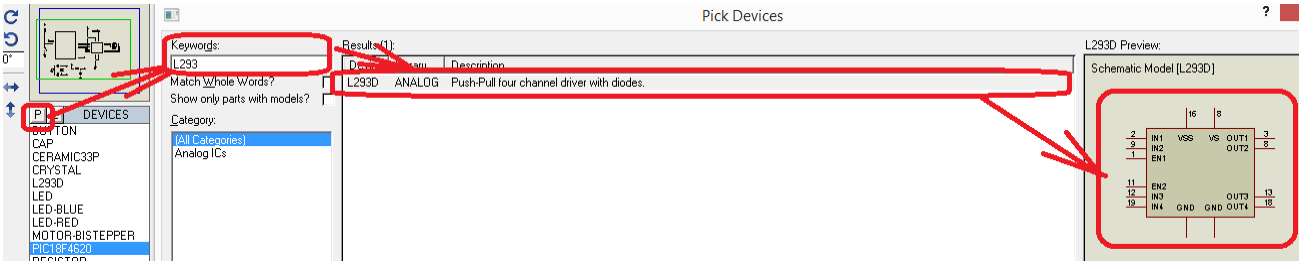
Agora que você já conhece o funcionamento do motor de passo crie uma pasta para o seu projeto, desenho o circuito abaixo no ISIS, depois escreva o programa no ARDUINO mostrado abaixo.

O diagrama abaixo mostra um circuito típico para o ARDUINO controlar um motor de passo usando o CI L293D como driver, monte o circuito para testar o programa! Para encontrar o motor de passo digite "step motor" no campo "keyword" da procura!

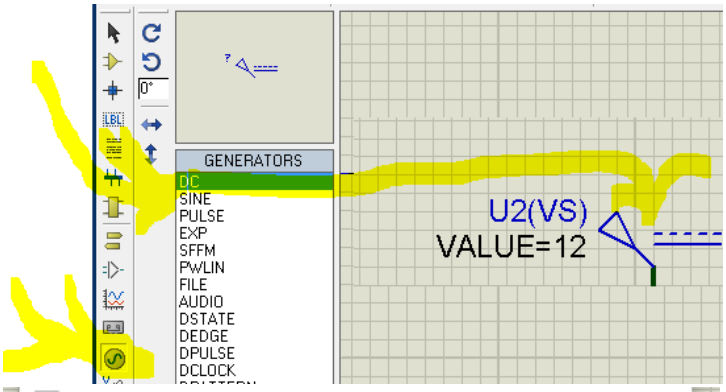


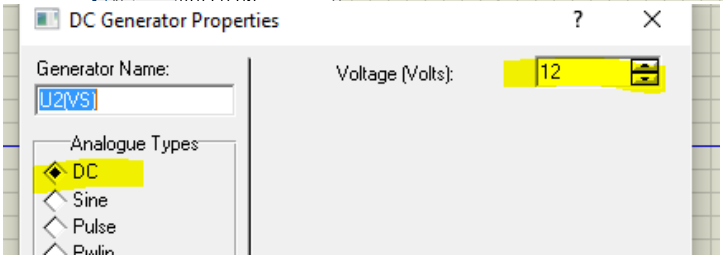
Ajuste a propriedade do motor como descrito abaixo, a tensão de alimentação deverá ser de 12V.

Para encontrar o CI L293D digite o seu código no campo "keyword" da procura!



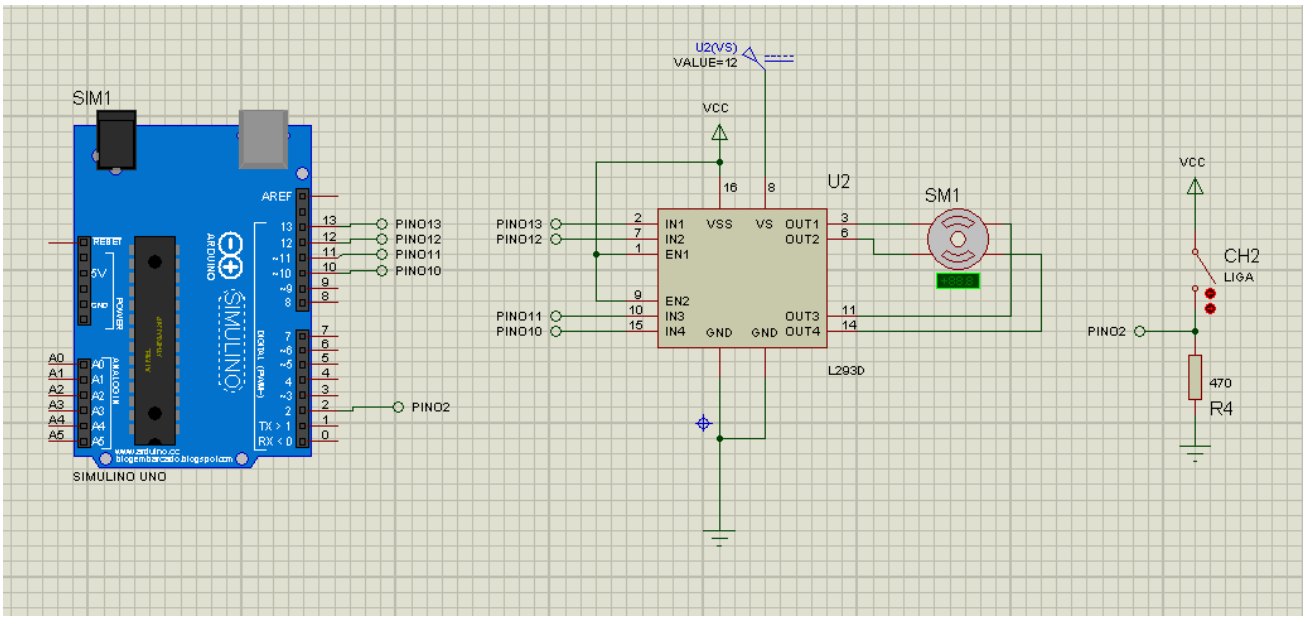
Para inserir o pino de alimentação de 12 V use o menu GENERATOR, ajuste as propriedades Voltage(V) para 12V, e Analogue Types para DC!





O circuito final deve ficar como na figura abaixo.

As bobinas A+ está ligada na saída OUT1 do driver, a bobina A- está ligada na saída OUT2 do driver, a bobina B- está ligada na saída OUT3 do driver e a bobina B- está ligada na saída OUT4 do driver.



O programa é mostrado abaixo:

```
int a1=13;//bobina A+
int a0=12;//bobina A-
int b1=11;//bobina B+
int b0=10;//Bobina B-
int ch2=2;//chave liga desliga motor
int s=0;//controla a sequencia
int velocidade=500;//velocidaded o giro quanto menor maior o giro
void setup() {
  // put your setup code here, to run once:
   pinMode(a1, OUTPUT);
   pinMode(a0, OUTPUT);
   pinMode(b1, OUTPUT);
   pinMode(b0, OUTPUT);
   pinMode(ch2, INPUT);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  //sequencai de giro a 90 graus
  if (digitalRead(ch2)){//liga motor
    //sequencia
    if (s==0){
      digitalWrite(a1, HIGH);
      digitalWrite(a0, LOW);
      digitalWrite(b1, LOW);
      digitalWrite(b0, LOW);
      delay(velocidade);
      s=1;//troca o passo
    }
    if (s==1){
      digitalWrite(a1, LOW);
      digitalWrite(a0, LOW);
      digitalWrite(b1, HIGH);
      digitalWrite(b0, LOW);
      delay(velocidade);
      s=2;//troca o passo
    }
```

```
    if (s==2){
      digitalWrite(a1, LOW);
      digitalWrite(a0, HIGH);
      digitalWrite(b1, LOW);
      digitalWrite(b0, LOW);
         delay(velocidade);
      s=3;//troca o passo
    }
    if (s==3){
      digitalWrite(a1, LOW);
      digitalWrite(a0, LOW);
      digitalWrite(b1, LOW);
      digitalWrite(b0, HIGH);
         delay(velocidade);
      s=0;//troca o passo volta para o inicio
    }
  }

  else{
    s=0;//devia par ao primeiro passo
    digitalWrite(a1, LOW);
    digitalWrite(a0, LOW);
    digitalWrite(b1, LOW);
    digitalWrite(b0, LOW);
  }
}
```

Detalhes do programa:

O nome das bobinas não pode ser escrito a+ ou A- porque o compilador pode entender o sinal mais e menos como operação.

A variável s é a variável do passo da sequência.

A variável velocidade é usada para configurar o tempo da função delay(), quanto menor este tempo mais rápido gira o motor, pois menor será o tempo que a rotina fica presa no delay.

Desafio 6

DESAFIO 7

Coloque uma segunda chave no circuito do motor de passo para inversão da rotação do motor.

Quando a chave de inversão estiver ligada o motor gira para um lado quando estiver desligada gira para o outro lado.

O giro do motor pode ser classificado em Horário e Anti-Horário!

Funções

Uma função é uma rotina escrita fora da função principal loop() e que pode ser usada dentro do loop() ou mesmo em outras funções.

O Arduino já possui uma série de funções prontas que você já vem usando a algum tempo, por exemplo, delay(), digtalWrite() etc.

Você também pode escrever suas próprias funções.

O principal objetivo de escrever uma função é simplificar o programa já que as instruções da função ficam fora do loop() e podem ser usados mais de uma vez!

A estrutura de uma função é descrita abaixo.

```
tipo  Nome  () {tipo parâmetro, ...){
    Corpo da função;

    return variavel;//fim da função
}
```

Onde:

Onde “tipo Nome” declara o tipo da variável de retorno da função, se não tiver variável de retorno o tipo deve ser declarado como “void” que é um tipo coringa.

O “tipo parâmetro” especifica o tipo da variável de entrada que será processada dentro da função, esta variável só será usada dentro da função. Se a função não tiver variável de entrada este campo deve ser mantido em branco.

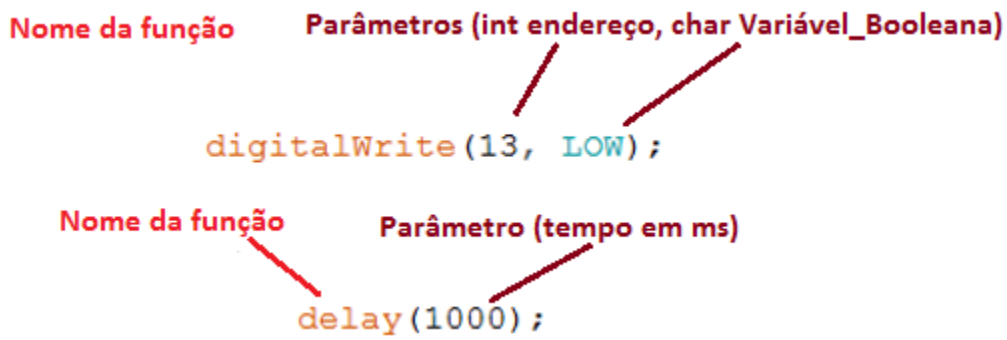
No corpo da função você escreve a rotina da sua função.

Na linha return variável a instrução return termina a função, e se tiver uma variável de retorno o campo variável deverá ser preenchido com esta variável.

Uma função é um desvio da sequencia de processamento, ao final da função a sequencia do processamento volta para a linha seguinte a chamada da função.

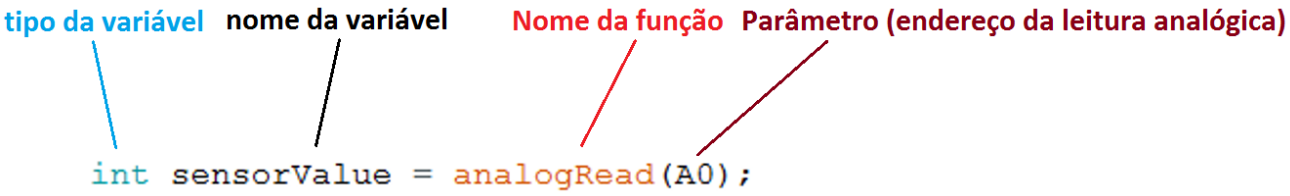
Você chama uma função do seu programa simplesmente escrevendo o nome da função com os seus parâmetros, se tiver!

Se a função não retorna valor você simplesmente escreve o nome da função com o seu parâmetro, você pode ver nas linhas abaixo do programa pisca-pisca a chamada da função do Arduino delay().



A função digitalWrite() tem dois parâmetros e a função delay() somente um!

Se a função tem uma variável de retorno você deve transferir o valor desta variável para uma variável do mesmo tipo da função, um exemplo desta aplicação é a função do Arduino analogRead()!



O valor da entrada analógica de endereço “A0” é passado para a variável “sensorValue”!

Você deve escrever uma função fora da função loop() ou setup().

A função deve sempre ser escrita antes de ser usada, assim se você vai usar a função dentro do loop() você pode escrever a função antes do loop(), mas esta não a forma normal de escrever uma função você pode avisar ao programa que existe uma função escrevendo o protótipo da função.

O protótipo da função funciona como uma declaração de variável e deve ser colocada antes da função setup(). Para declarar protótipo da função você escreve uma linha com o tipo nome e parâmetros da função, mas não precisa escrever toda a função, usando o protótipo você fica livre para escrever a função onde quiser, normalmente depois do loop()!

Veja no exemplo a seguir o uso do protótipo.

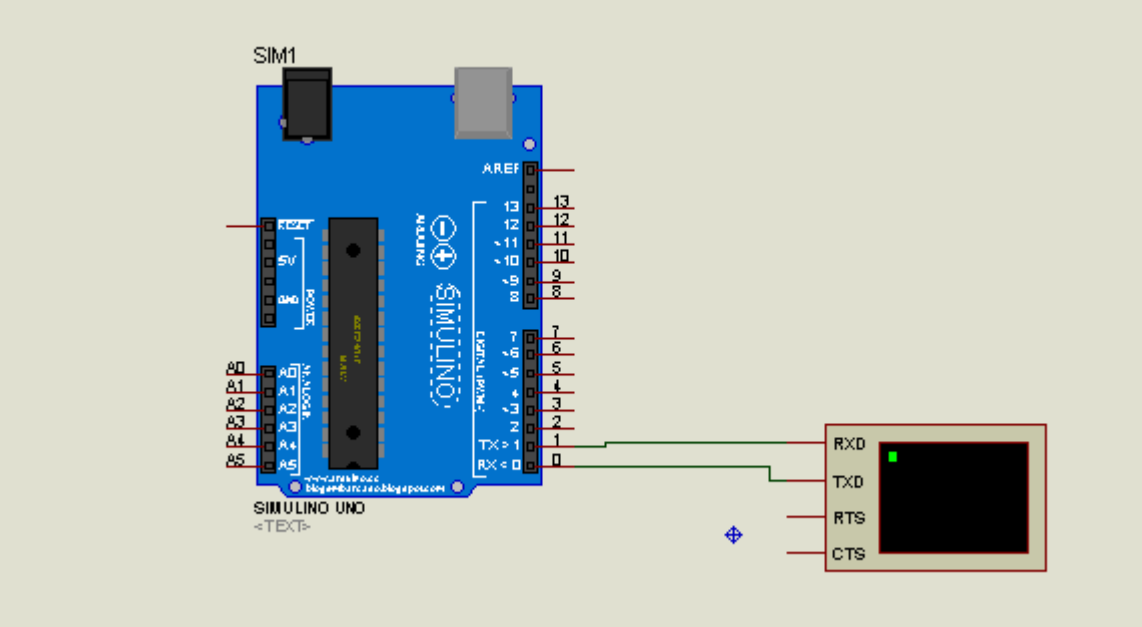
Note que a variável do parâmetro declarado quando você monta o programa só funciona dentro da função, e pode ter qualquer nome. Quando a função é chamada o valor da variável descrita na chamada é passado para o valor da variável da função!

Veja no exemplo a seguir.

Como exemplo monte o programa abaixo onde a função int Quadrado(int entrada) calcula o quadrado do número passado para a variável “entrada” e retorna para a variável resultado!

O resultado é mostrado na serial.

O circuito é mostrado abaixo.



O programa é mostrado abaixo.

```
1  byte byteRead;
2  int Quadrado(int num);//protótipo da função
3  int entrada;
4  void setup() {
5    // initialize serial communication at 9600 bits per second:
6    Serial.begin(9600);//Inicia a serial
7   }
8  void loop() {
9     if (Serial.available()) {//espera digital qualquer coisa no terminal
10       byteRead=Serial.read();
11       entrada=8; //valor da entrada
12       int resultado = Quadrado(entrada);//calcula o quadrado e passa para o resultado
13       Serial.print("resultado=");//mostra o resultado na serial
14       Serial.print(entrada);//mostra o resultado na serial
15       Serial.print("^2=");//mostra o resultado na serial
16       Serial.println(resultado);//mostra o resultado na serial
17    }
18  }
19  // função que calacula o quadrado do parâmetro "num"!
20 int Quadrado(int num){
21     int res;
22     res= num * num;//calcula o quadrado de forma simples
23     return res;//retorna o resultado
24 }
```

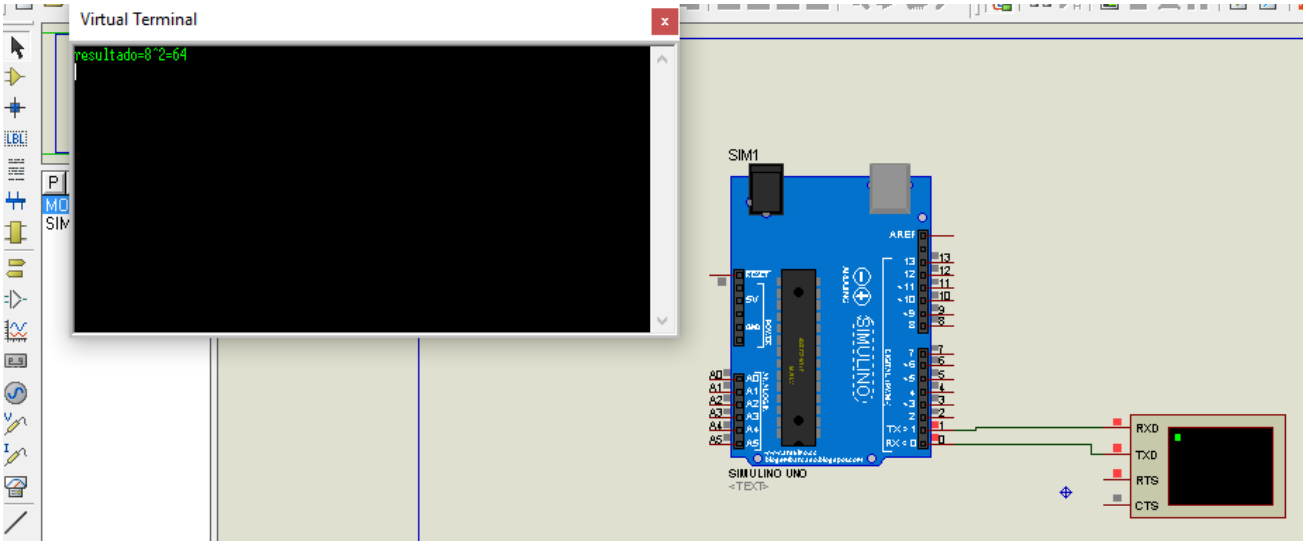
Observe na linha 2 a declaração do protótipo.

A função está escrita a partir da linha 20 depois do loop(0, isto só é possível devido a declaração do protótipo.

A variável usada entrada da função tem o nome de “num”, esta variável será usada somente dentro da função! A variável “res” é resultado do cálculo do quadrado, esta variável só funciona dentro da função!

O único cuidado que você deve ter é que a variável do programa que pega o resultado do quadrado deve ser do mesmo tipo da função, neste caso as duas são do tipo “int”!

O resultado.


Virtual Terminal
resultado=8^2=64
SIM1
SIMULINO UNO
RXD
TXD
RTS
CTS

Interrupção no Arduino.

Uma interrupção é um recurso de programação que desvia a sequência de processamento para uma função criada por você de forma imperativa, interrompendo o programa que está sendo executado dentro da função loop()!

Uma interrupção para tudo e vai para a função de interrupção!

Você usa uma interrupção quando tem uma entrada digital que deve ser atendida logo que é acionada, por exemplo um botão de emergência, quando uma interrupção é chamada todos os temporizadores do processamento normal não são interrompidos e todas as funções em execução são interrompidas e a função de interrupção é chamada e executada, ao final o programa volta exatamente para o passo que estava sendo executado antes da interrupção.

Na prática existem dois tipos de interrupção, uma chamada de externa e que é disparada por uma entrada digital especial, cada microcontrolador tem um ou mais pino para esta função. A outra forma de interrupção é chamada de interna que é disparada por um temporizador interno que fica rodando o tempo sem parar o programa normal e no tempo programado para tudo e desvia para a função de interrupção.

Você deve configurar a interrupção dizendo qual o tipo que está sendo usado, você ainda deve escrever a função dizendo o que você quer fazer na chamada da interrupção, esta rotina deve ser o mais simples possível.

Interrupção Externa:

Uma interrupção é disparada por um pino especial da placa Arduino os pinos disponíveis são mostrados na figura abaixo para cada um dos tipos de placa.

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g Leonardo, Micro)	3	2	0	1	7	
Due, Zero			(see below)			

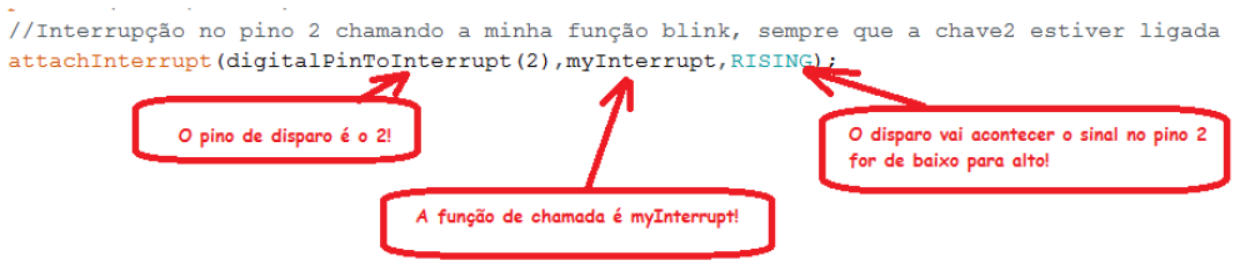
Os passos para usar uma interrupção externa são descrito a seguir.
Defina a interrupção na função setup () usando a função)! attachInterrupt(
Escreva a função a ser chamada pela interrupção este tipo de função recebe o código ISR (Interrupt Service Request).

Como escrever instrução de chamada função attachInterrupt()?

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode):

No campo “pin” você deve colocar o número do pino a ser usado na interrupção.
No campo “ISR” você deve escrever o nome da função chamada quando a interrupção for ativada, esta função você deve escrever depois da função loop().
No campo “mode” você deve escrever uma constante que indica como a interrupção será chaveada (ligada) e são listadas abaixo.

LOW: Enquanto o sinal do pino for baixo.
CHANGE: Quando o sinal do pino mudar.
RISING: Quando o sinal do pino for de baixo para alto.
FALLING: Quando o sinal do pino for de alto para baixo.
HIGH: Enquanto o sinal do pino for alto.
Exemplo:



Porque usar a função digitalPinToInterrupt(pin)?

Esta é uma função especial do Arduino e no campo pin você escreve o número do pino que irá disparar a interrupção, usando esta função o programa verifica se o pino escolhido é o correto para a placa que está sendo usado, por isto, você deve usar esta função como padrão para definir o pino da interrupção.
No exemplo abaixo est função diz que o pino a ser usado no chamado da interrupção é o pino 2.

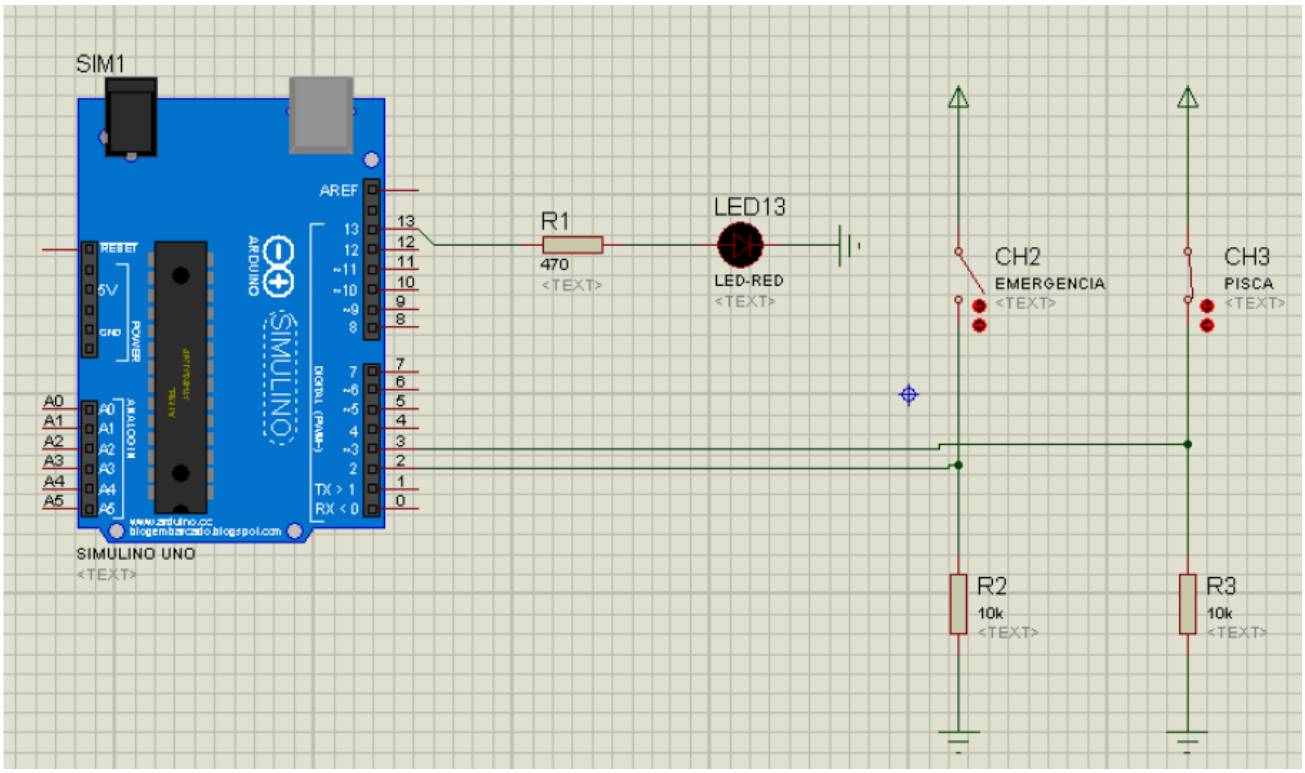
```
//Interrupção no pino 2 chamando a minha função blink, sempre que a chave2 estiver ligada
attachInterrupt(digitalPinToInterrupt(2),myInterrupt,HIGH);
```

Como usar uma variável na interrupção.

Você deve passar uma variável para a interrupção ou ainda receber um valor alterado por uma variável dentro da interrupção.
Quando você escrever a função de interrupção está não deve receber parâmetros e não devolve parâmetros, então você deve usar uma variável definida no seu programa principal, este tipo de variável é chamada de global, você deve colocar a definição “volatile” antes do tipo para garantir que ela não se perca ao ser usada na interrupção!
Na instrução a variável emergência é usada somente para sinalizar ao restante do programa que o botão foi pressionado, este tipo de uso para uma variável é chamado de flag (bandeira de sinalização)!

```
volatile int emergencia= 0;//flag que indica emergencia foi pressionada
```

O exemplo abaixo mostra uma aplicação típica para a interrupção.
Neste exemplo uma chave CH3 liga o pisca-pisca do LED, como o tempo de pisca é dado pela função delay() que para o programa durante a sua execução qundo a chave CH3 é desligada o LED só apaga ao final do tempo de pisca de 5000ms (5segundos). A chave CH2 Emergência está ligada ao pino de interrupção, assim, logo que ela é pressionado o programa para tudo que tá fazendo, inclusive a contagem do tempo na função delay() e desvia para a rotina de interrupção (myinterrupt) e desliga o LED imediatamente e ainda liga o flag emergência para indicar esta situação, depois de executar estas duas linhas o programa volta para o delay novamente, mas agora o LED já está apagado e ao passar pela intrução if (emergencia==LOW) o resultado é uma negação e a rotina de ligar o led não é executada. Quando a chave CH3 é desligada o flag é desligado também e tudo volta ao normal, ao pressionar a chave CH3 o pisca volta a funcionar. Note que o tempo de liga foi colocado em 5 segundos de propósito para você observar como a interrupção é rápida. Observe o botão de emergência só atua na transição, isto ao você pressioná-lo!
O programa completo é mostrado abaixo, crie uma pasta para o seu projeto, crie um novo projeto do ARDUINO escreva o programa abaixo e salve como na pasta criada par ao seu projeto. Cria um novo projeto no ISIS desenhe o circuito descrito abaixo e salve como (save design as...) na pasta do seu projeto.
Exporte o arquivo compilado no ARDUINO e depois carregue no Simulino do seu circuito, rode e observe o resultado.
O diagrama é mostrado abaixo:



O programa completo é mostrado abaixo:

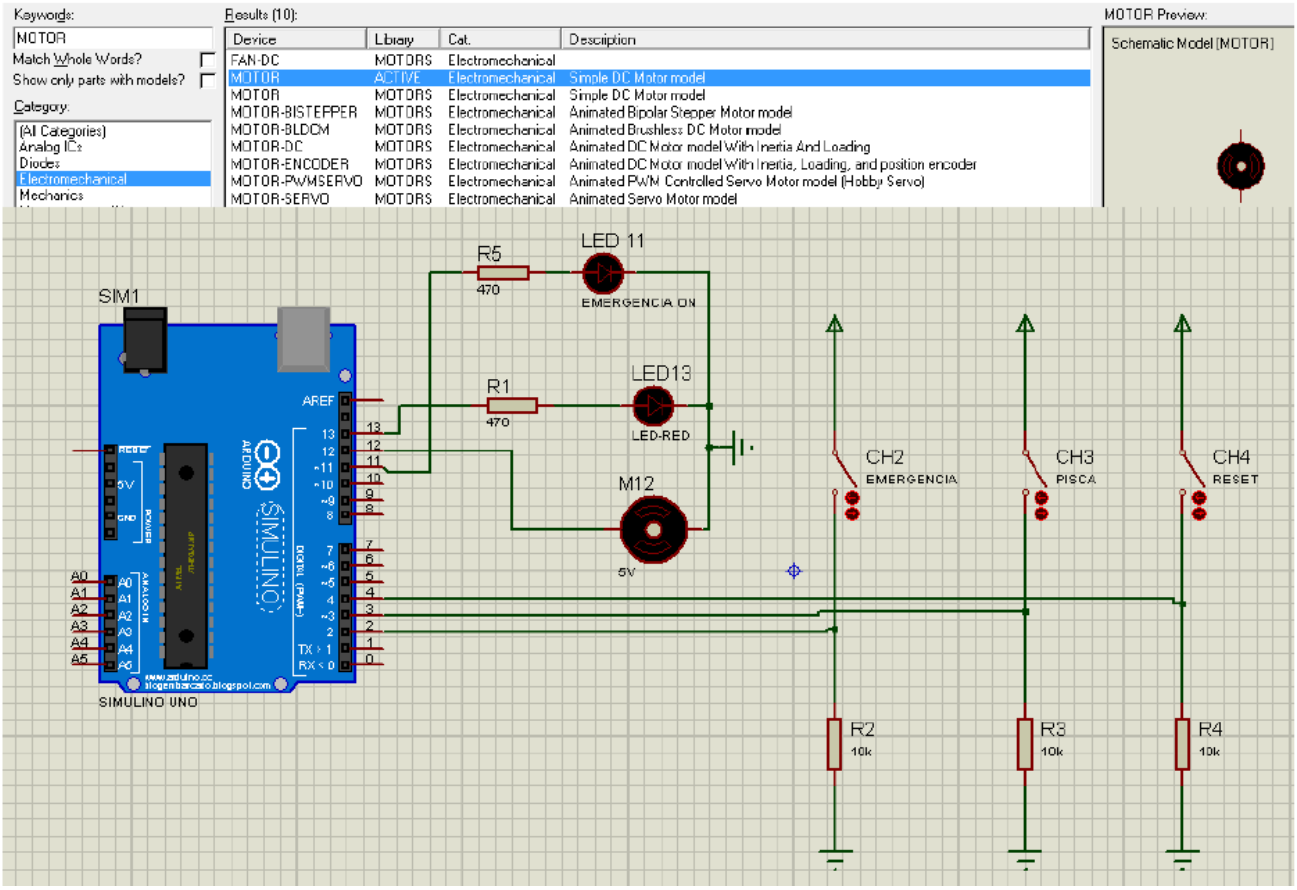
```
//exemplo de interrupção externa
int led13=13;
int ch3=3;//chave liga pisca
volatile byte emergencia= LOW;//flag que indica emergencia foi pressionada

void setup() {
  // put your setup code here, to run once:
  pinMode(led13,OUTPUT);
  pinMode(ch3,INPUT);
  //Interrupção no pino 2 chamando a minha função myInterrupt,
  //sempre que a chave2 ligar.
  attachInterrupt(digitalPinToInterrupt(2),myInterrupt,RISING);
}

void loop() {
  // put your main code here, to run repeatedly:

  if(digitalRead(ch3)){
    if (emergencia==LOW){
      //se ligou a chave 4 e não foi apertada a emergencia liga pisca
      digitalWrite(led13,HIGH);
      delay(5000);
      digitalWrite(led13,LOW);
      delay(100);
    }

  }
  else{
    //senão desliga pisca
    digitalWrite(led13,LOW);//desliga o led
    emergencia=LOW;//desliga o flag de emergencia
  }
}

void myInterrupt(){
  emergencia=HIGH;//liga o flag de emergencia
  digitalWrite(led13,LOW);//desliga o led imediatamente
}
```

Desafio 7:

DESAFIO 8

Altere o projeto do exemplo incluindo um MOTOR ACTIVE DC (5V, 1200Ohm) uma chave de reset como é descrito no diagrama abaixo.

O funcionamento do circuito deverá ser o seguinte: quando a chave CH3 pisca estiver ligada o LED13 deverá piscar e o MOTOR12 deverá liga de forma contínua, quando a chave CH3 for desligada o motor e o pisca deverão desligar, isto irá ocorrer após o tempo de delay. Quando a chave de emergência for pressionada o motor e o LED do pisca deverão ser desligados imediatamente, e o LED 11 deverá ligar indicando que o botão de emergência foi pressionado. Em circuitos reais sempre que o botão de emergência for pressionado deverá haver uma chave de reset que o operador deverá pressionar para desligar o alarme, esta é a função da chave CH4 RESET, depois de tudo parado para voltar a funcionar o botão de reset deverá ser pressionado, quando este botão for pressionado o led de emergência deverá apagar e o flag de emergência deverá ser desligado permitindo que o equipamento volte a operar normalmente, ligando e desligando o motor pela chave CH3 pisca! Observe que o operador deverá ficar pressionando o botão de RESET até o LED de emergência apagar!



ANEXOS:

ANEXO 1: _BV(bit)

/ *

O _BV (bit) é um compilador de macro definido como:

#define _BV (bit) (1 << (bit))

No arquivo <Avr / sfr_defs.h> é incluído indiretamente através de <avr / io.h>.

Só funciona com o Arduino!

Ele significa Bit Value onde você liga o bit descrito nos parênteses.

Esta é uma instrução muito prática para alterar somente um bit de um registrador, como por exemplo, os registradores dos temporizadores internos, interrupções etc.

Para colocar o valor zero (desligar o bit) você deverá colocar o símbolo de inversão na frente ~(_BV(bit))

Exemplo.

Ajusta o temporizador 2 do Arduino UNO para “fast PWM”

TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);

TCCR2B = _BV(CS22);

* /